ICMP Usage in Scanning[1]

Or

Understanding some of the ICMP Protocol's Hazards

Ofir Arkin

The Sys-Security Group

Founder

http://www.sys-security.com
ofir.arkin@sys-security.com

ITCon – Information Technology Consultants[2]

Senior Security Analyst

http://www.itcon-ltd.com
ofir@itcon-ltd.com

Version 2.0

September 2000

[1] This is part of "Network Scanning Techniques", by Ofir Arkin. To be published during 2000 (http://www.sys-security.com).

[2] IT Con is a leading information security consultancy company in the E-Commerce area. For more information please contact global@itcon-ltd.com.

Table of Contents

Figures List

Table List

Diagram List

1.0 Introduction

1.1 Introduction to Version 1.0

The Internet Control Message Protocol is one of the *debate full* protocols in the TCP/IP protocol suite regarding its security hazards. There is no consent between the experts in charge for securing Internet networks (Firewall Administrators, Network Administrators, System Administrators, Security Officers, etc.) regarding the actions that should be taken to secure their network infrastructure in order to prevent those risks.

In this paper I have tried to outline what can be done with the ICMP protocol regarding scanning.

Scanning can be defined as: The determination of the characteristics of the target network such as identifying which systems are alive and reachable via the Internet, and what services they offer, using techniques such as ping sweeps, port scans, firewalking, trace routing, and operating system identification.

This operation eventually leads to the discovery of the network topology map of the attacked network (although we will cover methods directly aimed at network topology mapping).

The kind of information collected using scanning methods can be summarized with a few simple questions:

- "What hosts are alive?"
- "What services are running on those hosts?"
- "How those hosts are organized?"
- "What are the operating systems used on those hosts?"
- "What is the role of each host?"

The data collected allow a malicious computer attacker to identify the hosts (if any) on a target network that are running a network service, which may have a known vulnerability that may allow a remote exploit.

The sections in this paper are divided according to the various methods in scanning; Host Detection using the ICMP protocol; Advanced Host Detection using the ICMP protocol - Host Detection using ICMP error messages generated from probed machines; Inverse Mapping Using ICMP; Using Trace Route with ICMP ECHO; and The usage of ICMP in the Operating System Finger Printing process. In the last section I have described which ICMP traffic should be filtered on the Border Router and/or Firewall in order to eliminate/reduce the risks outlined in this paper.

The paper introduces new methods for Host Detection using ICMP error messages generated from probed machines and a new method for OS Finger Printing using ICMP.

I hope that this paper would educate people to eliminate some of the security hazards the ICMP protocol carries.

1.2 Introduction to Version 2.0

Quite a large number of new OS fingerprinting methods using ICMP, which I have found are introduced with this revision. Among those methods two can be used in order to identify Microsoft Windows 2000 machines; one would allow us to distinguish between Microsoft Windows operating system machines and the rest of the world, and another would allow us to distinguish

between SUN Solaris machines and the rest of the world[3]. I have also tried to be accurate as possible with data presented in this paper. Few tables have been added to the paper mapping the behavior of the various operating systems I have used. These tables describe the results I got from the various machines after querying them with the various tests introduced with this paper.

See section 1.3 for a full Changes list.

1.3 Changes from version 1.0

2.0 Host Detection Using the ICMP Protocol

- 2.3 Broadcast ICMP
  - Added a table describing which operating systems would answer an ICMP ECHO request aimed at the Broadcast address of the network they reside on.
- 2.4 Non-ECHO ICMP
  - Added Information Request and Reply as a valid Host Detection method.
- 2.4.2 ICMP Information Request and Reply
  - The actual Information (added a section).
- 2.4.3 ICMP Address Mask Request and Reply
  - Added SUN Solaris and networking devices examples.

- 2.5 Non-Echo ICMP Sweep
  - Added a table summarizing which operating systems would answer those queries.
- 2.6 Non-ECHO ICMP Broadcasts
  - Added the fact that “Hosts running an operating system, which answers requests aimed at the IP broadcast address…”
  - Added two tables describing which operating systems would answer to which type of ICMP queries aimed at the broadcast address of the network they reside on?

3.0 Host Detection Using ICMP Error messages generated from the probed machines

- 3.1 IP datagrams with bad IP Header fields
  - Added more information on various other fields which can be used for this purpose.

6.0 The Usage of ICMP in the operating system Finger Printing Process

- 6.1 Using Wrong Codes within ICMP Datagrams
  - 6.1.1 Using ICMP Timestamp Requests with Codes different than 0
  - 6.1.2 Listing ICMP query message types sent to different operating systems with the Code field !=0 and the answers (is any) we got.
- 6.2 Using ICMP Address Mask Requests (Identifying Solaris Machines)
- 6.3 TOSing OSs out of the Window / Fingerprinting Microsoft Windows 2000
- 6.7 Using ICMP Address Mask Requests
- 6.8 Using ICMP Information Requests
- 6.9 Identifying operating systems according to their replies for non-ECHO ICMP requests aimed at the broadcast address.
- 6.10 IP TTL Field Value with ICMP
  - 6.10.1 IP TTL Field Value with ICMP ECHO Replies
  - 6.10.2 IP TTL Field Value with ICMP ECHO Requests
- 6.11 DF Bit
- 6.12 DF Bit Echoing

[3] See Section 6 for more information.

2.0 Host Detection using the ICMP Protocol[4]

The Host Detection stage gives a malicious computer attacker crucial information by identifying the computers on the targeted network that are reachable from the Internet. This process belongs to the scanning stage, which is one of the first stages in the Information Gathering process. The information collected during this stage could later lead to an attempt to break in to one (or more) of the targeted network computers. This, if the information gathered would be sufficient for the malicious computer attacker.

2.1 ICMP ECHO (Type 8) and ECHO Reply (Type 0)

We can use an *ICMP ECHO* datagram to determine whether a target IP address is active or not, by simply sending an ICMP ECHO[5] (ICMP type 8) datagram to the targeted system and waiting to see if an *ICMP ECHO Reply* (ICMP type 0) is received. If an ICMP ECHO reply is received, it would indicate that the target is alive (few firewalls spoof ICMP ECHO replies from protected hosts); No response means the target is down or a filtering device is preventing the incoming ICMP ECHO datagram from getting inside the protected network or the filtering device prevents the initiated reply from reaching the Internet.

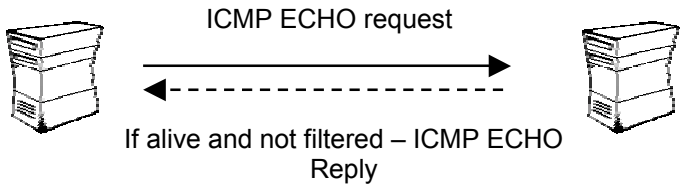


Figure 1: ICMP ECHO Mechanism

This mechanism is used by the Ping command to determine if a destination host is reachable.

In the next example two LINUX machines demonstrate the usage of Ping:

```
[root@stan /root]# ping 192.168.5.5
PING 192.168.5.5 (192.168.5.5) from 192.168.5.1 : 56(84) bytes of data.
64 bytes from 192.168.5.5: icmp_seq=0 ttl=255 time=4.4 ms
64 bytes from 192.168.5.5: icmp_seq=1 ttl=255 time=5.9 ms
64 bytes from 192.168.5.5: icmp_seq=2 ttl=255 time=5.8 ms

--- 192.168.5.5 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 4.4/5.3/5.9 ms
```

A Snort trace[6]:

```
01/26-13:16:25.746316 192.168.5.1 -> 192.168.5.5
```

[4] For more information about the ICMP Protocol please read “Appendix A: The ICMP Protocol”.

[5] From a technical point of view: The sending side initializes the identifier (used to identify ECHO requests aimed at different destination hosts) and sequence number (if multiple ECHO requests are sent to the same destination host), adds some data (arbitrary) to the data field and sends the ICMP ECHO to the destination host. In the ICMP header the code equals zero. The recipient should *only change* the type to ECHO Reply and return the datagram to the sender.

[6] Snort, written by Martin Roesch, can be found at http://www.snort.org.

```
ICMP TTL:64 TOS:0x0 ID:6059
ID:5721   Seq:1  ECHO
89 D7 8E 38 27 63 0B 00 08 09 0A 0B 0C 0D 0E 0F  ...8'c..........
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F  ................
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F   !"#$%&'()*+,-./
30 31 32 33 34 35 36 37                          01234567

01/26-13:16:25.746638 192.168.5.5 -> 192.168.5.1
ICMP TTL:255 TOS:0x0 ID:6072
ID:5721   Seq:1  ECHO REPLY
89 D7 8E 38 27 63 0B 00 08 09 0A 0B 0C 0D 0E 0F  ...8'c..........
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F  ................
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F   !"#$%&'()*+,-./
30 31 32 33 34 35 36 37                          01234567
```

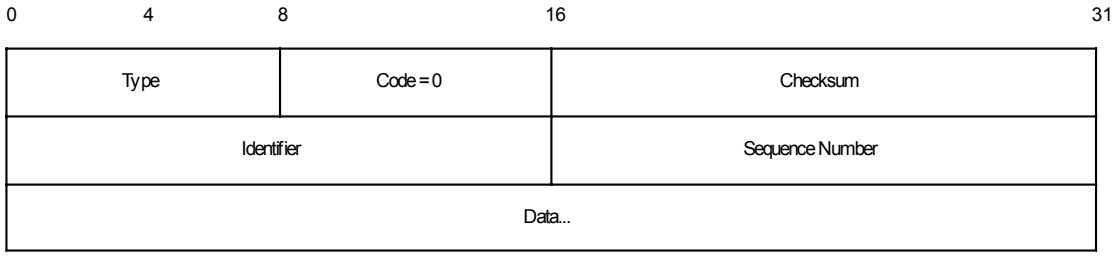


Figure 2: ICMP ECHO Request & Reply message format

Countermeasure: Block ICMP ECHO requests coming from the Internet towards your network at your border router and/or Firewall[7].

2.2 ICMP Sweep (Ping Sweep)

Querying multiple hosts using ICMP ECHO is referred to as *ICMP Sweep* (or *Ping Sweep*).

For a small to midsize network the Ping utility is an acceptable solution to this kind of host detection, but with large networks (such as Class A, or a full Class B) this kind of scan is fairly slow mainly because Ping waits for a reply (or a time out to be reached) from the probed host before proceeding to the next one.

fping[8] is a UNIX utility which sends parallel mass ECHO requests in a round robin fashion enabling it to be significantly faster than the usual Ping utility. It can also be fed with IP addresses with its accompanied tool *gping*. gping is used to generate a list of IP addresses which would be later fed into fping, directly or from a file, to perform the ICMP sweep. fping is also able to resolve hostnames of the probed machines if using the –d option.

Another UNIX tool that is able of doing an ICMP sweep in parallel, resolve the hostnames of the probed machines, save it to a file and a lot more is NMAP[9], written by Fyodor.

---

[7] It is better to filter unwanted traffic at your border router, reducing traffic rates for your firewall.

[8] ftp://ftp.tamu.edu/pub/Unix/src

[9] http://www.insecure.org

For the Microsoft Windows operating system a notable ICMP sweep tool is Pinger from Rhino9[10], able of doing what fping and NMAP do regarding this kind of scan.

Trying to resolve the names of the probed machines may discover the malicious computer attacker's IP number used for the probing, using the log of the authoritative DNS server.

The next example demonstrates the usage of NMAP to perform an ICMP sweep[11] against 20 IP addresses. Our test lab contains two LINUX machines running Redhat Linux v6.1, Kernel 2.2.12 (Stan & Kenny) and one Windows NT WRKS SP4 (Cartman). As it can be seen all of the machines answered the probe:

```
[root@stan /root]# nmap -sP -PI 192.168.5.1-20

Starting nmap V. 2.3BETA13 by fyodor@insecure.org (
www.insecure.org/nmap/ )
Host stan.sys-security.com (192.168.5.1) appears to be up.
Host kenny.sys-security.com (192.168.5.5) appears to be up.
Host cartman.sys-security.com (192.168.5.15) appears to be up.
Nmap run completed -- 20 IP addresses (3 hosts up) scanned in 3 seconds
```

If we wish to avoid the automatic resolving done by NMAP we should use the –n option to eliminate it.

ICMP sweeps are easily detected by IDS (Intrusion Detection Systems) whether launched in the regular way, or if used in a parallel way.

Countermeasure: Block ICMP ECHO requests coming from the Internet towards your network at your border router and/or Firewall.

2.3 Broadcast ICMP

A simpler way to map a targeted network for alive hosts is by sending an ICMP ECHO request to the broadcast address or to the network address of the targeted network.

The request would be broadcasted to all hosts on the targeted network. The alive hosts will send an ICMP ECHO Reply to the prober's source IP address (additional conditions apply here).

The malicious computer attacker has to send only one IP packet to produce this behavior.

This technique of host detection is applicable only to some of the UNIX and UNIX-like hosts of the targeted network. Microsoft Windows based machines will not generate an answer (ICMP ECHO Reply) to an ICMP ECHO request aimed at the broadcast address or at the network address. They are configured not to answer those queries out-of-the box (This applies to all Microsoft Windows operating systems accept for Microsoft Windows NT 4.0 with service pack below SP4). This is not an abnormal behavior as RFC 1122[12] states that if we send an ICMP ECHO request to an IP Broadcast or IP Multicast addresses it *may* be silently discarded by a host.

[10] The Rhino9 group no longer exists. Their tools are available from a number of sites on the Internet.

[11] The –sP –PI options enable NMAP to perform only an ICMP Sweep. The default behavior when using the –sP option is different and includes the usage of TCP ACK host detection technique as well.

[12] RFC 1122: Requirements for Internet Hosts - Communication Layers, http://www.ietf.org/rfc/rfc1122.txt.

The next example demonstrates the behavior expected from hosts when sending an ICMP ECHO request to the broadcast address of a network. The two LINUX machines on our test lab answered the query while the Microsoft Windows NT 4.0 Workstation with SP6a machine silently ignored it.

```
[root@stan /root]# ping -b 192.168.5.255
WARNING: pinging broadcast address
PING 192.168.5.255 (192.168.5.255) from 192.168.5.1 : 56(84) bytes of
data.
64 bytes from 192.168.5.1: icmp_seq=0 ttl=255 time=4.1 ms
64 bytes from 192.168.5.5: icmp_seq=0 ttl=255 time=5.7 ms (DUP!)

--- 192.168.5.255 ping statistics ---
1 packets transmitted, 1 packets received, +1 duplicates, 0% packet
loss
round-trip min/avg/max = 4.1/4.9/5.7 ms
```

In the next example I have sent an ICMP ECHO request to the network address of the targeted network. The same behavior was produced. The LINUX machines answered the ICMP ECHO request while the Microsoft Windows NT 4.0 with SP6a machine ignored it.

```
[root@stan /root]# ping -b 192.168.5.0
WARNING: pinging broadcast address
PING 192.168.5.0 (192.168.5.0) from 192.168.5.1 : 56(84) bytes of data.
64 bytes from 192.168.5.1: icmp_seq=0 ttl=255 time=7.5 ms
64 bytes from 192.168.5.5: icmp_seq=0 ttl=255 time=9.1 ms (DUP!)

--- 192.168.5.0 ping statistics ---
1 packets transmitted, 1 packets received, +1 duplicates, 0% packet
loss
round-trip min/avg/max = 7.5/8.3/9.1 ms
```

Note: Broadcast ICMP may result in a *Denial-Of-Service* condition if a lot of machines response to the query at once.

A more accurate table that lists which operating systems would answer to an ICMP ECHO request aimed at their Network / Broadcast address is given below:

Operating System	Echo Request Broadcast
Debian GNU/ LINUX 2.2, Kernel 2.4 test 2	+
Redhat LINUX 6.2 Kernel 2.2.14	+
FreeBSD 4.0	-
FreeBSD 3.4	-
OpenBSD 2.7	-
OpenBSD 2.6	-
NetBSD	
Solaris 2.5.1	+
Solaris 2.6	+
Solaris 2.7	+

Operating System	Echo Request Broadcast
Solaris 2.8	+
HP-UX v10.20	+
AIX	
ULTRIX	
Windows 95	-
Windows 98	-
Windows 98 SE	-
Windows ME	-
Windows NT 4 WRKS SP 3	-
Windows NT 4 WRKS SP 6a	-
Windows NT 4 Server SP4	-
Windows 2000 Professional (and SP1)	-
Windows 2000 Server (and SP1)	-

Table 1: Which Operating Systems would answer to an ICMP ECHO Request aimed at the Broadcast Address of the Network they resides on?

Countermeasure: Block the IP directed broadcast on the border router.

2.4 Non-ECHO ICMP

ICMP ECHO is not the only ICMP query message type available with the ICMP protocol.

Non-ECHO ICMP messages are being used for more advanced ICMP scanning techniques (not only probing hosts, but network devices, such as a router, as well).

The group of ICMP query message types includes the following:

ECHO Request (Type 8), and Reply (Type 0)
Time Stamp Request (Type 13), and Reply (Type 14)
Information Request (Type 15), and Reply (Type 16)
Address Mask Request (Type 17), and Reply (Type 18)
Router Solicitation (Type 10), and Router Advertisement (Type 9)

2.4.1 ICMP Time Stamp Request (Type 13) and Reply (Type 14)

The *ICMP Time Stamp Request and Reply* allows a node to query another for the current time. This allows a sender to determine the amount of latency that a particular network is experiencing. The sender initializes the identifier (used to identify Timestamp requests aimed at different destination hosts) and sequence number (if multiple Timestamp requests are sent to the same destination host), sets the originate time stamp and sends it to the recipient.

The receiving host fills in the receive and transmit time stamps, change the type of the message to time stamp reply and returns it to the recipient. The time stamp is the number of milliseconds elapsed since midnight UT (GMT).

The originate time stamp is the time the sender last touched the message before sending it, the receive time stamp is the time the recipient first touched it on receipt, and the Transmit time stamp is the time the receiver last touched the message on sending it.

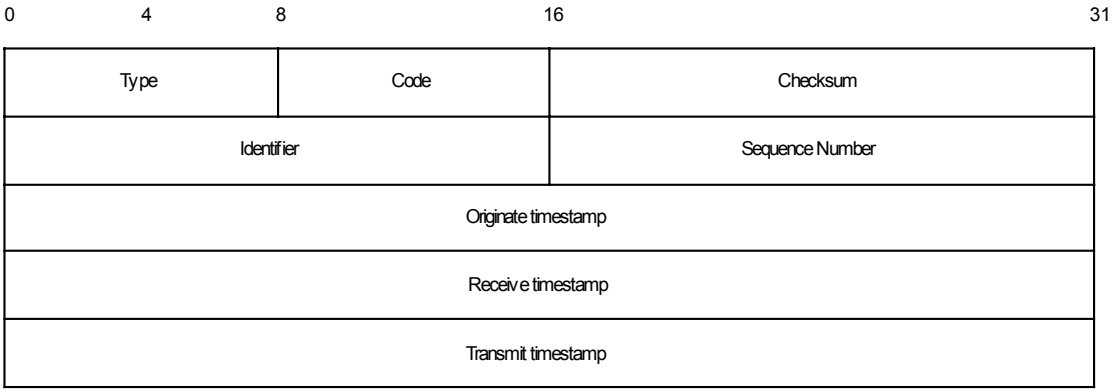


Figure 3: ICMP Time Stamp Request & Reply message format

As RFC 1122 state, a *host may* implement Timestamp and Timestamp Reply. If they are implemented a host must follow this rules:

- Minimum variability delay in handling the Timestamp request.
- The receiving host *must* answer to every Timestamp request that he receives.
- An ICMP Timestamp Request to an IP Broadcast or IP Multicast address *may* be silently discarded.
- The IP source address in an ICMP Timestamp reply *must* be the same as the specific-destination address of the corresponding Timestamp request message.
- If a source-route option is received in a Timestamp request, the return route *must* be reserved and used as a Source Route option for the Timestamp Reply option.
- If a Record Route and/or Timestamp option is received in a Timestamp request, this option(s) *should* be updated to include the current host and included in the IP header of the Timestamp Reply message.

Receiving an ICMP Timestamp Reply would reveal an alive host (or a networking device) that has implemented the ICMP Timestamp messages.

In the next example I have sent an ICMP Time Stamp Request, using the icmpush[13] tool, to a Redhat 6.1 LINUX, Kernel 2.2.12 machine:

```
[root@stan /root]# icmpush -tstamp 192.168.5.5
kenny.sys-security.com -> 13:48:07
```

Snort Trace:

```
01/26-13:51:29.342647 192.168.5.1 -> 192.168.5.5
ICMP TTL:254 TOS:0x0 ID:13170
TIMESTAMP REQUEST
88 16 D8 D9 02 8B 63 3D 00 00 00 00 00 00 00 00  ......c=........

01/26-13:51:29.342885 192.168.5.5 -> 192.168.5.1
ICMP TTL:255 TOS:0x0 ID:6096
```

[13] Icmpush was written by Slayer of hispahack.http://hispahack.ccc.de/ .

```
TIMESTAMP REPLY
88 16 D8 D9 02 8B 63 3D 02 88 50 18 02 88 50 18  ......c=..P...P.
2A DE 1C 00 A0 F9                                *.....
```

When I have sent an ICMP Time Stamp Request to a Windows NT WRKS 4.0 SP4 machine, I got no reply. Again, this is not an abnormal behavior from the Microsoft Windows NT machine, just an implementation choice as RFC 1122 states.

Countermeasure: Block ICMP Time Stamp Requests coming from the Internet on the border Router and/or Firewall.

2.4.2 ICMP Information Request (Type 15) and Reply (Type 16)

The *ICMP Information Request/Reply* pair was intended to support self-configuring systems such as diskless workstations at boot time, to allow them to discover their network address.

The sender fills in the request with the Destination IP address in the IP Header set to zero (meaning this network). The request may be sent with both Source IP Address and Destination IP Address set to zero. The sender initializes the identifier and the sequence number, both used to match the replies with the requests, and sends out the request. The ICMP header code field is zero.

If the request was issued with a non-zero Source IP Address the reply would only contain the network address in the Source IP Address of the reply. If the request had both the Source IP Address and the Destination IP Address set to zero, the reply will contain the network address in both the source and destination fields of the IP header.

From the description above one can understand that the ICMP Information request and reply mechanism was intended to be used locally.

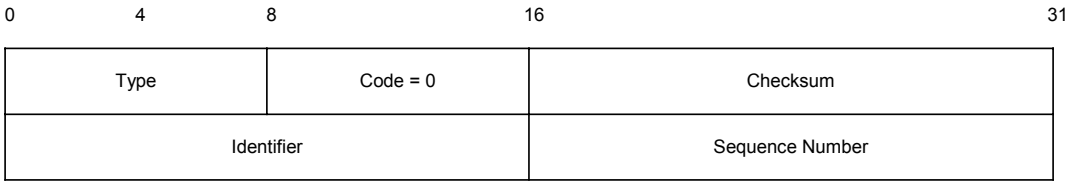


Figure 4: ICMP Information Request & Reply message format

The RARP, BOOTP & DHCP protocols provide better mechanisms for hosts to discover its own IP address.

The Information Request & Reply mechanism is now obsolete as stated in RFC 1122, and RFC 1812[14]. A *router should not* originate or respond to these messages; A *host should not* implement these messages.

Demands on one hand and reality on the other.

[14] RFC 1812: Requirements for IP Version 4 Routers, http://www.ietf.org/rfc/rfc1812.txt . As the RFC states this mechanism is now obsolete - A *router should not* originate or respond to these messages; A *host should not* implement these messages.

RFC 792 specifies that the Destination IP address should be set to zero, this mean that hosts that do not reside on the same network cannot send these ICMP query type.

But what would happen if we would send an ICMP Information Request with the Destination IP address set to a specific IP address of a host out in the void?

The next example illustrates that some operating systems would answer these queries even if not issued from the same network. The ICMP Information Request queries we are sending are not really RFC compliant because of the difference in the Destination IP address.

Those operating systems that answer our queries work in contrast to the RFC guidelines as well. We would see in the next example why.

In the next example I have sent an ICMP Information Request, using the SING[15] tool, to an AIX machine:

```
[root@aik icmp]# ./sing -info host_address[16]
SINGing to host_address (ip_address): 8 data bytes
8 bytes from ip_address: icmp_seq=0 ttl=238 Info Reply
8 bytes from ip_address: icmp_seq=1 ttl=238 Info Reply
8 bytes from ip_address: icmp_seq=2 ttl=238 Info Reply
8 bytes from ip_address: icmp_seq=3 ttl=238 Info Reply

--- host_address sing statistics ---
5 packets transmitted, 4 packets received, 20% packet loss
```

The tcpdump trace:

```
19:56:37.943679 ppp0 > slip139-92-208-21.tel.il.prserv.net >
host_address: icmp: information request
                        4500 001c 3372 0000 ff01 18a7 8b5c d015
                        xxxx xxxx 0f00 bee3 321c 0000
19:56:38.461427 ppp0 < host_address > slip139-92-208-
21.tel.il.prserv.net: icmp: information reply
                        4500 001c 661b 0000 ee01 f6fd xxxx xxxx
                        8b5c d015 1000 bde3 321c 0000
```

Lets do a quick analysis of the trace.

