

Thick Client Penetration Testing: Traffic Analysis

March 18, 2021 By Raj Chandel

Traffic analysis is one of the crucial parts of any successful penetration test. In this article, we're going to discuss some of the different techniques that can be used to analyze thick client applications. If a thick client using HTTP traffic then it is pretty straight forward to intercept the traffic. We can use tools like

To Intercept the HTTP like Traffic: –

- Burp Suite

To Intercept TCP like Traffic: –

- Wireshark
- MITM Relay + Burp Suite
- Echo Mirage (Properly Maintained)

As we're pen-testing Damn Vulnerable thick client applications and DVTA is using non-HTTP protocols for example, FTP. It doesn't make any HTTP connections so we can't use Burp Suite directly. So, we have another option to monitor the traffic by using a tool like Wireshark but it doesn't allow you to tamper with the traffic you can only monitor and understand the traffic. But if our goal is to intercept and modify the traffic, we will have to go with a tool called Echo Mirage.

So, without wasting much time let's begin the Traffic Analysis.

Table of Content

- Prerequisites
- What is Traffic Analysis
- Traffic Analysis Via Wireshark
- Traffic Analysis Via Echo Mirage
- Traffic Analysis Via Burp Suite + MITM Relay

Traffic Analysis

Any Thick client application communicating with the backend means they are sending some data to its backend components like web server, FTP Server, database server, etc. Analyzing the data during transfer is a very crucial part of the analysis of an application. Some applications perform data transit without enforcing any encryption. So, the concept of intercepting traffic of thick clients is not much different than thin clients, the tools will differ depending on the protocols used by the application also these applications are non-proxy aware as well the intercepting techniques also will slightly vary.

Prerequisites

- Wireshark
- Python 3
- Burp Suite
- Echo Mirage
- Python pip script
- MITM Relay

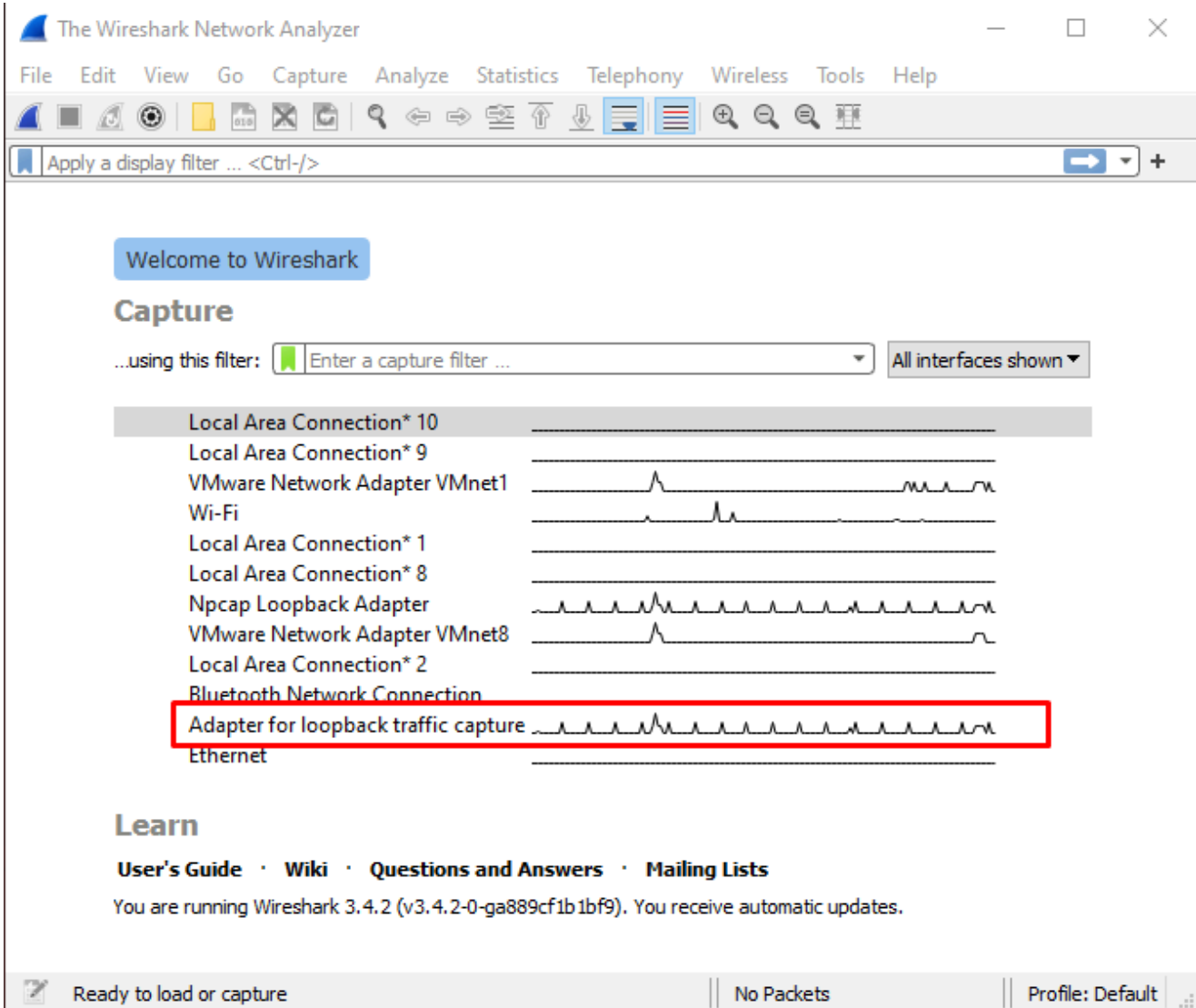
Traffic Analysis via Wireshark

As a penetration tester, you must have good knowledge of how to use a network packet sniffer is essential for day-to-day operations. Whether you are trying to understand a protocol, debug a network client or analyze traffic, you'll always end up needing a network sniffer.

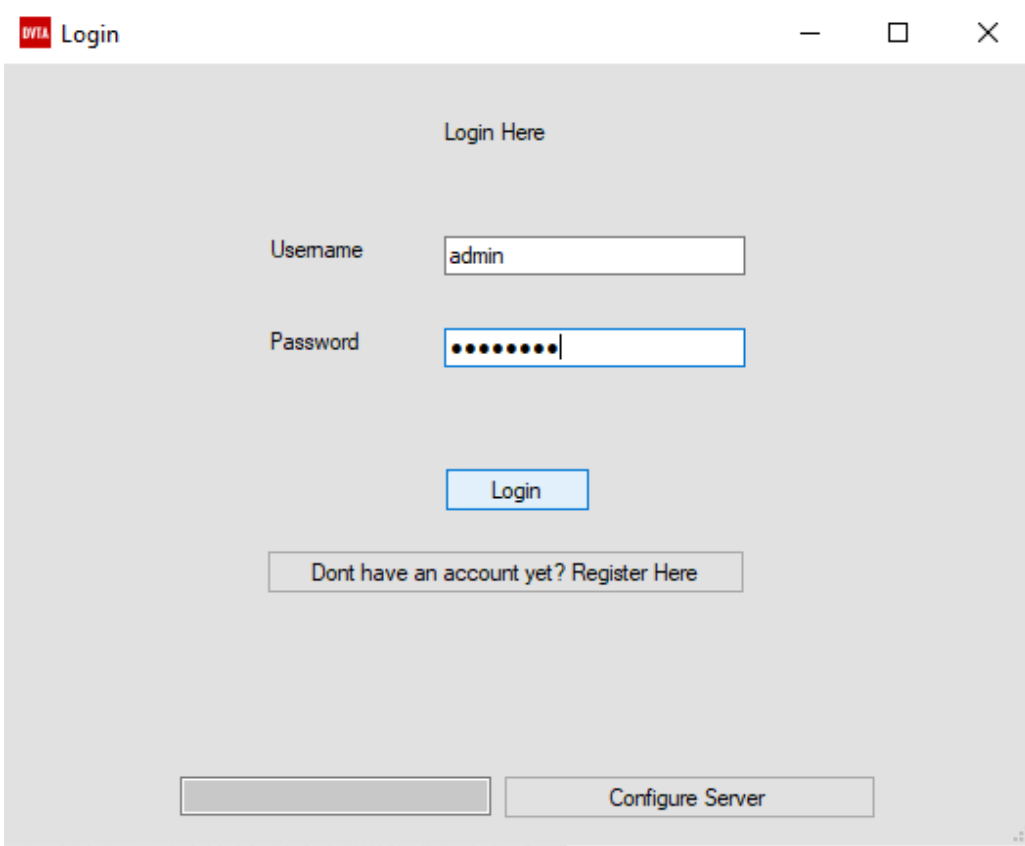
Examining the traffic between the tested thick client application and the server might reveal sensitive and unencrypted data such as:

- Most Sensitive data transferred over an unencrypted tunnel such as clear-text credentials/secrets/API Keys etc.
- HTTP and HTTPS web endpoints
- File blobs/chunks sent over the wire
- Proprietary protocols used by the program

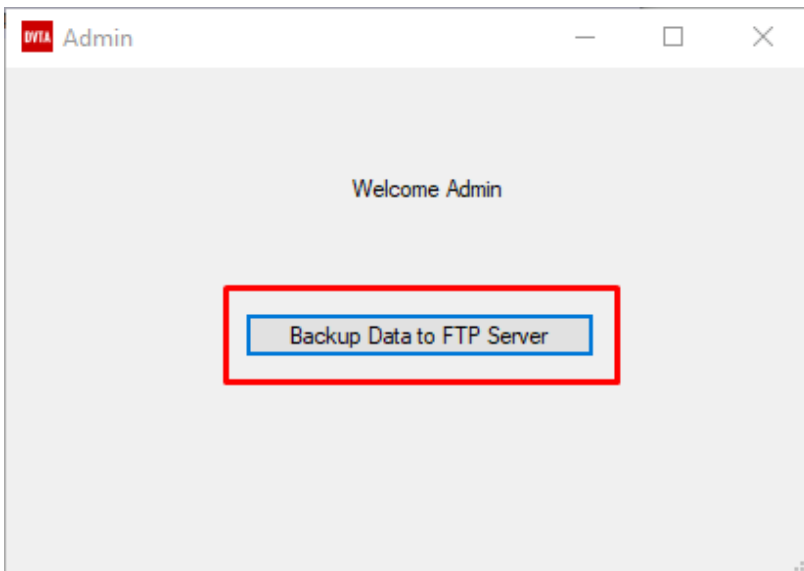
We are going to analyze FTP traffic that's generated by DVTA. For this, we are going to use the LoopBack address. So, first of all, Launch Wireshark and choose "Adapter for loopback Traffic Capture".



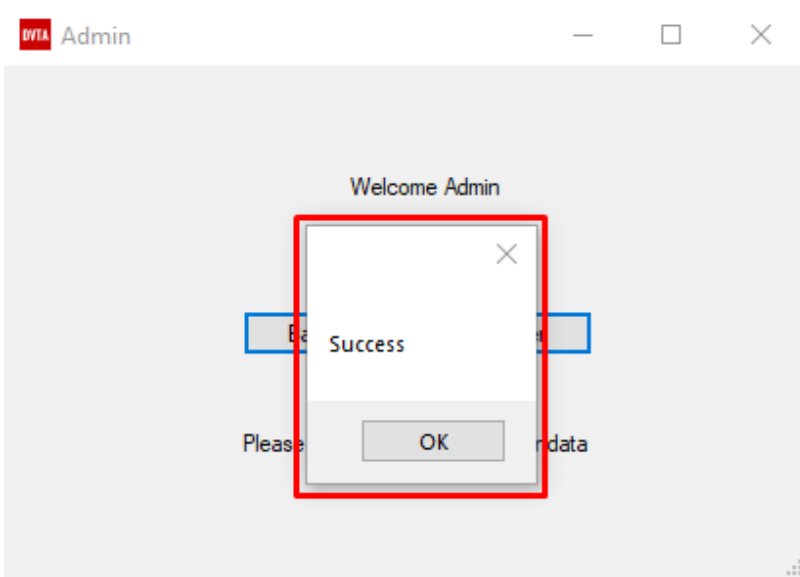
Then after launch modified DVTA application and login to the application by using admin credentials as shown below.



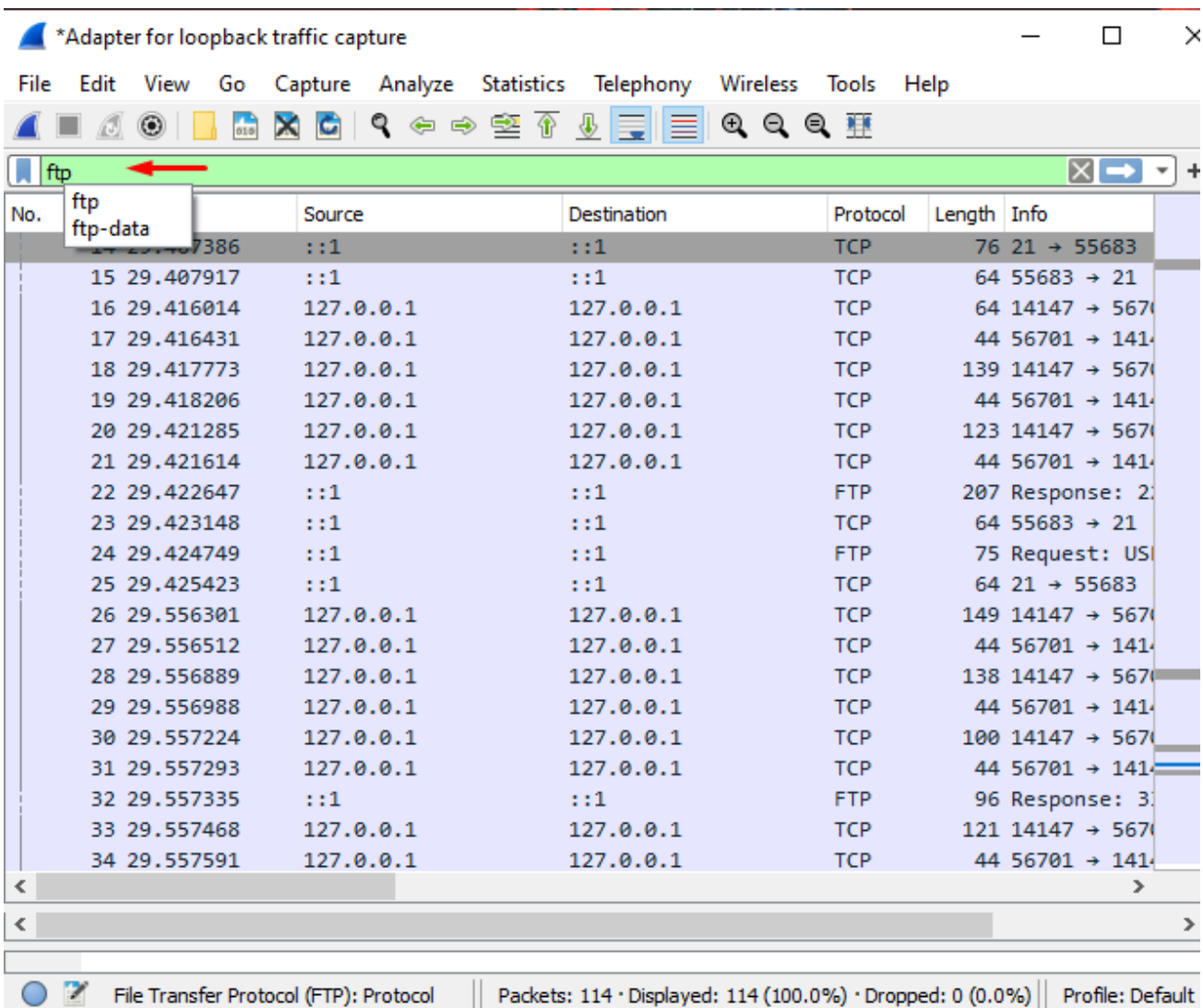
As you can see, we have successfully logged in and it showing Backup data to FTP server



