Progress report (Iteration IV)

Daria Vaskovskaya (d.vaskovskaya@innopolis.ru)

March 2020

1 Intro

- **Student Name**: Daria Vaskovskaya
- **Topic**: Man In The Middle Attacks: Execution and Detection
- **Supervisor**: Rasheed Hussain
- **Iteration number**: 4

2 Done during third iteration

- `task`
  - dig into mitmproxy and see what SSL library it is using and how it deals with it
  - dig into mitmengine and how to make it stronger
  - (spoof user-agent according to some TLS behavior)
- `hours spent` 12 hours
- `status` Partiallly done
- `results` will be available below

3 Current issues

- Seems to be that because of not enough information in mitmengine database it detects mitm even if there is no MITM (has not enough information about possible encryptions with received User Agent)

4 Results

4.1 Introduction

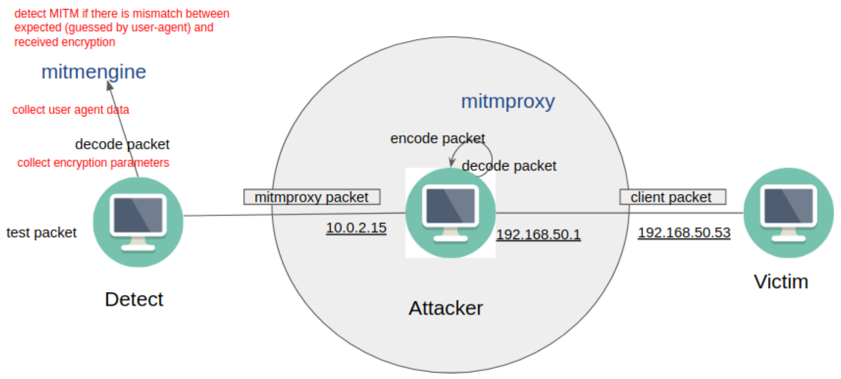
Firstly, I explored mitmproxy code a little bit. It was noticeable that it does not do anything smart with SSL.

I decided to start with exploring code of mitmengine to get better knowledge of how it works exactly. After exploring the code deeper, I realized that I wrongly understood the input and output formats of mitmengine. Therefore, I decided to fix those issues in order to check whether mitmengine will correctly detect MITM now.

I decided to write scripts to automate the process as well as I can. I wrote two scripts (one to emulate and test connection with MITM and another one to emulate and test connection without MITM).

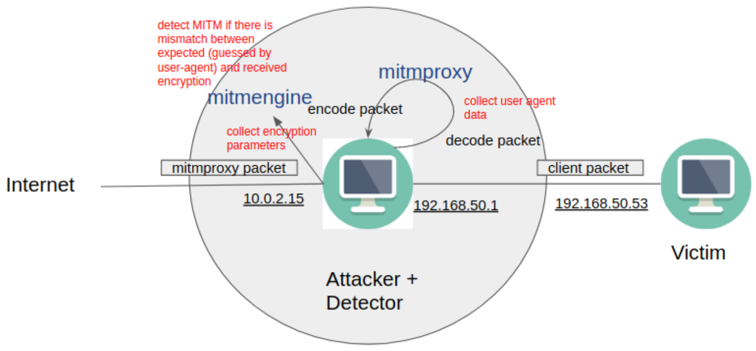
The expected result was that after completing the first script mitmengine will detect MITM, while after completing the second one it won't (because there will not be any MITM)

That is how mitmengine must be used:

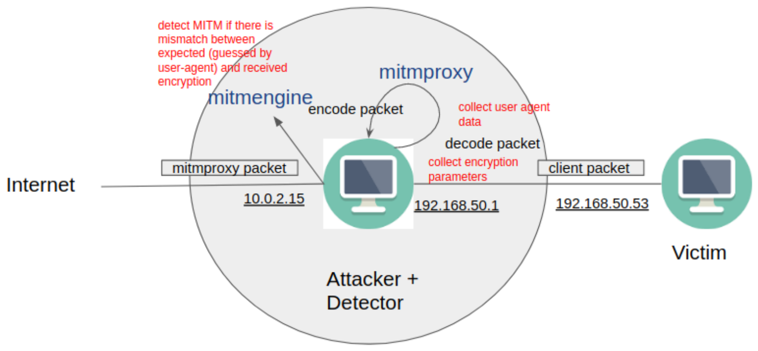


However, I decided not to involve real server with SSL and white IP (the one on which mitmengine is installed), because I thought that I can test everything faster If I use the existing environment (however, I spent much time even on that).

My scheme "with MITM":



My scheme "without MITM" (proxy here works as a server):



Created scripts catch any data from victim in 30 seconds.