The ICMP Information Request:

Value	Field	Additional Information
4	4-Bit Version	IP Version 4
5	4-Bit Header Length	4 x DWORD = 20 Bytes
00	8-Bit TOS	TOS=0
00 1c	16-Bit Total Length	
33 72	16-Bit Identification	
00 00	3-Bit Flags + 13-bit Fragment Offset	
ff	8-Bit TTL	TTL=255
01	8-Bit Protocol	1=ICMP

[15] SING written by Alfredo Andre‧s Omella, can be found at http://sourceforge.net/projects/sing.
[16] Since I have queried a production system for this test, with a permission of the owners, I do not wish to identify it.

18 a7	16-Bit Header Checksum	
8b 5c d0 15	32-bit Source IP Address	139.92.208.21
xx xx xx xx	32-Bit Destination IP Address	
0f	8-Bit Type	Type=15
00	8-Bit Code	Code=0
be e3	16-Bit Checksum	
32 1c	16-Bit Identifier	
00 00	16-Bit Sequence Number	

The ICMP Information Reply:

Value	Field	Additional Information
4	4-Bit Version	IP Version 4
5	4-Bit Header Length	4 x DWORD = 20 Bytes
00	8-Bit TOS	TOS=0
00 1c	16-Bit Total Length	
66 1b	16-Bit Identification	
00 00	3-Bit Flags + 13-bit Fragment Offset	
ee	8-Bit TTL	TTL=238
01	8-Bit Protocol	1=ICMP
F6 fd	16-Bit Header Checksum	
xx xx xx xx	32-bit Source IP Address	
8b 5c d0 15	32-Bit Destination IP Address	139.92.208.21
10	8-Bit Type	Type=16
00	8-Bit Code	Code=0
bd e3	16-Bit Checksum	
32 1c	16-Bit Identifier	
00 00	16-Bit Sequence Number	

Instead of having the network address in the Source IP Address we are getting the IP address of the host.

Does the reply compliant with RFC 792 regarding this issue? Basically yes, because the RFC does not specify an accurate behavior.

The RFC states: “To form a information reply message, the source and destination addresses are simply reversed, the type code changes to 16, and the checksum recomputed”.

This means that if the ICMP Information Request is coming from outside (Destination is not zero) of the network in question, the network address would not be revealed. But still a host could be revealed if he answers the request.

The request is not compliant with the RFC in my opinion because it does not fulfill its job – getting the network address.

Countermeasure: Block ICMP Information Requests coming from the Internet on the border Router and/or Firewall.

2.4.3 ICMP Address Mask Request (Type 17) and Reply (Type 18)

The *ICMP Address Mask Request* (and Reply) is intended for diskless systems to obtain its subnet mask in use on the local network at bootstrap time. Address Mask request is also used when a node wants to know the address mask of an interface. The reply (if any) contains the mask of that interface.

Once a host has obtained an IP address, it could than send an Address Mask request message to the broadcast address of the network they reside on (255.255.255.255). Any host on the network that has been configured to send address mask replies will fill in the subnet mask, change the type of the message to address mask reply and return it to the sender.

RFC 1122 states that the Address Mask request & reply query messages are entirely optional.

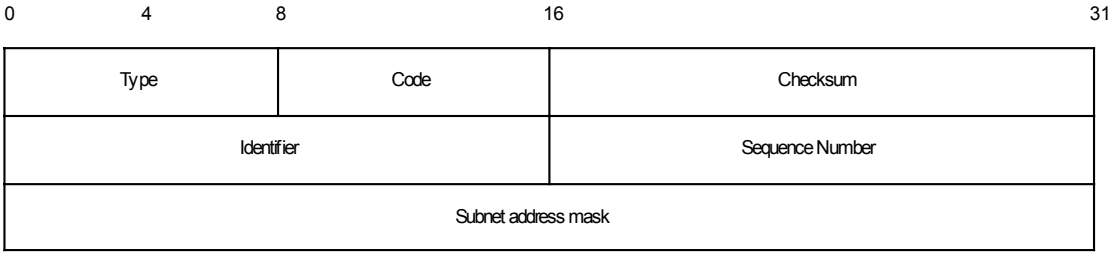


Figure 5: ICMP Address Mask Request & Reply message format

RFC 1122 also states that a system that has implemented ICMP Address Mask messages *must not* send an Address Mask Reply unless it is an authoritative agent for address masks.

Usually an Address Mask request would be answered by a gateway.

Receiving an Address Mask Reply from a host would reveal an alive host that is an authoritative agent for address masks. It will also allow a malicious computer attacker to gain knowledge about your network's configuration. This information can assist the malicious computer attacker in determining your internal network structure, as well as the routing scheme.

Please note that a Router *must* implement ICMP Address Mask messages. This will help identify routers along the path to the targeted network (it can also reveal internal routers if this kind of traffic is allowed to reach them).

If the Router is following RFC 1812 closely, it should not forward on an Address Mask Request to another network.

ICMP Address Mask Request aimed at a LINUX machine would not trigger an ICMP Address Mask Reply, nor a request aimed at a Microsoft Windows NT 4 Workstation SP 6a box.

In the next example I have sent an ICMP Address Mask Request to the broadcast address (192.168.5.255) of a class C network 192.168.5.0, spoofing the source IP to be 192.168.5.3:

```
[root@stan /root]# icmpush -vv -mask -sp 192.168.5.3 192.168.5.255
 -> ICMP total size = 12 bytes
 -> Outgoing interface = 192.168.5.1
 -> MTU = 1500 bytes
 -> Total packet size (ICMP + IP) = 32 bytes
ICMP Address Mask Request packet sent to 192.168.5.255 (192.168.5.255)

Receiving ICMP replies ...
-------------------------------------------------
192.168.5.3 ...
   Type = Address Mask Request (0x11)
   Code = 0x0     Checksum = 0xBF87
```

```
     Id = 0x3B7       Seq# = 0x3CB0
--------------------------------------------------
icmpush: Program finished OK
```

The snort trace:

```
-*> Snort! <*-
Version 1.5
By Martin Roesch (roesch@clark.net, www.clark.net/~roesch)
Kernel filter, protocol ALL, raw packet socket
Decoding Ethernet on interface eth0
02/15-13:47:37.179276 192.168.5.3 -> 192.168.5.255
ICMP TTL:254 TOS:0x0 ID:13170
ADDRESS REQUEST
B9 03 8E 49 00 00 00 00                         ...I....
```

No answer was received from the LINUX machines or from the Microsoft Windows NT Workstation 4 SP 6a machine on our test lab.

When I have tried to map which operating systems would answer (if at all) the ICMP Address Mask Requests, I have discovered that SUN Solaris is very cooperative with this kind of query[17]:

```
[root@aik icmp]# ./sing -mask -c 1 IP_Address[18]
SINGing to IP_Address (IP_Address): 12 data bytes
12 bytes from IP_Address: icmp_seq=0 ttl=241 mask=255.255.255.0

--- IP_Address sing statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
[root@aik icmp]#
```

The Tcpdump trace:

```
20:02:07.402229 ppp0 > slip139-92-208-21.tel.il.prserv.net >
Host_Address: icmp: address mask request
                        4500 0020 3372 0000 ff01 70a7 8b5c d015
                        xxxx xxxx 1100 afe3 3f1c 0000 0000 0000
20:02:07.831426 ppp0 < Host_Address > slip139-92-208-
21.tel.il.prserv.net: icmp: address mask is 0xffffff00 (DF)
                        4500 0020 3617 4000 f101 3c02 xxxx xxxx
                        8b5c d015 1200 afe2 3f1c 0000 ffff ff00
```

Our two last examples would be an ICMP Address Mask request aimed at a router (which must implement ICMP Address Mask Messages) and at a switch.

The following is an Address Mask Request sent to a Cisco Catalyst 5505 with OSS v4.5:

```
inferno:/tmp# sing -mask -c 1 10.13.58.240
```

[17] The –c 1 option enable SING to send only one ICMP datagram. The parameter can be changed to any desired value.
[18] The real IP Address and the Host address were replaced.

```
SINGing to 10.13.58.240 (10.13.58.240): 12 data bytes
12 bytes from 10.13.58.240: icmp_seq=0 ttl=60 mask=255.255.255.0

--- 10.13.58.240 sing statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
inferno:/tmp#

inferno:~# tcpdump -tnxv -s 1600 icmp
tcpdump: listening on xl0
10.13.58.199 > 10.13.58.240: icmp: address mask request (ttl 255, id
13170)
0000 :  4500 0020 3372 0000   FF01 FE99 0A0D 3AC7    E.. 3r........:.
0010 :  0A0D 3AF0 1100 6BF7   8308 0000 0000 0000    ..:...k.........

10.13.58.240 > 10.13.58.199: icmp: address mask is 0xffffff00 (ttl 60,
id 20187)
0000 :  4500 0020 4EDB 0000   3C01 A631 0A0D 3AF0    E.. N...<..1..:.
0010 :  0A0D 3AC7 1200 6BF6   8308 0000 FFFF FF00    ..:...k.........
0020 :  0000 0000 0000 0000   0000 0000 0000         ..............
^C
79 packets received by filter
0 packets dropped by kernel
inferno:~#
```

The last example is an ICMP Address Mask request sent to an Intel 8100 ISDN Router on our network:

```
[root@aik icmp]# ./sing -mask 10.0.0.254
SINGing to 10.0.0.254 (10.0.0.254): 12 data bytes
12 bytes from 10.0.0.254: icmp_seq=0 ttl=64 mask=255.255.255.0
12 bytes from 10.0.0.254: icmp_seq=1 ttl=64 mask=255.255.255.0
12 bytes from 10.0.0.254: icmp_seq=2 ttl=64 mask=255.255.255.0

--- 10.0.0.254 sing statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
[root@aik icmp]#
```

The tcpdump trace:

```
[root@aik /root]# tcpdump -x icmp
Kernel filter, protocol ALL, datagram packet socket
tcpdump: listening on all devices
16:34:30.666687 eth0 > 10.0.0.105 > 10.0.0.254: icmp: address mask
request
                  4500 0020 3372 0000 ff01 7304 0a00 0069
                  0a00 00fe 1100 0afd e402 0000 0000 0000
16:34:30.667961 eth0 < 10.0.0.254 > 10.0.0.105: icmp: address mask is
0xffffff00
                  4500 0020 2cb7 0000 4001 38c0 0a00 00fe
                  0a00 0069 1200 0afc e402 0000 ffff ff00
                  0000 0000 0000 0000 0000 0000 0000
```

Countermeasure: Block ICMP Address Mask Requests coming from the Internet on the border Router and/or Firewall.

2.5 Non-ECHO ICMP Sweeps

We can query multiple hosts using a Non-ECHO ICMP query message type. This is referred as a Non-ECHO ICMP sweep.

Who would answer our query?

Hosts that answer to the following:

- Hosts that are in a listening state.
- Hosts running an operating system that implemented the Non-ECHO ICMP query message type that was sent.
- Hosts that are configured to reply to the Non-ECHO ICMP query message type (few conditions here as well, for example: RFC 1122 states that a system that implemented ICMP Address Mask messages *must not* send an Address Mask Reply unless it is an authoritative agent for address masks).

Given the conditions above, which host(s) would answer our queries?

Operating System	Info. Request	Time Stamp Request	Address Mask Request
Debian GNU/ LINUX 2.2, Kernel 2.4 test 2	-	+	-
Redhat LINUX 6.2 Kernel 2.2.14	-	+	-
FreeBSD 4.0	-	+	-
FreeBSD 3.4	-	+	-
OpenBSD	-	+	-
NetBSD			
Solaris 2.5.1	-	+	+
Solaris 2.6	-	+	+
Solaris 2.7	-	+	+
Solaris 2.8	-	+	+
HP-UX v10.20	+	+	-
AIX v4.x	+	+	-
ULTRIX 4.2 – 4.5	+	+	+
Windows 95	-	-	+
Windows 98	-	+	+
Windows 98 SE	-	+	+
Windows ME	-	+	-
Windows NT 4 WRKS SP 3	-	-	+
Windows NT 4 WRKS SP 6a		-	
Windows NT 4 Server SP 4	-	-	-
Windows 2000 Professional	-	+	-
Windows 2000 Server	-	+	-

Networking Devices	Info. Request	Time Stamp Request	Address Mask Request
Cisco Catalyst 5505 with OSS v4.5	+	+	+
Cisco Catalyst 2900XL with IOS 11.2	+	+	-
Cisco 3600 with IOS 11.2	+	+	-
Cisco 7200 with IOS 11.3	+	+	-
Intel Express 8100 ISDN Router	-	-	+

Table 2: non-ECHO ICMP Query of different Operating Systems and Networking Devices

Countermeasure: Block ICMP Information Requests, ICMP Address Mask Requests & ICMP Time Stamp Requests coming from the Internet on the border Router and/or Firewall.

2.6 Non-ECHO ICMP Broadcasts

We can send a Non-ECHO ICMP query message type to the broadcast address or to the network address of the targeted network.

The request would be broadcasted to all listening hosts on the targeted network.

Who would answer our query?

- Hosts that are in a listening state
- Hosts running an operating system that implemented the Non-ECHO ICMP query message type that was sent.
- Hosts that are configured to reply to the Non-ECHO ICMP query message type (few conditions here as well, for example: a host may discard Non-ECHO ICMP query message type requests targeted at the broadcast address. For example an ICMP Timestamp Request to an IP Broadcast or IP Multicast address *may* be silently discarded).

Given the conditions above, the answering hosts would almost always be UNIX and UNIX-like machines. SUN Solaris, HP-UX, and LINUX are the only operating systems, from the group of operating systems I have tested, that would answer to an ICMP Timestamp Request aimed at the broadcast address of a network. HP-UX would answer Information Requests aimed at the broadcast address of a network. Non-would answer to an ICMP Address Mask Request aimed at the broadcast address of a network.

Operating System	Info. Request Broadcast	Time Stamp Request Broadcast	Address Mask Request Broadcast
Debian GNU/ LINUX 2.2, Kernel 2.4 test 2	-	+	-
Redhat LINUX 6.2 Kernel 2.2.14	-	+	-
FreeBSD 4.0	-	-	-
FreeBSD 3.4			
OpenBSD 2.7	-	-	-
OpenBSD 2.6	-	-	-

Operating System	Info. Request Broadcast	Time Stamp Request Broadcast	Address Mask Request Broadcast
NetBSD			
Solaris 2.5.1	*	+	-
Solaris 2.6	*	+	-
Solaris 2.7	*	+	-
Solaris 2.8	*	+	-
HP-UX v10.20	+	+	-
AIX 4.x			
ULTRIX 4.2 – 4.5			
Windows 95			
Windows 98	-	-	-
Windows 98 SE	-	-	-
Windows ME	-	-	-
Windows NT 4 WRKS SP 3	-	-	-
Windows NT 4 WRKS SP 6a			
Windows NT 4 Server SP 4	-	-	-
Windows 2000 Professional (& SP1)	-	-	-
Windows 2000 Server (& SP1)	-	-	-

Table 3: Operating Systems, which would answer to requests, aimed at the Broadcast address

Networking Devices	Info. Request Broadcast	Time Stamp Request Broadcast	Address Mask Request Broadcast
Cisco Catalyst 5505 with OSS v4.5	+	+	+
Cisco Catalyst 2900XL with IOS 11.2	+	-	-
Cisco 3600 with IOS 11.2	+	-	-
Cisco 7200 with IOS 11.3	+	-	-
Intel Express 8100 ISDN Router	-	-	-

Table 4: Networking Devices, which would answer to requests, aimed at the Broadcast address

Countermeasure: Block the IP directed broadcast on the border router. Block ICMP Information Requests, ICMP Address Mask Requests & ICMP Time Stamp Requests coming from the Internet on the border Router and/or Firewall.

3.0 Advanced Host Detection using the ICMP Protocol (using ICMP Error Messages generated from the probed machines)

The advanced host detection methods rely on the idea that we can use various methods in order to elicit an ICMP Error Message back from a probed machine and discover its existence. Some of the methods described here are:

- Mangling IP headers
  - Header Length Field
  - IP Options Field
- Using non-valid field values in the IP header
  - Using valid field values in the IP header
- Abusing Fragmentation
- The UDP Scan Host Detection method

With the first method we are using bad IP headers in the IP datagram that would generate an ICMP Parameter Problem error back from the probed machine to the source IP address of the probing datagram. The second method use non-valid field values in the IP header in order to force the probed machine to generate ICMP Destination Unreachable error message back to the malicious computer attacker. The third method discussed uses fragmentation to trigger an ICMP Fragment Reassembly Time Exceeded error message from the probed machine. The last method uses the UDP Scan method to elicit ICMP Port Unreachable error message back from a closed UDP port(s) on the probed host(s).

When using some of those methods we can determine if a filtering device is present and some can even discover the Access Control List a Filtering Device is forcing on the protected network.

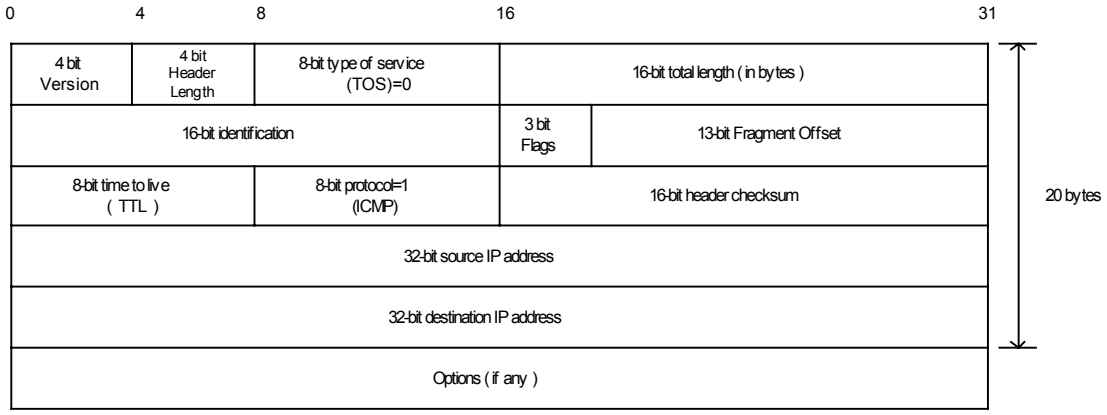


Figure 6: The IP Header

3.1 Sending IP Datagrams with bad IP headers fields – generating ICMP Parameter Problem error message back from probed machines

An ICMP Parameter Problem error message is sent when a router (*must* generate this message) or a host (*should* generate this message) process a datagram and finds a problem with the IP header parameters, which is not specifically covered by another ICMP error message. The ICMP Parameter Problem error message is only sent if the error caused the datagram to be discarded.

We have some variants with this type of Host Detection. We send an illegal forged datagram(s) with bad IP header field(s), that no specific ICMP error message is sent for this field(s). It will

force a Host to send back an ICMP Parameter Problem Error message with either Code 0 or Code 2 (When code 0 is used, the pointer field will point to the exact byte in the original IP Header, which caused the problem. Code 2 is sent when the Header length or the total packet length values of the IP datagram do not appear to be accurate) to the source IP address of the bad IP datagram and reveal its existence. With this type of host detection it is not relevant what would be the protocol (TCP/UDP/ICMP) embedded inside the IP datagram. All we care about is the ICMP Error messages generated by the probed machine (if any).

This method is very powerful in detecting host(s) on the probed network with direct access from the Internet, since a host should generate this error message. Routers must generate the ICMP Parameter Problem error message as well, but not all of them check the correctness of some fields inside the IP header like a host does (processing of some fields is done on the host only).

According to RFC 1122 a host should check for validity of the following fields when processing a packet[19]:

- Version Number – if not 4 a host must silently discard the IP packet.
- Checksum – a host should verify the IP header checksum on every received datagram and silently discard every datagram that has a bad checksum.

A router should check for the validity of the following fields when processing a packet[20]:

- Checksum – a router must verify the IP checksum of any packet it received, and must discard messages containing invalid checksums.

The conditions outlined eliminate the usage of this method to a limited number of fields only.

It is possible to send an IP datagram with bad field(s) in the IP header, which will get routed without getting dropped in the way to the probed machine. It should be noted that different routers perform different checks regarding the IP header (different implementation and interpretation of RFC 1812). When a router, because of a bad IP header, drops an IP packet and sends an ICMP Parameter Problem error message, it is possible to identify the manufacture of the router, and to adjust the wrong IP header field correctly according to a field, which is not checked by the manufacture of that particular router.

A router may be more forgiving than a Host regarding the IP header. This may result from the fact that a router is a vehicle for delivering the IP datagram and a Host is the Destination and the place where more processing on the datagram is done.

The downside for this method is the detection. Intrusion Detection Systems *should* alert you about abnormalities in the attacked network traffic, since not every day you receive IP packets with bad IP Header field(s).

We can use this type of Host Detection to sweep through the entire IP range of an organization and get back results, which will map all the alive hosts on the probed network with direct access from the Internet.

Even if a firewall or another filtering device is protecting the probed network we can still try to send those forged packets to an IP addresses with ports that are likely to be opened. For

[19] RFC 1122 – Requirements for Internet Host, http://www.ietf.org/rfc/rfc1122.txt.
[20] RFC 1812 – Requirements for IPv4 Routers, http://www.ietf.org/rfc/rfc1812.txt.

example - TCP ports 21,25,80; UDP port 53; and even try to send an ICMP message presumably coming back from a Host/Router who generated it upon receiving data from the attacked network.

In my opinion Firewalls/Filtering Devices should check the validity of those fields used to elicit the ICMP Parameter Problem error message and disallow this kind of traffic.

An example is given here using the *ISIC* tool written by Mike Frantzen[21]. ISIC sends randomly generated packets to a target computer. Its primary uses are to stress test an IP stack, to find leaks in a firewall, and to test the implementation of Intrusion Detection Systems and firewalls. The user can specify how often the packets will be fragmented; have IP options, TCP options, an urgent pointer, etc.

In the next example I have sent 20 IP Packets from a LINUX machine to a Microsoft Windows NT WRKS 4 SP4 machine. The datagrams were not fragmented nor bad IP version numbers were sent. The only weird thing sent inside the IP headers was random IP Header length, which have produced ICMP Parameter Problem Code 2 error message as I anticipated.

```
[root@stan packetshaping]# ./isic -s 192.168.5.5 -d 192.168.5.15 -p 20
-F 0 -V 0 -I 100
Compiled against Libnet 1.0
Installing Signal Handlers.
Seeding with 2015
No Maximum traffic limiter
Bad IP Version  = 0%            Odd IP Header Length    = 100%
Frag'd Pcnt     = 0%

Wrote 20 packets in 0.03s @ 637.94 pkts/s
```

tcpdump trace:

```
12:11:05.843480 eth0 > kenny.sys-security.com > cartman.sys-
security.com: ip-proto-110 226 [tos 0xe6,ECT]  (ttl 110, id 119,
optlen=24[|ip])

12:11:05.843961 eth0 P cartman.sys-security.com > kenny.sys-
security.com: icmp: parameter problem - octet 21 Offending pkt:
kenny.sys-security.com > cartman.sys-security.com: ip-proto-110 226
[tos 0xe6,ECT]  (ttl 110, id 119, optlen=24[|ip]) (ttl 128, id 37776)
```

Other fields we can use inside the IP Header
In the last example we have used a bad Header Length field value to generate an ICMP Parameter Problem code 2-error message.

An ICMP Parameter Problem would almost always result from an incorrect usage of the *IP option* field as well.

3.1.1 ACL Detection using IP Datagrams with bad IP headers fields

If we probe the entire IP range of the targeted network with all combinations of protocols and ports, it would draw us the targeted network topology map, and will allow us to determine the access list (ACL) a Filtering Device (If present, and not blocking outgoing ICMP Parameter Problem Error messages) is forcing.

[21] http://expert.cc.purdue.edu/~frantzen/

This, if the filtering device does not check the validity of the mangled IP header fields, and allows the specified traffic.

3.1.1.1 How we determine the ACL (ICMP Protocol embedded inside)?

When the embedded protocol is ICMP, we send various ICMP message types encapsulated inside IP datagrams with bad IP header(s). If we receive a reply from a Destination IP address we have a host that is alive and an ACL, which allows this type of message of ICMP to get to the host who generated the ICMP error message (and the Parameter Problem ICMP error message is allowed from the destination host to the Internet).

If we are not getting any reply than one of three possibilities:

- The Filtering Device disallows datagrams with the kind of bad field we are using.
- The Filtering Device is filtering the type of the ICMP message we are using.
- The Filtering Device blocks ICMP Parameter Problem error messages initiated from the protected network destined to the Internet.