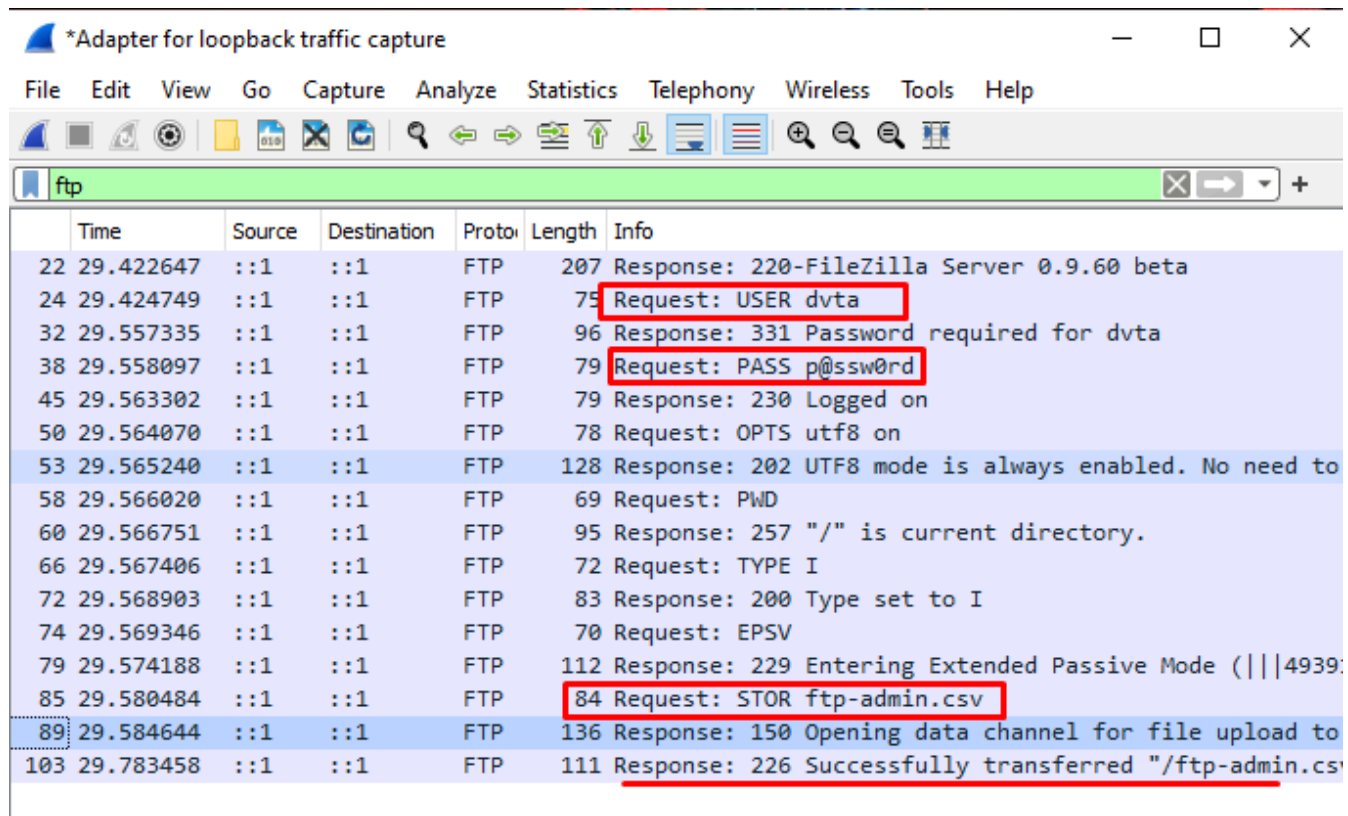
When you click backup data to FTP server it will make an FTP connection with the FTP server and it will try to upload some data to FTP server. After uploading the data, it will show you a Success message as shown below.



Just after Backup data to the FTP server come back to Wireshark and stop capturing the data so that it won't capture unnecessary data as shown in the below image. Apply the filter of FTP protocol so that only shows the traffic captured of DVTA application while uploading data to the FTP server.

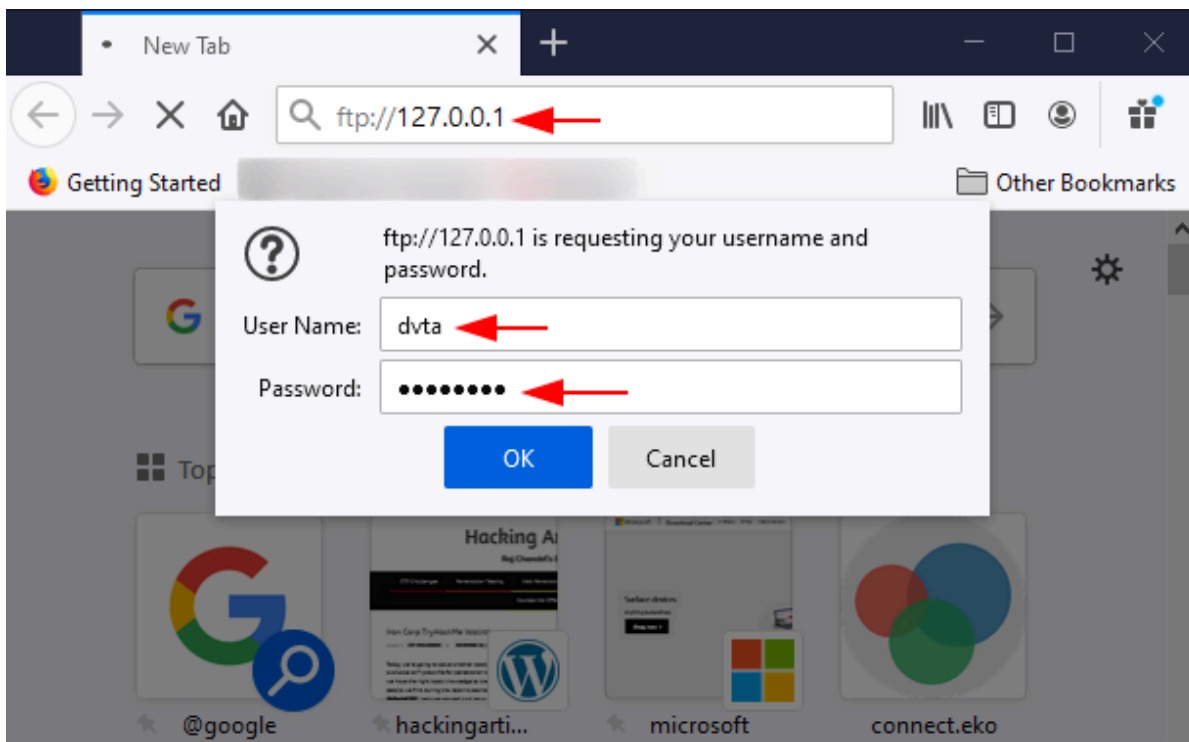


Now what we are interested in is the credentials that might be used by the DVTA application to login to the FTP server. We can check them by observing the captured FTP traffics. If you notice the first packet is the response from the FileZilla server and then we have a command **“USER and the USER name: DVTA”**. Then after in the third line, there is a response from the FTP server saying the password is Required for DVTA and in the next line, there is a **“Password: p@ssw0rd”** that was sent by the client.

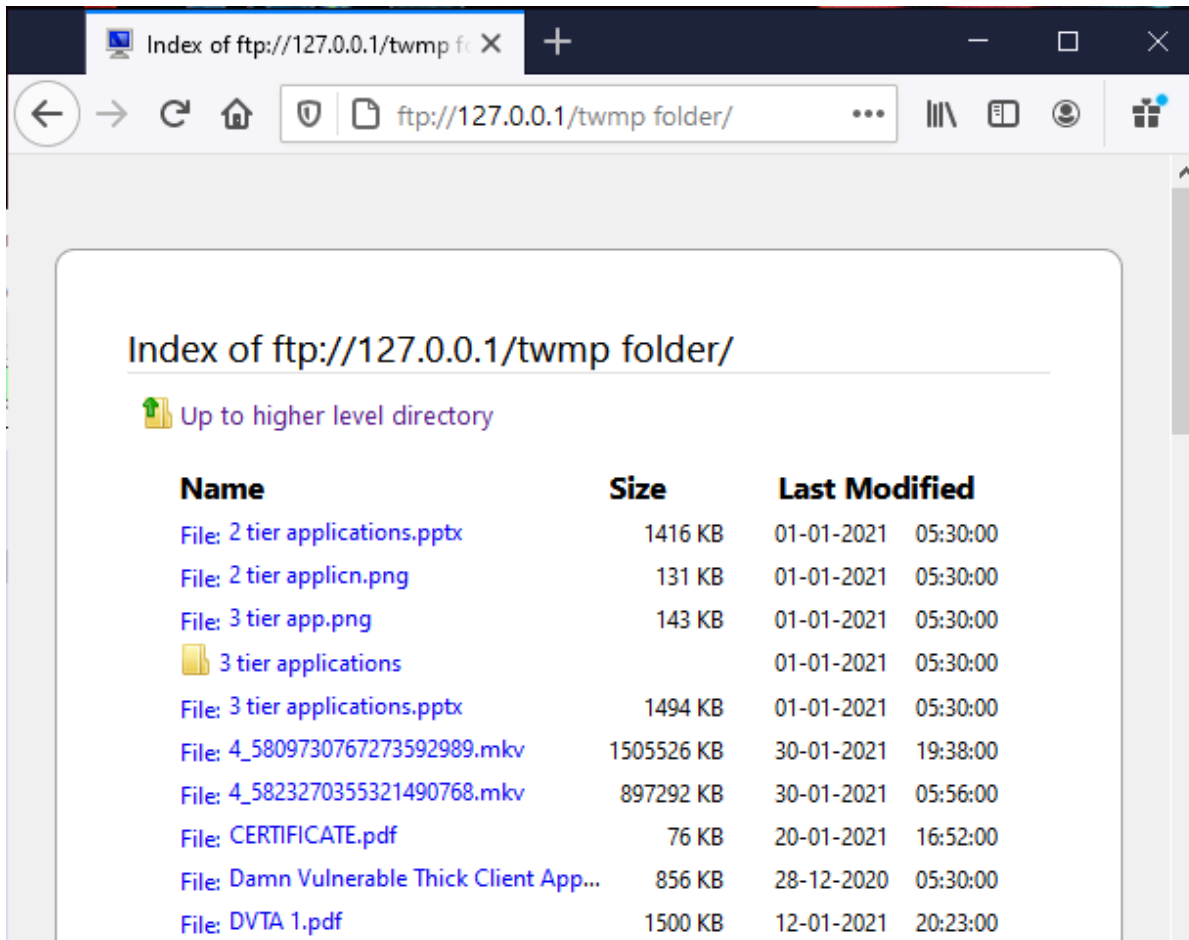


	Time	Source	Destination	Proto	Length	Info
22	29.422647	::1	::1	FTP	207	Response: 220-FileZilla Server 0.9.60 beta
24	29.424749	::1	::1	FTP	75	Request: USER dvta
32	29.557335	::1	::1	FTP	96	Response: 331 Password required for dvta
38	29.558097	::1	::1	FTP	79	Request: PASS p@ssw0rd
45	29.563302	::1	::1	FTP	79	Response: 230 Logged on
50	29.564070	::1	::1	FTP	78	Request: OPTS utf8 on
53	29.565240	::1	::1	FTP	128	Response: 202 UTF8 mode is always enabled. No need to
58	29.566020	::1	::1	FTP	69	Request: PWD
60	29.566751	::1	::1	FTP	95	Response: 257 "/" is current directory.
66	29.567406	::1	::1	FTP	72	Request: TYPE I
72	29.568903	::1	::1	FTP	83	Response: 200 Type set to I
74	29.569346	::1	::1	FTP	70	Request: EPSV
79	29.574188	::1	::1	FTP	112	Response: 229 Entering Extended Passive Mode (4939:
85	29.580484	::1	::1	FTP	84	Request: STOR ftp-admin.csv
89	29.584644	::1	::1	FTP	136	Response: 150 Opening data channel for file upload to
103	29.783458	::1	::1	FTP	111	Response: 226 Successfully transferred "/ftp-admin.csv"

Now it's pretty clear that the user name and password are available for us. So, let's quickly open a browser and surf to **ftp://127.0.0.1** and try to use the credentials here that we have captured as shown below.



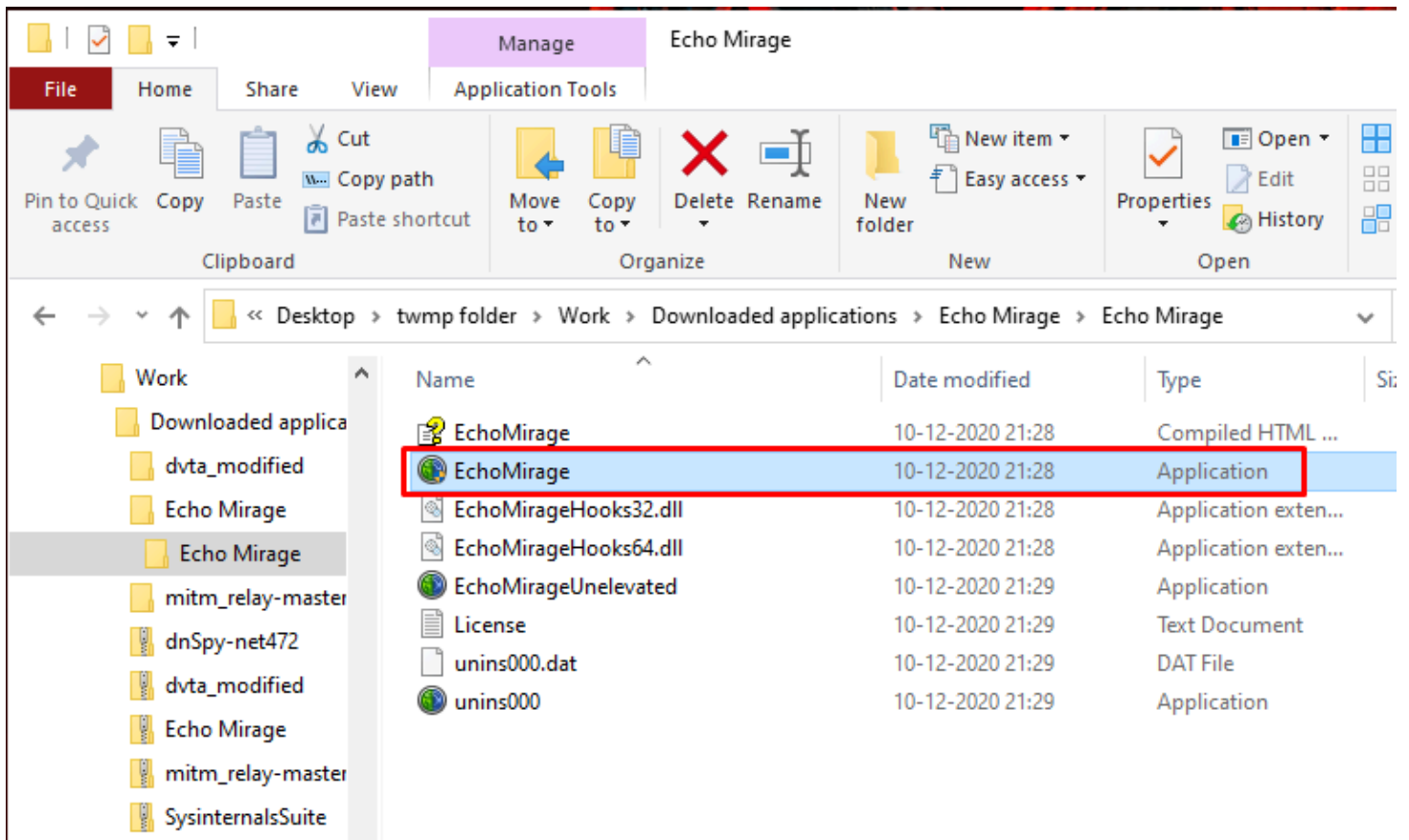
And here you go..., as you can see, we have successfully connected to the FTP server and we can download everything that is available in that folder as shown below. This is how an attacker grab credentials that are being used by the application in the client site. They can probably do some Wireshark capture and if the credentials are found in clear text, they can easily login to the FTP server and can tamper or download the data easily.



