# Unverified Fields - A Problem with Firewalls & Firewall Technology Today

**Ofir Arkin**


**The Sys-Security Group**

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

**Version 1.0**
**October 2000**

## Introduction

The following problem (as discussed in this paper) has not yet been identified. Certain firewalls today, will not authenticate the validity of certain protocol fields, within the packet they are processing.

The risk is exposure of information. What kind of information can be exposed?
Mainly it will be unique patterns of behavior produced by the probed machines answering our crafted queries (or other kind of network traffic initiated in order to elicit a reply).

In my research paper "ICMP Usage In Scanning[1]" I have introduced new operating system fingerprinting methods based on changing values inside certain fields of the ICMP datagram. Using some of these methods I will demonstrate the risk.

It is important to understand that I am using the ICMP protocol as an example. Other protocols can be used as well for this task.

[1] ICMP Usage In Scanning, by Ofir Arkin. Available from http://www.sys-security.com.

## The Tests Performed

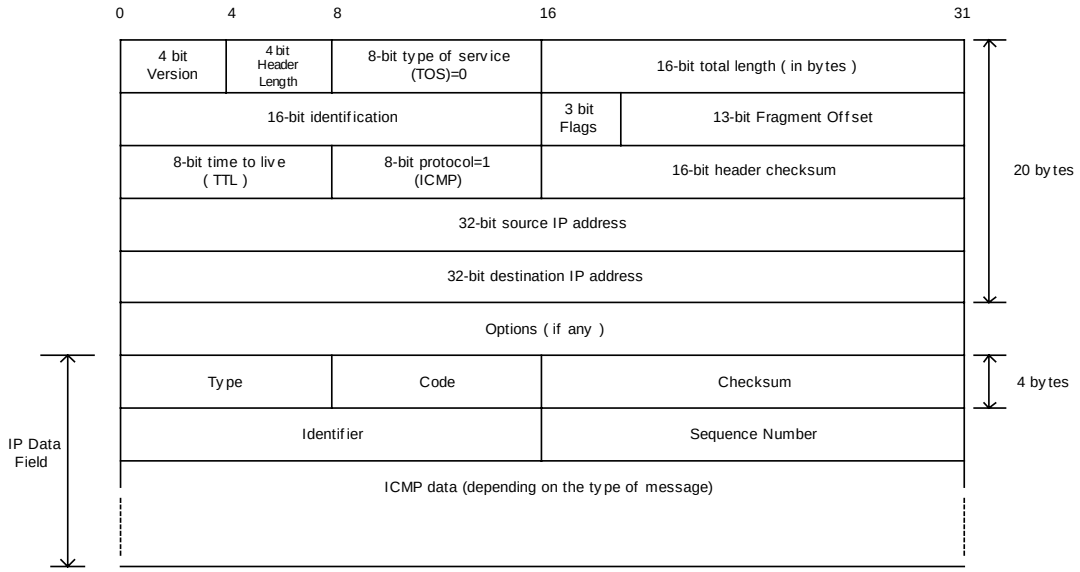


Figure 1: ICMP Echo Request Datagram

The following tests were performed against a machine protected by a Check Point Firewall-1, version 4.1 sp2 (versions 3.0b and 4.0 were also checked). We have allowed ICMP Echo Request traffic to reach the tested machine, and ICMP Echo Replies to be sent back to the Internet. All other traffic was blocked.