3.1.1.2 How we determine the ACL (TCP or UDP Protocol embedded inside)?

We can probe for every combination of protocol and port values inside an IP packet with bad IP header(s). If we would receive an answer it would indicate that the protocol and port we used are allowed to the probed host from the Internet, and the ICMP Parameter Problem error message is allowed from the destination host in the protected network out to the Internet. It would also indicate that the filtering device used on the targeted network is not validating the correctness of the fields we have used in order to elicit the ICMP Parameter Problem error message.

If the embedded protocol were either TCP or UDP, a reply would not be generated if:

- The Filtering Device disallows packets with the kind of bad field we are using.
- The Filtering Device filters the Protocol used.
- The Filtering Device is filtering the specific port we are using for the probe.
- The Filtering Device blocks ICMP Parameter Problem error messages initiated from the protected network destined to the Internet. In our case, the filtering device may be blocking the specific host we are probing for outgoing ICMP Parameter Problem datagrams.

Note: If we are using the IP Header Length field in order to elicit ICMP Parameter Problem error message back from the probed host(s) than the host processing the datagram may not be able to access the Protocol information embedded inside. The reason would be the faulty calculation that would be made – where the header ends and the data portion begins.

Countermeasure: Block outgoing ICMP Parameter Problem from the protected network to the Internet on the Firewall & on the border Router.

Check with the manufacture of your filtering device which fields it validates on the IP header when processing a datagram.

3.2 IP Datagrams with non-valid field values

This Host Detection method is based on different IP header fields within the crafted IP datagram that would have non-valid field values, which would trigger an ICMP Destination Unreachable Error message back from the probed machines.

Note that some hosts (AIX, HP-UX, Digital UNIX) may not send ICMP Protocol Unreachable messages.

3.2.1 The Protocol Field example

3.2.1.1 Using non-Valid (not used) IP protocol values

One such field within the IP header is the protocol field. If we will put a value, which does not represent a valid protocol number, the probed machine would elicit an ICMP Destination Unreachable – Protocol Unreachable error message back to the probed machine.

By sending this kind of crafted packets to all IP addresses within the IP address range of the probed network we can map the hosts that are directly connected to the Internet (assuming that no filtering device is present, or filtering the specific traffic).

3.2.1.1.1 Detecting if a Filtering Device is present

A packet sent with a protocol value, which does not represent a valid protocol number, should elicit an ICMP Destination Unreachable – Protocol Unreachable from the probed machine. Since this value is not used (and not valid) all hosts probed, unless filtered or are AIX, HP-UX, Digital UNIX machines, should send this reply. If a reply is not received we can assume that a filtering device prevents our packet from reaching our destination or from the reply to reach us back.

3.2.1.2 Using all combination of the IP protocol filed values

The difference with this variant is that we use all of the combinations available for the IP protocol field – since the IP protocol field has only 8 bits in length, there could be 256 combinations available.

NMAP 2.54 Beta 1 has integrated this variant and Fyodor have named it - IP Protocol scan. NMAP sends raw IP packets *without any further protocol header* (no payload) to each specified protocol on the target machine. If an ICMP Protocol Unreachable error message is received, the protocol is not in use. Otherwise it is assumed it is opened (or a filtering device is dropping our packets).

If our goal was Host Detection only, than using the NMAP implementation would be just fine. But if we wish to use this scan type for other purposes, such as ACL detection, than we would need the payload data as well (the embedded protocol's data).

We can determine if a filtering device is present quite easily using this scan method. If a large number of protocols (non valid values could be among those) seems to be "opened"/used (not receiving any reply – ICMP Protocol Unreachable) than we can assume a filtering device is blocking our probes (if using a packet with the protocol headers as well). If the filtering device is blocking the ICMP Protocol Unreachable error messages initiated from the protected network towards the Internet than nearly all of the 256 possible protocol values would be seemed "opened"/used.

With the current implementation with NMAP the 256 possible protocol values should be "opened" when a scan is performed against a machine inside a protected network, because a packet filter firewall (or other kind of firewall) *should* block the probe since it lacks information to validate the traffic against its rule base (information in the protocol headers such as ports for example).

In the next example I have used NMAP 2.54 Beta 1 in order to scan a Microsoft Windows 2000 Professional machine:

```
[root@catman /root]# nmap -vv -sO 192.168.1.1

Starting nmap V. 2.54BETA1 by fyodor@insecure.org (
www.insecure.org/nmap/ )
Host  (192.168.1.1) appears to be up ... good.
Initiating FIN,NULL, UDP, or Xmas stealth scan against  (192.168.1.1)
The UDP or stealth FIN/NULL/XMAS scan took 4 seconds to scan 254 ports.
Interesting protocols on  (192.168.1.1):
(The 250 protocols scanned but not shown below are in state: closed)
Protocol   State       Name
1          open        icmp
2          open        igmp
6          open        tcp
17         open        udp

Nmap run completed -- 1 IP address (1 host up) scanned in 4 seconds
```

A tcpdump trace of some of the communication exchanged:

```
17:44:45.651855 eth0 > localhost.localdomain > 192.168.1.1: ip-proto-50
0 (ttl 38, id 29363)
17:44:45.652169 eth0 < 192.168.1.1 > localhost.localdomain: icmp:
192.168.1.1 protocol 50 unreachable Offending pkt:
localhost.localdomain > 192.168.1.1: ip-proto-50 0 (ttl 38, id 29363)
(ttl 128, id 578)
17:44:45.652431 eth0 > localhost.localdomain > 192.168.1.1: ip-proto-
133 0 (ttl 38, id 18)
17:44:45.652538 eth0 > localhost.localdomain > 192.168.1.1: ip-proto-
253 0 (ttl 38, id 36169)
17:44:45.652626 eth0 > localhost.localdomain > 192.168.1.1: ip-proto-92
0 (ttl 38, id 26465)
17:44:45.652727 eth0 < 192.168.1.1 > localhost.localdomain: icmp:
192.168.1.1 protocol 133 unreachable Offending pkt:
localhost.localdomain > 192.168.1.1: ip-proto-133 0 (ttl 38, id 18)
(ttl 128, id 579)
17:44:45.652760 eth0 > localhost.localdomain > 192.168.1.1: ip-proto-
143 0 (ttl 38, id 14467)
17:44:45.652899 eth0 > localhost.localdomain > 192.168.1.1: ip-proto-30
0 (ttl 38, id 30441)
17:44:45.652932 eth0 < 192.168.1.1 > localhost.localdomain: icmp:
192.168.1.1 protocol 253 unreachable Offending pkt:
localhost.localdomain > 192.168.1.1: ip-proto-253 0 (ttl 38, id 36169)
(ttl 128, id 580)
```

3.2.2 ACL Detection using the Protocol field

First we need to determine if a filtering device is present using a non-valid (not used) protocol number probe. If a filtering device exists then no answer (ICMP Protocol Unreachable) will be received from the probed machine, assuming it is not AIX, HP-UX or Digital UNIX[22].

[22] You can determine this using OS finger printing methods.

If a certain protocol were not allowed through the filtering device we would not receive any ICMP error message from the probed machine. Probing for all combinations of protocols and ports against an IP range of a targeted network using non-valid and valid protocol values can determine the ACL a filtering device is forcing on the protected network, along with the topology map of a targeted network (hosts reachable from the Internet).

A reply would not be generated if:

- The Filtering Device filters the Protocol we are using
- The Filtering Device is filtering the specific port we are using for the probe.
- The Filtering Device blocks ICMP Destination Unreachable - Protocol Unreachable error messages initiated from the protected network destined to the Internet. In our case, the filtering device may be blocking the specific host we are probing for outgoing ICMP Destination Unreachable - Protocol Unreachable error messages.

Note: We can use this method for ACL detection but if the protocol we are using is not used on the target machine it should be blocked on the filtering device. Than, only opened TCP/UDP ports and allowed ICMP traffic could traverse the filtering device. If this kind of traffic is allowed we can have better ACL detection solutions then we outlined here.

Countermeasure: Block outgoing ICMP Protocol Unreachable error messages coming from the protected network to the Internet on your Firewall and/or Border Router. If you are using a firewall check that your firewall block protocols which are not supported (deny all stance).

3.3 Host Detection using IP fragmentation to elicit Fragment Reassembly Time Exceeded ICMP error message.

When a host receives a fragmented datagram with some of its pieces missing, and does not get the missing part(s) within a certain amount of time the host will discard the packet and generate an ICMP Fragment Reassembly Time Exceeded error message back to the sending host.

We can use this behavior as a Host Detection method, by sending fragmented datagrams with missing fragments to a probed host, and wait for an ICMP Fragment Reassembly Time Exceeded error message to be received from a live host(s), if any.

When we are using this method against all of the IP range of a probed network, we will discover the network topology of that targeted network.

3.3.1 ACL Detection using IP fragmentation

This method can be used not only to map the entire topology map of the targeted network, but also to determine the ACL a firewall or a filtering device is forcing on the protected network.

Simply using all combinations of TCP and UDP with different ports, with the IP addresses from the IP range of the probed network will do it. When we receive a reply it means a host we queried is alive, the port we have used is opened on that host, and the ACL allows the protocol type and the port that was used to get to the probed machine (and the ICMP Fragment Reassembly Time Exceeded error message back from the probed machine to the Internet).

If we were not getting any reply back from the probed machine it can mean:

- The Filtering Device filters the Protocol used.
- The Filtering Device is filtering the specific port we are using for the probe.

- The Filtering Device blocks ICMP Fragment Reassembly Time Exceeded error messages initiated from the protected network destined to the Internet. In our case, the filtering device may be blocking the specific host we are probing for outgoing ICMP Parameter Problem datagrams.

3.3.1.1 An Example with UDP (Filtering Device Detection)

Since UDP is a stateless protocol it may be better suited for our needs here. The first datagram would be fragmented including enough UDP information in the first fragmented datagram that would be enough to verify the packet against a Firewall's Rule base. The second part of the datagram would not be sent. It would force any host that gets such a packet to send us back an ICMP Fragment Reassembly Time Exceeded error message when the time for reassembly exceeds.

If the port we were using were an open port, than the ICMP Fragment Reassembly Time Exceeded error message would be generated. If the port were closed then an ICMP Port Unreachable error message would be produced.

If a firewall is blocking our probed than *no reply* would be generated.

No reply would be an indication that traffic to the Host we probed is filtered.

3.3.1.2 An example with TCP

We can divide the first packet of the TCP handshake into two fragments. We would put enough TCP information in the first packet that would be enough to verify the packet against the Firewall's Rule base (this means the port numbers we are using are included in the packet). We will not send the second part of the packet, forcing any host that gets such a packet to send us back an ICMP Fragment Reassembly Time Exceeded error message when the time for reassembly exceeds. This would indicate the host is accessible by this kind of traffic, which is allowed using the port we have specified as the destination port[23].

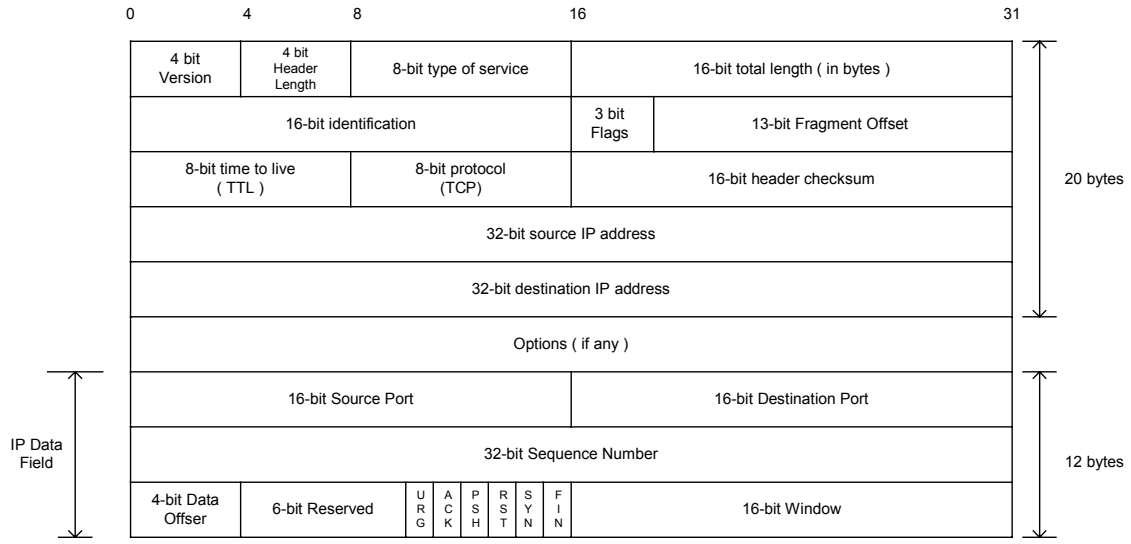


Figure 7: An Example: A TCP packet fragmented after only 12 bytes of TCP information

[23] In a case were a firewall is validating that the first packet is not fragmented, we can fragment another one instead. But than this scanning method would not be any different from any other scanning method using TCP flags combinations.

If the port we are using is open, than the ICMP error message would be generated. If the port is closed than a TCP RST packet should be sent back. If a filtering device were to block our probes than no reply would be generated. No reply would be an indication that traffic to the host we probed is filtered or the filtering device requires that the first TCP packet would not be fragmented (which is a legitimate requirement).

3.3.1.3 An Example with ICMP

We can do the same with encapsulating the ICMP protocol. When doing so the ICMP fragmented packets should sound the sirens when an Intrusion Detection system (if deployed) sees them. There is no reason to fragment an ICMP datagram.

If we think of sending fragmented ICMP through a bad filtering device product than we should at least include the first 4 bytes of the ICMP header with the IP datagram.

Countermeasure: Block outgoing ICMP Fragment Reassembly Time Exceeded Error messages.

3.4 Host Detection using UDP Scans, or why we wait for the ICMP Port Unreachable

How can we determine if a host is alive using a UDP probe? – We use the UDP scan method that uses ICMP Port Unreachable error message that may be generated from probed hosts as indicator of alive hosts. With this method we are sending a UDP datagram with 0 bytes of data to a UDP port on the attacked machine. If we have sent the datagram to a closed UDP port we will receive an ICMP Port Unreachable error message. If the port is opened, we would not receive any reply.

When a filtering device is blocking UDP traffic aimed at the attacked machine, it would copycat the behavior pattern as with opened UDP ports.

If we probe a large number of UDP ports on the same host and we do not receive a reply from a large number of ports, it would look like that a large number of probed UDP ports are opened. While a filtering device is probably blocking the traffic and nearly all of the ports are closed.

How can we remedy this?
We can set a threshold number of non-answering UDP ports, when reached we will assume a filtering device is blocking our probes.

Fyodor has implemented a threshold with NMAP 2.3 BETA 13, so when doing a UDP scan and not receiving an answer from a certain number of ports, it would assume a filtering device is monitoring the traffic, rather than reporting those ports as opened.

3.4.1 A Better Host Detection Using UDP Scan

We will take the UDP scan method and tweak it a bit for our needs. We know that a closed UDP port will generate an ICMP Port Unreachable error message indicating the state of the port - closed UDP port. We will choose a UDP port that should be definitely closed (according to the IANA list of assigned ports ftp://ftp.isi.edu/in-notes/iana/assignments/port-numbers). For example we can use port 0 (but it would reveal our probe pretty easily).

Based on the fact that sending a UDP datagram to a closed port should elicit an ICMP Port Unreachable, we would send one datagram to the port we have chosen, than:

- If no filtering device is present we will receive an ICMP Port Unreachable error message, which will indicate that the Host is alive.
- If no answer is given – a filtering device is covering that port.

In the next example I have used the HPING2[24] tool to send one UDP datagram to host 192.168.5.5 port 50, which was closed:

```
[root@stan /root]# hping2 -2 192.168.5.5 -p 50 -c 1
default routing not present
HPING 192.168.5.5 (eth0 192.168.5.5): udp mode set, 28 headers + 0 data
bytes
ICMP Port Unreachable from 192.168.5.5  (kenny.sys-security.com)

--- 192.168.5.5 hping statistic ---
1 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms


-*> Snort! <*-
Version 1.5
By Martin Roesch (roesch@clark.net, www.clark.net/~roesch)
Kernel filter, protocol ALL, raw packet socket
Decoding Ethernet on interface eth0
03/12-12:54:47.274096 192.168.5.1:2420 -> 192.168.5.5:50
UDP TTL:64 TOS:0x0 ID:57254
Len: 8

03/12-12:54:47.274360 192.168.5.5 -> 192.168.5.1
ICMP TTL:255 TOS:0xC0 ID:0
DESTINATION UNREACHABLE: PORT UNREACHABLE
00 00 00 00 45 00 00 1C DF A6 00 00 40 11 0F D4  ....E.......@...
C0 A8 05 01 C0 A8 05 05 09 74 00 32 00 08 6A E1  .........t.2..j.
```

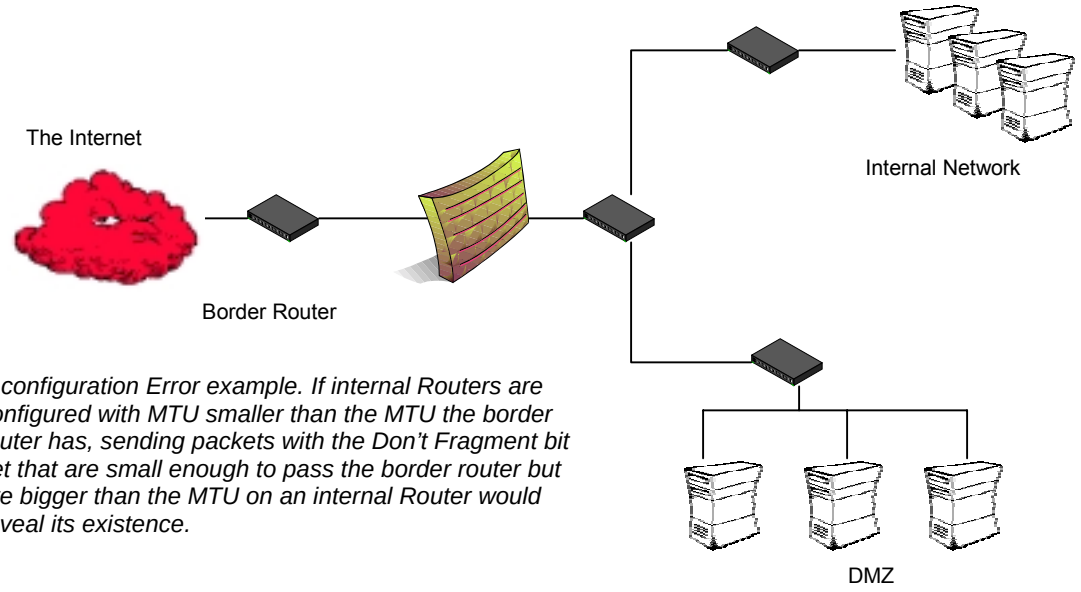
We can use the port number we have chosen, or a list of UDP ports that are likely not being used, and query all the IP range of an attacked network. Getting a reply back would reveal a live host. No reply would mean a filtering device is covering those hosts UDP traffic, and probably other protocols and hosts as well.

3.5 Using Packets bigger than the PMTU of internal routers to elicit an ICMP Fragmentation Needed and Don't Fragment Bit was Set (configuration problem)

If internal routers have a PMTU that is smaller than the PMTU for a path going through the border router, those routers would elicit an ICMP "Fragmentation Needed and Don't Fragment Bit was Set" error message back to the initiating host if receiving a packet too big to process that has the Don't Fragment Bit set on the IP Header, discovering internal architecture of the router deployment of the attacked network.

This is in my opinion a configuration problem causing a security hazard.

[24] HPING2 written by antirez, http://www.kyuzz.org/antirez/hping/ .



A configuration Error example. If internal Routers are configured with MTU smaller than the MTU the border router has, sending packets with the Don't Fragment bit set that are small enough to pass the border router but are bigger than the MTU on an internal Router would reveal its existence.

Figure 8: Using Packets bigger than the PMTU of internal routers to elicit an ICMP Fragmentation Needed and Don't Fragment Bit was Set

4.0 Inverse Mapping Using ICMP (ECHO & ECHO Reply)

Inverse Mapping is a technique used to map internal networks or hosts that are protected by a filtering devices/firewall. Usually some of those systems are not reachable from the Internet. We use routers, which will give away internal architecture information of a network, even if the question they were asked does not make any sense, for this scanning type. We compile a list of IP's that list what is not there and use it to conclude were things probably are.

A router looks at the IP address and makes decisions based on that solely.

We use two ICMP message types in order to use this technique. ICMP ECHO and ICMP ECHO Reply. We send a number of ICMP ECHO / ICMP ECHO Reply datagrams to different IP's we suspect are in the IP range of the network we are probing. When a router, either an exterior or interior, gets those ICMP message types for further processing, it looks at the IP address and makes decisions of routing based on it solely. When a router gets a datagram with an IP which is not used in the IP space / network segment of the part of the probed network he serves, the router will elicit an ICMP Host Unreachable (Generated by a router if a route to the destination host on a directly connected network is not available - does not respond to ARP) or ICMP Time Exceeded (Because the amount of time the Router waits for determining the destination host is unavailable have not been reached yet, but the TTL timer have turned 0 because of the time we wait for an answer) error message(s) back to the originator of the datagram. If we do not get an answer about a certain IP we can assume this IP exist inside the probed network[25].

We are using the ICMP ECHO Reply datagrams because most of the firewalls will let them pass through.

```
[root@cartman]# ./icmpush -vv -echo Target_IP[26]
 -> Outgoing interface = 192.168.1.5
 -> ICMP total size = 12 bytes
 -> Outgoing interface = 192.168.1.5
 -> MTU = 1500 bytes
 -> Total packet size (ICMP + IP) = 32 bytes
ICMP Echo Request packet sent to Target_IP (Target_IP)

Receiving ICMP replies ...
-------------------------------------------------------
Routers_IP  ...
         Type = Time Exceeded (0xB)
   Code = 0x0     Checksum = 0xF98F
     Id = 0x0         Seq# = 0x0
-------------------------------------------------------
./icmpush: Program finished OK


CMP TTL:254 TOS:0x0 ID:13170
ID:12291   Seq:317  ECHO


02/13-09:16:31.724400 Routers_IP -> 192.168.1.5
ICMP TTL:57 TOS:0x0 ID:7410
TTL EXCEEDED
```

[25] There is also a possibility that a filtering device is blocking our probes, or the replies.

[26] The real IP's of the targeted host and the Router were replaced because of legal problems that might arise when the ISP's personal that was used would understand it was one of their Routers used for this experiment.

```
00:13:12 prober> 192.168.2.5: icmp: echo reply
00:13:13 router> prober: icmp: host unreachable
```

Theoretically speaking, using any ICMP type in order to inverse map a network using a Router is possible. The downside would be that some Routers would filter unwanted traffic of certain ICMP types.

5.0 Using traceroute to Map a Network Topology

Traceroute is a Network debugging utility, which attempts to map all the hosts on a route to a certain destination host/machine.

The program sends UDP (by default) or ICMP ECHO Request[27] datagrams in sets of three, to a certain destination host. The first three datagram's to be sent have a Time-to-Live field value in the IP Header equals to one. The program lies on the fact that a router should decrement the TTL field value just before forwarding the datagram to another router/gateway.

If a router discovers that the Time-To-Live field value in an IP header of a datagram he process equals zero (or less) he would discard the datagram and generate an ICMP Time Exceeded Code 0 – transit TTL expired error message back to the originating host.

This is when a successful round is completed and another set of three datagrams is sent, this time with a Time-to-Live field value greater by one than the last set.

The originating host would know at which router the datagram expired since it receives this information with the ICMP Time Exceeded in Transit error message (Source IP address of the ICMP error message would be the IP address of the router/gateway; inside the IP header + 64 bits of original data of the datagram field we would have additional informaiton that would bound this ICMP error message to our issued traceroute command).

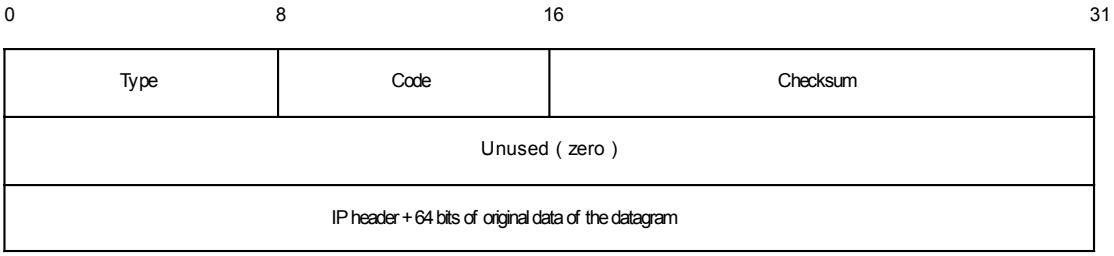


Figure 9: ICMP Time Exceeded message format

Since we increment the TTL field starting from one for each successful round (again - a round is finished when the ICMP Time Exceeded in Transit error message is received) until we receive an ICMP Port Unreachable error message (or ICMP ECHO Reply if we are using the ICMP ECHO request datagrams) from the destined machine, we map every router/gateway/host along the path to our destination.