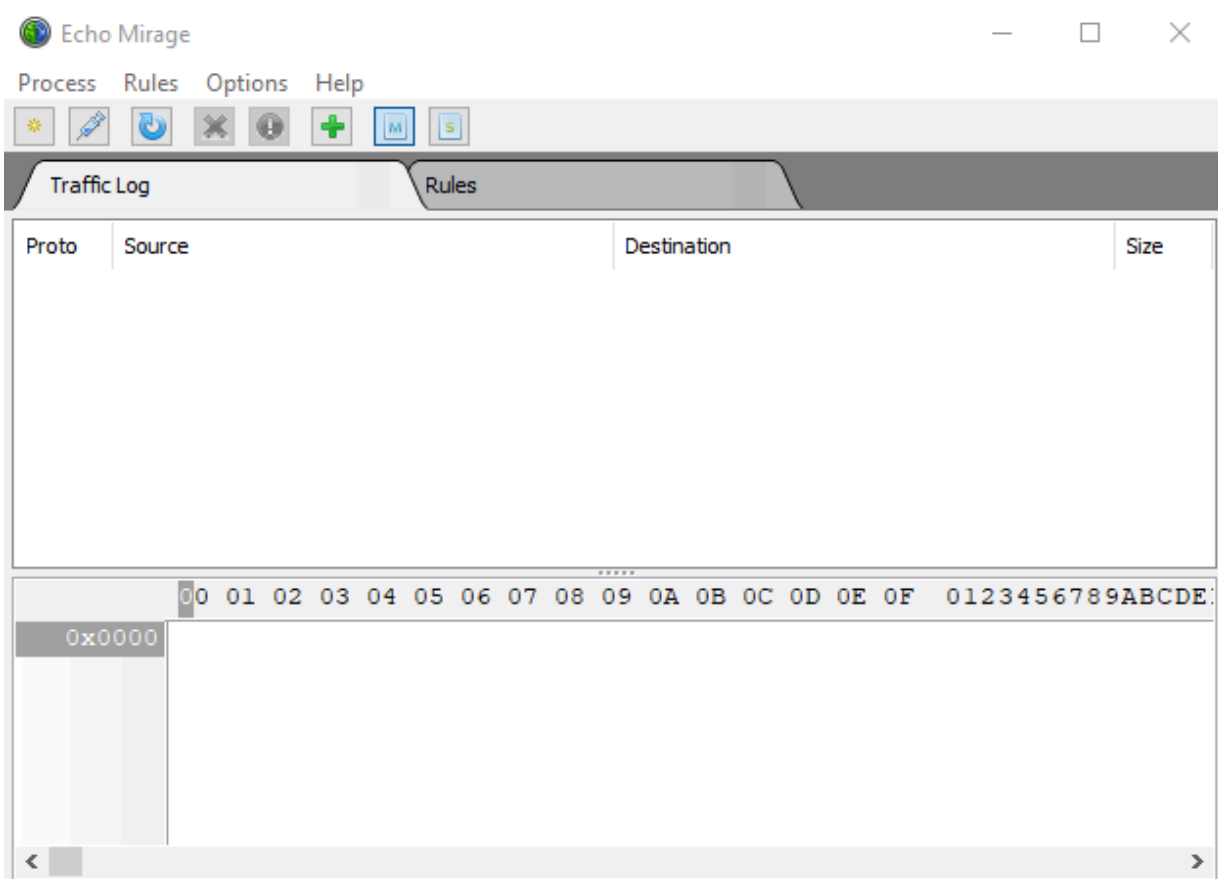
Traffic Analysis via Echo mirage

Download and Extract Echo Mirage into your desirable folder. You can download Echo Mirage from here: <https://mega.nz/file/bHhQ3QrA#14d-lc3bL1tm8AOUcMzlkbn6SaMYFPXNjRr1caD0m1E>

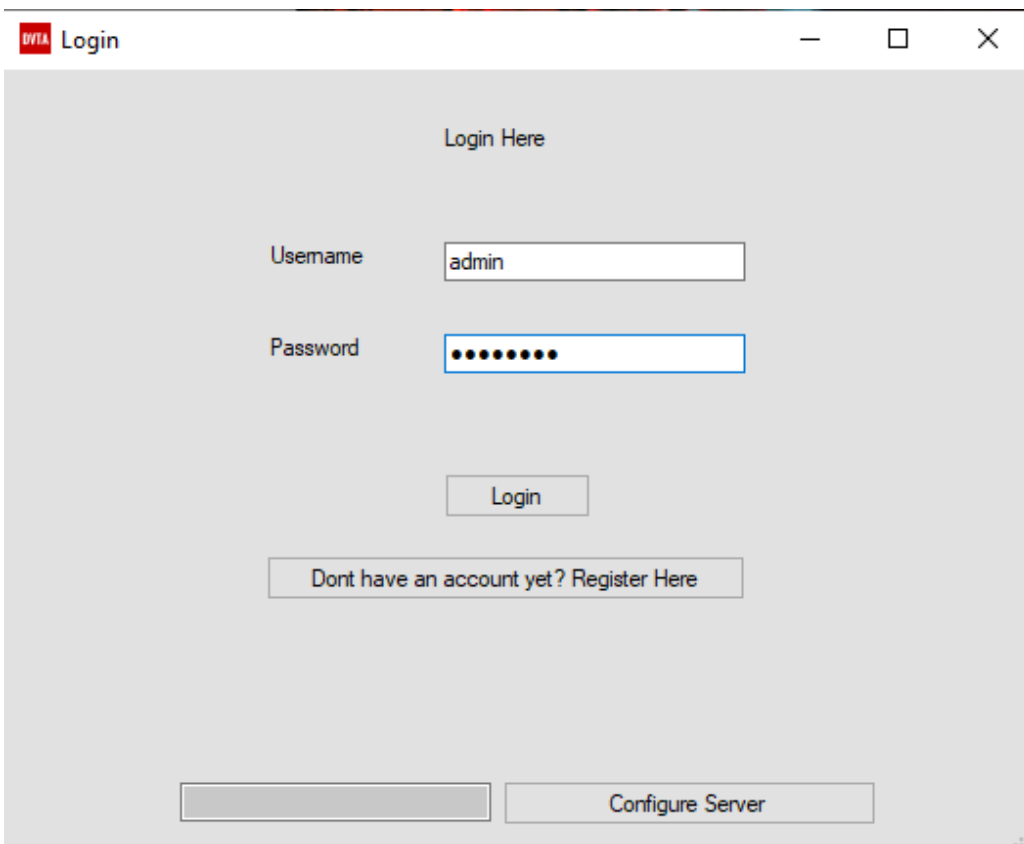
Let's open up the EchoMirage application.



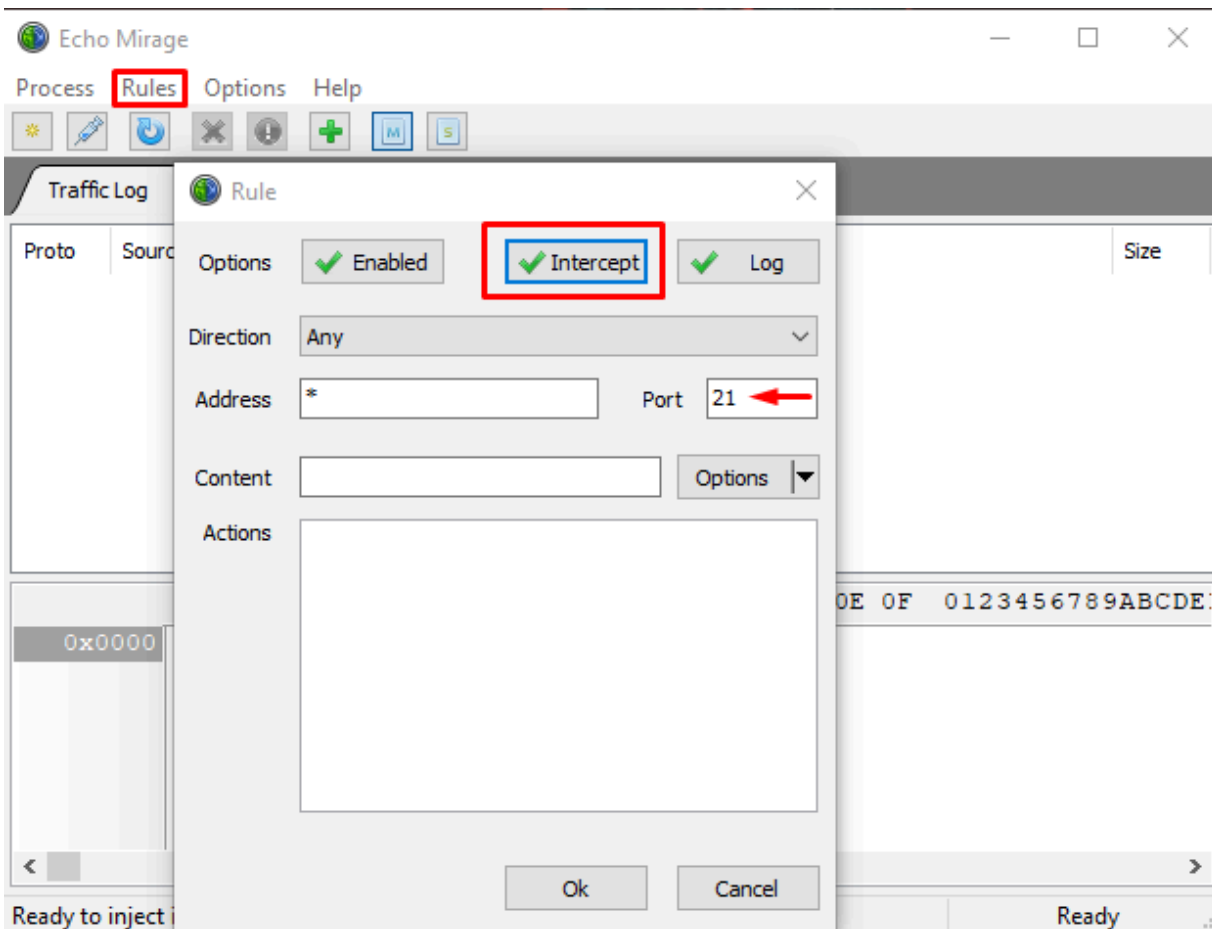
and it will open EchoMirage for us



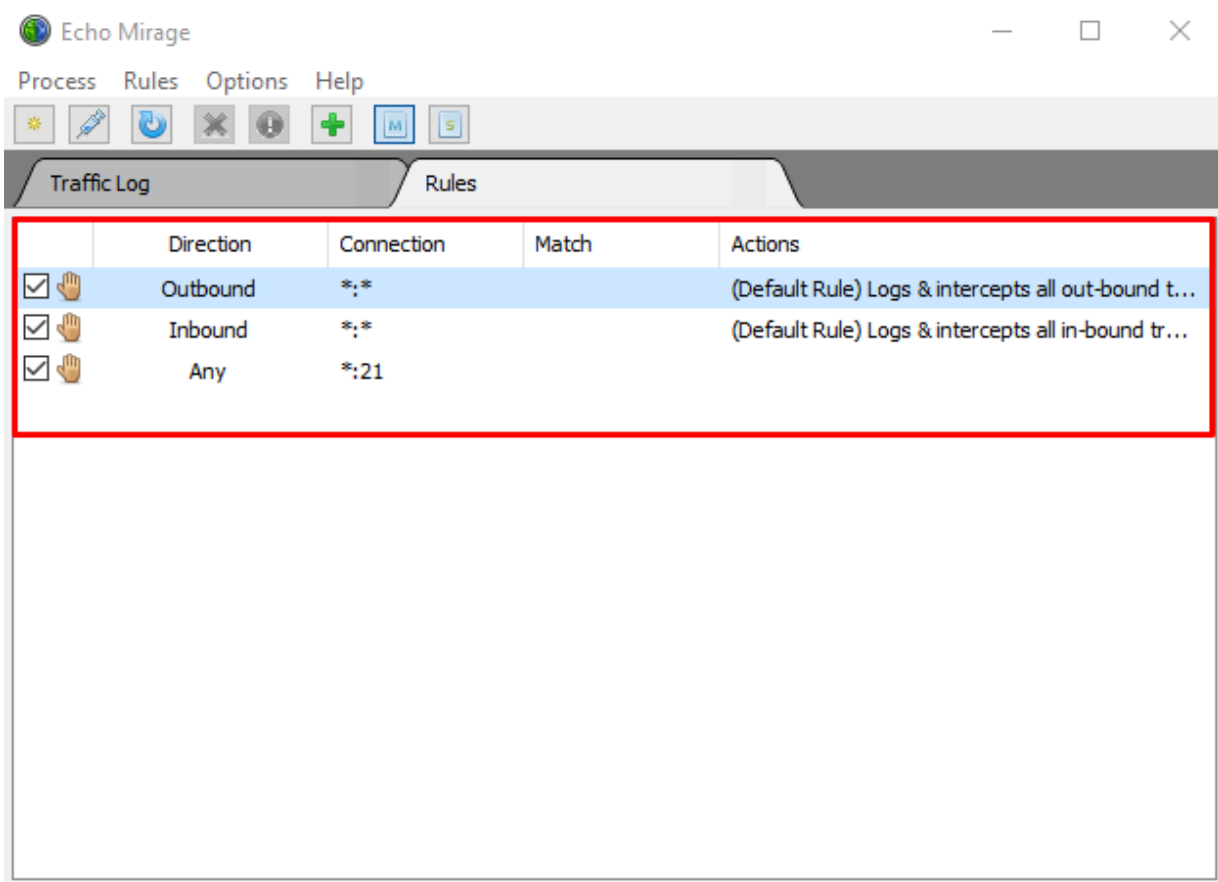
But also remember one thing don't forget to delete the FTP CSV file that you have uploaded earlier by using DVTA application so that we can upload a new copy again. Now again open the DVTA application and login into the Application by using admin credentials.