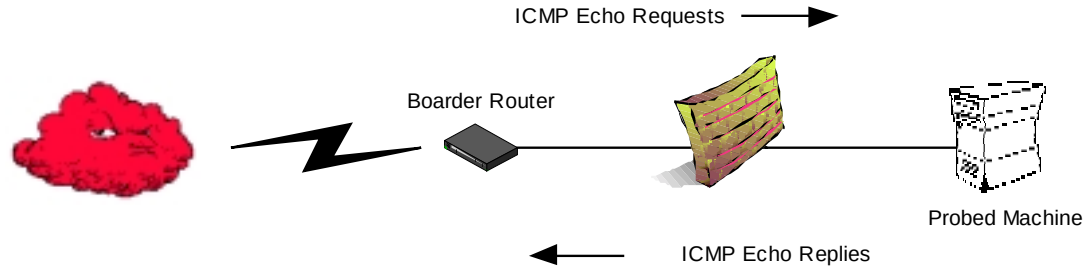


Figure 2: Initial setup

### Using Wrong Code Field Values

The first method to be tested was “Using wrong code values with ICMP Echo Requests”. With this method, a code field value different than the normal (0) is sent with the ICMP Echo request. Typically, Microsoft Windows operating systems will zero out this field in their ICMP Echo replies. UNIX and UNIX-like operating systems will leave this value intact in their ICMP Echo replies.

```
[root@godfather /root]# sing[2] -x 38 -c 5 Host_Address
SINGing to Host_Address (IP_Address): 16 data bytes
16 bytes from IP_Address: seq=0 ttl=232 TOS=0 time=690.118 ms
16 bytes from IP_Address: seq=1 ttl=232 TOS=0 time=680.129 ms
16 bytes from IP_Address: seq=2 ttl=232 TOS=0 time=680.120 ms
16 bytes from IP_Address: seq=3 ttl=232 TOS=0 time=670.719 ms
```

[2] SING, written by Alfredo Andres Omella, can be downloaded from http://www.sourceforge.net/projects/sing.

```
16 bytes from IP_Address: seq=4 ttl=232 TOS=0 time=671.754 ms

--- Host_Address sing statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 670.719/678.568/690.118 ms
[root@godfather /root]#
```

The tcpdump trace:

```
16:07:08.668464 ppp0 > x.x.x.x > y.y.y.y: icmp: echo request (ttl
255, id 13170)
                  4500 0024 3372 0000 ff01 0237 xxxx xxxx
                  yyyy yyyy 0826 580e 2604 0000 7c5a e839
                  0b33 0a00
16:07:09.356601 ppp0 < y.y.y.y > x.x.x.x: icmp: echo reply (ttl 232,
id 20802)
                  4500 0024 5142 0000 e801 fb66 yyyy yyyy
                  xxxx xxxx 0026 600e 2604 0000 7c5a e839
                  0b33 0a00
```

The forged code field was echoed back. The operating system in question is probably a UNIX or a UNIX-like operating system.

## Using the Unused

We can set the unused bit with our ICMP Echo Requests. HPUX 11.0 and Sun Solaris will echo back the unused bit, when generating an ICMP Echo reply. Other operating systems will set the value of this field according to the RFC – zero.

```
[root@godfather bin]# ./sing -echo -U host_address
SINGing to host_address (IP_Address): 16 data bytes
16 bytes from IP_Address: seq=1 RF! ttl=234 TOS=0 time=430.036 ms
16 bytes from IP_Address: seq=2 RF! ttl=234 TOS=0 time=430.042 ms
16 bytes from IP_Address: seq=3 RF! ttl=234 TOS=0 time=470.073 ms
16 bytes from IP_Address: seq=4 RF! ttl=234 TOS=0 time=430.039 ms
16 bytes from IP_Address: seq=5 RF! ttl=234 TOS=0 time=431.087 ms

--- host_address sing statistics ---
6 packets transmitted, 5 packets received, 16% packet loss
round-trip min/avg/max = 430.036/438.255/470.073 ms
[root@godfather bin]#
```

The tcpdump trace:

```
11:25:44.689692 ppp0 > x.x.x.x > y.y.y.y: icmp: echo request (ttl
255, id 13170)
                  4500 0024 3372 8000 ff01 88c1 xxxx xxxx
                  yyyy yyyy 0800 91af f003 0500 888c dd39
                  0186 0a00
11:25:45.119650 ppp0 < y.y.y.y > x.x.x.x: icmp: echo reply (ttl 234,
id 50096)
                  4500 0024 c3b0 8000 ea01 0d83 yyyy yyyy
                  xxxx xxxx 0000 99af f003 0500 888c dd39
                  0186 0a00
```

The Unused bit is echoed with the ICMP Echo reply. Who echos back the Unused Bit?