By default, when sending UDP packets we use a destination port which is probably not used by the destination host so the UDP datagram would not be processes and an ICMP Port Unreachable error message would be generated from the destined machine. The destination port would be incremented with each probe sent.

We get ICMP responses provided there is no prohibitive filtering or any packet loss.

[27] Microsoft Windows NT and Microsoft Windows 2000 are using the tracert command, which use ICMP ECHO Request datagrams as its default.

The output we see is a line showing the Time-To-Live, the address of the gateway, and the round trip time of each probe. If we do not get a response back within 5 seconds an "*" is printed, which represents no answer.

A regular traceroute example with ICMP would be[28]:

```
zuul:~>traceroute –I 10.0.0.10
traceroute to 10.0.0.10 (10.0.0.10), 30 hops max, 40 byte
packets
1 10.0.0.1 (10.0.0.1) 0.540 ms 0.394 ms 0.397 ms
2 10.0.0.2 (10.0.0.2) 2.455 ms 2.479 ms 2.512 ms
3 10.0.0.3 (10.0.0.3) 4.812 ms 4.780 ms 4.747 ms
4 10.0.0.4 (10.0.0.4) 5.010 ms 4.903 ms 4.980 ms
5 10.0.0.5 (10.0.0.5) 5.520 ms 5.809 ms 6.061 ms
6 10.0.0.6 (10.0.0.6) 9.584 ms 21.754 ms 20.530 ms
7 10.0.0.7 (10.0.0.7) 89.889 ms 79.719 ms 85.918 ms
8 10.0.0.8 (10.0.0.8) 92.605 ms 80.361 ms 94.336 ms
9 10.0.0.9 (10.0.0.9) 94.127 ms 81.764 ms 96.476 ms
10 10.0.0.10 (10.0.0.10) 96.012 ms 98.224 ms 99.312 ms
```

Lets assume that a network is protected by a firewall, which blocks all incoming traffic except for traffic aimed at the DNS Machine's UDP port 53. If we would perform a regular traceroute aimed for the DNS machine's IP address, our UDP datagrams would be sent with a destination port, which is probably not used on the targeted machine, and probably blocked by a Firewall or another filtering device. The traces would stop at the firewall at the entrance point to the probed network.

```
zuul:~>traceroute 10.0.0.10
traceroute to 10.0.0.10 (10.0.0.10), 30 hops max, 40 byte
packets
1 10.0.0.1 (10.0.0.1) 0.540 ms 0.394 ms 0.397 ms
2 10.0.0.2 (10.0.0.2) 2.455 ms 2.479 ms 2.512 ms
3 10.0.0.3 (10.0.0.3) 4.812 ms 4.780 ms 4.747 ms
4 10.0.0.4 (10.0.0.4) 5.010 ms 4.903 ms 4.980 ms
5 10.0.0.5 (10.0.0.5) 5.520 ms 5.809 ms 6.061 ms
6 10.0.0.6 (10.0.0.6) 9.584 ms 21.754 ms 20.530 ms
7 10.0.0.7 (10.0.0.7) 89.889 ms 79.719 ms 85.918 ms
8 10.0.0.8 (10.0.0.8) 92.605 ms 80.361 ms 94.336 ms
9 * * *
10 * * *
```

We need to set the port number to 53 in order to reach the DNS server. Since the traceroute program increases the port number every time it sends a UDP datagram, we need to calculate the port number to start with, so when a datagram would be processed by the Firewall[29] and would be examined, it would have the appropriate port and other information needed to fit with the Access Control List. If we use a simple equation we can calculate the starting port:

(Target port – (number of hops * number of probes)) -1

The number of hops (gateways) from our probing machine to the firewall is taken from our earlier traceroute. We use three probes for every query with the same TTL value, each one of them uses a different destination port number.

[28] All examples taken from "A Traceroute-Like Analysis of IP Packet Responses to Determine Gateway Access Control Lists" by David Goldsmith and Michael Shiffman. No real examples were provided because of legal issues.

[29] A firewall should not elicit any reply for any traffic destined directly for him.

```
zuul:~>traceroute -p28 10.0.0.10
traceroute to 10.0.0.10 (10.0.0.10), 30 hops max, 40 byte packets
1 10.0.0.1 (10.0.0.1) 0.501 ms 0.399 ms 0.395 ms
2 10.0.0.2 (10.0.0.2) 2.433 ms 2.940 ms 2.481 ms
3 10.0.0.3 (10.0.0.3) 4.790 ms 4.830 ms 4.885 ms
4 10.0.0.4 (10.0.0.4) 5.196 ms 5.127 ms 4.733 ms
5 10.0.0.5 (10.0.0.5) 5.650 ms 5.551 ms 6.165 ms
6 10.0.0.6 (10.0.0.6) 7.820 ms 20.554 ms 19.525 ms
7 10.0.0.7 (10.0.0.7) 88.552 ms 90.006 ms 93.447 ms
8 10.0.0.8 (10.0.0.8) 92.009 ms 94.855 ms 88.122 ms
9 10.0.0.9 (10.0.0.9) 101.163 ms * *
10 * * *
```

But with the regular traceroute program we now face another difficulty. After the datagram have passed the ACL of the Firewall (and we assume the firewall lets ICMP TTL Exceeded messages out) and listed the outer leg of the Firewall itself as the next hop, the next UDP datagram sent would be with a different port number - Than again it would be blocked by the firewall.

A modification to the traceroute program has been made by Michael Shiffman[30] in order to stop the port incrementation. One side affect from sending traceroutes with a fixed port number, which is allowed on the firewalls ACL, is the final datagram, which normally would generate an ICMP Port Unreachable message now would not be generated since the UDP port would be in a listening state on the probed machine and would not provide an answer.

```
zuul:~>traceroute -S –p53 10.0.0.15
traceroute to 10.0.0.15 (10.0.0.15), 30 hops max, 40 byte
packets
1 10.0.0.1 (10.0.0.1) 0.516 ms 0.396 ms 0.390 ms
2 10.0.0.2 (10.0.0.2) 2.516 ms 2.476 ms 2.431 ms
3 10.0.0.3 (10.0.0.3) 5.060 ms 4.848 ms 4.721 ms
4 10.0.0.4 (10.0.0.4) 5.019 ms 4.694 ms 4.973 ms
5 10.0.0.5 (10.0.0.5) 6.097 ms 5.856 ms 6.002 ms
6 10.0.0.6 (10.0.0.6) 19.257 ms 9.002 ms 21.797 ms
7 10.0.0.7 (10.0.0.7) 84.753 ms * *
8 10.0.0.8 (10.0.0.8) 96.864 ms 98.006 ms 95.491 ms
9 10.0.0.9 (10.0.0.9) 94.300 ms * 96.549 ms
10 10.0.0.10 (10.0.0.10) 101.257 ms 107.164 ms 103.318 ms
11 10.0.0.11 (10.0.0.11) 102.847 ms 110.158 ms *
12 10.0.0.12 (10.0.0.12) 192.196 ms 185.265 ms *
13 10.0.0.13 (10.0.0.13) 168.151 ms 183.238 ms 183.458 ms
14 10.0.0.14 (10.0.0.14) 218.972 ms 209.388 ms 195.686 ms
15 10.0.0.15 (10.0.0.15) 236.102 ms 237.208 ms 230.185 ms
```

[30] http://www.packetfactory.net

6.0 The usage of ICMP in Active Operating System Fingerprinting Process

Finger Printing is the art of Operating System Detection.

A malicious computer attacker needs few pieces of information before lunching an attack. First, a target, a host detected using a host detection method. The next piece of information would be the services that are running on that host. This would be done with one of the Port Scanning methods. The last piece of information would be the operating system used by the host.

The information would allow the malicious computer attacker to identify if the targeted host is vulnerable to a certain exploit aimed at a certain service version running on a certain operating system.

In this section I have outlined the ICMP methods for this type of scan. Few methods are new and were discovered during this research.

6.1 Using Wrong Codes within ICMP datagrams (the ICMP ECHO request example)

An interesting detail I have discovered during the lab experiments I did when I have researched ICMP scanning is when a wrong code is sent along with the correct type of ICMP query message, different operating systems would send different code values back.

In the next example I have sent an ICMP Timestamp Request with code 38 instead of code 0 to a LINUX machine running Redhat LINUX 6.2 Kernel 2.2.14 (it was experimented with kernel 2.2.12 as well). The LINUX machine processed the query and sent the reply, with the code value set to 38. I was thinking that a check for the validity of the code field would be done on the targeted machine. Obviously I was wrong.

```
[root@stan /root]# icmpush -vv -tstamp -c 38 192.168.5.5
 -> Outgoing interface = 192.168.5.1
 -> ICMP total size = 20 bytes
 -> Outgoing interface = 192.168.5.1
 -> MTU = 1500 bytes
 -> Total packet size (ICMP + IP) = 40 bytes
ICMP Timestamp Request packet sent to 192.168.5.5 (192.168.5.5)

Receiving ICMP replies ...
kenny.sys-security.com -> Timestamp Reply transmited at 18:06:40
icmpush: Program finished OK


02/14-18:10:31.951977 192.168.5.1 -> 192.168.5.5
ICMP TTL:254 TOS:0x0 ID:13170
TIMESTAMP REQUEST
1D 04 9D 20 03 78 8C 8B 00 00 00 00 00 00 00 00  ... .x..........

02/14-18:10:31.952233 192.168.5.5 -> 192.168.5.1
ICMP TTL:255 TOS:0x0 ID:220
TIMESTAMP REPLY
1D 04 9D 20 03 78 8C 8B 03 75 03 00 03 75 03 00  ... .x...u...u..
8C 21 01 00 8C 21                                .!...!
```

I was looking for other ICMP query types, which the Microsoft Windows machine I had on my test lab could answer, since Microsoft Windows NT 4 Workstation SP 6a machines do not answer ICMP Timestamp request messages, I used ICMP ECHO Request instead.

I have queried my LINUX box (Redhat 6.2 with kernel 2.2.12) with ICMP ECHO request with the code field value set to 38 - LINUX Replied with code value set to 38 again. We can look at the tcpdump trace, the type and code fields are in bold type:

```
10:06:02.329509   lo < localhost.localdomain > localhost.localdomain:
icmp: echo request
                          4500 0020 3372 0000 fe01 0610 c0a8 0105
                          c0a8 0105 0826 675a 7402 0e20 0186 0cd7

10:06:02.329639   lo > localhost.localdomain > localhost.localdomain:
icmp: echo reply
                          4500 0020 096d 0000 ff01 2f15 c0a8 0105
                          c0a8 0105 0026 6f5a 7402 0e20 0186 0cd7
```

If we examine what RFC 792 requires, we see that LINUX does exactly that.

The sending side initializes the identifier (used to identify ECHO requests aimed at different destination hosts) and sequence number (if multiple ECHO requests are sent to the same destination host), adds some data (arbitrary) to the data field and sends the ICMP ECHO Request to the destination host. *In the ICMP header the code equals zero*. The recipient should *only change* the type to ECHO Reply and return the datagram to the sender.

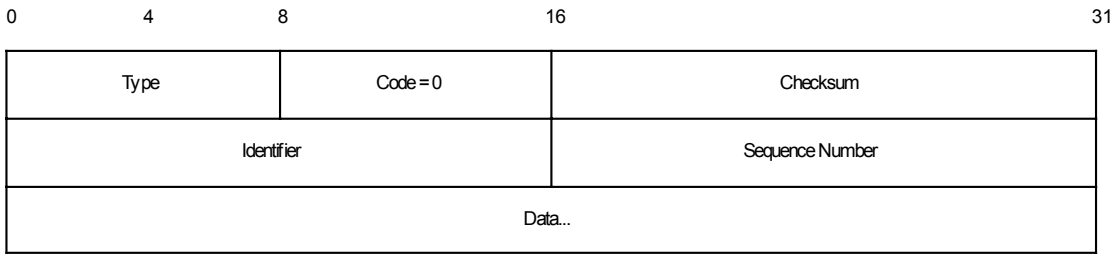


Figure 10: ICMP ECHO Request & Reply message format

This also means that we trust another machine to behave correctly.

LINUX changes the type field value to 0 and sends the reply. The code field is unchanged.

I have checked the behavior of my Microsoft Windows 2000 Professional box. I have sent the same ICMP ECHO Request message to the Microsoft Windows box (the code field is in bold type):

```
10:03:33.860212 eth0 > localhost.localdomain > 192.168.1.1: icmp: echo
request
                          4500 0020 3372 0000 fe01 0614 c0a8 0105
                          c0a8 0101 0826 d618 6102 f658 0183 c8e2
```

```
10:03:33.860689 eth0 < 192.168.1.1 > localhost.localdomain: icmp: echo
reply
                        4500 0020 2010 0000 8001 9776 c0a8 0101
                        c0a8 0105 0000 de3e 6102 f658 0183 c8e2
                        0000 0000 0000 0000 0000 0000 0000
```

The Microsoft Windows 2000 Professional operating system changed the code field value on the ICMP ECHO Reply to 0.

I have tested this method with various operating systems including LINUX Kernel 2.4t1-6, IBM AIX 4.x & 3.2, SUN Solaris 2.51, 2.6, 2.7 & 2.8, OpenBSD 2.6 & 2.7, NetBSD 1.4.2, BSDI BSD/OS 4.0 & 3.1, HP-UX 10.20 & 11.0, Compaq Tru64 v5.0, Irix 6.5.3 & 6.5.8, Ultrix 4.2-4.5, OpenVMS, FreeBSD 3.4, 4.0 & 4.1 and they produced the same results as the LINUX box (Kernel 2.2.x) did.

Microsoft Windows 4.0 Server SP4, Microsoft Windows NT 4.0 Workstation SP 6a, Microsoft Windows NT 4.0 Workstation SP3, Microsoft Windows 95 / 98 / 98 SE / ME have produced the same behavior as the Microsoft Windows 2000 Professional (Server & Advanced Server).

We have a method to differentiate between a Microsoft Windows box to the rest of the world using wrong codes inside ICMP ECHO Requests.

6.1.1 Using Wrong Codes with ICMP Datagrams (The ICMP Timestamp Request Example)

I have decided to map which operating systems would answer to an ICMP Timestamp Request that would have its code field not set to zero.

Interesting results were produced. The Microsoft Windows 98/98 SE/ME, and the Microsoft Windows 2000 Professional that have answered to ICMP Timestamp requests with the code field set to zero, now did not produce any reply back.

This enables us to group together certain versions of the Windows Operating System.

The next diagram shows how we can distinguish between the different Microsoft Windows operating systems using two datagrams of ICMP Timestamp request. The first one is a regular one; the Microsoft Windows machines that do not answer are Microsoft Windows 95 and Microsoft Windows NT 4.0 Workstation with SP 6a. All other operating systems answer the ICMP Time stamp request. The second stage is sending another datagram, this time with the Code field set to a value, which is not equal to zero. The operating systems that would not answer would include Windows 98/98 SE/ME/2000 Family, which are the newer versions of Microsoft Windows operating systems.

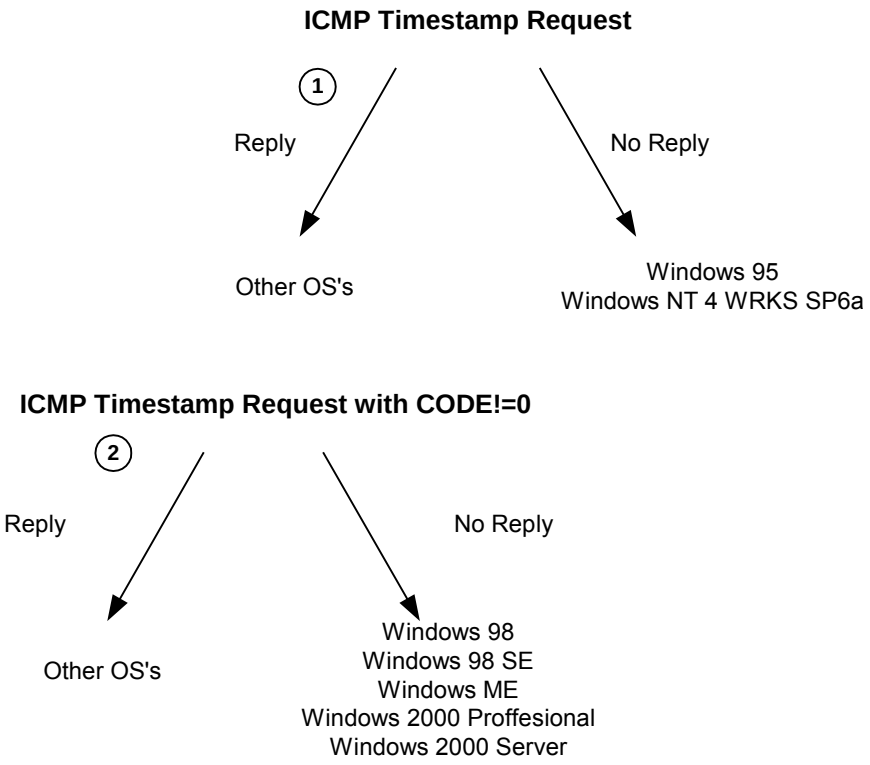


Diagram 1: Finger Printing Using ICMP Timestamp Request and Wrong Codes

It is quite obvious that Microsoft have tried to change some of their newer operating systems fingerprinting in later TCP/IP implementations of their operating systems. For example, the default for answering an ICMP Timestamp request was changed from "no answer" to "answer", like UNIX and UNIX-like operating systems. But the Microsoft programmers / designers / architects / security engineers did not think about every thing apparently.

6.1.2 Listing ICMP query message types sent to different operating systems with the Code field !=0 and the answers (is any) we got[31]

Operating System	Info. Request	Time Stamp Request	Address Mask Request
Debian GNU/ LINUX 2.2, Kernel 2.4 test 2	-	+	-
Redhat LINUX 6.2 Kernel 2.2.14	-	+	-
FreeBSD 4.0	-	+	-
FreeBSD 3.4	-	+	-
OpenBSD 2.7	-	+	-
OpenBSD 2.6	-	+	-
NetBSD	-	+	-
Solaris 2.5.1	*	+	+
Solaris 2.6	*	+	+
Solaris 2.7	*	+	+
Solaris 2.8	*	+	+

[31] Please see “Appendix D: ICMP Query Message types with Code field !=0 (table)”.

Operating System	Info. Request	Time Stamp Request	Address Mask Request
HP-UX 10.20	+	+	-
AIX 4.x	+	+	-
ULTRIX 4.2 – 4.5	+	+	+
Windows 95	-	-	+
Windows 98	-	- (CHANGE)	+
Windows 98 SE	-	- (CHANGE)	+
Windows ME	-	- (CHANGE)	-
Windows NT 4 WRKS SP 3	-	-	+
Windows NT 4 WRKS SP 6a	-	-	-
Windows NT 4 Server SP 4	-	-	-
Windows 2000 Professional	-	- (CHANGE)	-

Table 5: Using Wrong Codes when probing Non-ECHO Query ICMP Types

6.2 Using Fragmented ICMP Address Mask Requests (Identifying Solaris boxes)[32]

It appears that only some of the operating systems would answer an ICMP Address Mask Request as it is outlined in Table 2 in section 2.5. Those operating systems include - ULTRIX OpenVMS, Windows 95/98/98 SE/ME, NT below SP 4, and SUN Solaris. How can we distinguish between those who answer the request?

This is a regular ICMP Address Mask Request sent by SING to a SUN Solaris 2.7 machine:

```
[root@aik icmp]# ./sing -mask IP_Address
SINGing to IP_Address (IP_Address): 12 data bytes
12 bytes from IP_Address: icmp_seq=0 ttl=236 mask=255.255.255.0
12 bytes from IP_Address: icmp_seq=1 ttl=236 mask=255.255.255.0
12 bytes from IP_Address: icmp_seq=2 ttl=236 mask=255.255.255.0
12 bytes from IP_Address: icmp_seq=3 ttl=236 mask=255.255.255.0
12 bytes from IP_Address: icmp_seq=4 ttl=236 mask=255.255.255.0

--- IP_Address sing statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
```

All operating systems that would answer with ICMP Address Mask Reply would reply with the Address Mask of the network they reside on.

What would happen if we would introduce a little twist? Lets say we would send those queries fragmented?

In the next example, I have sent ICMP Address Mask Request to the same SUN Solaris 2.7 box, this time fragmented to pieces of 8 bytes of IP data. As we can see the answer I got was unusual:

```
[root@aik icmp]# ./sing -mask -c 2 -F 8 IP_Address
SINGing to IP_Address (IP_Address): 12 data bytes
12 bytes from IP_Address: icmp_seq=0 ttl=241 mask=0.0.0.0
12 bytes from IP_Address: icmp_seq=1 ttl=241 mask=0.0.0.0
```

[32] The Solaris portion was also discovered by Alfredo Andres Omella.

```
--- IP_Address sing statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
[root@aik icmp]#
```

The tcpdump trace:

```
20:02:48.441174 ppp0 > slip139-92-208-21.tel.il.prserv.net >
Host_Address: icmp: address mask request (frag 13170:8@0+)
                        4500 001c 3372 2000 ff01 50ab 8b5c d015
                        xxxx xxxx 1100 aee3 401c 0000
20:02:48.442858 ppp0 > slip139-92-208-21.tel.il.prserv.net >
Host_Address: (frag 13170:4@8)
                        4500 0018 3372 0001 ff01 70ae 8b5c d015
                        xxxx xxxx 0000 0000
20:02:49.111427 ppp0 < Host_Address > slip139-92-208-
21.tel.il.prserv.net: icmp: address mask is 0x00000000 (DF)
                        4500 0020 3618 4000 f101 3c01 xxxx xxxx
                        8b5c d015 1200 ade3 401c 0000 0000 0000

20:02:49.441492 ppp0 > slip139-92-208-21.tel.il.prserv.net >
Host_Address: icmp: address mask request (frag 13170:8@0+)
                        4500 001c 3372 2000 ff01 50ab 8b5c d015
                        xxxx xxxx 1100 ade3 401c 0100
20:02:49.442951 ppp0 > slip139-92-208-21.tel.il.prserv.net >
Host_Address: (frag 13170:4@8)
                        4500 0018 3372 0001 ff01 70ae 8b5c d015
                        xxxx xxxx 0000 0000
20:02:50.011433 ppp0 < Host_Address > slip139-92-208-
21.tel.il.prserv.net: icmp: address mask is 0x00000000 (DF)
                        4500 0020 3619 4000 f101 3c00 xxxx xxxx
                        8b5c d015 1200 ace3 401c 0100 0000 0000
```

The same SUN Solaris box now replies with a 0.0.0.0 as the Address Mask for the Network it resides on.

What would happen with the other operating systems?

They all would respond with the real Address Mask in their replies.

Here we got a distinction between SUN Solaris machines and the other operating systems that would answer those queries.

Important notice: When I have tested this method I have encountered some problems replicating the results with different ISPs. As it seems from analyzing the information I got, certain ISPs would block fragmented ICMP datagrams. This behavior would not enable this method to succeed. One way of testing this is to send a regular ICMP Echo request. We should watch for a response from the probed machine. If received, than we should send ICMP Echo request, this time fragmented. If no reply is received than your ISP is blocking ICMP fragments probably.

Note: When I have published this information in Bugtraq (August 5, 2000) Peter J. Holzer notified me that HP-UX 11.00 produce the same behavior as the SUN Solaris boxes. Darren Reed also noted that because SUN Solaris and HP-UX 11.0 share the same third party (Mentat) implementation for some of their TCP/IP stacks this behavior is produced by both.

We can further try to distinguish between the remaining operating systems. This, if we would use the !=0 code method I have introduced in section 6.1.