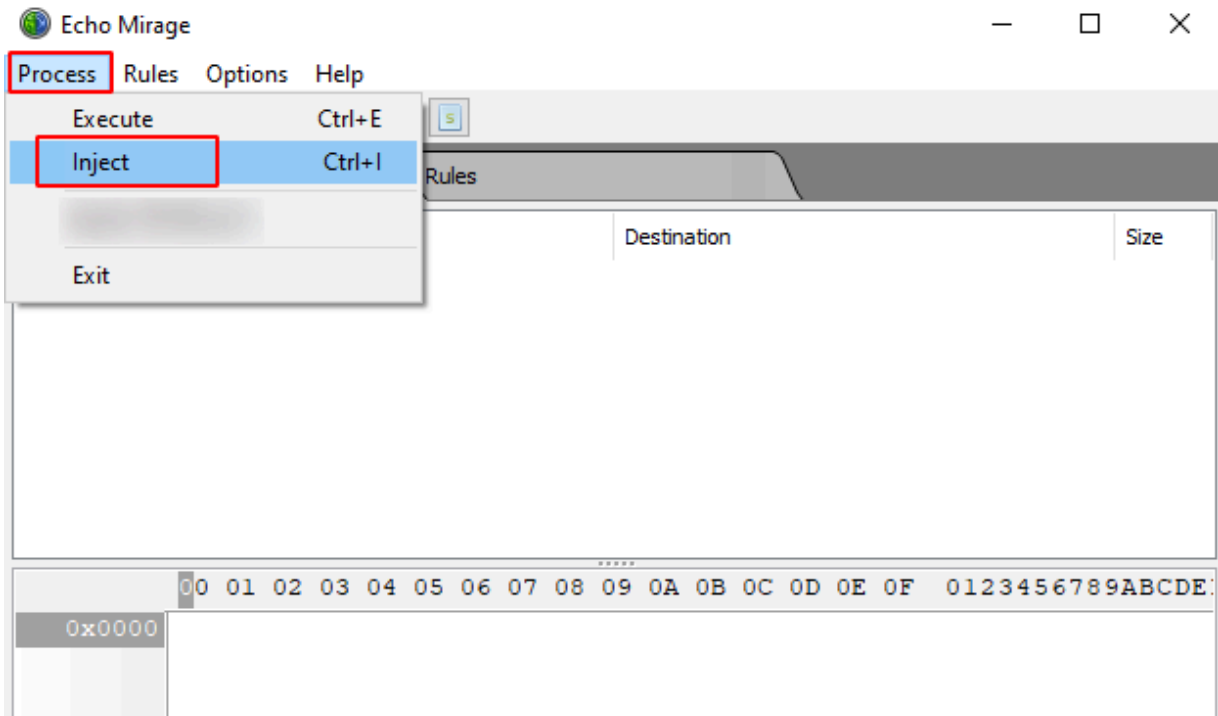
So, we have successfully logged into DVTA application and when we click on ***“Backup data to FTP server”*** then we want to be able to see what’s going on using EchoMirage. To do this create a new Rule by going to ***“Rule > New Rules”*** choose the direction to “Any” as we want to capture traffic only from the port of 21 so ***“set the Port to Port no. 21”*** then after click on ***“intercept”*** and then save the configuration by hitting ***“ok”*** as shown below



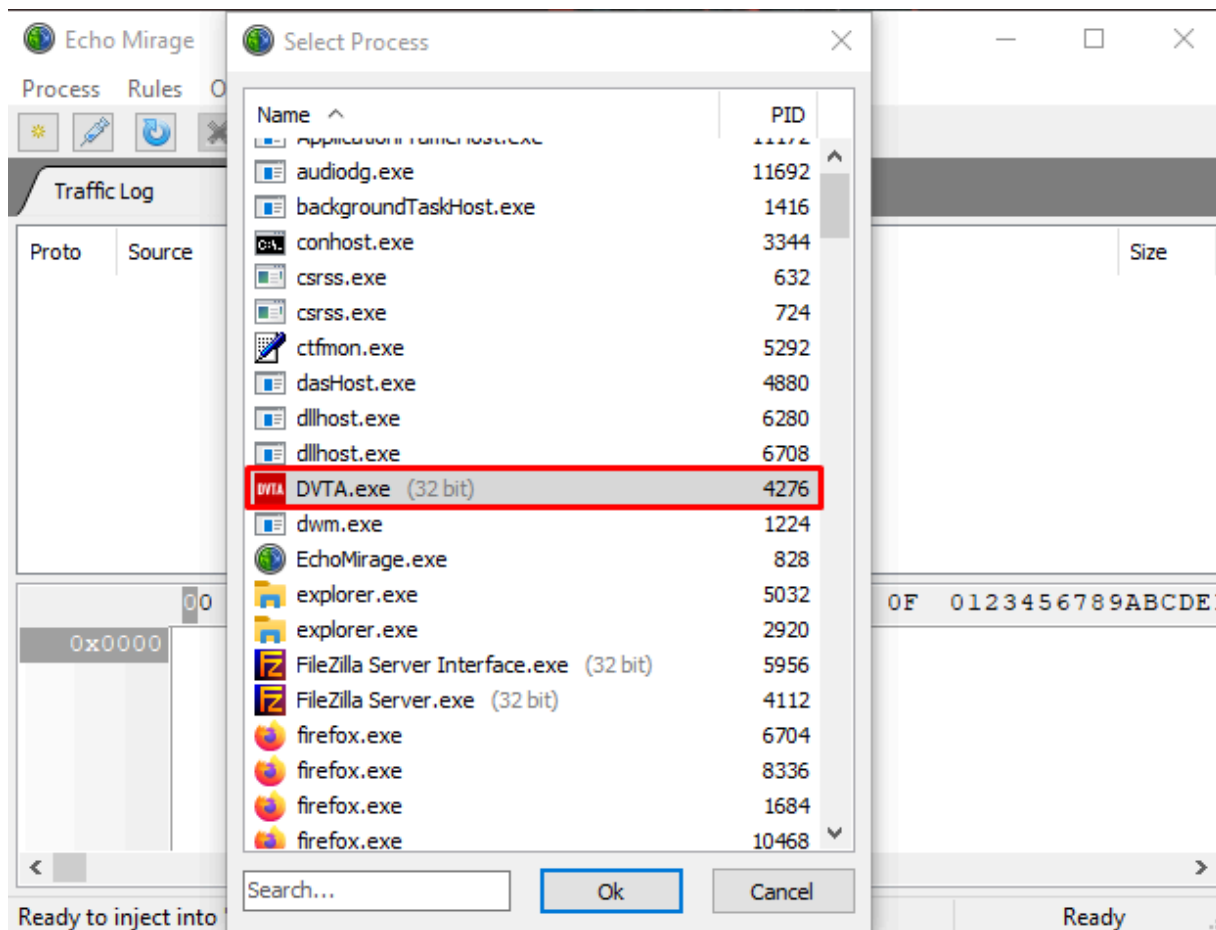
Now here we can see the Rules created by us, as we can see we are going to intercept all the traffic that’s going to outbound and inbound also we are going to intercept the traffic of port 21 only.



Now, navigate to Process and click on inject



and we are going to see all the process that is running on your machine. But we are interested to see the traffic only from DVTA.exe so select it and inject it into a process



Navigate to the tab of Traffic.Log then go to the DVTA application and click on Backup Data to FTP Server. Come back to Echo Mirage and here If you see it started capturing traffic from DVTA.exe.

But we're not interested in this so just hit OK and forward the traffic.

Echo Mirage

Process Rules Options Help

Traffic Log Rules Intercept

Outbound ??? data to 0.0.0.0:

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x0000	10	01	01	4D	00	00	01	00	45	01	00	00	04	00	00	74	.	.	M	E	
0x0010	40	1F	00	00	00	00	00	06	B4	10	00	00	00	00	00	00	@		
0x0020	E0	03	00	10	00	00	00	00	00	00	00	00	5E	00	0F	00	à	^		
0x0030	7C	00	02	00	80	00	08	00	90	00	1C	00	C8	00	14	00		E			
0x0040	F0	00	04	00	F4	00	1C	00	2C	01	00	00	2C	01	04	00	ø	.	.	ô		
0x0050	D1	BA	1D	83	C6	B6	34	01	00	00	34	01	00	00	34	01	Ñ	°	.	E	4			
0x0060	00	00	00	00	00	00	44	00	45	00	53	00	4B	00	54	00	D	.	E	.	S	.	K		
0x0070	4F	00	50	00	2D	00	4B	00	4D	00	38	00	32	00	35	00	O	.	P	K	.	M	.	8	.	2		
0x0080	32	00	44	00	73	00	61	00	A2	A5	A1	A5	92	A5	92	A5	2	.	D	.	s	.	a	.	c	¥	;	¥	?			
0x0090	D2	A5	A6	A5	82	A5	E3	A5	2E	00	4E	00	65	00	74	00	Ò	¥	;	¥	¥	ã	¥	.	N	.	e	.	.			
0x00A0	20	00	53	00	71	00	6C	00	43	00	6C	00	69	00	65	00	.	S	.	q	.	l	.	C	.	l	.	i	.			

0x00B0 6E 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x00C0 50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x00D0 6C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Disable rule