Sun Solaris, HP-UX 11.0 (probably version 10.30 as well).

## The DF Bit Echoing Test

With this test, we set the DF Bit with our probes. Some operating systems will not echo back this bit with their ICMP Echo replies. This will help us limit the range of operating systems on the target machine we are trying to identify.

```
[root@godfather bin]# ./sing -echo -G host_address
SINGing to host_address (IP_Address): 16 data bytes
16 bytes from IP_Address: seq=0 DF! ttl=234 TOS=0 time=458.907 ms
16 bytes from IP_Address: seq=1 DF! ttl=234 TOS=0 time=460.047 ms
16 bytes from IP_Address: seq=2 DF! ttl=234 TOS=0 time=460.040 ms
16 bytes from IP_Address: seq=3 DF! ttl=234 TOS=0 time=468.954 ms
16 bytes from IP_Address: seq=4 DF! ttl=234 TOS=0 time=460.061 ms
16 bytes from IP_Address: seq=6 DF! ttl=234 TOS=0 time=470.045 ms

--- host_address sing statistics ---
7 packets transmitted, 6 packets received, 14% packet loss
round-trip min/avg/max = 458.907/463.009/470.045 ms
[root@godfather bin]#
```

The tcpdump trace:

```
11:26:18.119691 ppp0 > x.x.x.x > y.y.y.y: icmp: echo request (DF)
(ttl 255, id 13170)
                  4500 0024 3372 4000 ff01 c8c1 xxxx xxxx
                  yyyy yyyy 0800 f661 ff03 0600 aa8c dd39
                  73d3 0100
11:26:18.589652 ppp0 < y.y.y.y > x.x.x.x: icmp: echo reply (DF) (ttl
234, id 50102)
                  4500 0024 c3b6 4000 ea01 4d7d yyyy yyyy
                  xxxx xxxx 0000 fe61 ff03 0600 aa8c dd39
                  73d3 0100
```

The DF bit was echoed back. This rule out LINUX machines based on Kernel 2.2.x and 2.4.x, Ultrix, Novell Netware, and Microsoft Windows 2000 family of operating systems from being used by the probed machine.

## The TOS Echoing Test

The last test that was performed used the TOS field. Usually an ICMP Echo request will use the TOS default value of 0x00. Another value may be used. An ICMP Echo reply responding to this request should use the same TOS value as with the request.

```
[root@godfather bin]# ./sing -echo -TOS 8  host_address
SINGing to host_address (IP_Address): 16 data bytes
16 bytes from IP_Address: seq=1 ttl=234 TOS=8 time=450.089 ms
16 bytes from IP_Address: seq=2 ttl=234 TOS=8 time=470.057 ms
16 bytes from IP_Address: seq=3 ttl=234 TOS=8 time=470.049 ms
16 bytes from IP_Address: seq=4 ttl=234 TOS=8 time=450.070 ms
16 bytes from IP_Address: seq=5 ttl=234 TOS=8 time=450.061 ms

--- host_address sing statistics ---
6 packets transmitted, 5 packets received, 16% packet loss
round-trip min/avg/max = 450.061/458.065/470.057 ms
```

```
[root@godfather bin]#
```

The tcpdump trace:

```
11:28:28.839695 ppp0 > x.x.x.x > y.y.y.y: icmp: echo request [tos
0x8]  (ttl 255, id 13170)
                  4508 0024 3372 0000 ff01 08ba xxxx xxxx
                  yyyy yyyy 0800 e564 0204 0500 2c8d dd39
                  f5cf 0c00
11:28:29.289664 ppp0 < y.y.y.y > x.x.x.x: icmp: echo reply [tos 0x8]
(ttl 234, id 15779)
                  4508 0024 3da3 0000 ea01 1389 yyyy yyyy
                  xxxx xxxx 0000 ed64 0204 0500 2c8d dd39
                  f5cf 0c00
```

As shown in the tcpdump, the TOS field value was echoed back. This behavior rules out the following operating systems, which do not echo back the TOS field value with their ICMP Echo replies: Microsoft Windows 2000 operating system family, Novell Netware, and Ultrix.

