Graham Dungan
February 1, 2025
UIN: 332001764

CSCE 413: Software Security
PoC 5

Development of Web Application for MITM Attack (20 Points)

For this assignment, I created a simple HTTP banking server that allows users to check their balance using their account id and token. In order to demonstrate a man-in-the-middle attack, I have also created three docker containers, `client`, `server`, and `mitm`.

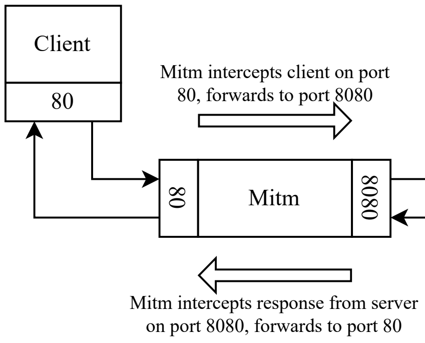

Client
80
Mitm intercepts client on port 80, forwards to port 8080
Server
8080
80
Mitm
8080
Mitm intercepts response from server on port 8080, forwards to port 80

Figure 1: Mitm intercept behavior

The `server` docker container runs the banking web application on port 8080. The `client` docker container will execute curl commands and is set to use port 80 as its proxy. The `mitm` container, running mitmproxy, will listen on port 80 for any requests from the client.
An example mitm passive attack scenario is available by running the `./exploit.sh` script. To do this,

1. Install Docker Engine on your machine (required)
2. Enter the PoC5 directory
   `cd PoC5`
3. Run the `exploit.sh` script in the PoC5 directory.
   `./exploit.sh`

The script will demonstrate a simple user login with an account id and a token. It will then display the MITM's logs, showing that the request and response of the server have been intercepted.

Configuration of MITM Tool (25 Points)

The MITM tool's configuration can be found in its Dockerfile, `./mitm/Dockerfile`, and within the docker-compose `./docker-compose.yml`.

The Dockerfile imports the mitmproxy image from Docker Hub, as seen on line 1. This is the tool that we will use to perform our MITM attack. It runs the command-line, non-interactive form of mitmproxy ”mitmdump” and is set to listen for traffic on port 80. The `-v` argument has been included for verbose output.

```
1 FROM mitmproxy/mitmproxy
2 CMD ["mitmdump", "--listen-port", "80", "-v"]
```

The docker-compose.yml builds the MITM container on a shared network with other containers, `mitm_network`, and exposes port 80 for listening.

```
mitm:
    build: ./mitm
    networks:
      - mitm_network
    ports:
      - "80:80"
```

The client will not naturally attempt to communicate to the web server hosted on port 8080 through port 80, so we manually set the environment variables in `./client/Dockerfile` to use `mitm:80` as a HTTP proxy,

```
FROM curlimages/curl:latest
ENV http_proxy=http://mitm:80
ENV https_proxy=http://mitm:80
CMD ["sh"]
```

This ensures that the client will communicate through the MITM to send requests to the HTTP server.

The server Dockerfile simply runs the Python `webapp.py` program which runs on port 8080. The docker-compose similarly joins it to the `mitm_network` network to speak to the MITM and client and exposes port 8080.

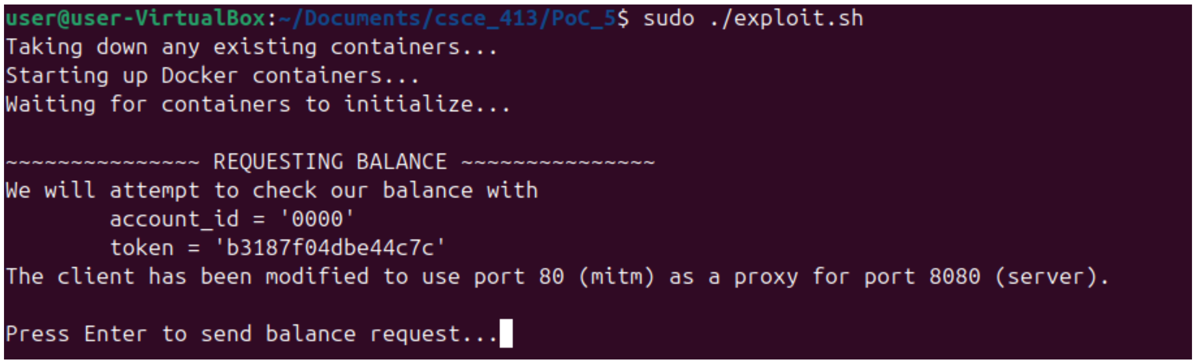
```
server:
    build: ./server
    networks:
      - mitm_network
    ports:
      - "8080:8080"
```

Demonstration of MITM Attack (30 Points)

This attack can be demonstrated via `./exploit.sh`, as explained by the directions earlier.