Injected into "DVTA.e

Admin

Welcome Admin

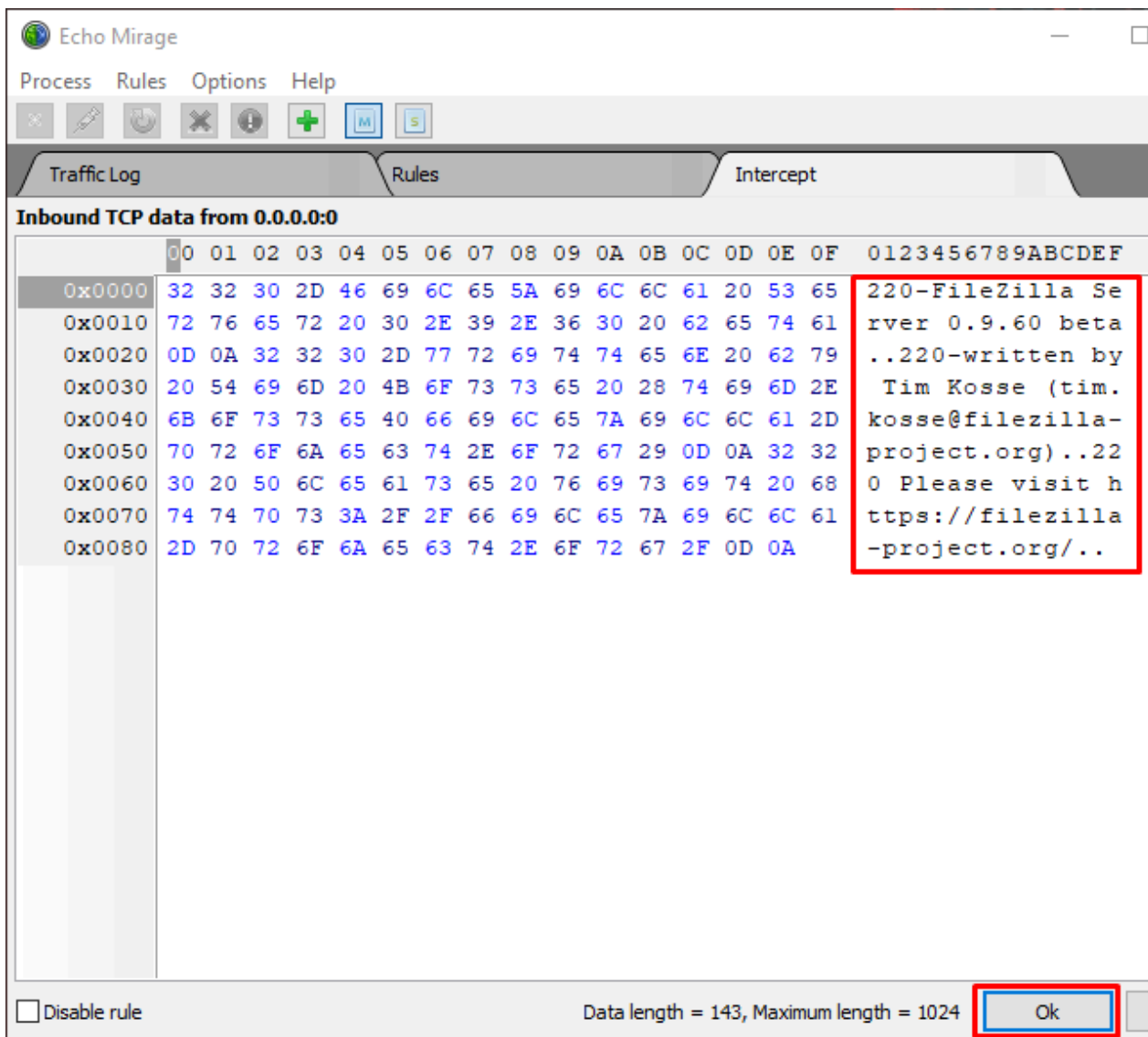
Backup Data to FTP Server

Cancel

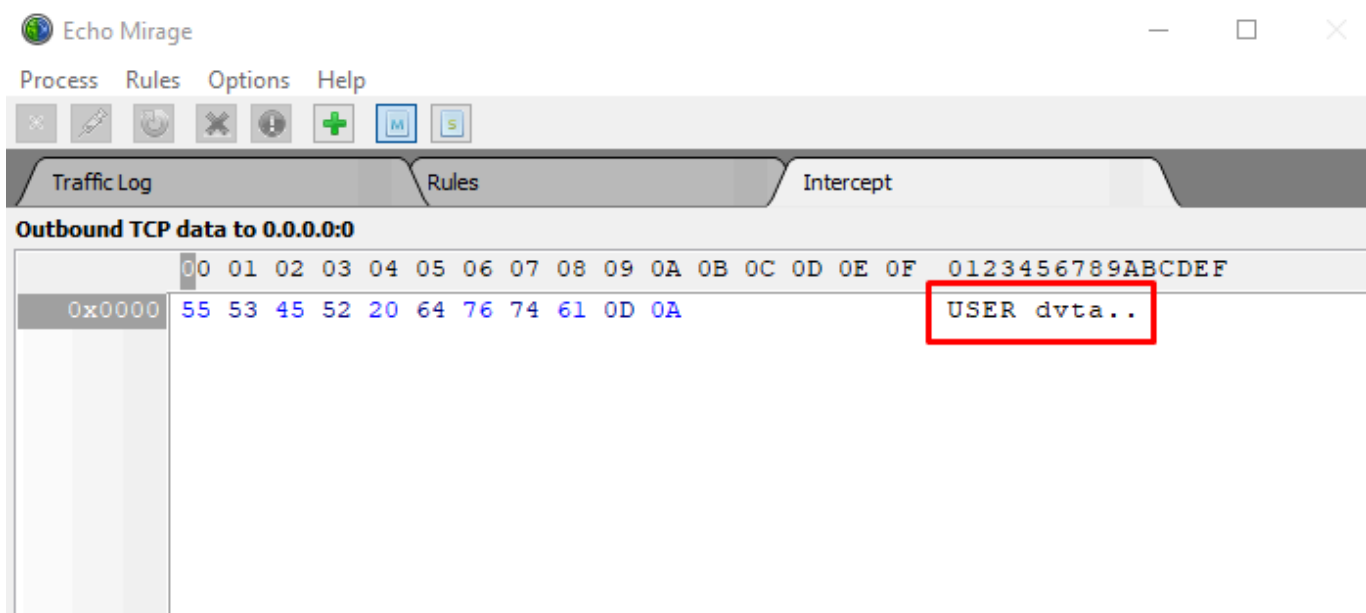
Intercept

Please wait while unloading your data

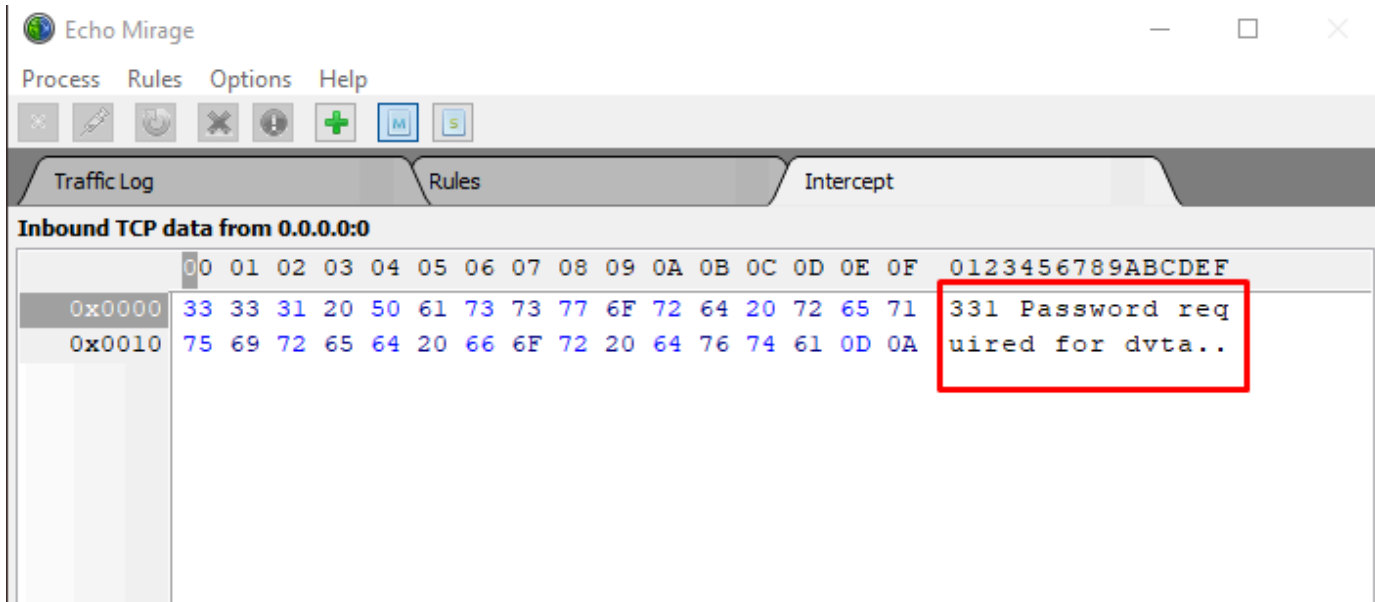
If you see there is a response coming from files FileZilla server and click ok once again..



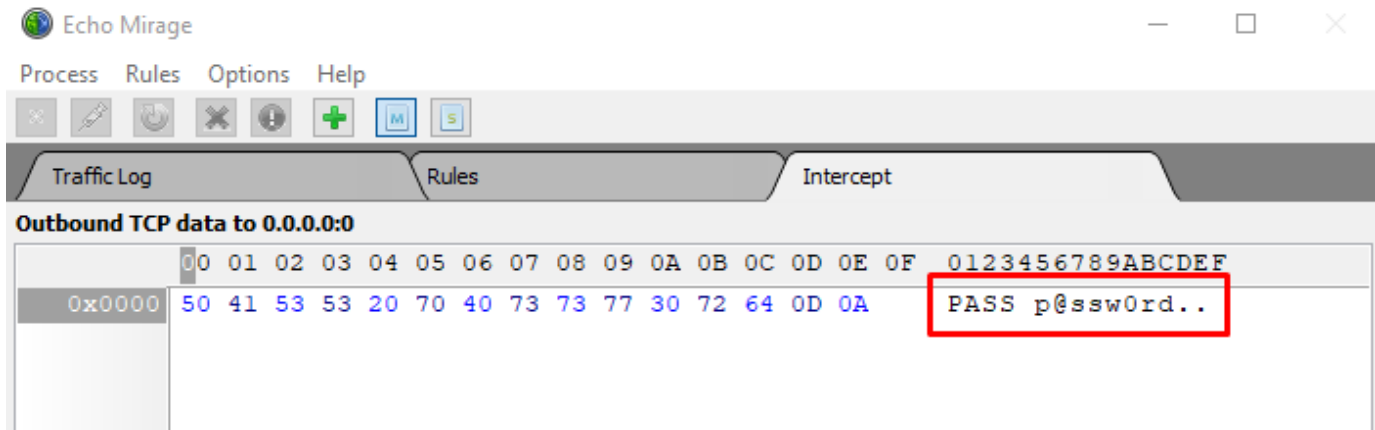
And if you see here the user name is being sent to the FTP server



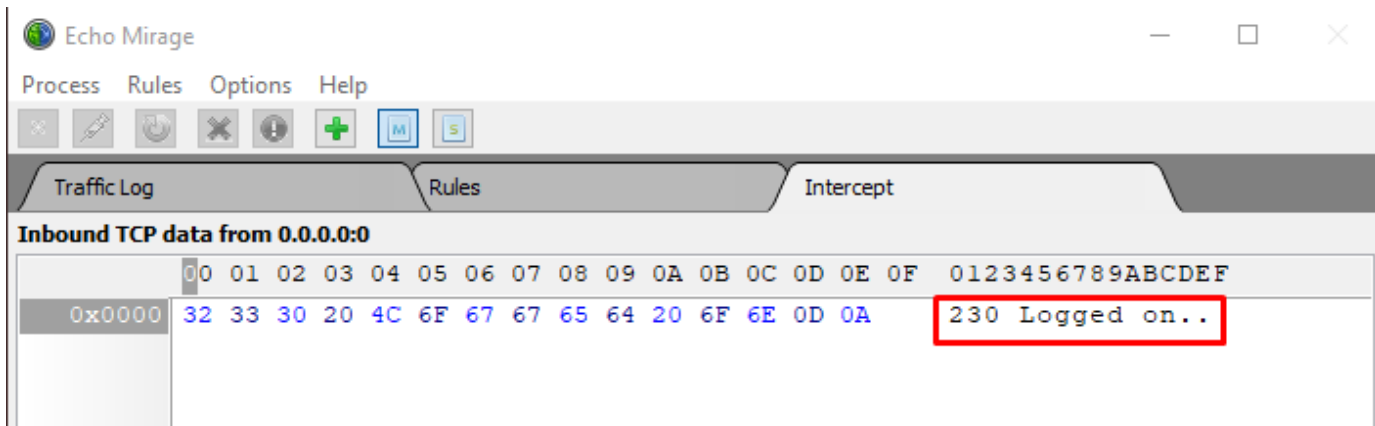
And the response from FTP server says that password required for DVTA. Hit ok and forward the Traffic.



As you can see the password can be seen in clear text *"p@ssw0rd"*. The DVTA client is sending its password to the FTP server



And as you can see, you're logged in now

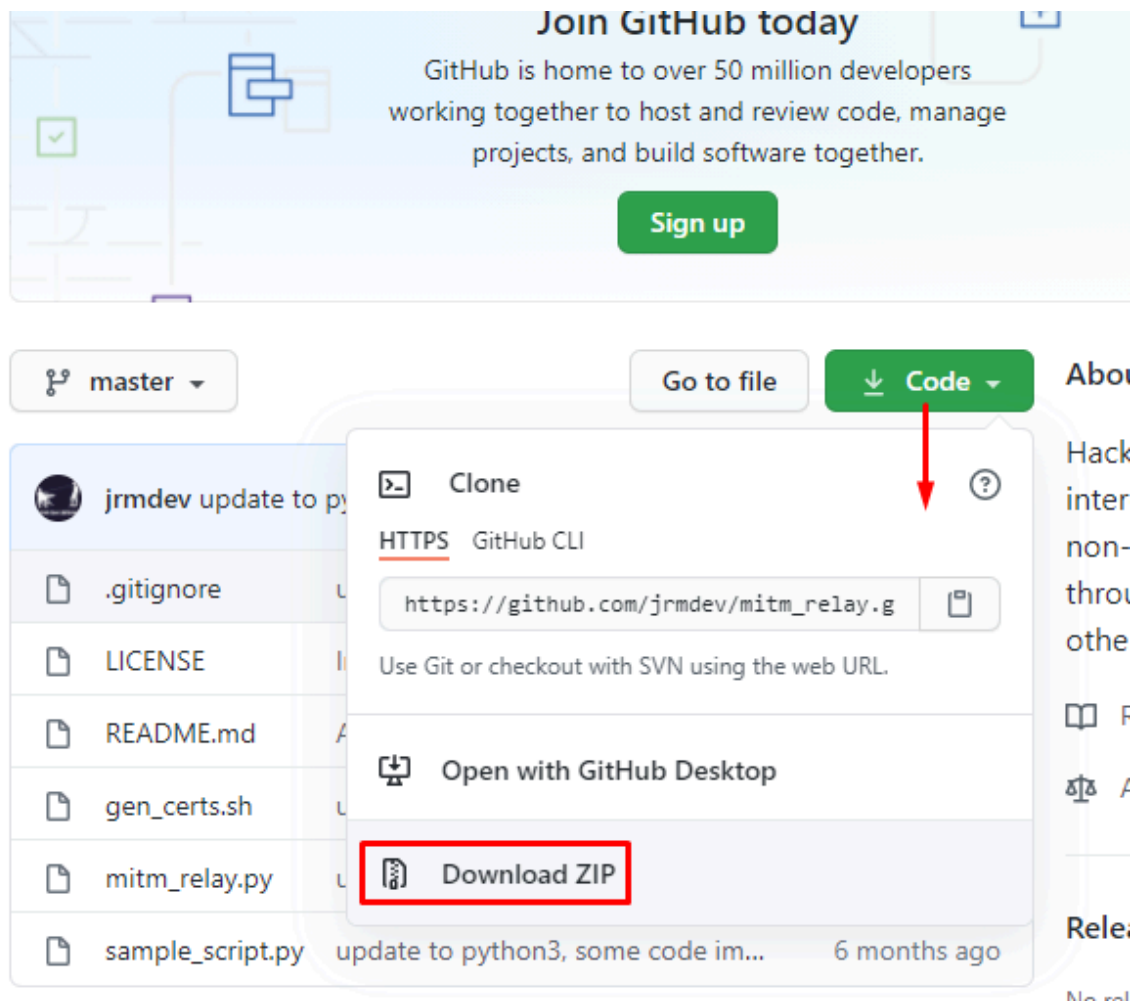


If you are still observing the traffic then you should see some file being uploaded onto their FTP server just hit ok and forward the traffic.

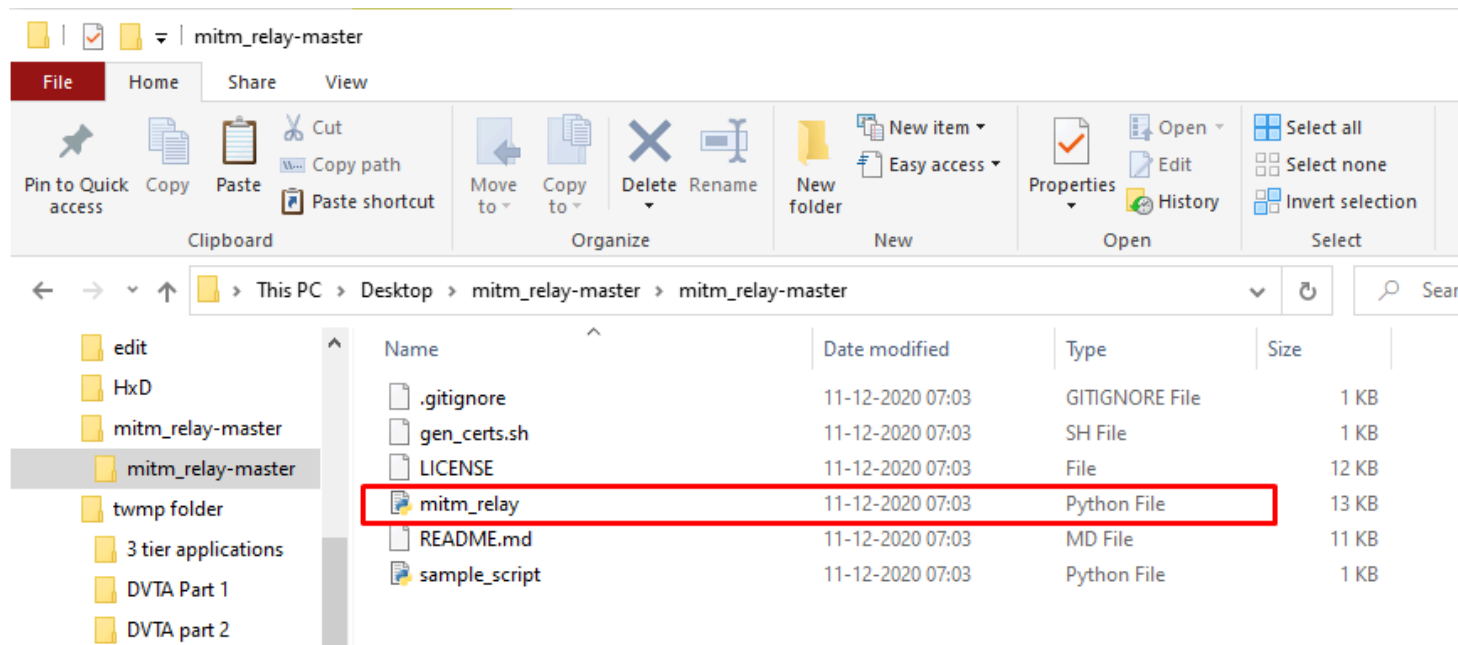
Traffic Analysis via Burp Suite + MITM Relay