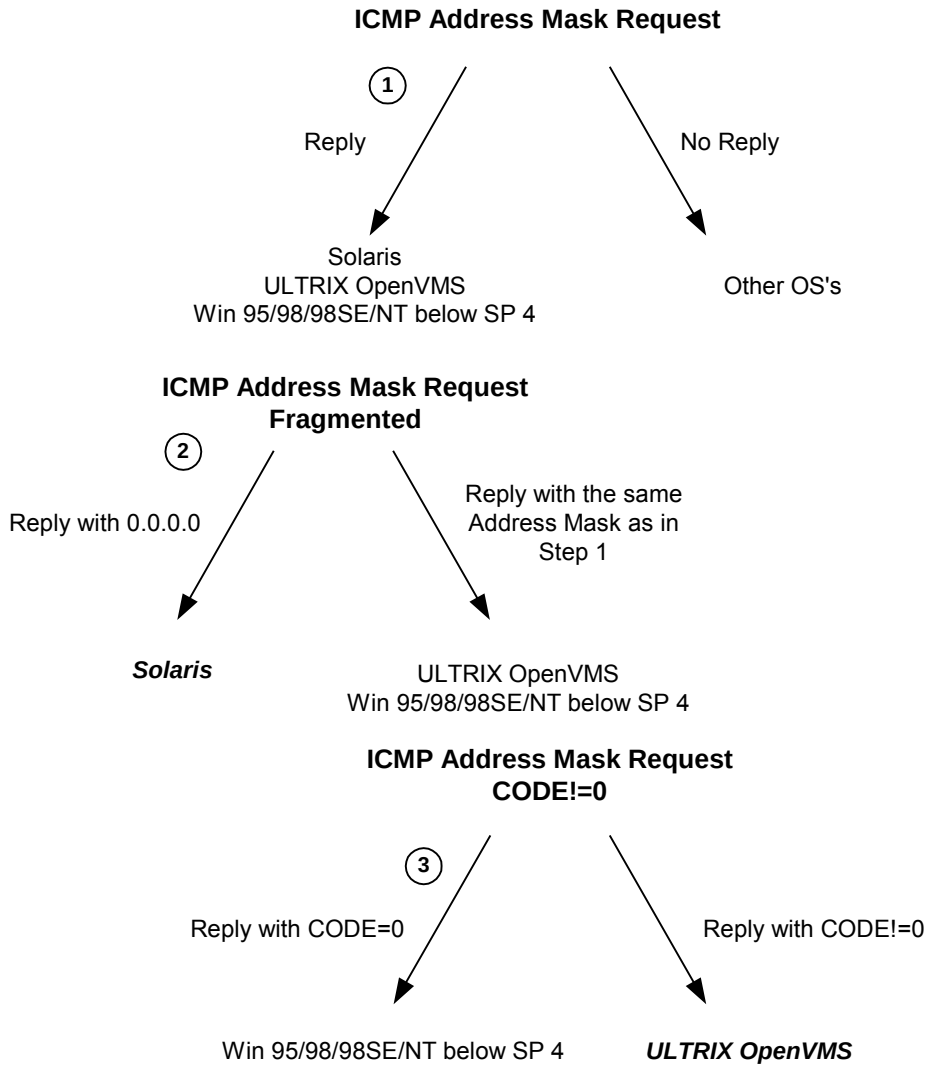


Diagram 2: Finger Printing Using ICMP Address Mask Requests

6.3 TOSing OSs out of the Window / Fingerprinting Microsoft Windows 2000

Each IP Datagram has an 8-bit field called the “TOS Byte”, which represents the IP support for prioritization and Type-of-Service handling.

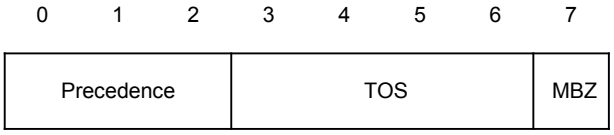


Figure 11: The Type of Service Byte

The “TOS Byte” consists of three fields.

The “Precedence field”, which is 3-bit long, is intended to prioritize the IP Datagram. It has eight levels of prioritization[33]:

Precedence	Definition
0	Routine (Normal)
1	Priority
2	Immediate
3	Flash
4	Flash Override
5	Critical
6	Internetwork Control
7	Network control

Table 6: Precedence Field Values

Higher priority traffic should be sent before lower priority traffic.

The second field, 4 bits long, is the “Type-of-Service” field. It is intended to describe how the network should make tradeoffs between throughput, delay, reliability, and cost in routing an IP Datagram.

RFC 1349[34] has defined the “Type-of-Service” field as a single enumerated value, thus interpreted as a numeric value rather than independent flags (with RFC 791 the 4 bits were distinct options, allowing combinations as well). The 4 bits represents a maximum of 16 possible values.

Value (Hex)	Value (Binary)	Service
0	0000	Normal
1	1000	Minimize Delay
2	0100	Maximize Throughput
4	0010	Maximize Reliability
8	0001	Minimize Cost
15		Maximize Security[35]

Table 7: Type-of-Service Field Values

What about the other 10 value possibilities?

RFC 1349 refer to this issue and states that “although the semantics of values other than the five listed above are not defined by this memo, they are perfectly legal TOS values, and hosts and routers must not preclude their use in any way”…”A host or a router need not make any distinction between TOS values who’s semantics are defined by this memo and those that are not”.

The last field, the “MBZ” (most be zero), is unused and most be zero. Routers and hosts ignore this last field. This field is 1 bit long.

Combining Type-of-Service flags with the different prioritization values, dictates very explicit types of behavior with certain types of data.

[33] RFC 791 – Internet Protocol, http://www.ietf.org/rfc/rfc791.txt.
[34] RFC 1349 - Type of Service in the Internet Protocol Suite, http://www.ietf.org/rfc/rfc1349.txt.
[35] RFC 1455 - Physical Link Security Type of Service, http://www.ietf.org/rfc/rfc1455.txt.

Please note the not all TCP/IP implementations would use this values (nor offer a mechanism for setting those values) and some will not handle datagrams which have Type-of-Service and/or Precedence values other than the defaults, differently.

6.3.1 The use of the Type-of-Service field with the Internet Control Message Protocol

RFC 1349 also define the usage of the Type-of-Service field with the ICMP messages. It distinguishes between ICMP error messages (Destination Unreachable, Source Quench, Redirect, Time Exceeded, and Parameter Problem), ICMP query messages (Echo, Router Solicitation, Timestamp, Information request, Address Mask request) and ICMP reply messages (Echo reply, Router Advertisement, Timestamp reply, Information reply, Address Mask reply).

Simple rules are defined:

- An ICMP error message is always sent with the default TOS (0x00)
- An ICMP request message may be sent with any value in the TOS field. A mechanism to allow the user to specify the TOS value to be used would be a useful feature in many applications that generate ICMP request messages.

  The RFC further specify that although ICMP request messages are normally sent with the default TOS, there are sometimes good reasons why they would be sent with some other TOS value.

- An ICMP reply message is sent with the same value in the TOS field as was used in the corresponding ICMP request message.

Using this logic I have decided to check if certain operating systems react correctly to an ICMP Query messages with a Type-of-Service field value, which is different than the default (0x00).

The check out was produced with all ICMP query message types sent with a Type-of-Service field set to a known value, than set to an unknown value (the term known and unknown are used here because I was not experimenting with non-legit values, and since any value may be sent inside this field).

The following example is an ICMP Echo request sent to my FreeBSD 4.0 machine. The tool used here is HPING2 beta 54. The –o option with HPING2 enable it to insert Type-of-Service values.

```
[root@aik /root]# hping2 -1 -o 8 192.168.1.15
default routing not present
HPING 192.168.1.15 (eth0 192.168.1.15): icmp mode set, 28 headers + 0
data bytes46 bytes from 192.168.1.15: icmp_seq=0 ttl=255 id=16 rtt=1.1
ms
46 bytes from 192.168.1.15: icmp_seq=1 ttl=255 id=17 rtt=0.4 ms
46 bytes from 192.168.1.15: icmp_seq=2 ttl=255 id=18 rtt=0.3 ms
46 bytes from 192.168.1.15: icmp_seq=3 ttl=255 id=19 rtt=0.3 ms
46 bytes from 192.168.1.15: icmp_seq=4 ttl=255 id=20 rtt=0.3 ms
…

--- 192.168.1.15 hping statistic ---
11 packets tramitted, 11 packets received, 0% packet loss
round-trip min/avg/max = 0.3/0.4/1.1 ms
```

Snort trace:

```
Initializing Network Interface...
Decoding Ethernet on interface eth0

-*> Snort! <*-
Version 1.6
By Martin Roesch (roesch@clark.net, www.clark.net/~roesch)
08/09-21:48:37.280337 192.168.1.200 -> 192.168.1.15
ICMP TTL:64 TOS:0x8 ID:60783
ID:48899   Seq:0  ECHO

08/09-21:48:37.280928 192.168.1.15 -> 192.168.1.200
ICMP TTL:255 TOS:0x8 ID:16
ID:48899   Seq:0  ECHO REPLY
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00                                            ..
```

This is the second test I have produced, sending ICMP Echo request with the Type-of-Service field set to a 10 Hex value, a value that is not a known Type-of-Service value:

```
[root@aik /root]# hping2 -1 -o 10 192.168.1.15
default routing not present
HPING 192.168.1.15 (eth0 192.168.1.15): icmp mode set, 28 headers + 0
data bytes46 bytes from 192.168.1.15: icmp_seq=0 ttl=255 id=27 rtt=0.4
ms
46 bytes from 192.168.1.15: icmp_seq=1 ttl=255 id=28 rtt=0.4 ms
46 bytes from 192.168.1.15: icmp_seq=2 ttl=255 id=29 rtt=0.4 ms
46 bytes from 192.168.1.15: icmp_seq=3 ttl=255 id=30 rtt=0.3 ms
…

--- 192.168.1.15 hping statistic ---
10 packets tramitted, 10 packets received, 0% packet loss
round-trip min/avg/max = 0.3/0.4/0.4 ms
[root@aik /root]#
```

Snort trace:

```
Initializing Network Interface...
Decoding Ethernet on interface eth0

-*> Snort! <*-
Version 1.6
By Martin Roesch (roesch@clark.net, www.clark.net/~roesch)
08/09-21:48:58.626840 192.168.1.200 -> 192.168.1.15
ICMP TTL:64 TOS:0x10 ID:53895
ID:49667   Seq:0  ECHO

08/09-21:48:58.627170 192.168.1.15 -> 192.168.1.200
ICMP TTL:255 TOS:0x10 ID:27
ID:49667   Seq:0  ECHO REPLY
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00                                            ..
```

As it can be seen from the trace, the ICMP Echo reply message have maintained the Type-of-Service value as was used in the corresponding ICMP request message.

FreeBSD 4.0 does not respond to ICMP Information request, or to ICMP Address Mask requests. I had to verify with ICMP Timestamp requests with the same Type-of-Service values as with the previous ICMP Echo requests that this behavior is produced with ICMP Timestamp request and replies as well.

This time I had to use another tool since HPING2 did not provide me with the ability to send ICMP Timestamp request. I switched to nemesis:[36]

```
[root@aik /root]# nemesis-icmp -v -i 13 -t 8 -S x.x.x.x -D y.y.y.y

ICMP Packet Injection -=- The NEMESIS Project 1.1
(c) 1999, 2000 obecian <obecian@celerity.bartoli.org>

[IP]  x.x.x.x > y.y.y.y
[Type] TIMESTAMP REQUEST
[Sequence number] 0
[IP ID] 0
[IP TTL] 254
[IP TOS] 0x8
[IP Frag] 0x4000

Wrote 48 bytes

ICMP Packet Injected
[root@aik /root]#
```

Snort trace:

```
Initializing Network Interface...
Decoding raw data on interface ppp0

-*> Snort! <*-
Version 1.6
By Martin Roesch (roesch@clark.net, www.clark.net/~roesch)
08/15-21:05:37.570078 x.x.x.x -> y.y.y.y
ICMP TTL:254 TOS:0x8 ID:24  DF
TIMESTAMP REQUEST
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00                          ........

08/15-21:05:38.101883 y.y.y.y -> x.x.x.x
ICMP TTL:241 TOS:0x8 ID:31017  DF
TIMESTAMP REPLY
00 00 00 00 00 00 00 00 03 DB 2F 7A 03 DB 2F 7A  ............/z../z
00 00 00 00 00 00 00 00                          ........
```

The same behavior was produced.

[36] Nemesis, written by obescian, can be downloaded from, http://celerity.bartoli.org.

Ok. I was curious again. I imagined that the Microsoft Windows implementation of the things might be a little different.

When I was examining ICMP Echo requests I noticed something is wrong with Microsoft:

```
[root@aik /root]# hping2 -1 -o 10 192.168.1.1
default routing not present
HPING 192.168.1.1 (eth0 192.168.1.1): icmp mode set, 28 headers + 0
data bytes
46 bytes from 192.168.1.1: icmp_seq=0 ttl=128 id=74 rtt=0.9 ms
46 bytes from 192.168.1.1: icmp_seq=1 ttl=128 id=75 rtt=0.5 ms
46 bytes from 192.168.1.1: icmp_seq=2 ttl=128 id=76 rtt=0.5 ms
…

--- 192.168.1.1 hping statistic ---
8 packets tramitted, 8 packets received, 0% packet loss
round-trip min/avg/max = 0.5/0.6/0.9 ms
[root@aik /root]#
```

Snort trace:

```
Initializing Network Interface...
Decoding Ethernet on interface eth0

-*> Snort! <*-
Version 1.6
By Martin Roesch (roesch@clark.net, www.clark.net/~roesch)
08/09-21:43:53.257483 192.168.1.200 -> 192.168.1.1
ICMP TTL:64 TOS:0x10 ID:34638
ID:45571   Seq:0  ECHO

08/09-21:43:53.258294 192.168.1.1 -> 192.168.1.200
ICMP TTL:128 TOS:0x0 ID:86
ID:45571   Seq:0  ECHO REPLY
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00                                            ..
```

Oops! Some one zero out my Type-of-Service field!

Before I would let you know who of all Microsoft Windows operating systems did that, I am going to list the Microsoft operating systems who behave correctly – Microsoft Windows 98/SE/ME, Microsoft Windows NT 4 Workstation SP3, Microsoft Windows NT 4 Server SP4, Microsoft Windows NT 4 Workstation SP6a.

The Microsoft Windows 2000 family (Professional, Server, Advanced Server) zero out this field on the ICMP Echo reply.

Is this makes those Microsoft Windows 2000 machines identified easily and uniquely?

99.9% yes. The other 0.01 % belongs to Ultrix.

From the operating systems I have checked (Linux Kernel 2.2.x, Linux Kernel 2.4 test 2/4/5, FreeBSD 4.0 & 4.1, OpenBSD 2.6 & 2.7, NetBSD 1.4.2, SUN Solaris 2.7 & 2.8, Compaq Tru64

UNIX 5.0, AIX 4.1 & 3.2, OpenVMS v7.2, Irix 6.5.3 & 6.5.8, Ultrix 4.2-4.5, Microsoft Windows 98/SE/ME, Microsoft Windows NT 4 Workstation & Server with various service packs, Microsoft Windows 2000 Professional, Server & Advanced Server) only Ultrix behaved like the Microsoft Windows 2000 machines.

How can we distinguish between those?
First, there are much fewer Ultrix machines out there than Microsoft's Windows 2000 (I see your faces – not convincing enough).

The fast track in distinguishing between Ultrix and Microsoft Windows 2000 is simply by looking at the TTL field value. Microsoft Windows 2000 family uses 128 as their default TTL value in ICMP ECHO replies while Ultrix uses 255. For more information about the TTL field value see sections 6.10 and 6.11.

Another method would be sending an ICMP Information request or an ICMP Address Mask request - than only Ultrix would answer our request (if not filtered of course) and not the Microsoft Windows 2000 machines.

Other ICMP query message types help us to identify a unique group of Microsoft operating systems. As a rule all operating systems except the named Microsoft windows operating systems here, maintain a single behavior regarding the Type-of-Service field. All would maintain the same values with different types of ICMP requests. But, again, Microsoft have some of the "top" people understanding TCP/IP to the degree we humans do not understand so we have the following Microsoft operating systems zero out (0x00) the Type-of-Service field on the replies for ICMP Timestamp requests: Microsoft Windows 98/98SE/ME. Microsoft Windows 2000 machines would zero out this field as well.

This means that Microsoft Windows 98/98SE/ME would not zero out the Type-of-Service field value with ICMP Echo requests but will do so with ICMP Timestamp requests.

Here we got a way to fingerprint Microsoft Windows 2000 machines from the rest of the world and from the rest of the Microsoft Windows operating systems.

Operating System	Information Request With TOS!=0x00	Time Stamp Request With TOS!=0x00	Address Mask Request With TOS!=0x00	Echo Request With TOS!=0x00
Debian GNU/ LINUX 2.2, Kernel 2.4 test 2 (*)	Not Answering	!=0x00	Not Answering	!=0x00
Redhat LINUX 6.2 Kernel 2.2.14 (*)	Not Answering	!=0x00	Not Answering	!=0x00
FreeBSD 4.0 (*)	Not Answering	!=0x00	Not Answering	!=0x00
FreeBSD 3.4	Not Answering		Not Answering	
OpenBSD 2.7 (*)	Not Answering	!=0x00	Not Answering	!=0x00
OpenBSD 2.6	Not Answering	!=0x00	Not Answering	!=0x00
NetBSD	Not Answering	!=0x00	Not Answering	!=0x00
BSDI BSD/OS 4.0 (*)	Not Answering	!=0x00	Not Answering	!=0x00
BSDI BSD/OS 3.1 (*)	Not Answering	!=0x00	Not Answering	!=0x00
Solaris 2.5.1	Not Implemented			
Solaris 2.6	Not Implemented			
Solaris 2.7 (*)	Not Implemented	!=0x00	!=0x00	!=0x00
Solaris 2.8 (*)	Not Implemented	!=0x00	!=0x00	!=0x00
HP-UX v10.20			Not Answering	

Operating System	Information Request With TOS!=0x00	Time Stamp Request With TOS!=0x00	Address Mask Request With TOS!=0x00	Echo Request With TOS!=0x00
HP-UX v11.0	Not Answering	Not Answering	!=0x00	!=0x00
Compaq Tru64 v5.0 (*)		!=0x00	Not Answering	!=0x00
Irix 6.5.3 (*)	Not Answering	!=0x00	Not Answering	!=0x00
Irix 6.5.8 (*)	Not Answering	!=0x00	Not Answering	!=0x00
AIX 4.1 (*)		!=0x00	Not Answering	!=0x00
AIX 3.2 (*)		!=0x00	Not Answering	!=0x00
ULTRIX 4.2 – 4.5 (*)		0x00	0x00	0x00
OpenVMS v7.1-2 (*)		!=0x00	!=0x00	!=0x00
Novell Netware 5.1 SP1 (*)	Not Answering	Not Answering	Not Answering	0x00
Novell Netware 5.0 (*)	Not Answering	Not Answering	Not Answering	0x00
Novell Netware 3.12 (*)	Not Answering	Not Answering	Not Answering	0x00
Windows 95	Not Answering	Not Answering		
Windows 98 (*)	Not Answering	0x00	0x00	!=0x00
Windows 98 SE (*)	Not Answering	0x00		!=0x00
Windows ME (*)	Not Answering	0x00	Not Answering	!=0x00
Windows NT 4 WRKS SP 3 (*)	Not Answering	Not Answering		!=0x00
Windows NT 4 WRKS SP 6a (*)	Not Answering	Not Answering	Not Answering	!=0x00
Windows NT 4 Server SP4	Not Answering	Not Answering	Not Answering	!=0x00
Windows 2000 Professional (*)	Not Answering	0x00	Not Answering	0x00
Windows 2000 Server (*)	Not Answering	0x00	Not Answering	0x00

Table 8: ICMP Query Message Types with TOS! = 0

6.4 ICMP error Message Quenching

RFC 1812 suggests limiting the rate at which various error messages are sent.
Only few operating systems are known to follow this RFC.

An attacker can use this to send UDP packets to a random, high UDP port and count the number of ICMP Destination unreachable messages received within a given amount of time.

6.5 ICMP Message Quoting

Every ICMP error message includes the Internet Protocol (IP) Header and *at least* the first 8 data bytes of the datagram that triggered the error; more than 8 octets (bytes) *may* be sent.

Except for LINUX and Solaris almost all implementations will quote 8 bytes of the datagram that triggered the error message. Solaris sends more information than is needed and *Linux even more*.

The following example is a snort log of a LINUX machine (LINUX 6.1 Kernel 2.2.12) that have generated a Port Unreachable ICMP error message:

```
03/01-12:29:39.259510 192.168.5.5 -> 192.168.5.1
ICMP TTL:255 TOS:0xDE ID:149
DESTINATION UNREACHABLE: PROTOCOL UNREACHABLE
00 00 00 00 45 7E 04 32 00 0D 00 00 89 70 A1 7A  ....E~.2.....p.z
```

```
C0 A8 05 01 C0 A8 05 05 FE 94 6C 95 59 F2 D9 3C  ..........l.Y..<
8D AA B6 0B 2B 80 CB 8B 89 4D C9 59 19 D6 0F A0  ....+....M.Y....
D3 67 D1 0F CB ED 84 8C 91 7E 24 00 70 B9 D7 E4  .g.......~$.p...
6E AA 91 8F CF 5C ED 86 1B A2 40 1D 93 10 73 4B  n....\....@...sK
49 5B A8 D5 91 99 47 F0 15 6B EB 8B 21 2D A2 15  I[....G..k..!-..
A1 97 4C AD 6D A1 2B E5 15 07 86 77 3A 85 E9 6E  ..L.m.+....w:..n
58 87 05 73 6D FB E9 05 29 73 DD B4 C0 EA 98 1D  X..sm...)s......
6E 44 8F 47 85 A4 89 E6 CF 64 18 B5 FD 31 19 C0  nD.G.....d...1..
C0 8A 8E CB 60 B0 D5 F5 79 57 81 DD 78 0B 1B EF  ....`...yW..x...
CE 8A E5 AC 46 D4 E3 91 6C 24 80 59 CC 00 C4 AB  ....F...l$.Y....
86 CC 39 FC AD B1 AF 3F 16 B1 6D 9C 47 5D 85 F5  ..9....?..m.G]..
FC E3 CC 01 0E DC CC 48 E4 B6 0B 0E E5 08 A5 41  .......H.......A
9A D9 45 B9 7A 37 13 31 C7 96 F2 42 2E 20 95 21  ..E.z7.1...B. .!
D8 EF 74 F4 78 B3 44 14 F5 4D 45 B4 08 C0 7B 1A  ..t.x.D..ME...{.
7E B0 B5 71 2A 5A 95 61 22 0E 72 B7 1A 57 1E F2  ~..q*Z.a".r..W..
3E B9 28 33 EA 3A 23 70 34 41 CF 43 C8 B1 CE 1A  >.(3.:#p4A.C....
15 FD 42 E9 E1 4B DC 93 35 2C 10 6C 71 B5 0D 1C  ..B..K..5,.lq...
84 60 E9 68 51 30 79 AE 2E 1D 59 F0 F4 C8 AD CD  .`.hQ0y...Y.....
0E 62 1F 23 42 2F 30 70 91 DA 5C 86 4E 62 CF 93  .b.#B/0p..\.Nb..
84 B9 39 9D F2 03 B8 FA 08 E1 BA B5 86 15 1D DE  ..9.............
FD 9E 68 61 F9 71 32 CB 78 CD 6A 27 3F E7 FC 2D  ..ha.q2.x.j'?..-
54 90 90 17 76 DC 82 AD E9 07 6A A5 2F 7B F7 69  T...v.....j./{.i
89 C8 71 AA 27 DA 1A A3 CD 30 75 3C EA 36 52 EA  ..q.'....0u<.6R.
AE D9 DC 3A 0A E5 B7 BA 97 F0 91 FA D4 98 94 8F  ...:............
F9 5B CE 0A C6 5A 71 29 38 32 05 42 6D 57 8C C2  .[...Zq)82.BmW..
95 59 E3 33 0F 70 7E 61 4E D9 3E EB 75 CB D7 A1  .Y.3.p~aN.>.u...
B0 95 9C A5 F2 44 7D C6 11 E2 DC 7B CF B0 C0 BB  .....D}....{....
B8 B6 DA 95 77 76 4F A7 6B 90 4B 0F E3 36 64 EC  ....wvO.k.K..6d.
19 1A A9 91 D5 15 52 4C AE D3 42 6D DE 0E 43 2D  ......RL..Bm..C-
26 A1 ED 7E C1 8E 74 7A 2C 6A 36 5A 4B 1C DC FF  &..~..tz,j6ZK...
D2 FF 3D 61 59 C6 E4 E1 19 DD 29 77 A4 9D D2 93  ..=aY.....)w....
03 0D 1B 14 21 3B 6E 9D 66 23 05 72 D2 89 80 3D  ....!;n.f#.r...=
AE 03 A7 9F D2 89 5D D7 E9 0C B0 98 A0 04 0F AE  ......].........
9E 17 62 93 83 28 CA 81                          ..b..(..
```

This technique allows us to identify Solaris & LINUX machines even if there is no port opened.

6.6 ICMP Error Message Echoing Integrity

When sending back an ICMP error message, some stack implementations may alter the IP header.

If an attacker examines the types of alternation that have been made to the headers, he may be able to make certain assumptions about the target operating system.

Fyodor gives the following examples in his article “Remote OS detection via TCP/IP Stack Finger Printing”[37]:

> “For example, AIX and BSDI send back an IP 'total length' field that is 20 bytes too high. Some BSDI, FreeBSD, OpenBSD, ULTRIX, and VAXen change the IP ID that you sent them. While the checksum is going to change due to the changed TTL anyway, there are some machines (AIX, FreeBSD, etc.) which send back an inconsistent or 0 checksum. Same thing goes with the UDP checksum."

