BAB 4
HASIL PENELITIAN

4.1 RINGKASAN HASIL PENELITIAN

Penelitian ini bertujuan untuk menganalisis kerentanan penggunaan OpenSSL pada server *localhost* yang menggunakan protokol HTTPS, namun mengalami kendala dengan sertifikat CA yang tidak valid. *Server* yang digunakan dalam penelitian ini dibangun menggunakan *Node.js*. Analisis dilakukan dengan metode *sniffing* untuk mengidentifikasi potensi ancaman serangan SSL *hijacking*.

4.2 KONFIGURASI SERVER MENGGUNAKAN OPENSSL DAN NODE.JS

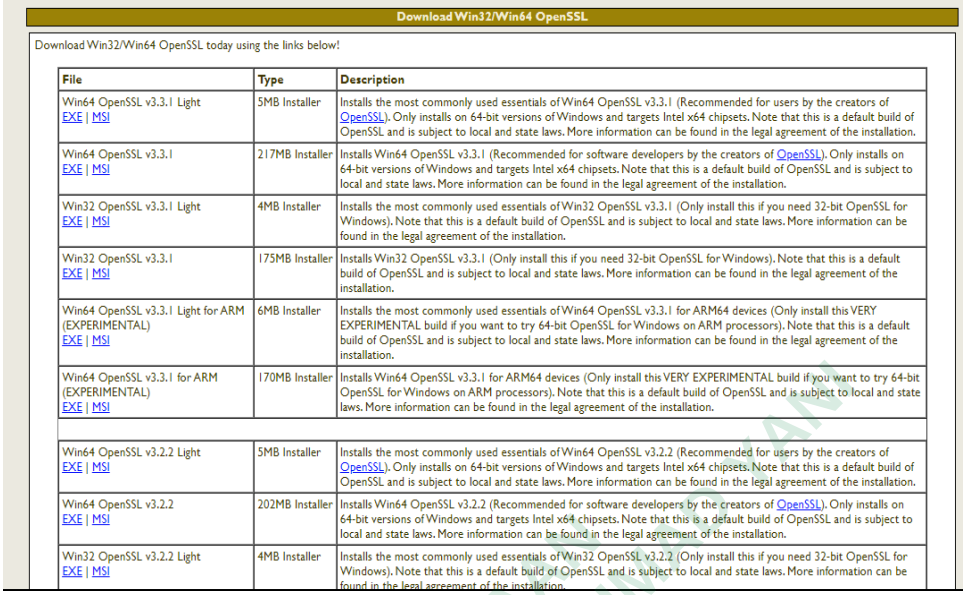
Untuk memastikan keamanan komunikasi data pada *server*, konfigurasi HTTPS menggunakan OpenSSL dan Node.js menjadi pilihan yang tepat. Proses ini melibatkan pembuatan sertifikat SSL/TLS dengan OpenSSL dan pengaturan *server Node.js* untuk menggunakan sertifikat tersebut, sehingga koneksi antara klien dan *server* dapat terenkripsi dengan baik.

Menggunakan *Windows* sebagai sistem operasi, langkah-langkah konfigurasi dimulai dengan instalasi OpenSSL dan *Node.js*. Selanjutnya, sertifikat SSL/TLS yang dibuat menggunakan OpenSSL dan kemudian diterapkan pada server Node.js. Tahap ini memastikan bahwa data yang dikirimkan melalui jaringan tidak dapat dengan mudah diakses atau dimodifikasi oleh pihak yang tidak berwenang. Panduan ini akan menjelaskan langkah-langkah detail untuk menyelesaikan konfigurasi tersebut.

4.2.1 Instalasi OpenSSL

Langkah pertama adalah menginstal OpenSSL pada sistem operasi yang digunakan. Dalam penelitian ini, sistem operasi yang digunakan adalah *Windows* 11. Berikut langkah-langkahnya:

1. Kunjungi situs resmi OpenSSL dan unduh versi terbaru dari OpenSSL *Installer* untuk *Windows*.
    a. Unduh OpenSSL v3.3.1 dapat dilihat pada Gambar 4.1.