First of all, you need to download MITM Relay from GitHub or also you can directly download it from here: –

https://github.com/jrmdev/mitm_relay

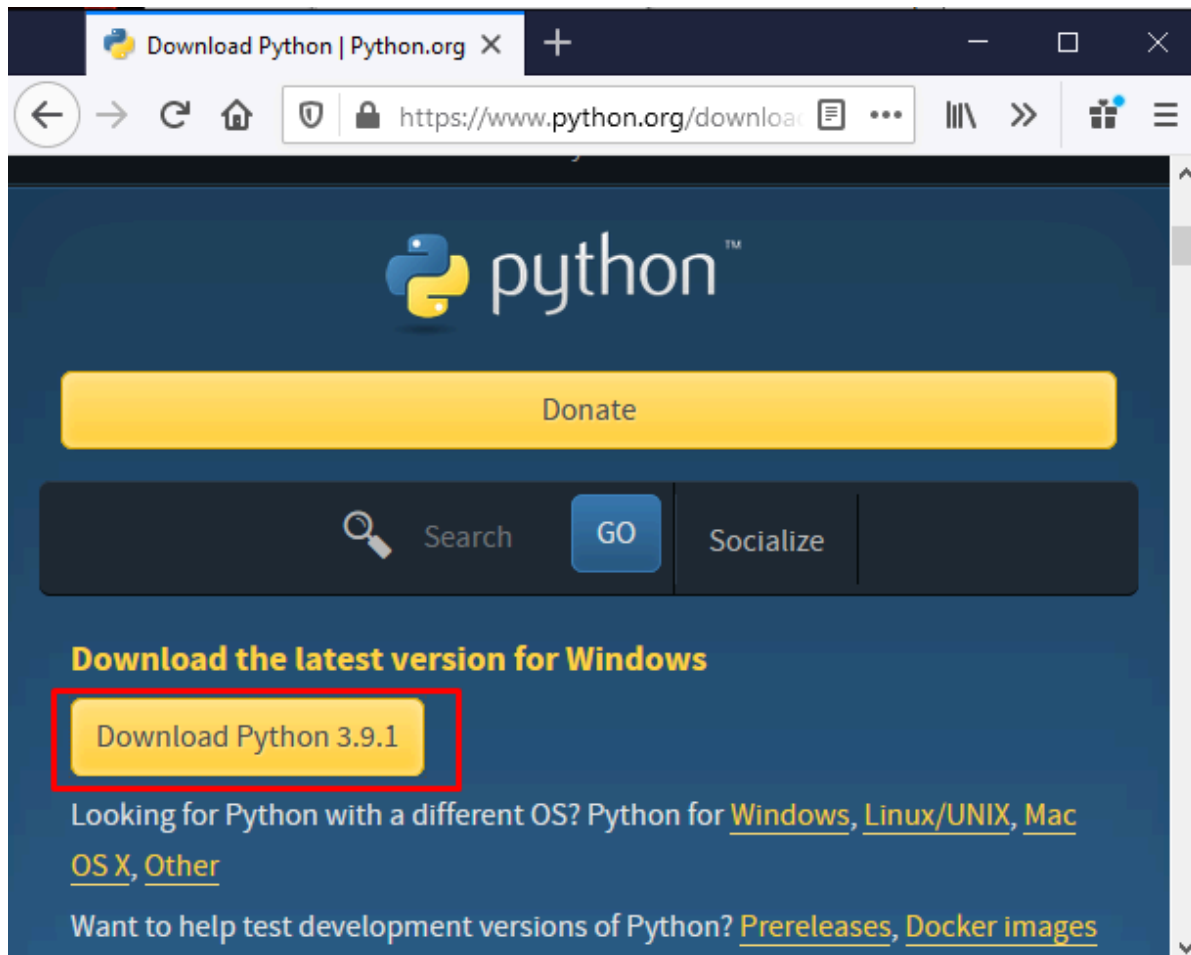


Download and Extract in your work folder



MITM Relay is a python script and it works fine with Python 3 so download and install it by going to the official website of python or you can also download it from here: –

<https://www.python.org/ftp/python/3.9.2/python-3.9.2-amd64.exe>



In manner to download repositories from python, “*get-pip*” needs to be available in your system you can get it from here: – <https://bootstrap.pypa.io/get-pip.py>

All you need to do is save the source code by giving it a **right-click** and **select** the option “**save as**” and save it into your working folder by the name of get-pip as shown below

```
#!/usr/bin/env python
#
# Hi There!
# You may be wondering what this giant blob of binary data here is, you might
# even be worried that we're up to something nefarious (good for you for being
# paranoid!). This is a base85 encoding of a zip file, this zip file contains
# an entire copy of pip (version 20.2.4).
#
# Pip is a thing that installs packages, pip itself is a package that someone
# might want to install, especially if they're looking to run this get-pip.py
# script. Pip has a lot of code to deal with the security of installing
# packages, various edge cases on various platforms, and other such sort of
# "tribal knowledge" that has been encoded in its code base. Because of this
# we basically include an entire copy of pip inside this blob. We do this
# because the alternatives are attempt to implement a "minipip" that probably
# doesn't do things correctly and has weird edge cases, or compress pip itself
# down into a single file.
#
# If you're wondering how this is created, it is using an invoke task located
# in tasks/generate.py called "installer". It can be invoked by using
# ``invoke generate.installer``.

import os.path
import pkgutil
import shutil
import sys
import struct
import tempfile

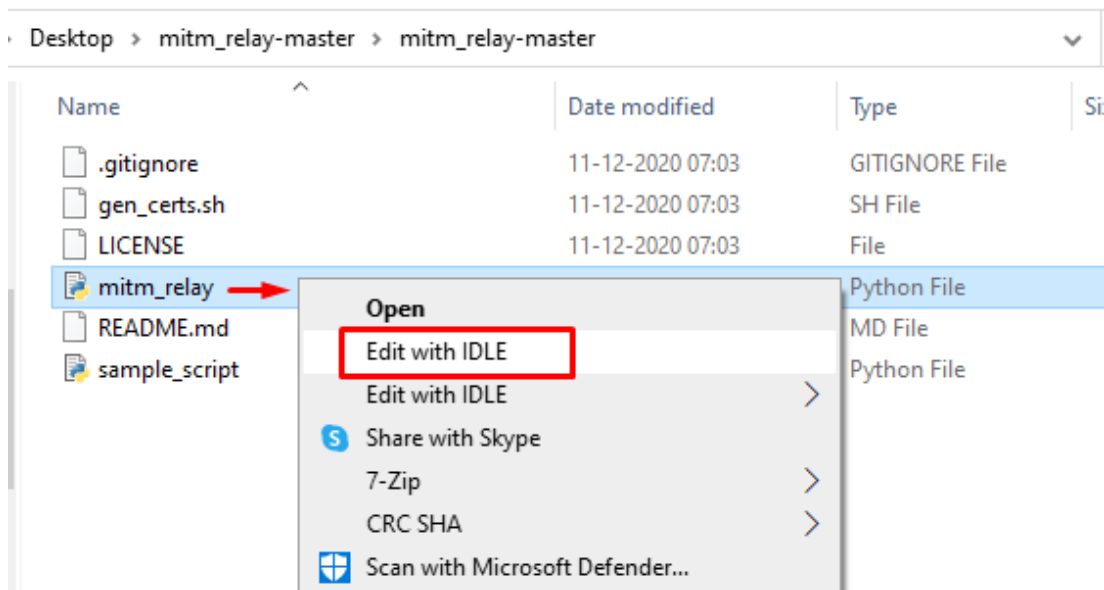
# Useful for very coarse version
PY2 = sys.version_info[0] == 2
PY3 = sys.version_info[0] == 3
```



Let's quickly open up the Power Shell and navigate to the directory of python3. It can easily be founded in c drive and then run get-pip.py script from here as shown below

```
Windows PowerShell
PS C:\> cd .\Python3\
PS C:\Python3> .\python3.exe "C:\Users\vijvi\OneDrive\Desktop\get-pip.py"
C:\Python3\lib\site-packages\setuptools\distutils_patch.py:25: UserWarning: Distutils was imported before Setuptools. This usage is discouraged and may exhibit undesirable behaviors or errors. Please use Setuptools' objects directly or at least import Setuptools first.
  warnings.warn(
Collecting pip
  Using cached pip-21.0.1-py3-none-any.whl (1.5 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 21.0.1
    Uninstalling pip-21.0.1:
      Successfully uninstalled pip-21.0.1
  Successfully installed pip-21.0.1
PS C:\Python3>
```

As you can see pip is successfully downloaded in the above Image now let's install the Requests module because if you look at MITM Relay it makes use of a module called Requests. To check this all you need to do is go to the directory of MITM Relay and open MITM relay with the option of ***"Edit with IDLE"***



As you can see here MITM Relay makes use of a module named Requests

A screenshot of a Python IDE window titled 'mitm_relay.py - C:\Users\vijvi\OneDrive\Desktop\mitm_relay-master\mitm_relay-master\mit...'. The menu bar includes 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code is as follows:

```
#!/usr/bin/env python3

import sys
import socket
import ssl
import os
import requests
import argparse
import time
import string

from http.server import HTTPServer, BaseHTTPRequestHandler
from threading import Thread
from select import select

BIND_WEBSERVER = ('127.0.0.1', 49999)
BUFSIZE = 4096

__prog_name__ = 'mitm_relay'
__version__ = 1.0
```

The 'import requests' line is highlighted with a red rectangle.

And by default, the Requests module is not installed. This can be installed by running this command

```
python3 -m pip install requests
```

And it will install the requests module for us.

```
Windows PowerShell
PS C:\Python3> python3 -m pip install requests
Requirement already satisfied: requests in c:\python3\lib\site-packages (2.25.1)
Requirement already satisfied: chardet<5,>=3.0.2 in c:\python3\lib\site-packages (from requests) (4.0.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\python3\lib\site-packages (from requests) (1.26.3)
Requirement already satisfied: idna<3,>=2.5 in c:\python3\lib\site-packages (from requests) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in c:\python3\lib\site-packages (from requests) (2020.12.5)
PS C:\Python3>
```

Let's install Burp Suite

You can download the Burp Suite community edition from here: –

<https://portswigger.net/burp/releases/professional-community-2020-11-3>

Burp Suite Releases

Professional / Community 2020.11.3

01 December 2020 at 16:30 UTC

Burp Suite Community Edition Windows (64-bit)

Download show checksums

This release provides several bug fixes. Most notably:

- We have fixed a bug that occasionally caused issues with the new UI, such as Burp appearing to lock up.
- When you forward an intercepted request without making any changes, it is no longer erroneously marked as "Edited" in the proxy history.
- The "Getting Started" links on the Proxy Intercept tab are now only displayed until you intercept your first request.

The installation is straight forward you don't need to change anything during the installation process. After downloading the Burp Suite open it

ⓘ Welcome to Burp Suite Community Edition. Use the options below to create or open a project.



Note: Disk-based projects are only supported on Burp Suite Professional.

☒ **Temporary project**

☐ **New project on disk**

Name:

File:

☐ **Open existing project**

Name	File
<input type="text"/>	<input type="text"/>

File:

☒ Pause Automated Tasks

Start the Burp Suite and go to the proxy and select options and if you see it is listening on the *“port 8080”*

Burp Project Intruder Repeater Window Help

Sequencer	Decoder	Comparer	Extender	Project options	User options
Dashboard	Target	Proxy	Intruder	Repeater	
Intercept	HTTP history	WebSockets history	Options		

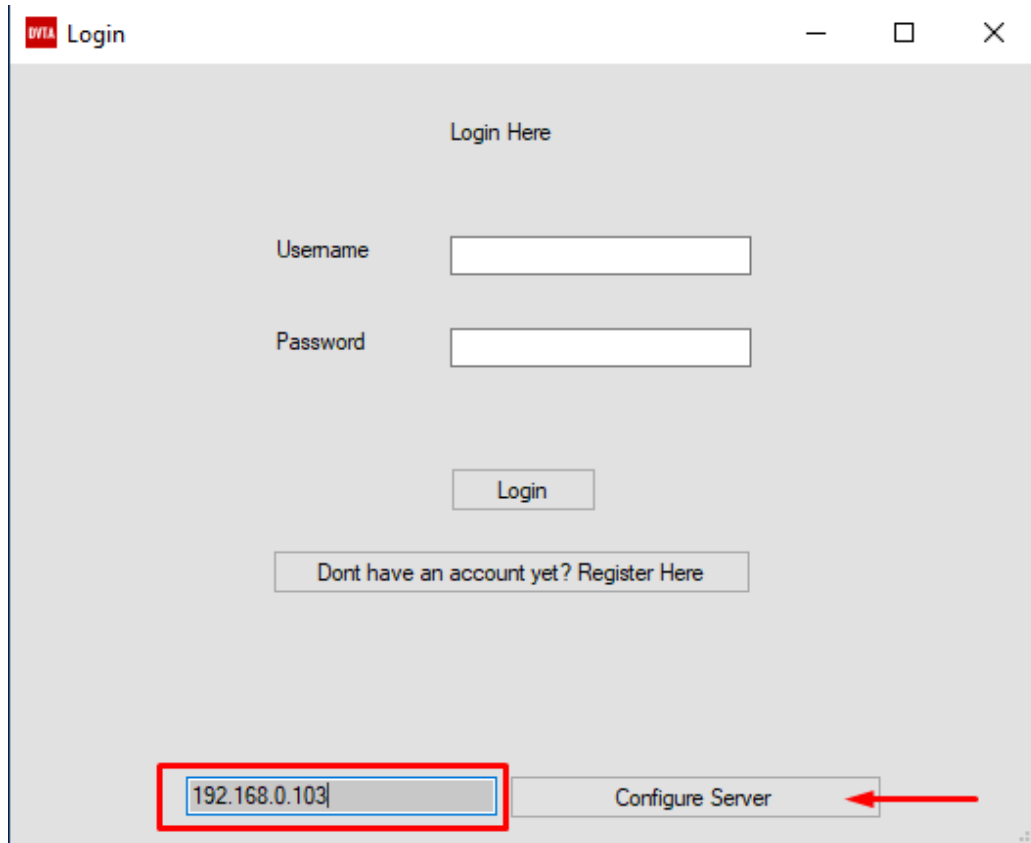
ⓘ **Proxy Listeners**

⚙ Burp Proxy uses listeners to receive incoming HTTP requests from your browser. You will need to configure your browser to use one

<input type="button" value="Add"/>	Running	Interface	Invisible	Redirect	Certificate	TLS Protocols
<input type="button" value="Edit"/>	<input checked="" type="checkbox"/>	127.0.0.1:8080			Per-host	Default
<input type="button" value="Remove"/>						

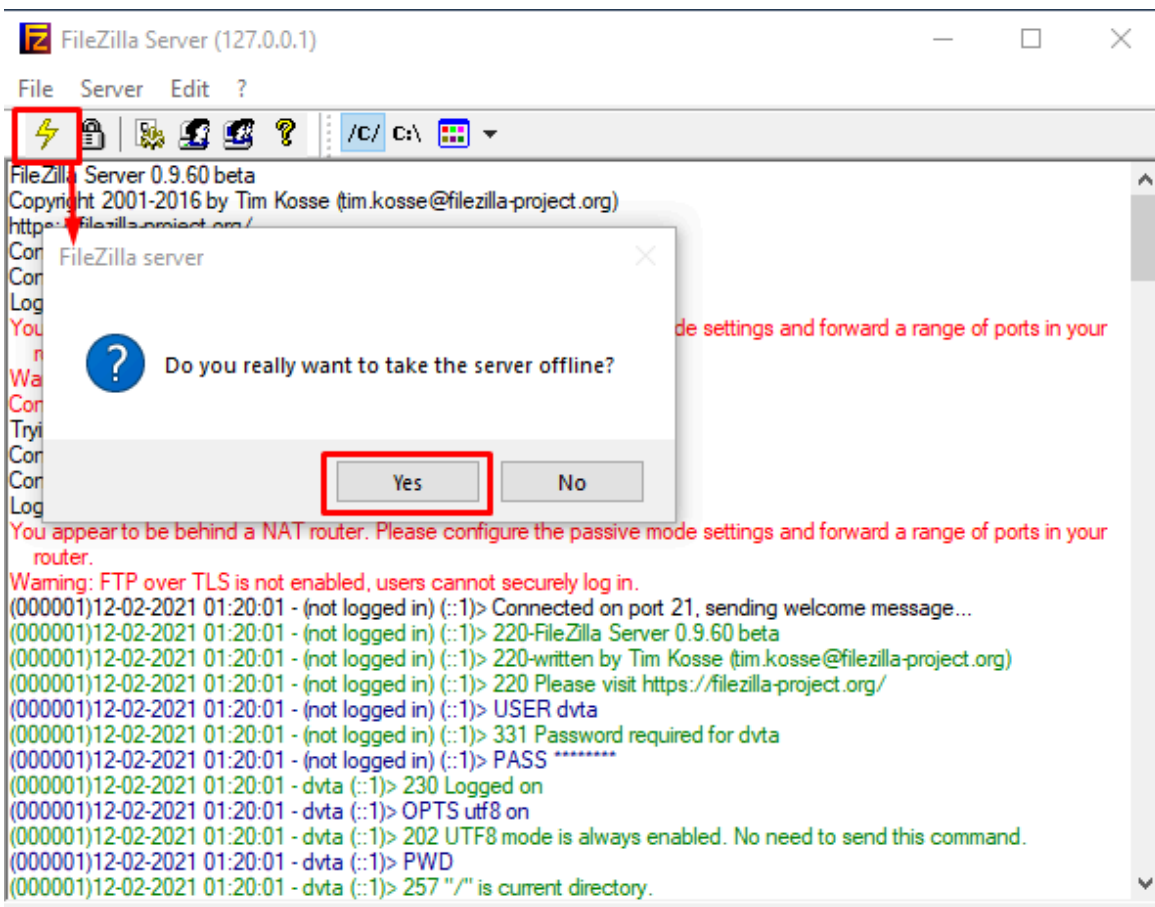
Each installation of Burp generates its own CA certificate that Proxy listeners can use when negotiating TLS connections. You can import or regenerate a CA certificate.