[37] http://www.insecure.org/nmap/nmap-fingerprinting-article.html

6.7 TOS Field in ICMP Port Unreachable Error Message

Nearly all stack implementations send back 0x00 as the TOS value when generating an ICMP Port Unreachable Message as RFC 1349 orders. All but LINUX, which sends the value of 0xc0.

In the next example we have sent one UDP packet destined to port 50 (which is closed on the destination machine) from one LINUX machine to another, both running Redhat LINUX 6.1:

```
[root@stan /root]# hping2 -2 192.168.5.5 -p 50 -c 1
default routing not present
HPING 192.168.5.5 (eth0 192.168.5.5): udp mode set, 28 headers + 0 data
bytes
ICMP Port Unreachable from 192.168.5.5  (kenny.sys-security.com)

--- 192.168.5.5 hping statistic ---
1 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

-*> Snort! <*-
Version 1.5
By Martin Roesch (roesch@clark.net, www.clark.net/~roesch)
Kernel filter, protocol ALL, raw packet socket
Decoding Ethernet on interface eth0
03/12-12:54:47.274096 192.168.5.1:2420 -> 192.168.5.5:50
UDP TTL:64 TOS:0x0 ID:57254
Len: 8

03/12-12:54:47.274360 192.168.5.5 -> 192.168.5.1
ICMP TTL:255 TOS:0xC0 ID:0
DESTINATION UNREACHABLE: PORT UNREACHABLE
00 00 00 00 45 00 00 1C DF A6 00 00 40 11 0F D4  ....E.......@...
C0 A8 05 01 C0 A8 05 05 09 74 00 32 00 08 6A E1  .........t.2..j.
```

6.8 Using ICMP Information Requests

Because of the fact, that only few operating systems would reply to these queries, we can group them together.

From table 2 in section 2.5 we can conclude that HP-UX 10.20, AIX, ULTRIX & Open-VMS would be the only operating systems (among those I have tested) that would answer these queries.

We can further distinguish between those operating systems if we would send an ICMP Address Mask Request and wait for the reply from the systems in question. AIX and HP-UX would not answer the query, while the ULTRIX & Open-VMS would.

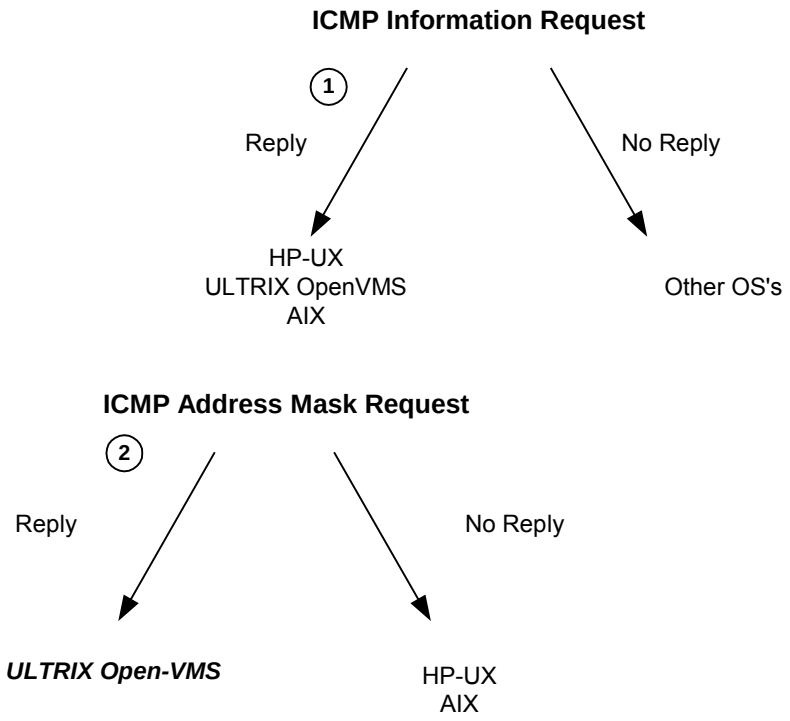


Diagram 3: Finger Printing Using ICMP Information Request Combines with ICMP Address Mask Request

6.9 Identifying operating systems according to their replies for non-ECHO ICMP requests aimed at the broadcast address.

If IP directed broadcasts are not blocked, than we can identify the answering machines quite easily.

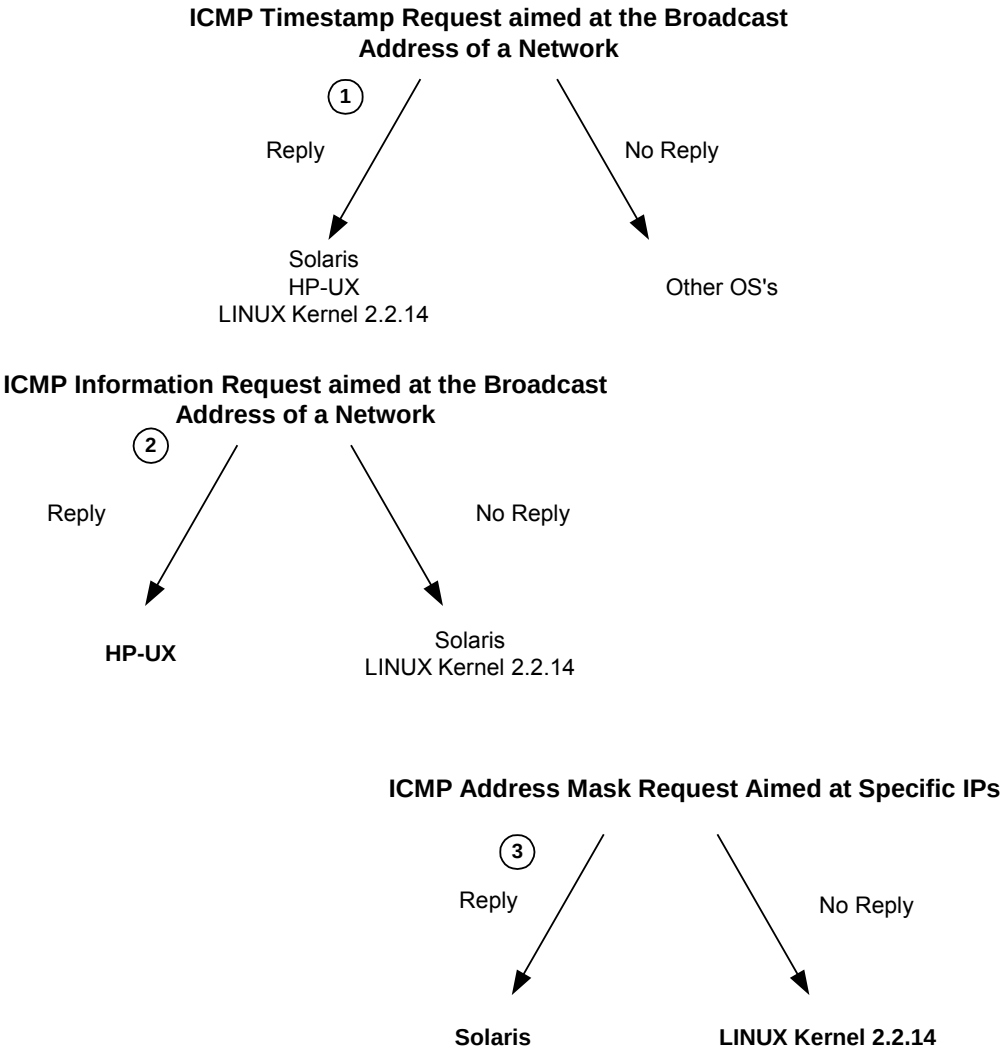


Diagram 4: Finger Printing Using non-ECHO ICMP Query Types aimed at the Broadcast Address of an Attacked Network

The first step is sending an ICMP Timestamp request aimed at the broadcast address of a network. The operating systems who would answer would include SUN Solaris, HP-UX 10.20, and LINUX (Kernel version 2.2.x). We can further identify those operating systems by sending an ICMP Information request aimed at the broadcast address of a network. HP-UX 10.20 would answer the query while SUN Solaris and LINUX would not. To distinguish between the two we would send an ICMP Address Mask request to the IPs that did not answer in the previous step. SUN Solaris would reply to the query while LINUX would not.

For complete information see Section 2.6.

6.10 IP TTL Field Value with ICMP

The IP TTL field value with ICMP has two separate values, one for ICMP query messages and one for ICMP query replies.

The TTL field value help us identify certain operating systems and groups of operating systems. It also provide us with the simplest means to add another check criteria when we are quering other host(s) or listening to traffic (sniffing).

6.10.1 IP TTL Field Value with ICMP Query Replies

We can use the IP TTL field value with the ICMP Query Reply datagrams to identify certain groups of operating systems. The method discussed in this section is very simple one. We send an ICMP Query message to a host. If we receive a reply, we would be looking at the IP TTL field value in the ICMP query reply. The next table describes the IP TTL field value with ICMP Echo replies. According to it we can distinguish between certain operating systems:

Operating System	IP TTL on ICMP datagrams - In Reply -
LINUX Kernel 2.4	255
Kernel 2.2.14	255
Kernel 2.0.x[38]	64
FreeBSD 4.0	255
FreeBSD 3.4	255
OpenBSD 2.7	255
OpenBSD 2.6	255
NetBSD	255
BSDI BSD/OS 4.0	255
BSDI BSD/OS 3.1	255
Solaris 2.5.1	255
Solaris 2.6	255
Solaris 2.7	255
Solaris 2.8	255
HP-UX v10.20	255
HP-UX v11.0	255
Compaq Tru64 v5.0	64
Irix 6.5.3 (*)	255
Irix 6.5.8 (*)	255
AIX 4.1 (*)	255
AIX 3.2 (*)	255
ULTRIX 4.2 – 4.5 (*)	255
OpenVMS v7.1-2 (*)	255
Windows 95	32
Windows 98 (*)	128
Windows 98 SE (*)	128
Windows ME (*)	128
Windows NT 4 WRKS SP 3	128
Windows NT 4 WRKS SP 6a	128
Windows NT 4 Server SP4	128
Windows 2000 Professional	128
Windows 2000 Server	128

Table 9: IP TTL Field Values in replies from Various Operating Systems

[38] Stephane Omnes provided information about LINUX Kernel 2.0.x.

If we would look at the ICMP Echo replies IP TTL field values than we see some patterns:

- UNIX and UNIX-like operating systems use 255 as their IP TTL field value with ICMP query replies.
- Compaq Tru64 5.0 and LINUX 2.0.x are the exception, using 64 as its IP TTL field value with ICMP query replies.
- Microsoft Windows operating system machines are using the value of 128.
- Microsoft Windows 95 is the only Microsoft operating system to use 32 as its IP TTL field value with ICMP query messages.

With the ICMP query replies we have an operating systems that is clearly distinguished from the other - Windows 95. Other operating systems are grouped into the 64 group (LINUX Kernel 2.0.x & Compaq Tru64 5.0), the 255 group (UNIX and UNIX-like), and into the 128 group (Microsoft operating systems).

We are not limited to ICMP ECHO replies only. We can use the other ICMP Query message types as well, and the results should be the same. In the next example an ICMP Timestamp request is sent to a Redhat 6.1 LINUX, Kernel 2.2.12 machine:

```
[root@stan /root]# icmpush -tstamp 192.168.5.5
kenny.sys-security.com -> 13:48:07
```

Snort Trace:

```
01/26-13:51:29.342647 192.168.5.1 -> 192.168.5.5
ICMP TTL:254 TOS:0x0 ID:13170
TIMESTAMP REQUEST
88 16 D8 D9 02 8B 63 3D 00 00 00 00 00 00 00 00  ......c=........

01/26-13:51:29.342885 192.168.5.5 -> 192.168.5.1
ICMP TTL:255 TOS:0x0 ID:6096
TIMESTAMP REPLY
88 16 D8 D9 02 8B 63 3D 02 88 50 18 02 88 50 18  ......c=..P...P.
2A DE 1C 00 A0 F9                                *.....
```

IP TTL field value is 255 (the machine is on the same LAN).

We can use this information with other tests as, to provide us extra criteria with zero effort.

6.10.2 IP TTL Field Value with ICMP ECHO Requests

The examination of the IP TTL field value is not limited to ICMP Query replies only. We can learn a lot from the ICMP requests as well.

Operating System	IP TTL on ICMP datagrams In Requests
LINUX Kernel 2.4 test 1-7	64
LINUX Kernel 2.2.x	64
LINUX Kernel 2.0.x	64
FreeBSD 4.0	255
FreeBSD 3.4	255

Operating System	IP TTL on ICMP datagrams In Requests
OpenBSD 2.7	255
OpenBSD 2.6	255
NetBSD	
Solaris 2.5.1	255
Solaris 2.6	255
Solaris 2.7	255
Solaris 2.8	255
HP-UX v10.20	255
Windows 95	32
Windows 98	32
Windows 98 SE	32
Windows ME	32
Windows NT 4 WRKS SP 3	32
Windows NT 4 WRKS SP 6a	32
Windows NT 4 Server SP4	32
Windows 2000 Professional	128
Windows 2000 Server	128

Table 10: IP TTL Field Values in requests from Various Operating Systems

The ICMP Query message type used was ICMP Echo request, which is common on all operating systems tested using the ping utility.

- LINUX Kernel 2.0.x, 2.2.x & 2.4.x use 64 as their IP TTL Field Value with ICMP Echo Requests.
- FreeBSD 4.1, 4.0, 3.4; Sun Solaris 2.5.1, 2.6, 2.7, 2.8; OpenBSD 2.6, 2.7, NetBSD and HP UX 10.20 use 255 as their IP TTL field value with ICMP Echo requests. With the OSs listed above the same IP TTL Field value with any ICMP message is given.
- Windows 95/98/98SE/ME/NT4 WRKS SP3,SP4,SP6a/NT4 Server SP4 - all using 32 as their IP TTL field value with ICMP Echo requests.
- A Microsoft window 2000 is using 128 as its IP TTL Field Value with ICMP Echo requests.

We can distinguish between LINUX, Microsoft Windows 2000, the other Microsoft operating systems group, and the 255 group.

6.10.3 Correlating the Information

Using the IP TTL field value with ICMP messages we can distinguish between Microsoft Windows 2000, certain Microsoft Windows Operating systems, LINUX Kernel 2.2.x & 2.4.x, LINUX Kernel 2.x.0, and the *BSD and Solaris group.

Operating System	IP TTL value in the ECHO Requests	IP TTL value in the ECHO Replies
Microsoft Windows Family	32	128
*BSD and Solaris	255	255
LINUX Kernel 2.2.x & 2.4.x	64	255
LINUX Kernel 2.0.x	64	64
Microsoft Windows 2000	128	128
Microsoft Windows 95	32	32

Table 11: Further dividing the groups of operating systems according to IP TTL field value in the ICMP ECHO Requests and in the ICMP ECHO Replies

One would expect that the IP TTL field value would be the same with both ICMP Query requests and ICMP Query replies. Apparently this is not true and provide us with valuable information about the operating system querying / being queried.

6.11 DF Bit

A few operating systems would set the DF bit with the replies they produce for ICMP Query messages they answer for.

In the next example an OpenBSD 2.7 box replies to an ICMP Echo Request with an ICMP Echo reply, which sets the DF bit on.

```
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
08/19-23:48:04.711978 139.92.185.88 -> 129.128.5.191
ICMP TTL:255 TOS:0x0 ID:13170
ID:1187   Seq:1  ECHO
84 F2 9E 39 15 DD 0A 00                          ...9....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
08/19-23:48:05.711955 129.128.5.191 -> 139.92.185.88
ICMP TTL:235 TOS:0x0 ID:26398  DF
ID:1187   Seq:1  ECHO REPLY
84 F2 9E 39 15 DD 0A 00                          ...9....

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

I have tested this behavior with a number of operating systems. I have queried them for all the ICMP Query messages they answer for to find out who reply with the DF bit set.

Solaris 2.6,2.7 & 2.8, OpenBSD and HP-UX 11.0 sets the DF bit in their replies to the ICMP Query messages they answer for. Other operating systems do not.

Operating systems that I have checked are: Linux Kernel 2.4 test 2,4,5,6; Linux Kernel 2.2.x; FreeBSD 4.0, 3.4; OpenBSD 2.7,2.6; NetBSD 1.4.1,1.4.2; BSDI BSD/OS 4.0,3.1; Solaris 2.6,2.7,2.8; HP-UX 11.0; Compaq Tru64 5.0; Aix 4.1,3.2; Irix 6.5.3, 6.5.8; Ultrix 4.2 – 4.5; OpenVMS v7.1-2; Novel Netware 5.1 SP1, 5.0, 3.12; Microsoft Windows 98/98SE/ME, Microsoft Windows NT WRKS SP6a, Microsoft Windows NT Server SP4, Microsoft Windows 2000 Family.

6.12 DF Bit Echoing

Some operating systems, when receiving an ICMP Query message with the DF bit set, would set the DF bit with their replies as well. Sometimes it would be in contrast with their regular behavior, which would be not setting the DF Bit in their replies for a regular query that comes with the DF bit not set.

6.12.1 DF Bit Echoing with the ICMP Echo request

The snort trace below illustrates an ICMP Echo request sent from a Linux box, using nemesis, to a SUN Solaris 2.7 machine:

```
[root@aik /root]# nemesis-icmp -i 8 x.x.x.x
08/10-15:24:21.625260 10.0.0.105 -> x.x.x.x
ICMP TTL:64 TOS:0x0 ID:13670 DF
ID:62979   Seq:0  ECHO

08/10-15:24:22.623507 10.0.0.105 -> x.x.x.x
ICMP TTL:64 TOS:0x0 ID:43567 DF
ID:62979   Seq:256  ECHO

08/10-15:24:23.318173 x.x.x.x -> 10.0.0.105
ICMP TTL:239 TOS:0x0 ID:221  DF
ID:62979   Seq:0  ECHO REPLY
08 8C 02 85 1C 2A 7F 32 AB 14 6C 79 F5 2E 53 84  .....*.2..ly..S.
AF 15                                            ..

08/10-15:24:23.555488 x.x.x.x -> 10.0.0.105
ICMP TTL:239 TOS:0x0 ID:222  DF
ID:62979   Seq:256  ECHO REPLY
BE 13 02 8F 90 8F 15 93 94 93 04 97 98 97 16 9B  ................
9C 9B                                            ..
```

Most of the operating systems that I have checked this behavior against did the same thing. In the reply they produced, the DF bit was set.

Which operating systems are the exceptional and do not echo back the DF bit?
Linux Kernel 2.2.x, Linux Kernel 2.4 with the various test kernels, Ultrix v4.2 – 4.5, and Novell Netware.

How can we distinguish between those operating systems?
Frankly it is quite simple. In the next example I have sent an ICMP Echo request to my Linux box loop back address:

```
[root@aik sbin]# ./nemesis-icmp -i 8 -S 127.0.0.1 -D 127.0.0.1

Initializing Network Interface...
   => Decoding LoopBack on interface lo

-*> Snort! <*-
Version 1.6.3
By Martin Roesch (roesch@clark.net, www.snort.org)
08/20-17:11:06.825971 127.0.0.1 -> 127.0.0.1
ICMP TTL:254 TOS:0x18 ID:104  DF
ID:0   Seq:0  ECHO
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00                                      ....


08/20-17:11:06.826007 127.0.0.1 -> 127.0.0.1
ICMP TTL:255 TOS:0x18 ID:105
ID:0   Seq:0  ECHO REPLY
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00                                      ....
```

Since LINUX and Ultrix are using a TTL field value of 255 in their ICMP Query replies, and Novell Netware uses 128, it is easy to distinguish between those groups.

6.12.2 DF Bit Echoing with the ICMP Address Mask request

With ICMP Address Mask requests we have a different story. Among the operating systems that I have checked that answer for an ICMP Address Mask request Sun Solaris & OpenVMS echo back the DF bit. Microsoft Windows 98, Microsoft Windows 98 SE, and Ultrix do not echo back the DF bit.

Again it is very simple to distinguish between the Microsoft Windows 98 family and between the Ultrix machines. Since the Microsoft Windows 98 family is using 128 as their TTL field value in their ICMP query replies and Ultrix uses 255, we can distinguish between those operating systems.

We have here a simple method to distinguish between Microsoft Windows 98 / 98 SE, and Ultrix machines to the rest of the operating systems world.

Another interesting piece of information is that the Microsoft Windows 98 family changed its behavior from DF echoing with the ICMP Echo request to not echoing with the ICMP Address Mask request. This inconsistency is a factor with all Microsoft operating systems (Echoing with ICMP Echo request, not echoing with the other types of ICMP query).

6.12.3 DF Bit Echoing with the ICMP Timestamp request

Since a lot more operating systems answer for an ICMP Timsestamp request than with the ICMP Address Mask request, we have a bit more difficulty in identifing those.

Linux with Kernel 2.2.x, Linux with Kernel 2.4, Ultrix, Microsoft Windows 98/98SE/ME, and the Microsoft Windows 2000 Family would not echo back the DF bit with ICMP Timestamp replies they produce for ICMP Timestamp request that sets their DF bit.

Here we can only distinguish between certain groups of operating systems; again it would be according to their TTL field value with their replies.

Linux would use 255 as its TTL field value for the ICMP Timestamp reply; Ultrix would use the same value. The Microsoft family of operating systems that would answer for this kind of query would use 128 as their TTL value.

Again we have Linux and Ultrix on the one hand and the Microsoft Family on the other hand. How can we further distinguish between those?

6.12.4 Using all of the Information in order to identify maximum of operating systems

We can group Linux and Ultrix with the ICMP Echo requests. We can do the same with Microsoft Windows 98 / 98 SE & Ultrix using the ICMP Address Mask requests. This would allow us to pinpoint the Linux boxes from the first stage. So when we would go into the third stage we would know which operating systems are Linux based, which are Microsoft Windows 98 / 98 SE based, and which are Ultrix based. This would leave us with Microsoft Windows ME and with the Microsoft Windows 2000 family machines.

6.12.5 Why this would work (for the skeptical)

All those skeptical would say that if they receive an ICMP Query request with the DF bit set than it should be clear that something is wrong and someone is probably trying to scan them. Think again. What would happen if a Solaris box would query your box? Than the same behavior would be produced.

This is an ICMP Echo request sent from a Solaris 2.6 box to a Linux box. We can see that the DF bit is set with the request and not set with the reply. But again if some one would mimic this behavior with a tool used on a Linux box to query the world, which is 100% mimicking Solaris than we would never know if this is a legit request or an attempt for scanning / fingerprinting.