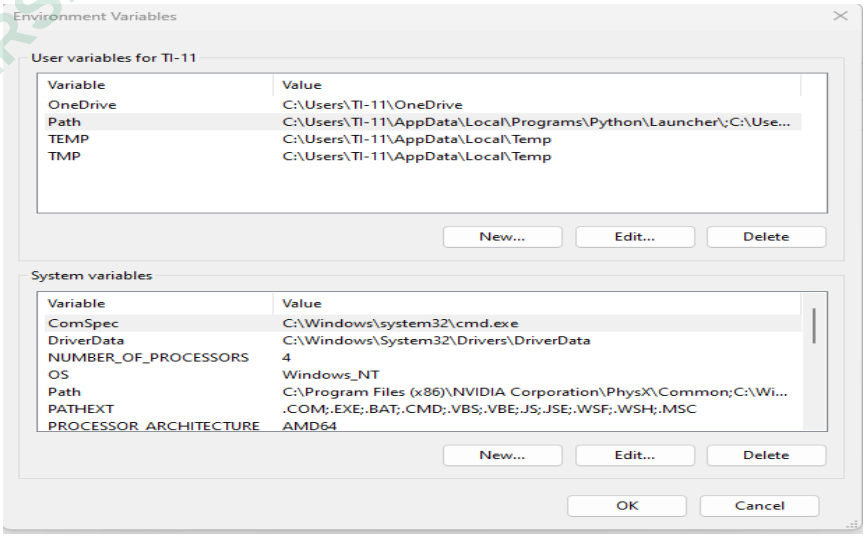
Download Win32/Win64 OpenSSL

Download Win32/Win64 OpenSSL today using the links below!

File	Type	Description
Win64 OpenSSL v3.3.1 Light EXE | MSI	5MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v3.3.1 (Recommended for users by the creators of OpenSSL). Only installs on 64-bit versions of Windows and targets Intel x64 chipsets. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v3.3.1 EXE | MSI	217MB Installer	Installs Win64 OpenSSL v3.3.1 (Recommended for software developers by the creators of OpenSSL). Only installs on 64-bit versions of Windows and targets Intel x64 chipsets. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v3.3.1 Light EXE | MSI	4MB Installer	Installs the most commonly used essentials of Win32 OpenSSL v3.3.1 (Only install this if you need 32-bit OpenSSL for Windows). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v3.3.1 EXE | MSI	175MB Installer	Installs Win32 OpenSSL v3.3.1 (Only install this if you need 32-bit OpenSSL for Windows). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v3.3.1 Light for ARM (EXPERIMENTAL) EXE | MSI	6MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v3.3.1 for ARM64 devices (Only install this VERY EXPERIMENTAL build if you want to try 64-bit OpenSSL for Windows on ARM processors). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v3.3.1 for ARM (EXPERIMENTAL) EXE | MSI	170MB Installer	Installs Win64 OpenSSL v3.3.1 for ARM64 devices (Only install this VERY EXPERIMENTAL build if you want to try 64-bit OpenSSL for Windows on ARM processors). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v3.2.2 Light EXE | MSI	5MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v3.2.2 (Recommended for users by the creators of OpenSSL). Only installs on 64-bit versions of Windows and targets Intel x64 chipsets. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v3.2.2 EXE | MSI	202MB Installer	Installs Win64 OpenSSL v3.2.2 (Recommended for software developers by the creators of OpenSSL). Only installs on 64-bit versions of Windows and targets Intel x64 chipsets. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v3.2.2 Light EXE | MSI	4MB Installer	Installs the most commonly used essentials of Win32 OpenSSL v3.2.2 (Only install this if you need 32-bit OpenSSL for Windows). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.

Gambar 4. 1 Unduh OpenSSL

2. Jalankan *file installer* yang telah diunduh dan ikuti petunjuk instalasi.
3. *Set Environment Variables*
    a. Setelah instalasi selesai, tambahkan *path* direktori *bin* OpenSSL ke *environment variables* sistem. Buka *"Control Panel" > "System and Security" > "System" > "Advanced system settings" > "Environment Variables". Konfigurasi dapat dilihat pada Gambar 4.2.*



Gambar 4. 2 Menambahkan *Path Variable*

b. Tambahkan *path* *C:\Program Files\OpenSSL-Win64\bin* (atau direktori instalasi yang sesuai) ke variabel *Path*.