4.2 Scripts

1st ("with MITM") script:

```
#!/bin/bash

#prepare inputs for the mitmengine
rm /tmp/01_encrypted.pcap
touch /tmp/01_encrypted.pcap

rm /tmp/02_decrypted.pcap
touch /tmp/02_decrypted.pcap

rm /tmp/03_headers.json
```

```
touch /tmp/03_headers.json


#prepare to launch mitmproxy
sysctl -w net.ipv4.ip_forward=1
sysctl -w net.ipv6.conf.all.forwarding=1
sysctl -w net.ipv4.conf.all.send_redirects=0
iptables -t nat -A PREROUTING -i enp0s8 -p tcp \
    --dport 80 -j REDIRECT --to-port 8080
iptables -t nat -A PREROUTING -i enp0s8 -p tcp \
    --dport 443 -j REDIRECT --to-port 8080
ip6tables -t nat -A PREROUTING -i enp0s8 -p tcp \
    --dport 80 -j REDIRECT --to-port 8080
ip6tables -t nat -A PREROUTING -i enp0s8 -p tcp \
    --dport 443 -j REDIRECT --to-port 8080

# launch mitmdump (same as mitmproxy, but can be
# launched in background) in transparent mode
MITMPROXY_SSLKEYLOGFILE=/tmp/ssl_key_log \
    ./mitmdump --mode transparent --showhost &
export MITMPROXY_PID=$!

#record data with tshark

#enp0s8 - interface before mitmproxy
tshark -i enp0s8 -w /tmp/01_encrypted.pcap &
export TSHARK_RX_CAP_PID=$!

#enp0s3 - interface which goes to internet
tshark -i enp0s3 -w /tmp/01_uplink_encrypted.pcap &
export TSHARK_TX_CAP_PID=$!

echo "waiting on requests"
sleep 30
echo "lol, ne uspel"

kill $MITMPROXY_PID
kill $TSHARK_RX_CAP_PID
kill $TSHARK_TX_CAP_PID

iptables -t nat -D PREROUTING 1
iptables -t nat -D PREROUTING 1

# decrypt victim's data in order to get user agents
tshark -nr /tmp/01_encrypted.pcap -Y http \
    -o tls.keylog_file:/tmp/ssl_key_log \
    -w /tmp/02_decrypted.pcap

# parse decrypted data to json
# (this json is used by mitmengine as an input)
tshark -nr /tmp/02_decrypted.pcap -T json -Y http \
    -e http.request.line > /tmp/03_headers.json

# run mitmengine with prepared data
chmod a+rw /tmp/01_uplink_encrypted.pcap /tmp/03_headers.json
sudo -i -u linuxlite /bin/bash \
    -c "(cd /home/linuxlite/mitmengine \
        && go run cmd/demo/my_main.go \
            -handshake /tmp/01_uplink_encrypted.pcap \
            -header /tmp/03_headers.json)"
```

2nd ("without MITM") script

```
#!/bin/bash
#Prepare inputs for mitmengine
rm /tmp/01_encrypted.pcap
touch /tmp/01_encrypted.pcap

rm /tmp/02_decrypted.pcap
touch /tmp/02_decrypted.pcap

rm /tmp/03_headers.json
touch /tmp/03_headers.json

#prepare to launch mitmproxy
sysctl -w net.ipv4.ip_forward=1
sysctl -w net.ipv6.conf.all.forwarding=1
sysctl -w net.ipv4.conf.all.send_redirects=0
iptables -t nat -A PREROUTING -i enp0s8 -p tcp \
    --dport 80 -j REDIRECT --to-port 8080
iptables -t nat -A PREROUTING -i enp0s8 -p tcp \
    --dport 443 -j REDIRECT --to-port 8080
ip6tables -t nat -A PREROUTING -i enp0s8 -p tcp \
    --dport 80 -j REDIRECT --to-port 8080
ip6tables -t nat -A PREROUTING -i enp0s8 -p tcp \
    --dport 443 -j REDIRECT --to-port 8080

# launch mitmdump (same as mitmproxy, but can be
# launched in background) in transparent mode
MITMPROXY_SSLKEYLOGFILE=/tmp/ssl_key_log \
    ./mitmdump --mode transparent --showhost &
export MITMPROXY_PID=$!
```