Client sends Request

The client will send a simple curl command to request that they can view the balance of the user account `0000` with token `b3187f04dbe44c7c`. It can be seen within `exploit.sh` that the client executes the curl command `curl -v "http://server:8080/balance?&account_id=0000&token=b3187f04dbe44c7c"`



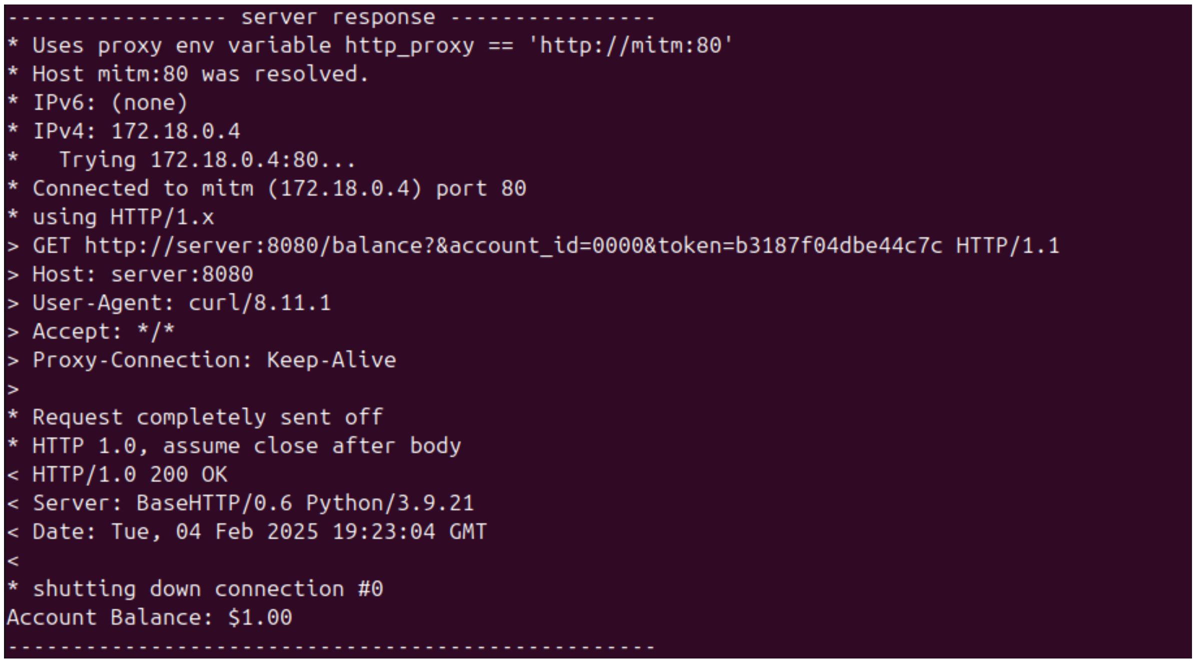
```
user@user-VirtualBox:~/Documents/csce_413/PoC_5$ sudo ./exploit.sh
Taking down any existing containers...
Starting up Docker containers...
Waiting for containers to initialize...

~~~~~~~~~~~~~~~ REQUESTING BALANCE ~~~~~~~~~~~~~~~
We will attempt to check our balance with
        account_id = '0000'
        token = 'b3187f04dbe44c7c'
The client has been modified to use port 80 (mitm) as a proxy for port 8080 (server).