4. Verifikasi Instalasi

a. Buka *Command Prompt* dan ketik *openssl version* untuk memverifikasi bahwa OpenSSL telah terinstal dengan benar.

```
C:\Program Files\OpenSSL-Win64\bin>openssl version
OpenSSL 3.3.1 4 Jun 2024 (Library: OpenSSL 3.3.1 4 Jun 2024)
```

Gambar 4.3 Versi OpenSSL

b. Pada Gambar 4.3 setelah memasukkan perintah *openssl version* pada *Command Prompt* maka akan muncul versi OpenSSL dan instalasi selesai.

4.2.2 Instalasi Node.js

Berikut adalah langkah-langkah untuk menginstal Node.js :

1. Kunjungi situs resmi Node.js dan unduh *installer* versi terbaru.

a. Unduh Node.js dapat dilihat pada Gambar 4.4.

Gambar 4. 4 Unduh Node.js

2. Jalankan file *installer* yang telah diunduh dan ikuti petunjuk instalasi.

3. Verifikasi Instalasi : Buka *Command Prompt* dan ketikan perintah *node -v* dan *npm –v* untuk memastikan Node.js dan *npm* telah terinstal dengan benar.

4.2.3 Pembuatan Sertifikat SSL

Setelah OpenSSL dan Node.js terinstal, langkah berikutnya adalah membuat sertifikat SSL *self-signed*:

1. Buka *Command Prompt* dan navigasikan ke direktori file untuk menyimpan sertifikat dan kunci.
2. Jalankan perintah berikut untuk membuat sertifikat *self-signed*:

   a. *openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365*

   b. Ikuti petunjuk untuk mengisi detail sertifikat seperti negara, organisasi, dan nama domain. Setelah selesai, dua file akan dihasilkan: *key.pem* (*private key*) dan *cert.pem* (*public certificate*).

4.2.4 Konfigurasi Server Node.js

Setelah sertifikat SSL dibuat, langkah selanjutnya adalah mengkonfigurasi server Node.js untuk menggunakan HTTPS:

1. Membuat file server

   Buat file *JavaScript* baru *server.js* dan tambahkan kode berikut:

server.js

C: > Windows > System32 > my-secure-server > server.js > ...

```
const fs = require('fs');
const https = require('https');
const express = require('express');
const bodyParser = require('body-parser');
const app = express();

// Sertifikat SSL
const privateKey = fs.readFileSync('localhost.key', 'utf8');
const certificate = fs.readFileSync('localhost.crt', 'utf8');
const credentials = { key: privateKey, cert: certificate };

// Middleware untuk mengurai data dari form POST
app.use(bodyParser.urlencoded({ extended: true }));

// Middleware untuk melayani file statis
app.use(express.static('public'));

// Route untuk halaman login
app.get('/', (req, res) => {
    res.sendFile(__dirname + '/public/login.html');
});

// Route untuk memproses login
app.post('/login', (req, res) => {
    const { username, password } = req.body;
    // Proses login sederhana
    if (username === 'admin' && password === 'password') {
        res.send('Login berhasil!');
    } else {
        res.send('Username atau password salah!');
    }
});

```

Gambar 4. 5 *JavaScript server.js*

a. Gambar 4.5 merupakan kode program *JavaScript* yang berfungsi sebagai perintah untuk menjalankan *server*.

2. Membuat tampilan *web server*

   Buat file *Html* baru login.html dan tambahkan kode berikut:

```
login.html
C: > Windows > System32 > my-secure-server > public > login.html > ...
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
</head>
<body>
    <h1>Login</h1>
    <form action="/login" method="post">
        <label for="username">Username:</label>
        <input type="text" id="username" name="username" required><br><br>
        <label for="password">Password:</label>
        <input type="password" id="password" name="password" required><br><br>
        <button type="submit">Login</button>
    </form>
</body>
</html>
```

Gambar 4. 6 *Login.html*

a. Gambar 4.6 berfungsi untuk merubah tampilan halaman *web*.