Let's open up the DVTA.exe application and configure the server to the IP address of the local machine such as 192.168.0.103.

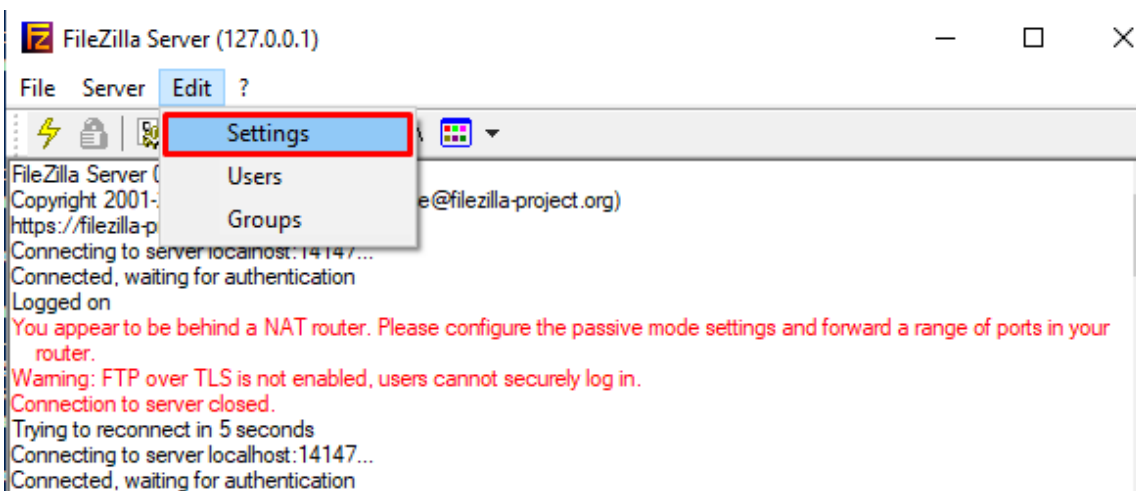


Now open up the FileZilla Server interface and stop the FTP server interface

and then go to ***“Edit > Settings”*** and change the value of “Listen on these ports: 2111” because in this scenario our MITM Relay is also running on port no. 21 and FileZilla server is also listening on port no. 21 and it is not possible to run both services on same port so, that's why we change the port 21 to 2111 here. This happens just because of we are running FileZilla server and MITM relay in same machine in the real-life penetration testing these services are running on different machines so we don't face these problems there. **There is another setting that you have to change in security settings that is to “Disable IP check”.**



Then go to **"Edit > Settings"**



and change the value of "Listen on these ports: 2111" because in this scenario our MITM Relay is also running on port no. 21 and FileZilla server is also listening on port no. 21 and it is not possible to run both services on same port so, that's why we change the port 21 to 2111 here. This happens just because of we are running FileZilla server and MITM relay in same machine in the real-life penetration testing these services are running on different machines so we don't face these problems there.

The screenshot shows the 'FileZilla Server Options' dialog box with the 'General settings' tab selected. On the left is a tree view of settings categories: General settings (selected), Welcome message, IP bindings, IP Filter, Passive mode settings, Security settings, Miscellaneous, Admin Interface settings, Logging, Speed Limits, Filetransfer compression, FTP over TLS settings, and Autoban. The main area is divided into three sections: Connection settings, Performance settings, and Timeout settings. In the Connection settings, 'Listen on these ports' is set to '2111' (highlighted with a red box) and 'Max. number of users' is set to '0'. In the Performance settings, 'Number of threads' is set to '2'. In the Timeout settings, 'Connections timeout' is '120', 'No Transfer timeout' is '600', and 'Login timeout' is '60'. Each timeout setting includes a descriptive text explaining the unit and range. At the bottom left are 'OK' and 'Cancel' buttons.

General settings FileZilla Server

Connection settings

Listen on these ports: List of ports between 1 and 65535. These ports are used both for plain FTP and explicit FTP over TLS. (Default port: 21)

Max. number of users: (0 for unlimited users)

Performance settings

Number of threads: This value should be a multiple of the number of processors installed on your system. Increase this value if your server is under heavy load.

Timeout settings

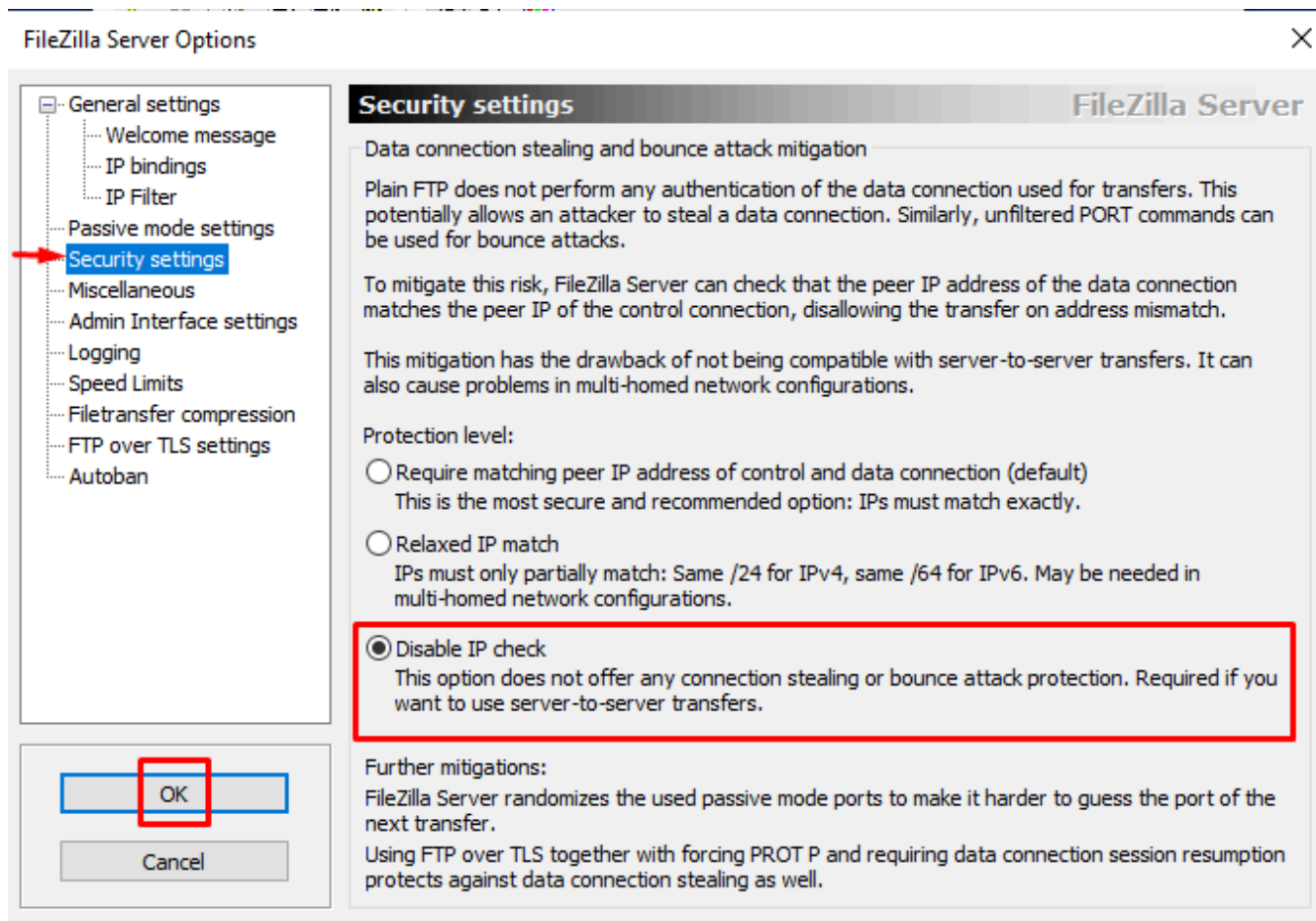
Connections timeout: in seconds (1-9999, 0 for no timeout).

No Transfer timeout: in seconds (600-9999, 0 for no timeout). This value specifies the time a user has to initiate a file transfer.

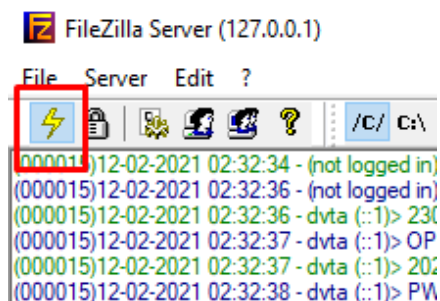
Login timeout: in seconds (1-9999, 0 for no timeout). This value specifies the time in which a new user has to login.

OK Cancel

There is another setting that you have to change in security settings that is to **“Disable IP check”**. If you don’t do this the file upload will fail because the source which is originating the file transfer and the address where the data connection is controlled are not going to match that’s why we disabled the IP check.



And at last start the FTP server.



Come back to the Power Shell and start the MITM Relay

All you need to do is go to the directory of MITM Relay and copy the location of MITM Relay and open it into a power shell by just running these commands.

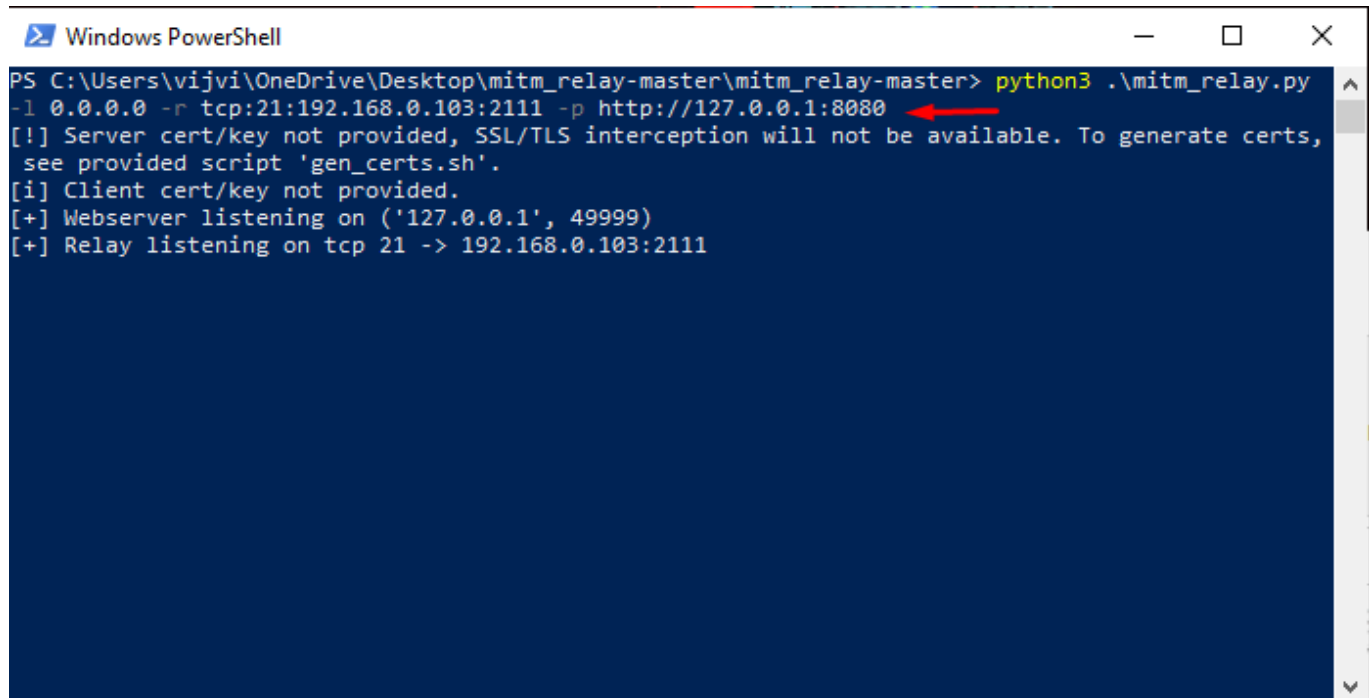
cd (directory of MITM Relay)

```
python3.exe .\mitm_relay.py -l 0.0.0.0 -r tcp:21:192.168.0.103:2111 -p http://127.
```

where -l is the listening address which is 0.0.0.0 so that we can listen to all the interfaces.

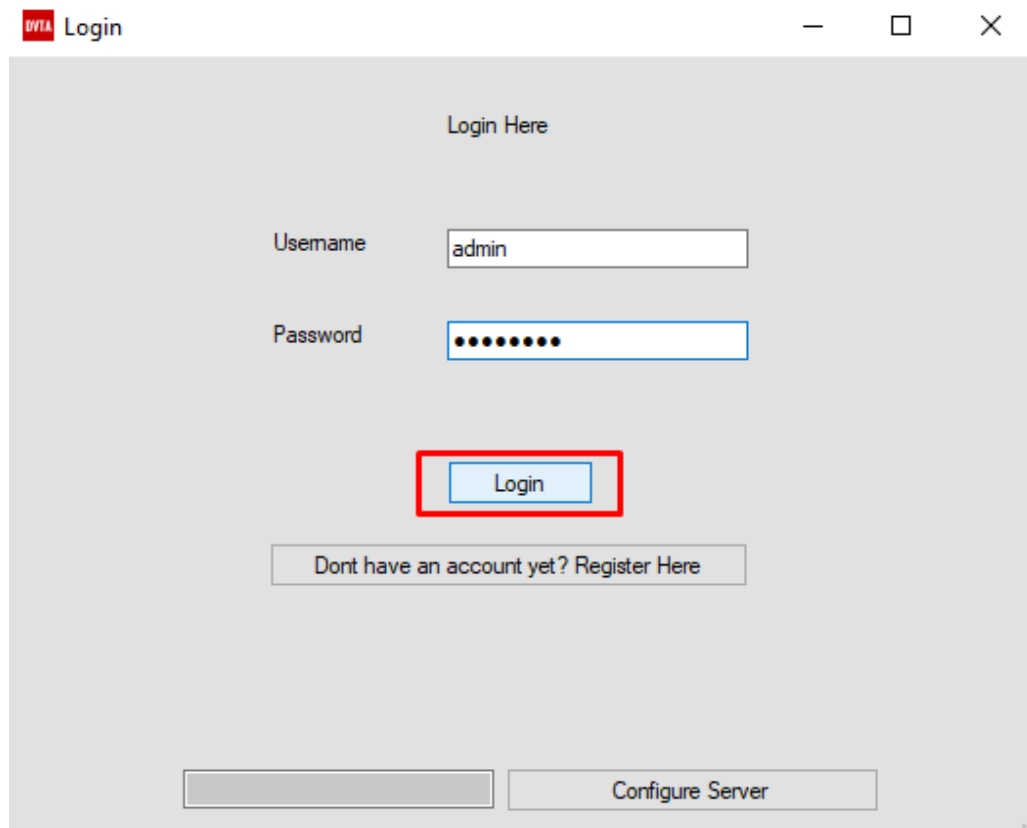
-r stands for the relay. We are setting up relay here using tcp:21 and the destination address is going to be 192.168.0.103 this is the destination address where FTP server is running and the port on which the FTP server on the destination server is running is 2111.