Press Enter to send balance request...
```

Server Reciprocates Request

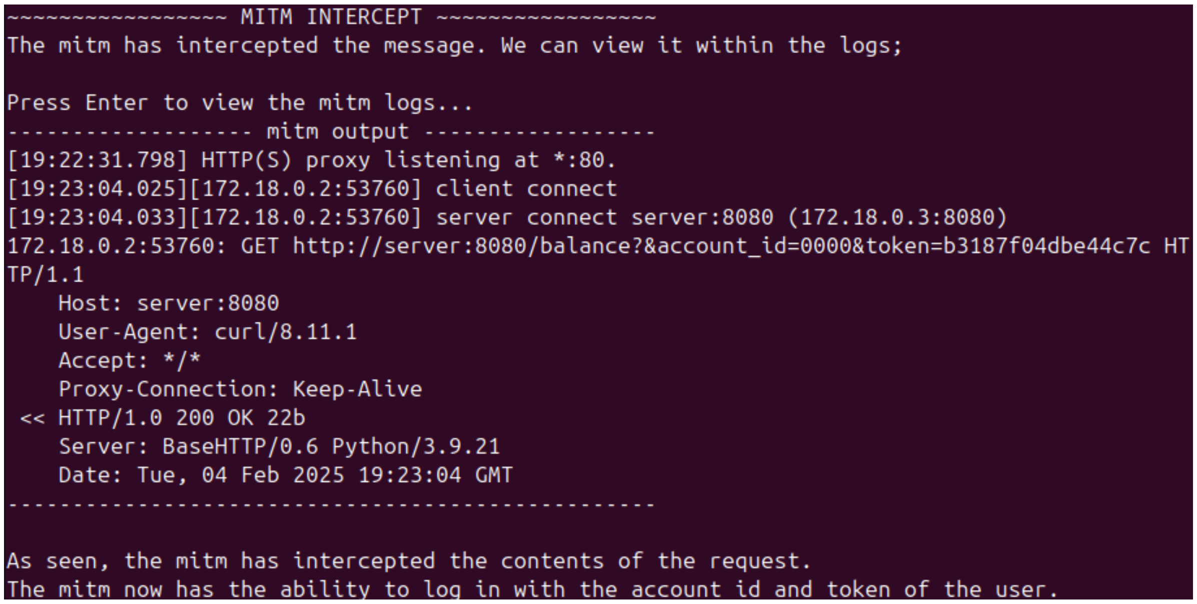
The user has sent the request and the server responds with the contents of the bank account. While the server is aware that it is using a proxy, a real-world implementation may not have this awareness. It is then feasible that the MITM could passively listen to data, unnoticed.



```
----------------- server response ----------------
* Uses proxy env variable http_proxy == 'http://mitm:80'
* Host mitm:80 was resolved.
* IPv6: (none)
* IPv4: 172.18.0.4
*   Trying 172.18.0.4:80...
* Connected to mitm (172.18.0.4) port 80
* using HTTP/1.x
> GET http://server:8080/balance?&account_id=0000&token=b3187f04dbe44c7c HTTP/1.1
> Host: server:8080
> User-Agent: curl/8.11.1
> Accept: */*
> Proxy-Connection: Keep-Alive
>
* Request completely sent off
* HTTP 1.0, assume close after body
< HTTP/1.0 200 OK
< Server: BaseHTTP/0.6 Python/3.9.21
< Date: Tue, 04 Feb 2025 19:23:04 GMT
<
* shutting down connection #0
Account Balance: $1.00
--------------------------------------------------
```

MITM Acquires Data

Given that the client and server used the MITM as a proxy, the MITM had seen the entire transaction. The user's request and the server's response (the 22 byte response 200 response) can be seen during the connection.



```
~~~~~~~~~~~~~~~~ MITM INTERCEPT ~~~~~~~~~~~~~~~~
The mitm has intercepted the message. We can view it within the logs;

Press Enter to view the mitm logs...
------------------ mitm output -----------------
[19:22:31.798] HTTP(S) proxy listening at *:80.
[19:23:04.025][172.18.0.2:53760] client connect
[19:23:04.033][172.18.0.2:53760] server connect server:8080 (172.18.0.3:8080)
172.18.0.2:53760: GET http://server:8080/balance?&account_id=0000&token=b3187f04dbe44c7c HT
TP/1.1
    Host: server:8080
    User-Agent: curl/8.11.1
    Accept: */*
    Proxy-Connection: Keep-Alive
 << HTTP/1.0 200 OK 22b
    Server: BaseHTTP/0.6 Python/3.9.21
    Date: Tue, 04 Feb 2025 19:23:04 GMT
-------------------------------------------------

As seen, the mitm has intercepted the contents of the request.
The mitm now has the ability to log in with the account id and token of the user.
```

Analysis of Captured Data and Implications (15 Points)

It is seen that the MITM has captured the client and server's connection, the GET request including the user's credentials (account ID and token), and the server's response.

This attack implies that, since the MITM was used a proxy, to the server's knowledge, the request from the MITM was legitimately from the user. Any communication, unless encrypted, would be exposed for a third party to view or manipulate. An account ID, password, and the balance of the user was stolen in this instance, but other sensitive data such as personally identifiable information, credentials, or communications could be viewed and even changed.

This MITM attack compromises confidentiality as the server and client have no knowledge that the sensitive data (passwords, tokens, IDs) had been exposed to some unauthorized party. Following, this MITM attack compromises integrity because the MITM could easily manipulate request information (deposit amounts, messages, etc.) or manipulate response information (savings, account information, etc.) without the knowledge of the server or client.

Documentation (10 Points)

See above.