```
Initializing Network Interface...
Decoding raw data on interface ppp0

-*> Snort! <*-
Version 1.6
By Martin Roesch (roesch@clark.net, www.clark.net/~roesch)
08/10-23:32:52.201612 y.y.y.y -> 139.92.207.58
ICMP TTL:239 TOS:0x0 ID:48656  DF
ID:2080   Seq:0  ECHO
39 93 10 A3 00 03 F0 E5 08 09 0A 0B 0C 0D 0E 0F  9...............
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F  ................
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F   !"#$%&'()*+,-./
30 31 32 33 34 35 36 37                          01234567

08/10-23:32:52.201649 139.92.207.58 -> y.y.y.y
ICMP TTL:255 TOS:0x0 ID:349
ID:2080   Seq:0  ECHO REPLY
39 93 10 A3 00 03 F0 E5 08 09 0A 0B 0C 0D 0E 0F  9...............
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F  ................
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F   !"#$%&'()*+,-./
30 31 32 33 34 35 36 37                          01234567
```

Operating System	Info. Request	Time Stamp Request	Address Mask Request	Echo Request
Debian GNU/ LINUX 2.2, Kernel 2.4 test 2 (*)	Not Answering	+ (**- DF**)	Not Answering	+ (**- DF**)
Redhat LINUX 6.2 Kernel 2.2.14 (*)	Not Answering	+ (**- DF**)	Not Answering	+ (**- DF**)
FreeBSD 4.0 (*)	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
FreeBSD 3.4	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
OpenBSD 2.7	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
OpenBSD 2.6	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
NetBSD	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
BSDI BSD/OS 4.0	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
BSDI BSD/OS 3.1	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
Solaris 2.5.1	Not Answering			
Solaris 2.6	Not Answering	+ (+ DF)	+ (+ DF)	+ (+ DF)
Solaris 2.7 (*)	Not Answering	+ (+ DF)	+ (+ DF)	+ (+ DF)
Solaris 2.8	Not Answering	+ (+ DF)	+ (+ DF)	+ (+ DF)
HP-UX v10.20			Not Answering	
HP-UX v11.0	Not Answering	Not Answeting	+ (+ DF)	+ (+ DF)
Compaq Tru64 v5.0 (*)		+ (+ DF)	Not Answering -	+ (+ DF)
Irix 6.5.3 (*)	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
Irix 6.5.8 (*)	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
AIX 4.1 (*)		+ (+ DF)	Not Answering	+ (+ DF)
AIX 3.2 (*)		+ (+ DF)	Not Answering	+ (+ DF)

Operating System	Info. Request	Time Stamp Request	Address Mask Request	Echo Request
ULTRIX 4.2 – 4.5 (*)		+ (- DF)	+ (- DF)	+ (- DF)
OpenVMS v7.1-2 (*)		+ (+ DF)	+ (+ DF)	+ (+ DF)
Novell Netware 5.1 SP1 (*)	Not Answering	Not Answering	Not Answering	+ (- DF)
Novell Netware 5.0 (*)	Not Answering	Not Answering	Not Answering	+ (- DF)
Novell Netware 3.12	Not Answering	Not Answering	Not Answering	+ (- DF)
Windows 95	Not Answering	Not Answering		
Windows 98 (*)	Not Answering	+ (- DF)	+ (- DF)	+ (+ DF)
Windows 98 SE (*)	Not Answering	+ (- DF)	+ (- DF)	+ (+ DF)
Windows ME (*)	Not Answering	+ (- DF)	Not Answering	+ (+ DF)
Windows NT 4 WRKS SP 3 (*)	Not Answering	Not Answering		
Windows NT 4 WRKS SP 6a (*)	Not Answering	Not Answering	Not Answering	+ (+ DF)
Windows NT 4 Server SP4	Not Answering	Not Answering	Not Answering	+ (+ DF)
Windows 2000 Professional (*)	Not Answering	+ (- DF)	Not Answering	+ (+ DF)
Windows 2000 Server (*)	Not Answering	+ (- DF)	Not Answering	+ (+ DF)

Table 12: DF Bit Echoing

6.12.6 Combining all together

If we combine every thing together than we can start from sending ICMP Echo requests with the DF bit set probing the attacked systems/network. The OSs, which will not echo the DF bit, would be Linux with kernel 2.2.x, Linux with kernel 2.4.x, Novel Netware, and Ultrix. We can distinguish the novel Netware machines from the rest of the OSs according to the TTL field values with the ICMP echo replies. The second stage would be sending ICMP Address Mask requests with the DF bit set. Microsoft Windows 98/98 SE and Ultrix would not echo the DF bit. We can distinguish

between the Ultrix machines and the Microsoft Windows machines, because of the different TTL field values in the ICMP Address Mask replies. We can now also identify the Ultrix machines with the first step – we know their IPs now. Than it leaves us with only the Linux boxes. Within two steps we are able of fingerprinting Novel Netware, Ultrix, Microsoft Windows 98/98 SE and Linux with kernel 2.2.x and kernel 2.4.x. In the last step we are sending ICMP Timestamp requests with the DF bit set to the same group of IPs we are probing. The OSs which do not echo back the DF bit are Linux with Kernel 2.2.x, Linux with Kernel 2.4, Ultrix, Microsoft Windows 98/98SE, Microsoft Windows ME, and Microsoft Windows 2000 Family. Since we already fingerprinted most of the OSs that do not echo back the DF bit in the first two steps, this enable us to fingerprint Microsoft Windows ME, and Microsoft Windows 2000 family.

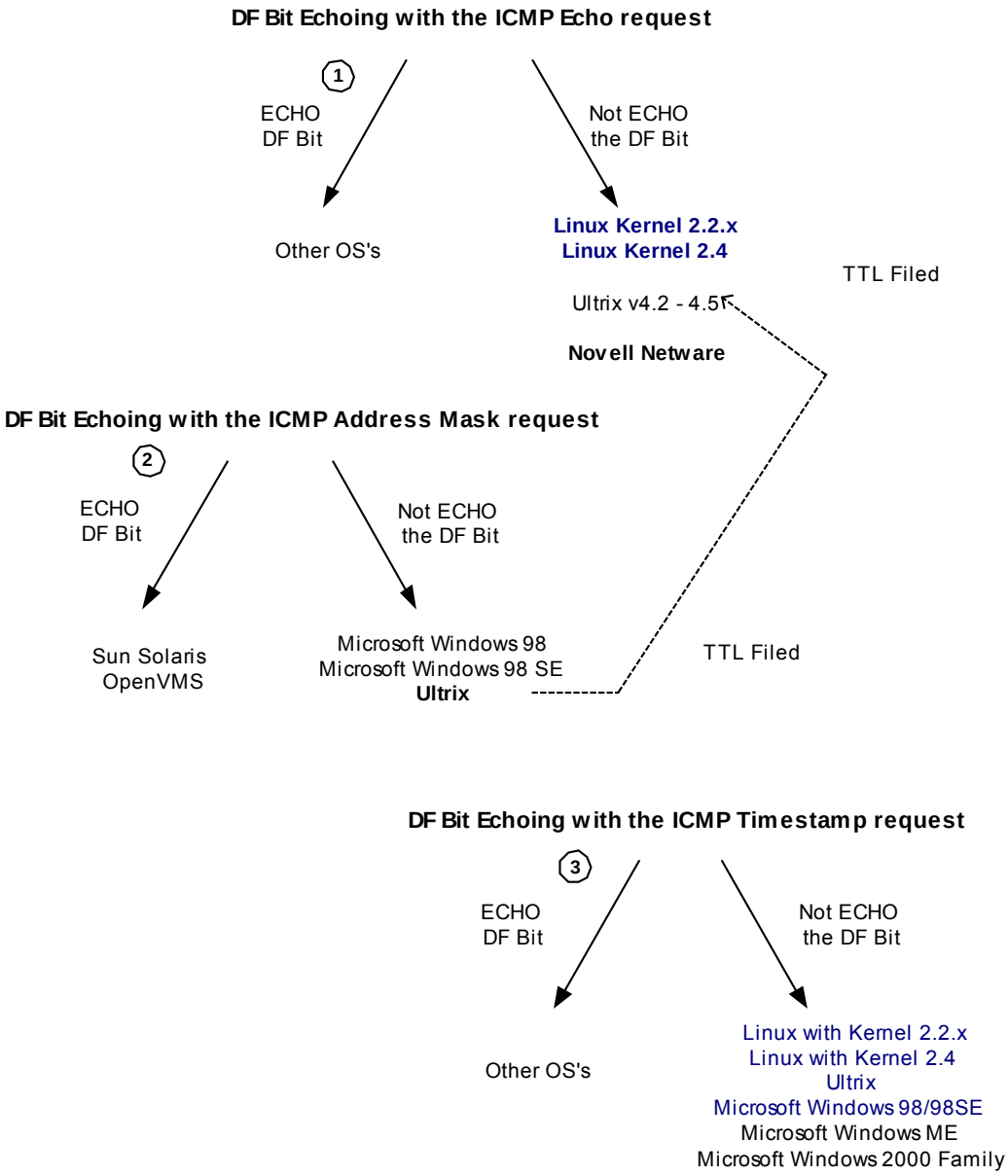


Diagram 5: DF Bit Echoing

6.13 What will not produce any gain compared to the effort and the detection ability?

6.13.1 Unusual Big ICMP ECHO Messages

What would happen if we would send unusual big ICMP ECHO message that would require its fragmentation? Would the queried operating systems will process the query correctly and produce an accurate reply?

```
[root@aik /root]# ping -s 1500 x.x.x.x
PING x.x.x.x (x.x.x.x) from y.y.y.y : 1500(1528) bytes of data.
1508 bytes from x.x.x.x: icmp_seq=0 ttl=241 time=1034.7 ms
1508 bytes from host_address (x.x.x.x): icmp_seq=2 ttl=241 time=1020.0
ms
1508 bytes from host_address (x.x.x.x): icmp_seq=3 ttl=241 time=1090.4
ms
1508 bytes from host_address (x.x.x.x): icmp_seq=5 ttl=241 time=1060.0
ms