3. Jalankan *server*

   Buka *Command Prompt*, navigasikan ke direktori tempat *file server.js* berada, dan jalankan perintah berikut: *node server.js*

```
C:\Windows\System32\my-secure-server>node server.js
Server berjalan di https://localhost
```

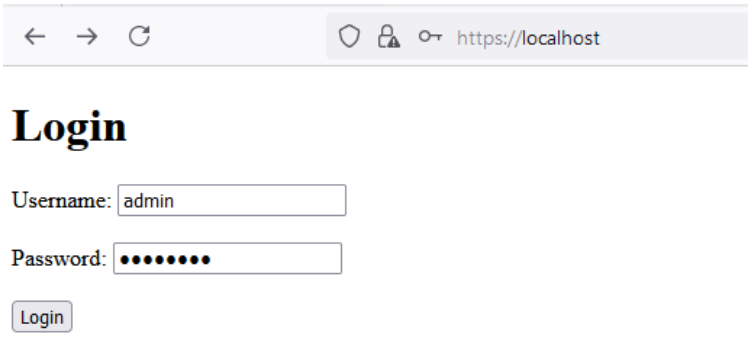
Gambar 4. 7 *Node server.js*

a. Gambar 4.7 merupakan tampilan *Command Prompt* pada saat menjalankan Node.js.

4.3 ANALISIS KERENTANAN

4.3.1 Permasalahan Sertifikat CA Tidak Valid

Sertifikat *self-signed* yang digunakan pada *server* tidak diterbitkan oleh *Certificate Authority* (CA) yang terpercaya. Ketika sertifikat *self-signed* digunakan, browser akan menampilkan peringatan keamanan yang menunjukkan bahwa sertifikat tidak dapat dipercaya. Hal ini dapat dilihat pada Gambar 4.8 dimana muncul peringatan keamanan pada browser.

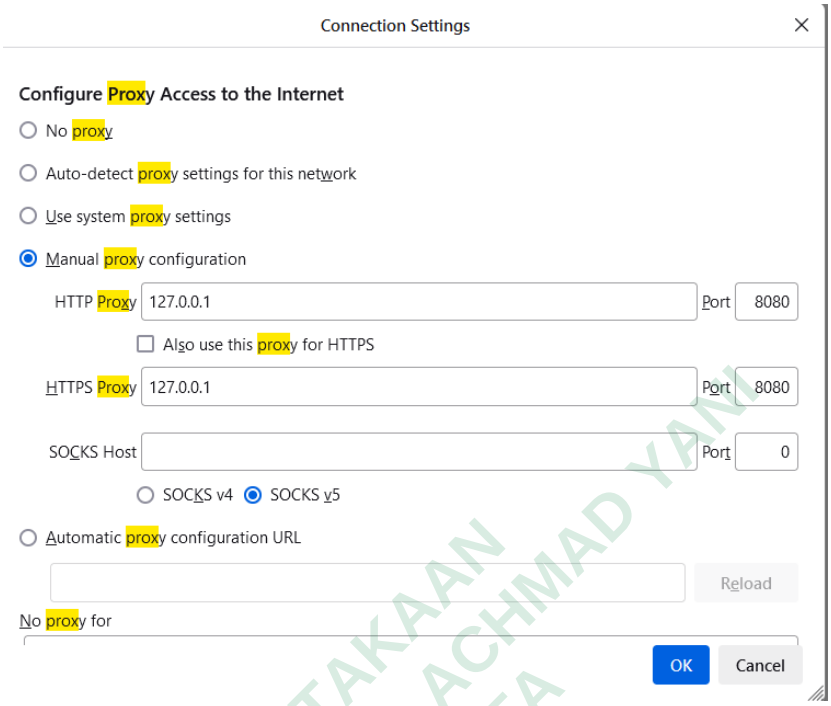


Gambar 4. 8 Tampilan *web server*

4.3.2 Instalasi Mitmproxy

Metode *sniffing* dilakukan untuk menganalisis lalu lintas jaringan dan mengidentifikasi potensi serangan. SSL *hijacking* dapat terjadi ketika berhasil mencegat dan memodifikasi lalu lintas HTTPS antara klien dan *server*. Berikut langkah-langkahnya:

1. Unduh *Python* dan instal ikuti petunjuk dengan benar.
2. Instal *mitmproxy*
    a. Buka *Command Prompt* dan jalankan perintah berikut untuk menginstal mitmproxy: *pip install mitmproxy*
    b. Setelah instalasi selesai, verifikasi bahwa mitmproxy telah terinstal dengan menjalankan perintah berikut: *mitmproxy –version*
    c. Jalankan *mitmproxy* dengan perintah berikut di *Command Prompt: mitmproxy*
    d. Atur browser *klien* untuk menggunakan *mitmproxy* sebagai proxy. *Setting proxy* HTTP dan HTTPS ke 127.0.0.1 dengan port 8080 (*port default mitmproxy*). *Setting* dapat dilihat pada Gambar 4.9.



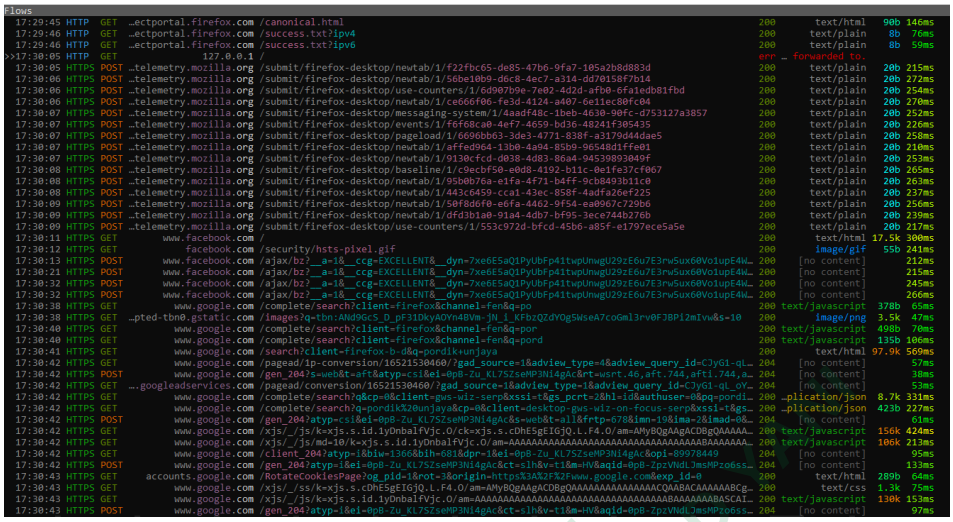
Gambar 4. 9 *Setting proxy browser user*

3. Saat menggunakan *mitmproxy* untuk mencegat lalu lintas HTTPS, *device user* perlu menginstal sertifikat CA *mitmproxy* di browser. Setelah menjalankan *mitmproxy*, buka *http://mitm.it* di browser *user* dan ikuti petunjuk untuk mengunduh dan menginstal sertifikat CA.

4.3.3 Simulasi Serangan SSL Hijacking Menggunakan Metode Sniffing

Tujuan dari simulasi ini adalah untuk menunjukkan bagaimana serangan dapat dilakukan dan untuk mengidentifikasi potensi kerentanan dalam penggunaan OpenSSL pada *server* Node.js dengan sertifikat *self-signed*. Berikut langkah-langkahnya:

1. Jalankan *server* Node.js.
2. Jalankan *mitmproxy* untuk menangkap lalu lintas.
3. Buka *browser user* yang sudah di *setting* dan buka halaman *https://localhost.*
4. Analasis paket data.