## Conclusion

The various tests outlined that the system in question is probably a SUN Solaris machine or an HPUX 11.0 machine. We could not differentiate between the two, because ICMP Echo requests initiated from the probed machine were blocked by the Firewall. If we had been seeing those, it would give us an indication about the usage of ICMP Echo requests in the PMTU discovery process HPUX 11.0 & 10.30 machines take.

We know for certain that the queried machine is not a LINUX operating system based on Kernel 2.2.x or 2.4; A Microsoft Windows 2000 based machine; an Ultrix machine; or a Novel Netware based machine.

The fields within the ICMP datagram that we have used were:

- ICMP Code Field
- The Unused Bit (IP Header)
- The DF Bit (IP Header)
- The TOS field (IP Header)

The tests we've made did not take into consideration the default behavior taken by several operating systems. For example, setting the DF bit in replies for any ICMP Query message by Sun Solaris as its global PMTU discovery process (and maintenance). And we have limited ourselves only to the usage of ICMP Echo request.

The firewall in this case was used as a tunnel. It verified the fields it needs for checking the datagram against the firewall's rule base, and did not verify the integrity of the other fields of the datagram. We gave extra meaning for a simple ICMP Echo request, and the firewall failed to cope with it.

## Why this is a bit different from other methods?

Other methods might use legitimate parameters in order to try to fingerprint the operating system used on the machine probed. One example might be using the TCP Options. With the method I have introduced in this paper, fields, which have to have a default (some fields should be predefined) value, were used for the same purpose. This highlights the problem with firewalls today – too much is not taken into consideration. The number of probing methods for a malicious computer attacker to use against machines hidden by a firewall still allows him to get a clear indication of the operating systems and give him crucial information (if those machines are accessible to the Internet).

Sure, there are other methods we can use to learn about one-machine operating systems. I can name banner grabbing for example. But what would happen if some of those methods will fail or be blocked? Than the idea I have presented in this paper will allow a malicious computer attacker to get the same results.

I did expect that I would not able to use those fields. My initial assumption was that firewalls verify those fields as their standard behavior. Obviously I was wrong.

### Why will this work with other types of firewalls as well?

Reverse Proxies, Dynamic firewalls, Packet Filtering Firewalls, they all verify the fields they need in order to match the packets characteristics (SOCKETS pair) to their ACL's (or tables). This is done to determine if the traffic examined is allowed through the firewall (or not).

Most of them look at the destination IP Address, destination Port, Source IP Address, Source Port to decide the packet's future routing.

Taking proxies for example, some of them will not use brand new parameters for their connection with the target. They will change the Sockets pair information, and will be used as a tunnel for the probing.

The information the reply will carry will not be harmed. It will be intact, just the sockets pair will change. A simple tunnel for our probes.

Some might question the solution – checking and validating those fields' values; in saying it will slow down firewalls and other filtering devices activities.

I say for them – Why do we have firewalls / filtering devices in the first place? Is it just to protect my ports, or just to introduce a barrier for the hacker? Stop and think for a minute. Could I have the same defense with a host facing the Internet and no commercial firewall's protection? Of course I can.

The firewall devices presented in today's market simply do not supply us with the solution.

Will a very strict filtering rule base help me?
It may help you. But all of your machines connected by an Internet host will be in danger for this probing method.

Bare In mind: This is not only limited to the ICMP protocol only. Other protocols could be used easily as well.