And in last -p stands for the proxy. We set it up to 127.0.0.1:8080 this is the address where the Burp Suite is running.



```
Windows PowerShell
PS C:\Users\vijvi\OneDrive\Desktop\mitm_relay-master\mitm_relay-master> python3 .\mitm_relay.py
-l 0.0.0.0 -r tcp:21:192.168.0.103:2111 -p http://127.0.0.1:8080
[!] Server cert/key not provided, SSL/TLS interception will not be available. To generate certs,
    see provided script 'gen_certs.sh'.
[i] Client cert/key not provided.
[+] Webserver listening on ('127.0.0.1', 49999)
[+] Relay listening on tcp 21 -> 192.168.0.103:2111
```

Let's come back to the DVTA application and log in to the application using the admin credentials



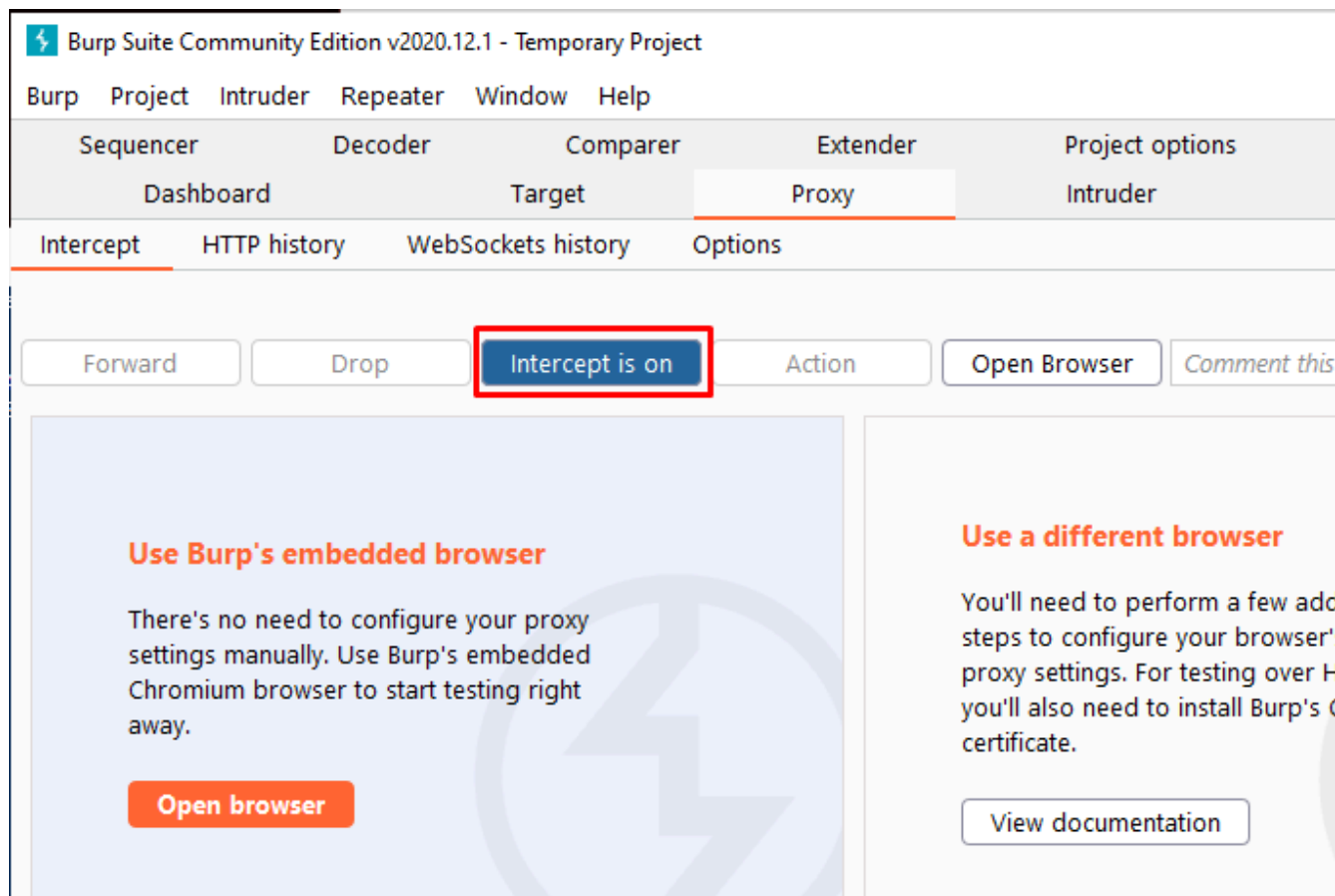
DVTA Login

Login Here

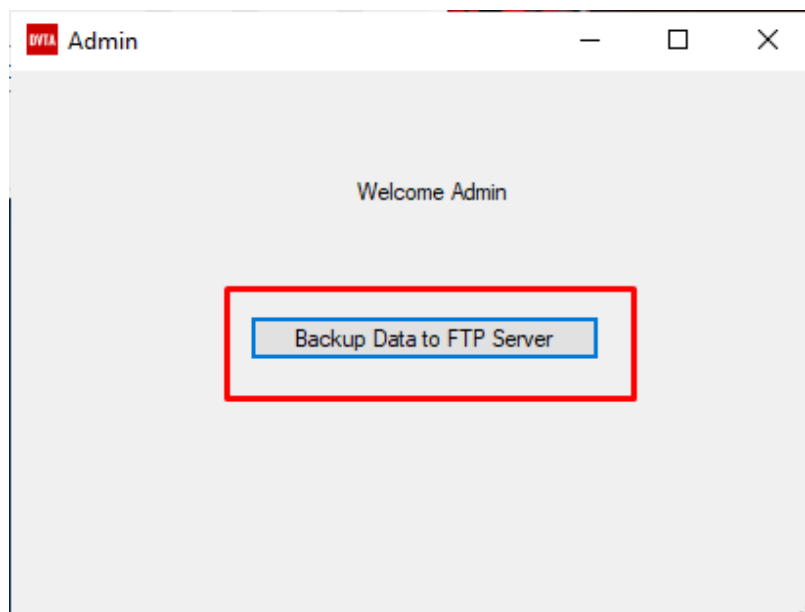
Username:

Password:

And then come back to Burp Suite and navigate to the tab of “*proxy > Intercept*” and “*Turn on the Intercept*” and so that it will start capturing the traffic coming from DVTA.



Now, come back to DVTA application and hit “*Backup Data to FTP Server*”



Come back to the Burp Suite and look at where it started intercepting the Traffic from DVTA that is a response coming from the FTP server.

⚡ Burp Suite Community Edition v2021.2 - Temporary Project

— □ ×

Burp Project Intruder Repeater Window Help

Sequencer	Decoder	Comparer	Extender	Project options	User options
Dashboard		Target	Proxy	Intruder	Repeater
Intercept	HTTP history	WebSockets history	Options		

✎ Request to http://127.0.0.1:49999

Forward Drop Intercept... Action Open Br... Comment this item

Pretty Raw \n Actions

```
1 POST /SERVER_RESPONSE/from/192.168.0.103/2111 HTTP/1.1
2 Host: 127.0.0.1:49999
3 Accept-Encoding: gzip, deflate
4 X-Mitm_Relay-To: 192.168.0.103:50005
5 X-Mitm_Relay-From: 192.168.0.103:2111
6 Content-Length: 143
7 User-Agent: python-urllib3/1.26.3
8 Connection: close
9
10 220-FileZilla Server 0.9.60 beta
11 220-written by Tim Kosse (tim.kosse@filezilla-project.org)
12 220 Please visit https://filezilla-project.org/
13
```

INSPECTOR

ⓘ ⚙ ⬅ ➡ Search... 0 matches

Click forward and look on the next screen there is a user name with the user command being sent from the client. On the first line there you can see the client request and forward this

⚡ Burp Suite Community Edition v2021.2 - Temporary Project

— □ ×

Burp Project Intruder Repeater Window Help

Sequencer	Decoder	Comparer	Extender	Project options	User options
Dashboard		Target	Proxy	Intruder	Repeater

Intercept HTTP history WebSockets history Options

✎ Request to http://127.0.0.1:49999

Forward Drop Intercept... Action Open Br... Comment this item

Pretty Raw \n Actions

```
1 POST /CLIENT_REQUEST/to/192.168.0.103/2111 HTTP/1.1
2 Host: 127.0.0.1:49999
3 Accept-Encoding: gzip, deflate
4 X-Mitm_Relay-To: 192.168.0.103:2111
5 X-Mitm_Relay-From: 192.168.0.103:50005
6 Content-Length: 11
7 User-Agent: python-urllib3/1.26.3
8 Connection: close
9
10 USER dvta
11
```

INSPECTOR

0 matches

Now we're getting server response which is the password required for the DVTA. On the next screen, the client may send the password to see this just forward the Traffic

⚡ Burp Suite Community Edition v2021.2 - Temporary Project

— □ ×

Burp Project Intruder Repeater Window Help

Sequencer	Decoder	Comparer	Extender	Project options	User options
Dashboard	Target	Proxy	Intruder	Repeater	

Intercept HTTP history WebSockets history Options

✎ Request to http://127.0.0.1:49999

Forward Drop Intercept... Action Open Br... Comment this item

Pretty Raw \n Actions

```
1 POST /SERVER_RESPONSE/from/192.168.0.103/2111 HTTP/1.1
2 Host: 127.0.0.1:49999
3 Accept-Encoding: gzip, deflate
4 X-Mitm_Relay-To: 192.168.0.103:50090
5 X-Mitm_Relay-From: 192.168.0.103:2111
6 Content-Length: 32
7 User-Agent: python-urllib3/1.26.3
8 Connection: close
9
10 331 Password required for dvta
11
```

INSPECTOR

0 matches

Look at the bottom using the pass command it sending the password: – P@ssw0rd

⚡ Burp Suite Community Edition v2021.2 - Temporary Project

Burp Project Intruder Repeater Window Help

Sequencer	Decoder	Comparer	Extender	Project options	User options
Dashboard		Target	Proxy	Intruder	Repeater

Intercept HTTP history WebSockets history Options

✎ Request to http://127.0.0.1:49999

Forward Drop Intercept... Action Open Br... Comment this item

Pretty Raw \n Actions

```
1 POST /CLIENT_REQUEST/to/192.168.0.103/2111 HTTP/1.1
2 Host: 127.0.0.1:49999
3 Accept-Encoding: gzip, deflate
4 X-Mitm_Relay-To: 192.168.0.103:2111
5 X-Mitm_Relay-From: 192.168.0.103:50090
6 Content-Length: 15
7 User-Agent: python-urllib3/1.26.3
8 Connection: close
9
10 PASS p@sswOrd
11
```

INSPECTOR

0 matches

Now the next subsequent request will upload the file onto the server so we can see that here on next request the user is logged on forward the traffic

⚡ Burp Suite Community Edition v2021.2 - Temporary Project

Burp Project Intruder Repeater Window Help

Sequencer	Decoder	Comparer	Extender	Project options	User options
Dashboard		Target	Proxy	Intruder	Repeater

Intercept HTTP history WebSockets history Options

✎ Request to http://127.0.0.1:49999

Forward Drop Intercept... Action Open Br... Comment this item

Pretty Raw \n Actions ▾

```
1 POST /SERVER_RESPONSE/from/192.168.0.103/2111 HTTP/1.1
2 Host: 127.0.0.1:49999
3 Accept-Encoding: gzip, deflate
4 X-Mitm_Relay-To: 192.168.0.103:50132
5 X-Mitm_Relay-From: 192.168.0.103:2111
6 Content-Length: 15
7 User-Agent: python-urllib3/1.26.3
8 Connection: close
9
10 230 Logged on ←
11
```

INSPECTOR

ⓘ ⚙ ⬅ ➡ Search... 0 matches

After forwarding some of requests there is a store request which is uploading the “*ftp-admin.csv*” file. Forward the request to next

⚡ Burp Suite Community Edition v2021.2 - Temporary Project

Burp Project Intruder Repeater Window Help

Sequencer	Decoder	Comparer	Extender	Project options	User options
Dashboard		Target	Proxy	Intruder	Repeater

Intercept HTTP history WebSockets history Options

✎ Request to http://127.0.0.1:49999

Forward Drop Intercept... Action Open Br... Comment this item

Pretty Raw \n Actions

```
1 POST /CLIENT_REQUEST/to/192.168.0.103/2111 HTTP/1.1
2 Host: 127.0.0.1:49999
3 Accept-Encoding: gzip, deflate
4 X-Mitm_Relay-To: 192.168.0.103:2111
5 X-Mitm_Relay-From: 192.168.0.103:50132
6 Content-Length: 20
7 User-Agent: python-urllib3/1.26.3
8 Connection: close
9
10 STOR ftp-admin.csv
11
```

INSPECTOR

0 matches

And on the next request there is a response from the server that is opening the data channel for the file upload

⚡ Burp Suite Community Edition v2021.2 - Temporary Project

— □ ×

Burp Project Intruder Repeater Window Help

Sequencer	Decoder	Comparer	Extender	Project options	User options
Dashboard		Target	Proxy	Intruder	Repeater

Intercept HTTP history WebSockets history Options

✎ Request to http://127.0.0.1:49999

Forward Drop Intercept... Action Open Br... Comment this item

Pretty Raw \n Actions ▾

```
1 POST /SERVER_RESPONSE/from/192.168.0.103/2111 HTTP/1.1
2 Host: 127.0.0.1:49999
3 Accept-Encoding: gzip, deflate
4 X-Mitm_Relay-To: 192.168.0.103:50132
5 X-Mitm_Relay-From: 192.168.0.103:2111
6 Content-Length: 72
7 User-Agent: python-urllib3/1.26.3
8 Connection: close
9
10 150 Opening data channel for file upload to server of "/ftp-admin.csv"
11
```

INSPECTOR

ⓘ ⚙ ⏪ ⏩ Search... 0 matches

Let's forward the traffic and it is completed the upload.

⚡ Burp Suite Community Edition v2021.2 - Temporary Project

Burp Project Intruder Repeater Window Help

Sequencer	Decoder	Comparer	Extender	Project options	User options
Dashboard	Target	Proxy	Intruder	Repeater	

Intercept HTTP history WebSockets history Options

✎ Request to http://127.0.0.1:49999

Forward Drop Intercept... Action Open Br... Comment this item

Pretty Raw \n Actions

```
1 POST /SERVER_RESPONSE/from/192.168.0.103/2111 HTTP/1.1
2 Host: 127.0.0.1:49999
3 Accept-Encoding: gzip, deflate
4 X-Mitm_Relay-To: 192.168.0.103:50132
5 X-Mitm_Relay-From: 192.168.0.103:2111
6 Content-Length: 47
7 User-Agent: python-urllib3/1.26.3
8 Connection: close
9
10 226 Successfully transferred "/ftp-admin.csv"
11
```

INSPECTOR

0 matches

This is how we can use MITM Relay to intercept and analyze the TCP traffic using Burp Suite.

This is looking good we have grabbed the FTP username and password that is coming from the client. In the next part, we will perform some attacks on the thick client.