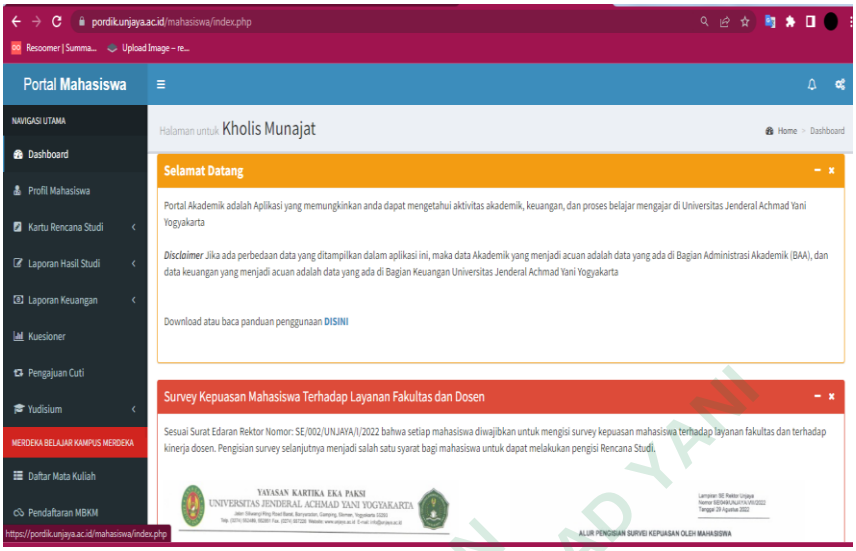
Gambar 4. 10 Tampilan *capture* data *mitmproxy*

a. Pada Gambar 4.10 data *https://localhost* tidak terdeteksi akibat dari Sertifikat *self-signed* yang digunakan pada *server* tidak diterbitkan oleh *Certificate Authority* (CA) yang terpercaya.

4.3.4 Uji Coba Serangan Pada Halaman Web Lain

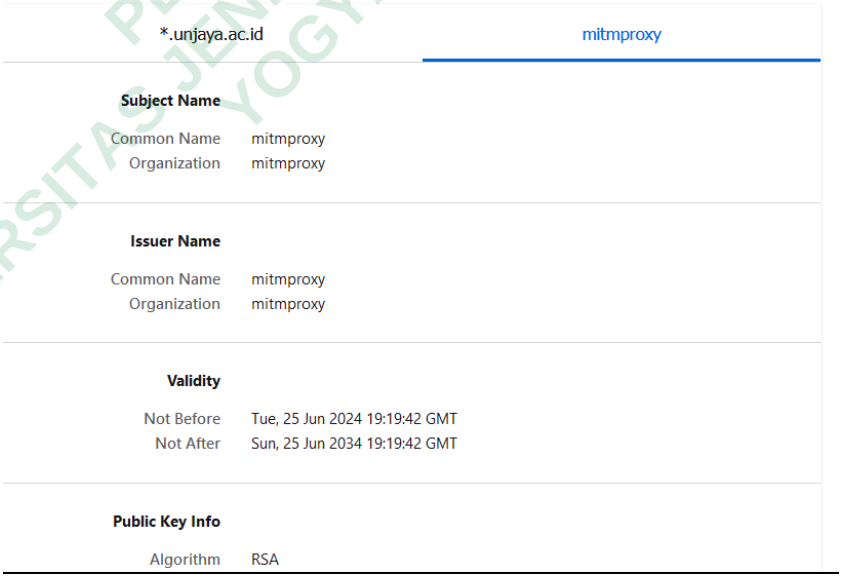
Setelah melakukan simulasi serangan SSL *hijacking* pada *server* lokal, langkah selanjutnya adalah melakukan uji coba serangan pada halaman *web* lain untuk mengidentifikasi potensi kerentanan lebih lanjut. Disini *user* menggunakan halaman web *https://pordik.unjaya.ac.id* sebagai uji coba. Berikut langkah-langkahnya:

1. Buka halaman *web https://pordik.unjaya.ac.id* dan *login*. Halaman *login* dapat dilihat pada Gambar 4.11.



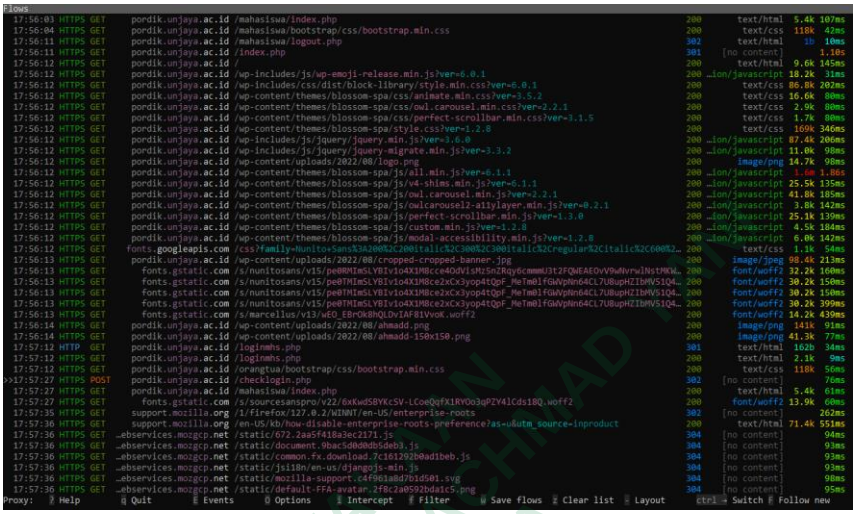
Gambar 4. 11 Halaman *https://pordik.unjaya.ac.id*