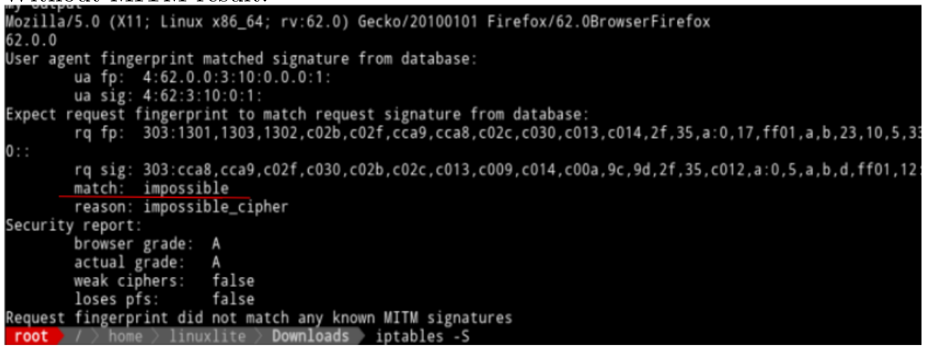
```
# record data with tshark

# enp0s8 - interface before mitmproxy
tshark -i enp0s8 -w /tmp/01_encrypted.pcap &
export TSHARK_RX_CAP_PID=$!

echo "waiting on requests"
sleep 30
echo "lol, ne uspel"

kill $MITMPROXY_PID
kill $TSHARK_RX_CAP_PID

iptables -t nat -D PREROUTING 1
iptables -t nat -D PREROUTING 1


#decrypt user data as If mitmproxy is a server
tshark -nr /tmp/01_encrypted.pcap -Y http \
    -o tls.keylog_file:/tmp/ssl_key_log \
    -w /tmp/02_decrypted.pcap

# parse decrypted data to json
# (this json is used by mitmengine as an input)
tshark -nr /tmp/02_decrypted.pcap -T json -Y http \
    -e http.request.line > /tmp/03_headers.json

# HERE FILE WITH MITM PROXY ENCRYPTION IS NOT USED,
# ONLY WITH REAL USER'S ENCRYPTION
chmod a+rw /tmp/01_encrypted.pcap /tmp/03_headers.json
sudo -i -u linuxlite /bin/bash \
    -c "(cd /home/linuxlite/mitmengine \
        && go run cmd/demo/my_main.go \
            -handshake /tmp/01_encrypted.pcap \
            -header /tmp/03_headers.json)"
```

4.3 Outputs

The first problem was that mitmengine does not have any info about the new browser versions, it could not guess whether there is MITM or not, because it did not know anything about my User Agents. So, I had to search which User Agent can be possibly found in the mitmengine database if I change browser/browser version. There were some variants there, I have chosen to download Firefox 62.0

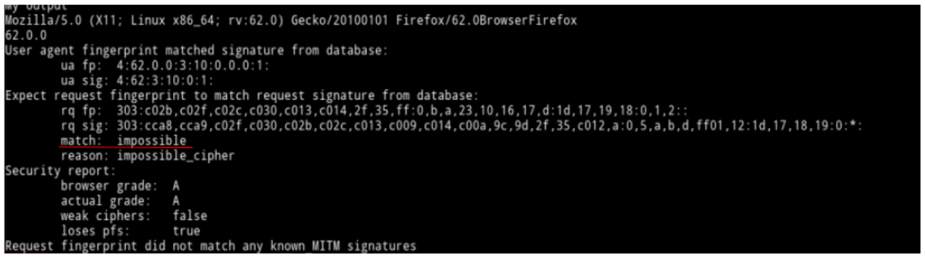
However, mitmengine still does not show expected output. It detects MITM in both cases (there is false positive)

Without MITM result:



```
Mozilla/5.0 (X11; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0BrowserFirefox
62.0.0
User agent fingerprint matched signature from database:
        ua fp:  4:62.0.0:3:10:0.0.0:1:
        ua sig: 4:62:3:10:0:1:
Expect request fingerprint to match request signature from database:
        rq fp:  303:1301,1303,1302,c02b,c02f,cca9,cca8,c02c,c030,c013,c014,2f,35,a:0,17,ff01,a,b,23,10,5,33
0::
        rq sig: 303:cca8,cca9,c02f,c030,c02b,c02c,c013,c009,c014,c00a,9c,9d,2f,35,c012,a:0,5,a,b,d,ff01,12
        match:  impossible
        reason: impossible_cipher
Security report:
        browser grade:  A
        actual grade:   A
        weak ciphers:   false
        loses pfs:      false
Request fingerprint did not match any known MITM signatures
 root  / home linuxlite Downloads  iptables -S
```

With MITM result:



```
Mozilla/5.0 (X11; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0BrowserFirefox
62.0.0
User agent fingerprint matched signature from database:
        ua fp:  4:62.0.0:3:10:0.0.0:1:
        ua sig: 4:62:3:10:0:1:
Expect request fingerprint to match request signature from database:
        rq fp:  303:c02b,c02f,c02c,c030,c013,c014,2f,35,ff:0,b,a,23,10,16,17,d:1d,17,19,18:0,1,2::
        rq sig: 303:cca8,cca9,c02f,c030,c02b,c02c,c013,c009,c014,c00a,9c,9d,2f,35,c012,a:0,5,a,b,d,ff01,12:1d,17,18,19:0:*:
        match:  impossible
        reason: impossible_cipher
Security report:
        browser grade:  A
        actual grade:   A
        weak ciphers:   false
        loses pfs:      true
Request fingerprint did not match any known MITM signatures
```

5 Future plans

Add my own signatures in mitmengine database and try mitmengine once more.

Hours: 12