--- x.x.x.x ping statistics ---
8 packets transmitted, 5 packets received, 37% packet loss
round-trip min/avg/max = 1000.2/1041.0/1090.4 ms
[root@aik /root]#
```

As it seems all the probed operating systems I have tested behaved correctly processing the query and sending the reply back.

What else can assist us with this kind of query?
The DF (Don't Fragment) bit.

Some operating systems would process the query and set the don't fragment bit on the fragments on the reply.

Operating System	DF bit set on the Reply?
Debian GNU/ LINUX 2.2, Kernel 2.4 test 2	-
Redhat LINUX 6.2 Kernel 2.2.14	-
FreeBSD 4.0	-
FreeBSD 3.4	-
OpenBSD 2.7	+ (DF set)
OpenBSD 2.6	
NetBSD	-
BSDI BSD/OS 4.0	-
BSDI BSD/OS 3.1	-
Solaris 2.5.1	
Solaris 2.6	+ (DF set)
Solaris 2.7	+ (DF set)
Solaris 2.8	+ (DF set)
HP-UX v10.20	
HP-UX v11.0	+ (DF set)

Operating System	DF bit set on the Reply?
Compaq Tru64 v5.0	-
Irix 6.5.3	-
Irix 6.5.8	-
AIX v4.1	-
AIX v3.2	-
ULTRIX v4.2 – v4.5	-
OpenVMS v7.1-2	-
Novell Netware 5.1 SP1	-
Novell Netware 5.0	-
Novell Netware 3.12	-
Windows 95	-
Windows 98	-
Windows 98 SE	-
Windows ME	-
Windows NT 4 WRKS SP 3	-
Windows NT 4 WRKS SP 6a	-
Windows NT 4 Server SP 4	-
Windows 2000 Professional SP1	-
Windows 2000 Server SP1	-
Windows 2000 Advanced Server	-

Table 13: DF Bit set on reply

7.0 Filtering ICMP on your Filtering Device to Prevent Scanning Using ICMP

7.1 Inbound

Incoming ICMP traffic that should be blocked in order to *prevent scanning techniques that were outlined in this paper are*:

- ICMP ECHO (used for Host Detection, traceroute & Inverse Mapping)
- ICMP ECHO Reply (used for Inverse Mapping)
- ICMP Time Stamp Request (used for Host Detection)
- ICMP Address Mask Request (used for Host Detection)
- All ICMP Message Types (Inverse Mapping Technique)

You should also block the IP directed broadcast on your border router.
Deny access to your Broadcast and Network addresses from the Internet.

7.2 Outbound

There are people who claim that any traffic type of ICMP should be allowed from a protected network to the Internet. This is not true. Filtering the incoming traffic does not mean we are protected from some of the security hazards I outlined in this paper.

7.2.1 ICMP ECHO Reply (Type 0)
Used to map a host using Host Detection.

7.2.2 ICMP Destination Unreachable Messages
I have demonstrated that host detection can be done with bad IP Header packets, which elicit various ICMP Parameter Problem and ICMP Destination Unreachable error messages from the probed machines and draw the attacked network topology.

7.2.3 ICMP “Fragmentation Needed and Don’t Fragment Bit was Set”
See section 3.5

7.2.4 ICMP ECHO (Type 8)
We have to have a Stateful filtering device that would perform Stateful inspection with ICMP in order to let ICMP ECHO Requests out, and receive only the corresponding ICMP ECHO Replies.

The current state with filtering devices is not that bright. Most of them do not perform Stateful inspection with the ICMP protocol. Allowing ICMP ECHO Replies inside our protected network is very dangerous and is not worth it.

Unless you use a Stateful filtering device with the ICMP protocol don’t let ICMP ECHO Replies into your protected network. This would make your requests useless so you better block them.

7.2.5 ICMP Time to Live Exceeded in Transit (Type 11 Code 0)
To eliminate traceroute and Reverse Mapping techniques we do not want to let a Time-to-Live Exceeded code 0 messages go back to the malicious computer attacker.

7.2.6 ICMP Fragmentation Reassembly Time Exceeded (Type 11 Code 1)

By blocking this ICMP type we eliminate the usage of a Host Detection technique, which sends only few fragments, form a fragmented datagram, and force the probed host to send us an ICMP Fragmentation Reassembly Time Exceeded error message back revealing his existence.

7.2.7 ICMP Parameter Problem

We have demonstrated that host detection can be made with bad IP Header packets, which would elicit various ICMP Parameter Problem and ICMP Destination Unreachable error messages from the probed machines.

7.2.8 ICMP Time Stamp Request & Reply

Time Stamp requests & replies can be used for Host Detection and Inverse Mapping.

7.2.9 ICMP Address Mask Request and Reply

Address Mask request & reply can be used for host detection and Inverse Mapping.

7.2.10 The liability Question

System administrator / Network administrator don't want to be held liable for an attack generated from there network by an abusive user (or a malicious computer attacker using a compromised system within the network). Therefore blocking some types of ICMP traffic from the protected network to the outside world is recommended for liability reasons:

- Destination Unreachable Codes 2-4
    - ICMP Destination Unreachable error messages 2-4 ("Port Unreachable", "Protocol Unreachable" and "Fragmentation Needed and DF Flag was Set") is a group of messages that are hard error conditions and when received should terminate a connection.

      This allow an attacker to send *fake* Destination Unreachable codes 2-4 to terminate valid connections between the attacked target and other hosts on the void.

      Old TCP/IP implementations terminat TCP connections when receiving those error messages. Modern TCP/IP implementations no longer terminate a TCP connection when receiving those error messages
- Source Quench messages
    - Since hosts still react to Source Quenches by slowing communication, they can be used as a Denial-of-Service measure.
- Redirect messages
    - If you can forge ICMP Redirect packets, and if your target host pays attention to them - ICMP Redirects may be employed for denial of service attacks, where a host is sent a route that loses it connectivity, or is sent an ICMP Network Unreachable packet telling it that it can no longer access a particular network.

This means that all outbound ICMP traffic should be disallowed.

7.3 Other Considerations

If you want to maintain strong ICMP filtering rules with your Firewall/Filtering-Device I suggest you block all incoming ICMP traffic except for Type 3 Code 4, which is used by the Path MTU Discovery process[39]. ICMP Type 3 Code 4 should be allowed from the Internet to your DMZ at least. Opening your Internal segmentation to this kind of traffic is questionable and depends on the facilities / activities / usage of the site and the level of filtering you wish to maintain.

If you will block incoming ICMP "Fragmentation Needed and Don't Fragment Bit was Set" your network performance will suffer from degradation. You should understand the security risks involving in opening this kind of traffic to your DMZ (& protected network) - The possibility of a Denial-of-Service, Inverse Mapping, Host Detection, and a one-way Covert communication channel (which was not been seen in the wild yet).

Another consideration could be the usage of network troubleshooting tools such as traceroute and ping. In the case of traceroute if the filtering device you are using does not support Stateful inspection with ICMP than allowing ICMP TTL Exceeded In Transit (Type 11, code 0) error messages inside the protected network could lead to various security hazards. The same goes with ping, where ICMP ECHO reply is even more dangerous when allowed inside the protected network (Inverse Mapping, Covert Channel and more security risks).

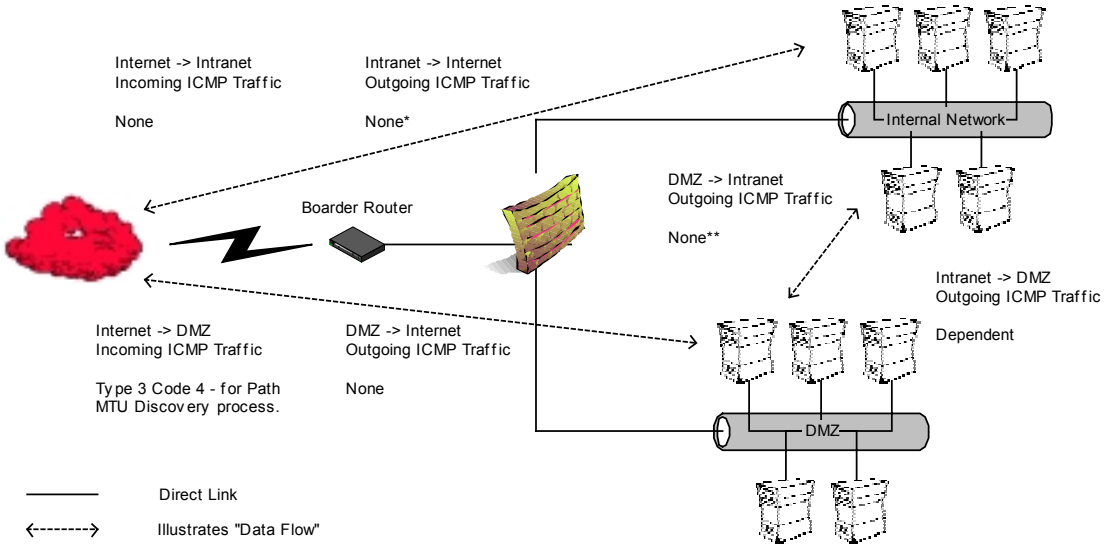
You can limit the number of systems that need to use the network troubleshooting tools with ACL, but bear in mind that those systems could be mapped from the Internet – and this is only the tip of the iceberg.

Internal Host(s) performance considerations – When blocking incoming ICMP Destination Unreachable Network/Host/Protocol/Port Unreachable ICMP error messages coming from the Internet, host(s) would hang when the destination system's network is unreachable/when a host is unreachable/when a protocol on the destination machine is not available/a port on a destination machine is closed. They all would hang until the timeout counter would reach zero. This little inconveniently is better than having the dangers other types of ICMP error messages inside your network can introduce.

Unless your filtering device is a real intelligence one, doing his work with dynamic tables and correlating correctly the ICMP replies with the requests, do not open your Internal network segment to no ICMP traffic type.

Some might offer to use a Proxy server with the ICMP protocol between the Internet and you protected network(s). A Proxy Server is only a tunnel – remember that.

[39] See Appendix B: "Fragmentation Needed but the Don't Fragment Bit was set" and the Path MTU Discovery Process.



* You can have a dedicated Management station that would be allowed to use ICMP for troubleshooting purposes only. The various ICMP replies should be allowed only by a statful inspection / Dynamic firewall. This means that no incoming ICMP is traffic is allowed to the management station, unless its correlated with a previous ICMP query this machine produced.

** If a malicious computer attacker breaks into the DMZ you do not want to provide him the means to scan internal machines & and the ability to query them directly.

Figure 12: Firewall ICMP Filtering Rules

8.0 Conclusion

The ICMP protocol is a very powerful tool in the hands of smart malicious computer attackers. Mapping, detecting, and fingerprinting of hosts and networking devices can be done in various ways as I have outlined in this paper.

It is extremely important to understand that ICMP traffic can be used for other malicious activities other than scanning, such as:

- Denial of Service Attacks
- Distributed Denial of Service Attacks
- Covert Channel Communications

Therefore filtering Inbound and Outbound ICMP traffic is very important and may help you in preventing risks to your computing environment.

9.0 Acknowledgment

9.1 Acknowledgment for version 1.0

I would like to thank the following people for their help with/during this research.

Ariel Pisetsky for going over this paper correcting my English, and for his moral support.

Christopher Tresco, Systems Administrator at the Massachusetts Institute of Technology provided necessary test systems to verify my findings.

Special thanks to mr2940 for his patience while I introduced my new ideas.

James Cudney, Michael, Pat, for their support when the times where bad.

9.1 Acknowledgment for version 2.0

I would like to thank Alfredo Andre's Omella author of SING for his help.

I would like to thank Fyodor for his help providing me with necessary test systems.

I would like to thank the people who provided feedback to the first version of this research paper, and to the people who provided feedback to my Bugtraq posts.

Appendix A: The ICMP Protocol[40]

Internet **C**ontrol **M**essage **P**rotocol (ICMP) is used for two types of operations: when a *router* or a *destination host* need to inform the source host about errors in a datagram processing, and for probing the network with request messages in order to determine general characteristics about the network (getting the information back, hopefully, with the reply messages).

Some of ICMP's characteristics are:

- ICMP uses IP as if it were a higher-level protocol, however, ICMP is already an internal part of IP, and must be implemented by every IP module.
- ICMP is used to provide feedback about some errors in a datagram processing, not to make IP reliable. Datagrams may still be undelivered without any report of their loss. If a higher level protocol that use IP need reliability he must implement it.
- No ICMP messages are sent in response to ICMP messages to avoid infinite repetitions. The exception is a response to ICMP query messages (ICMP Types 0,8-10,13-18. See Table 1 ICMP Query Messages).
- For fragmented IP datagrams ICMP messages are only sent about errors on fragment zero (first fragment).
- ICMP error messages are never sent in response to a datagram that is *destined* to a *broadcast* or a *multicast* address.
- ICMP error messages are never sent in response to a datagram sent as a link layer broadcast.
- ICMP error messages are never sent in response to a datagram whose source address does not represents a unique host – the source IP address cannot be *zero*, a *loopback* address, a *broadcast* address or a *multicast* address.
- ICMP Error messages are never sent in response to an IGMP massage of any kind.
- When an ICMP message of ***unknown type*** is received, it must be silently *discarded*.
- Routers will almost always generate ICMP messages but when it comes to a destination host(s), the number of ICMP messages generated is implementation dependent.

ICMP Query Messages	*ICMP error Messages*
Echo	Destination Unreachable
Router Advertisement	Source Quench
Router Solicitation	Redirect
Time Stamp	Time Exceeded
Information	Parameter Problem
Address Mask	

Table 14: ICMP message types

[40] ICMP is described in RFC 972 (http://www.ietf.org/rfc/rfc0972.txt) with updates in: RFC 896 (Source Quench), RFC 950 (Address Mask Extensions), RFC 1191 (Path MTU Discovery) & RFC 1256 (Router Discovery). Further clarifications about the ICMP protocol are included in RFC 1122 and in RFC 1812. STD 2 has redefine and clarified much of ICMP's core functionality.

A.1 ICMP Messages

ICMP messages are sent in IP datagrams. The protocol number will be always one (ICMP), and the Type-of-Service will be zero. The IP data field will contain the actual ICMP message:

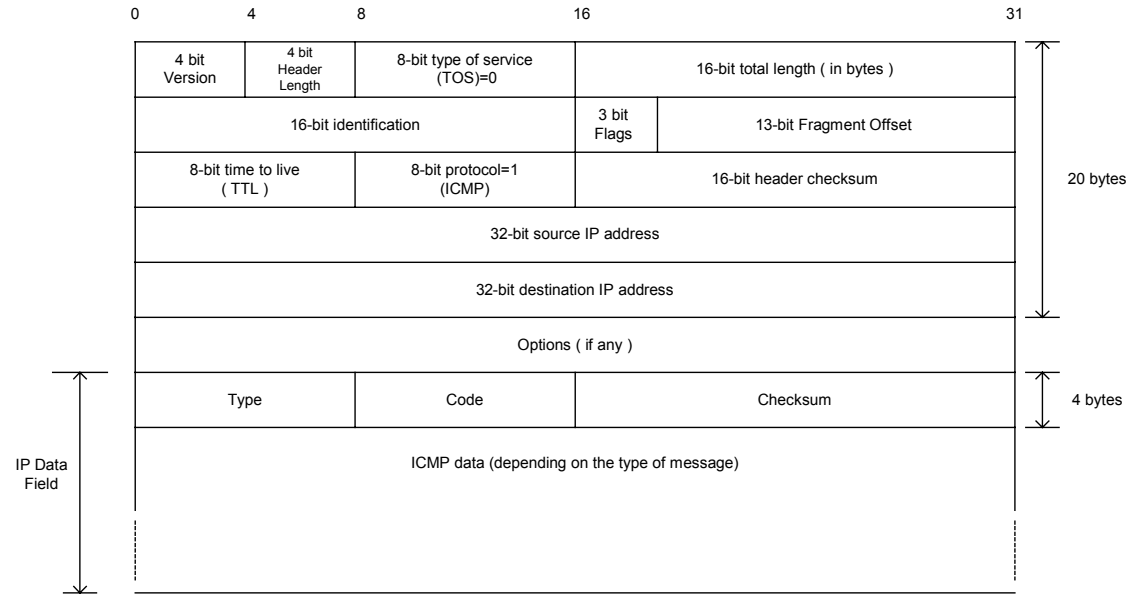


Figure 13: ICMP Message Format

ICMP error message length

Every ICMP error message includes the Internet (IP) Header and *at least* the first 8 data octets (bytes) of the datagram that triggered the error; more than 8 octets (bytes) *may* be sent; this header and data must be unchanged from the received datagram.

The **TYPE** field specifies the type of the message, while the error code for the datagram reported on by this ICMP message is contained in the **CODE** field. The code interpretation is dependent upon the message type.

Type	Name	Code
0	Echo Reply	0 No Code
1	Unassigned	
2	Unassigned	
3	Destination Unreachable[41]	
		0 Net Unreachable
		1 Host Unreachable
		2 Protocol Unreachable
		3 Port Unreachable
		4 Fragmentation Needed and Don't Fragment was Set
		5 Source Route Failed
		6 Destination Network Unknown
		7 Destination Host Unknown
		8 Source Host Isolated[42]
		9 Communication with Destination Network is Administratively Prohibited[43]
		10 Communication with Destination Host is Administratively Prohibited[44]
		11 Destination Network Unreachable for Type of Service.
		12 Destination Host Unreachable for Type of Service.
		13 Communication Administratively Prohibited.
		14 Host Precedence Violation
		15 Precedence cutoff in effect
4	Source Quench	0 No Code
5	Redirect	
		0 Redirect Datagram for the Network (or subnet)
		1 Redirect Datagram for the Host
		2 Redirect Datagram for the Type of Service and Network
		3 Redirect Datagram for the Type of Service and Host
6	Alternate Host Address	0 Alternate Address for Host
7	Unassigned	
8	Echo Request	0 No Code
9	Router Advertisement	0 No Code
10	Router Selection	0 No Code
11	Time Exceeded	
		0 Time to Live exceeded in Transit
		1 Fragment Reassembly Time Exceeded
12	Parameter Problem	
		0 Pointer indicates the error
		1 Missing a Required Option
		2 Bad Length
13	Timestamp	0 No Code
14	Timestamp Reply	0 No Code

[41] RFC 972 defines codes 1-5. RFC 1122 defines codes 6-12. RFC 1812 defines codes 13-15.
[42] Reserved for use by U.S. military agencies.
[43] Reserved for use by U.S. military agencies.
[44] Reserved for use by U.S. military agencies.

Type	Name	Code	
15	Information Request	0	No Code
16	Information Reply	0	No Code
17	Address Mask Request	0	No Code
18	Address Mask Reply	0	No Code
19	Reserved (for Security)	0	No Code
	20-29 reserved (for Robustness Experiment)		
30	Traceroute		
31	Datagram Conversion Error		
32	Mobile Host Redirect		
33	IPv6 Where-Are-You		
34	IPv6 I-Am-Here		
35	Mobile Registration Request		
36	Mobile Registration Reply		
39	SKIP		
40	Photuris		
		0	Reserved
		1	unknown security parameters index
		2	valid security parameters, but authentication failed
		3	valid security parameters, but decryption failed

Table 15: ICMP Types & Codes

Checksum – contains the 16bit one's complement of the one's complement sum of the ICMP message starting with the ICMP Type field. For computing this checksum, the checksum field is assumed to be zero.

Data

- With ICMP error messages it will contain a part of the original IP message for which this ICMP message was generated. The length of the DATA field equals the IP datagram length less the IP header length. Every ICMP error message includes the Internet (IP) Header and *at least* the first 8 data octets (bytes) of the datagram that triggered the error; more than 8 octets (bytes) *may* be sent; this header and data must be unchanged from the received datagram.

- With ICMP query messages the Data field will contain dependent information upon the query type.

Appendix B: ICMP "Fragmentation Needed but the Don't Fragment Bit was set" and the Path MTU Discovery Process [45]

When one host needs to send data to another host, the data is transmitted in a series of IP datagrams. We wish the datagrams be the largest size possible that does not require fragmentation[46] along the path from the source host to the destination host.

Fragmentation by the IP layer raises few problems:

- If one fragment from a packet is dropped, we need to retransmit the whole packet.
- Load on the routers, which needs to do the fragmentation.
- Some simpler firewalls would block all fragments because they do not contain the header information for a higher layer protocol needed for filtering.

The **M**aximum **T**ransfer **U**nit (***MTU***) is a link layer restriction on the maximum number of bytes of data in a single transmission. The smallest MTU of any link on the current path between two hosts is called the ***Path MTU***.

B.1 The PATH MTU Discovery Process

We use the Don't Fragment Bit Flag in the IP header to dynamically discover the Path MTU of a given route. The source host assumes that the PMTU of a path is the known MTU of its first hop. He will send all datagrams with that size, and set the Don't Fragment Bit. If along the path to the destination host, there is a router that needs to fragment the datagram in order to pass it to the next hop, an ICMP error message (Type 3 Code 4 "Fragmentation Needed and DF set") will be generated, since the Don't Fragment bit was set. When the sending host receives the ICMP error message he should reduce his assumed PMTU for the path.

The process can end when the estimated PMTU is low enough for the datagrams not to be fragmented. The source host itself can stop the process if he is willing to have the datagrams fragmented in some circumstances.

Usually the DF bit would be set in all datagrams, so if a route changes to the destination host, and the PMTU is lowered, than we would discover it.

The PMTU of a path might be increased over time, again because of a change in the routing topology. To detect it, a host should periodically increase its assumed PMTU for that link.

The link MTU field in the ICMP "Fragmentation Needed and DF set" error message, carries the MTU of the constricting hop, enabling the source host to know the exact value he needs to set the PMTU for that path to allow the voyage of the datagrams beyond that point (router) without fragmentation.

B.2 Host specification

A host must reduce his estimated PMTU for the relevant path when he receives the ICMP "Fragmentation Needed and the DF bit was set" error message. RFC 1191 does not outline a specific behavior that is expected from the sending host, because different applications may have different requirements, and different implementation architectures may favor different strategies.

---

[45] RFC 1191, http://www.ietf.org/rfc/rfc1191.txt, J. Mogul, S. Deering.

[46] When we send a packet that it is too large to be sent across a link as a single unit, a router needs to slice/split the packet into smaller parts, which contain enough information for the receiver to reassemble them. This is called fragmentation.

The only required behavior is that a host *must* attempt to avoid sending more messages with the same PMTU value in the near future. A host can either cease setting the Don't Fragment bit in the IP header (and allow fragmentation by the routers in the way) or reduce the datagram size. The better strategy would be to lower the message size because fragmentation will cause more traffic and consume more Internet resources.

A host using the PMTU Discovery process *must* detect decreases in Path MTU as fast as possible. A host *may* detect increases in Path MTU, by sending datagrams larger than the current estimated PMTU, which will usually be rejected by some router on the path to a destination since the PMTU usually will not increase. Since this would generate traffic back to the host, the check for the increases must be done at infrequent intervals. The RFC specify that an attempt for detecting an increasment *must not* be done less than 10 minutes after a datagram Too Big has been received for the given destination, or less than 2 minute after a previously successful attempt to increase.

The sending host must know how to handle an ICMP "Fragmentation Needed and the DF bit was set" error message that was sent by a device who does not know how to handle the PMTU protocol and does not include the next-hop MTU in the error message. Several strategies are available:

- The PMTU should be set to the minimum between the currently assumed PMTU and 576[47]. The DF bit should not be set in future datagrams for that path.
- Searching for the accurate value for the PMTU for a path. We keep sending datagrams with the DF bit set with lowered PMTU until we do not receive errors.

A host must not reduce the estimation of a Path MTU value below 68 bytes.

A host MUST not increase its estimate of the Path MTU in response to the contents of a Datagram Too Big message.

B.3 Router Specification

When a router cannot forward a datagram because it exceeded the MTU of the next-hop network and the Don't Fragment bit was set, he is required to generate an ICMP Destination Unreachable message to the source of the datagram., with the appropriate code indicating "Fragmentation needed and the Don't Fragment Bit was set". In the error message the router *must* include the MTU of the next-hop in a 16bit field inside the error message.

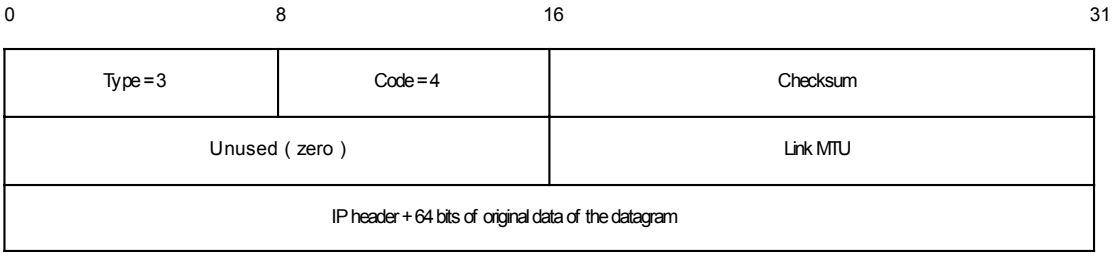


Figure 14: ICMP Fragmentation Required with Link MTU

[47] The usage of the lesser between 576 and the first-hop MTU as the PMTU for a destination, which is not connected to the same network was the old implementation. The results were the use of smaller datagrams than necessary, waste of Internet resources, and not being optimal.

The value of the next-hop MTU field should be set to the size in bytes of the largest datagram that could be forwarded, along the path of the original datagram, without being fragmented by this router. The size includes IP header plus IP data and no lower level headers should be included.

Because every router should be able to forward a datagram of 68 bytes without fragmenting it, the link MTU field should not contain a value less than 68.

B.4 The TCP MSS (Maximum Segment Size) Option and PATH MTU Discovery Process

The RFC specify that a host that is doing Path MTU Discovery *must not* send datagrams larger than 576 bytes unless the receiving host grants him permission.

When we are establishing a TCP connection both sides announce the maximum amount of data in one packet that should be sent by the remote system – The maximum segment size, MSS (if one of the ends does not specify an MSS, it defaults to 536 – there is no permission from the other end to send more than this amount). The packet generated would be, normally, 40 bytes larger than the MSS; 20 bytes for the IP header and 20 bytes for the TCP header. Most systems announce an MSS that is determined from the MTU on the interface that the traffic to the remote system passes out from the system through.

Each side upon receiving the MSS of the other side should not send any segments larger than the MSS received, regardless of the PMTU. After receiving the MSS value the Path MTU Discovery process will start to take affect. We will send our IP packets with the DF bit set allowing us to recognize points in the path to our destination that cannot process packets larger as the MSS of the destination host plus 40 bytes. When such an ICMP error message arrives, we should lower the PMTU to a path (according to the link MTU field, or if not used, to use the rules regarding the old implementation) and retransmit. The value of the link MTU cannot be higher than the MSS of the destination host. When retransmission occurs resulting from ICMP type 3 code 4 error message, the congestion windows should not change, but slow start should be initiated. The process continues until we adjust the correct PMTU of a path (not receiving ICMP error messages from the intermediate routers) which will allow us to fragment at the TCP layer which is much more efficient than at the IP layer.

Appendix C: Mapping Operating Systems for answering/discarding ICMP query message types

Operating System	Info. Request	Time Stamp Request	Address Mask Request	Address Mask Request Frag.	IP TTL on ICMP datagrams - In Reply -	IP TTL on ICMP datagrams - In Req. -
Debian GNU/ LINUX 2.2, Kernel 2.4 test 2 (*)	-	+	-	-	255	64
Redhat LINUX 6.2 Kernel 2.2.14 (*)	-	+	-	-	255	64
LINUX Kernel 2.0.x					64	64
FreeBSD 4.0 (*)	-	+	-	-	255	255
FreeBSD 3.4	-	+	-	-	255	255
OpenBSD 2.7	-	+	-	-	255	255
OpenBSD 2.6	-	+	-	-	255	255
NetBSD	-	+	-	-	255	
BSDI BSD/OS 4.0	-	+	-	-	255	
BSDI BSD/OS 3.1	-	+	-	-	255	
Solaris 2.5.1	-	+	+	+ (0.0.0.0)	255	255
Solaris 2.6	-	+	+	+ (0.0.0.0)	255	255
Solaris 2.7 (*)	-	+	+	+ (0.0.0.0)	255	255
Solaris 2.8	-	+	+	+ (0.0.0.0)	255	255
HP-UX v10.20	+	+	-	-	255	255
HP-UX v11.0	-	-	+	+ (0.0.0.0)	255	
Compaq Tru64 v5.0 (*)	+	+	-	-	64	
Irix 6.5.3 (*)	-	+	-	-	255	
Irix 6.5.8 (*)	-	+	-	-	255	
AIX 4.1 (*)	+	+	-	-	255	
AIX 3.2 (*)	+	+	-	-	255	
ULTRIX 4.2 – 4.5 (*)	+	+	+	+	255	
OpenVMS v7.1-2 (*)	+	+	+	+	255	
Novell Netware 5.1 SP1 (*)	-	-	-	-	128	
Novell Netware 5.0 (*)	-	-	-	-	128	
Novell Netware 3.12	-	-	-	-	128	
Windows 95	-	-	+	+	32	32
Windows 98 (*)	-	+	+	+	128	32
Windows 98 SE (*)	-	+	+	+	128	32
Windows ME (*)	-	+	-	-	128	32
Windows NT 4 WRKS SP 3 (*)	-	-	+	+	128	32
Windows NT 4 WRKS SP 6a (*)	-	-	-	-	128	32
Windows NT 4 Server SP4	-	-	-	-	128	32
Windows 2000 Professional (*)	-	+	-	-	128	128
Windows 2000 Server (*)	-	+	-	-	128	128

Networking Devices	Info. Request	Time Stamp Request	Address Mask Request	Address Mask Request Frag.	IP TTL on ICMP datagrams - In Reply -	IP TTL on ICMP datagrams - In Req. -
Cisco Catalyst 5505 with OSS v4.5	+	+	+	-	60	60
Cisco Catalyst 2900XL with IOS 11.2	+	+	-	-	255	
Cisco 3600 with IOS 11.2	+	+	-	-	255	
Cisco 7200 with IOS 11.3	+	+	-	-	255	255
Intel Express 8100 ISDN Router (*)	-	-	+	+	64	

Appendix D: ICMP Query Message Types with Code field !=0

Operating System	Info. Request	Time Stamp Request	Address Mask Request	ECHO Request
Debian GNU/ LINUX 2.2, Kernel 2.4 test 2 (*)	-	+ (!=0)	-	+ (!=0)
Redhat LINUX 6.2 Kernel 2.2.14 (*)	-	+ (!=0)	-	+ (!=0)
FreeBSD 4.0 (*)	-	+ (!=0)	-	+ (!=0)
FreeBSD 3.4	-	+ (!=0)	-	
OpenBSD 2.7	-	+ (!=0)	-	+ (!=0)
OpenBSD 2.6	-	+ (!=0)	-	+ (!=0)
NetBSD	-	+ (!=0)	-	+ (!=0)
BSDI BSD/OS 4.0 (*)	-	+ (!=0)	-	+ (!=0)
BSDI BSD/OS 3.1 (*)	-	+ (!=0)	-	+ (!=0)
Solaris 2.5.1	*	+ (!=0)	+ (!=0)	+ (!=0)
Solaris 2.6	*	+ (!=0)	+ (!=0)	+ (!=0)
Solaris 2.7 (*)	*	+ (!=0)	+ (!=0)	+ (!=0)
Solaris 2.8	*	+ (!=0)	+ (!=0)	+ (!=0)
HP-UX v10.20	+ (!=0)	+ (!=0)	-	
HP-UX v11.0	-	-	+ (!=0)	+ (!=0)
Compaq Tru64 v5.0 (*)	+ (!=0)	+ (!=0)	-	+ (!=0)
Irix 6.5.3 (*)	-	+ (!=0)	-	+ (!=0)
Irix 6.5.8 (*)	-	+ (!=0)	-	+ (!=0)
AIX 4.1 (*)	+ (!=0)	+ (!=0)	-	+ (!=0)
Aix 3.2 (*)	+ (!=0)	+ (!=0)	-	
ULTRIX 4.2 - 4.5 (*)	+ (!=0)	+ (!=0)	+ (!=0)	+ (!=0)
OpenVMS v7.1-2 (*)	+ (!=0)	+ (!=0)	+ (!=0)	+ (!=0)
Novell Netware 5.1 SP1 (*)	-	-	-	+ (!=0)
Novell Netware 5.0 (*)	-	-	-	+ (!=0)
Novell Netware 3.12 (*)	-	-	-	+ (!=0)
Windows 95	-	-	+	+ (0)
Windows 98 (*)	-	- (CHANGE)	+	+ (0)
Windows 98 SE (*)	-	- (CHANGE)	+	+ (0)
Windows ME (*)	-	- (CHANGE)	-	+ (0)
Windows NT 4 WRKS SP 3 (*)	-	-	+	+ (0)
Windows NT 4 WRKS SP 6a (*)	-	-	-	+ (0)
Windows NT 4 Server SP4	-	-	-	+ (0)
Windows 2000 Professional (*)	-	- (CHANGE)	-	+ (0)
Windows 2000 Server (*)	-	- (CHANGE)	-	+ (0)

Networking Devices	Info. Request	Time Stamp Request	Address Mask Request	ECHO Request
Cisco Catalyst 5505 with OSS v4.5	+	+	+	+ (!0)
Cisco Catalyst 2900XL with IOS 11.2	+	+	-	+ (!0)
Cisco 3600 with IOS 11.2				+ (!0)
Cisco 7200 with IOS 11.3	+	+	-	+ (!0)
Intel Express 8100 ISDN Router (*)				

Appendix E: ICMP Query Message Types aimed at a Broadcast Address

Operating System	Info. Request Broadcast	Time Stamp Request Broadcast	Address Mask Request Broadcast	Echo Request Broadcast
Debian GNU/ LINUX 2.2, Kernel 2.4 test 2				+
Redhat LINUX 6.2 Kernel 2.2.14 (*)	-	+	-	+
FreeBSD 4.0 (*)	-	-	-	-
FreeBSD 3.4	-	-	-	-
OpenBSD 2.7	-	-	-	-
OpenBSD 2.6	-	-	-	-
NetBSD				
BSDI BSD/OS 4.0 (*)				
BSDI BSD/OS 3.1 (*)				
Solaris 2.5.1	*	+	-	+
Solaris 2.6	*	+	-	+
Solaris 2.7	*	+	-	+
Solaris 2.8	*	+	-	+
HP-UX v10.20	+	+	-	+
Compaq Tru64 v5.0 (*)				
Irix 6.5.3 (*)				
Irix 6.5.8 (*)				
AIX 4.1 (*)				
AIX 3.2 (*)				
ULTRIX 4.2 – 4.5 (*)				
OpenVMS v7.1-2 (*)				
Novell Netware 5.1 SP1 (*)				
Novell Netware 5.0 (*)				
Novell Netware 3.12 (*)				
Windows 95				
Windows 98	-	-	-	-
Windows 98 SE (*)	-	-	-	-
Windows ME (*)	-	-	-	-
Windows NT 4 WRKS SP 3 (*)	-	-	-	-
Windows NT 4 WRKS SP 6a (*)	-	-	-	-
Windows NT 4 Server SP4	-	-	-	-
Windows 2000 Professional (*)	-	-	-	-
Windows 2000 Server (*)	-	-	-	-

Networking Devices	Info. Request Broadcast	Time Stamp Request Broadcast	Address Mask Request Broadcast		Echo Broadcast	
Cisco Catalyst 5505 with OSS v4.5	+	+	+		+	
Cisco Catalyst 2900XL with IOS 11.2	+	-	-		+	
Cisco 3600 with IOS 11.2	+	-	-			
Cisco 7200 with IOS 11.3	+	-	-		+	
Intel Express 8100 ISDN Router (*)	-	-	-		-	Big Question Marks

Appendix F: ICMP Query Message Types with TOS! = 0

Operating System	Information Request With TOS!=0x00	Time Stamp Request With TOS!=0x00	Address Mask Request With TOS!=0x00	Echo Request With TOS!=0x00
Debian GNU/ LINUX 2.2, Kernel 2.4 test 2 (*)	Not Answering	!=0x00	Not Answering	!=0x00
Redhat LINUX 6.2 Kernel 2.2.14 (*)	Not Answering	!=0x00	Not Answering	!=0x00
FreeBSD 4.0 (*)	Not Answering	!=0x00	Not Answering	!=0x00
FreeBSD 3.4	Not Answering		Not Answering	
OpenBSD 2.7 (*)	Not Answering	!=0x00	Not Answering	!=0x00
OpenBSD 2.6	Not Answering	!=0x00	Not Answering	!=0x00
NetBSD	Not Answering	!=0x00	Not Answering	!=0x00
BSDI BSD/OS 4.0 (*)	Not Answering	!=0x00	Not Answering	!=0x00
BSDI BSD/OS 3.1 (*)	Not Answering	!=0x00	Not Answering	!=0x00
Solaris 2.5.1	Not Implemented			
Solaris 2.6	Not Implemented			
Solaris 2.7 (*)	Not Implemented	!=0x00	!=0x00	!=0x00
Solaris 2.8 (*)	Not Implemented	!=0x00	!=0x00	!=0x00
HP-UX v10.20			Not Answering	
HP-UX v11.0	Not Answering	Not Answering	!=0x00	!=0x00
Compaq Tru64 v5.0 (*)		!=0x00	Not Answering	!=0x00
Irix 6.5.3 (*)	Not Answering	!=0x00	Not Answering	!=0x00
Irix 6.5.8 (*)	Not Answering	!=0x00	Not Answering	!=0x00
AIX 4.1 (*)		!=0x00	Not Answering	!=0x00
AIX 3.2 (*)		!=0x00	Not Answering	!=0x00
ULTRIX 4.2 – 4.5 (*)		**0x00**	**0x00**	**0x00**
OpenVMS v7.1-2 (*)		!=0x00	!=0x00	!=0x00
Novell Netware 5.1 SP1 (*)	Not Answering	Not Answering	Not Answering	**0x00**
Novell Netware 5.0 (*)	Not Answering	Not Answering	Not Answering	**0x00**
Novell Netware 3.12 (*)	Not Answering	Not Answering	Not Answering	**0x00**
Windows 95	Not Answering	Not Answering		
Windows 98 (*)	Not Answering	**0x00**	**0x00**	!=0x00
Windows 98 SE (*)	Not Answering	**0x00**		!=0x00
Windows ME (*)	Not Answering	**0x00**	Not Answering	!=0x00
Windows NT 4 WRKS SP 3 (*)	Not Answering	Not Answering		!=0x00
Windows NT 4 WRKS SP 6a (*)	Not Answering	Not Answering	Not Answering	!=0x00
Windows NT 4 Server SP4	Not Answering	Not Answering	Not Answering	!=0x00
Windows 2000 Professional (*)	Not Answering	**0x00**	Not Answering	**0x00**
Windows 2000 Server (*)	Not Answering	**0x00**	Not Answering	**0x00**

Appendix G: DF Bit Echoing

Operating System	Info. Request	Time Stamp Request	Address Mask Request	Echo Request
Debian GNU/ LINUX 2.2, Kernel 2.4 test 2 (*)	Not Answering	+ (**- DF**)	Not Answering	+ (**- DF**)
Redhat LINUX 6.2 Kernel 2.2.14 (*)	Not Answering	+ (**- DF**)	Not Answering	+ (**- DF**)
FreeBSD 4.0 (*)	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
FreeBSD 3.4	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
OpenBSD 2.7	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
OpenBSD 2.6	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
NetBSD	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
BSDI BSD/OS 4.0	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
BSDI BSD/OS 3.1	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
Solaris 2.5.1	Not Answering			
Solaris 2.6	Not Answering	+ (+ DF)	+ (+ DF)	+ (+ DF)
Solaris 2.7 (*)	Not Answering	+ (+ DF)	+ (+ DF)	+ (+ DF)
Solaris 2.8	Not Answering	+ (+ DF)	+ (+ DF)	+ (+ DF)
HP-UX v10.20			Not Answering	
HP-UX v11.0	Not Answering	Not Answeting	+ (+ DF)	+ (+ DF)
Compaq Tru64 v5.0 (*)		+ (+ DF)	Not Answering -	+ (+ DF)
Irix 6.5.3 (*)	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
Irix 6.5.8 (*)	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
AIX 4.1 (*)		+ (+ DF)	Not Answering	+ (+ DF)
AIX 3.2 (*)		+ (+ DF)	Not Answering	+ (+ DF)
ULTRIX 4.2 – 4.5 (*)		+ (**- DF**)	+ (**- DF**)	+ (**- DF**)
OpenVMS v7.1-2 (*)		+ (+ DF)	+ (+ DF)	+ (+ DF)
Novell Netware 5.1 SP1 (*)	Not Answering	Not Answering	Not Answering	+ (**- DF**)
Novell Netware 5.0 (*)	Not Answering	Not Answering	Not Answering	+ (**- DF**)
Novell Netware 3.12	Not Answering	Not Answering	Not Answering	+ (**- DF**)
Windows 95	Not Answering	Not Answering		
Windows 98 (*)	Not Answering	+ (**- DF**)	+ (**- DF**)	+ (+ DF)
Windows 98 SE (*)	Not Answering	+ (**- DF**)	+ (**- DF**)	+ (+ DF)
Windows ME (*)	Not Answering	+ (**- DF**)	Not Answering	+ (+ DF)
Windows NT 4 WRKS SP 3 (*)	Not Answering	Not Answering		
Windows NT 4 WRKS SP 6a (*)	Not Answering	Not Answering	Not Answering	+ (+ DF)
Windows NT 4 Server SP4	Not Answering	Not Answering	Not Answering	+ (+ DF)
Windows 2000 Professional (*)	Not Answering	+ (**- DF**)	Not Answering	+ (**- DF**)
Windows 2000 Server (*)	Not Answering	+ (**- DF**)	Not Answering	+ (**- DF**)

For corrections/additions/suggestions for this research paper, please send email to ofir@itcon-ltd.com. Further Information and updates would be posted to http://www.sys-security.com.

Thank you for reading

Ofir Arkin

The Sys-Security Group

Founder

http://www.sys-security.com
ofir.arkin@sys-security.com

ITCon – Information Technology Consultants

Senior Security Analyst

http://www.itcon-ltd.com
ofir@itcon-ltd.com