2. Cek Sertifikat SSL pada halaman *https://pordik.unjaya.ac.id* . Pada halaman tersebut menampilkan dua sertifikat SSL dikarenakan *browser* sudah di *setting* dengan *mitmproxy*. Hal tersebut dapat dilihat pada Gambar 4.12.



Gambar 4. 12 Sertifikat SSL *double*

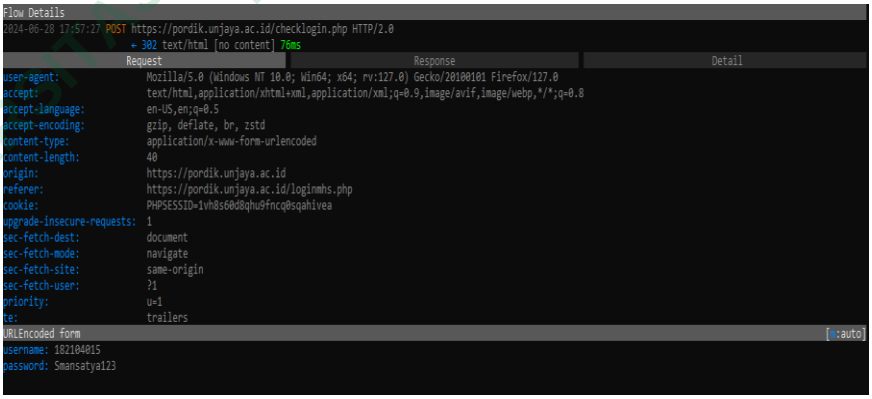
3. Cek *tools mitmproxy*



Gambar 4. 13 *Mitmproxy* membaca *https://pordik.unjaya.ac.id*

a. Pada Gambar 4.13 *tools mitmproxy* berhasil menangkap data *https://pordik.unjaya.ac.id* pada bagian HTTPS POST *https://pordik.unjaya.ac.id.*

4. Hasil



Gambar 4. 14 *Mitmproxy* menangkap *username* dan *password*

a. Pada Gambar 4.14 *tools mitmproxy* berhasil menangkap data *username* dan *password user* pada bagian HTTPS POST *https://pordik.unjaya.ac.id.*

4.4 EVALUASI HASIL UJI COBA

4.4.1 Hasil Pengamatan

Dengan menggunakan *mitmproxy*, lalu lintas HTTPS dari halaman *web* eksternal dapat dicegat dan dimodifikasi. Hal ini menunjukkan bahwa serangan MITM dapat dilakukan pada situs *web* yang menggunakan sertifikat valid jika penyerang berhasil menempatkan dirinya sebagai *proxy*.

4.4.2 Rekomendasi Keamanan

Serangan ini menyoroti pentingnya kepercayaan pada sertifikat SSL/TLS yang digunakan oleh situs *web*. Meskipun situs *web* menggunakan sertifikat yang valid, lalu lintas bisa dicegat oleh *proxy* dan data sensitif dapat diakses. Berikut rekomendasi dari penulis yang mungkin dapat meminimalisir ancaman serangan :

1. Gunakan mekanisme keamanan tambahan seperti *HTTP Strict Transport Security* (HSTS) untuk memastikan bahwa *browser* hanya melakukan koneksi ke *server* yang menggunakan HTTPS yang valid.
2. Gunakan alat keamanan seperti *Content Security Policy* (CSP) untuk mengurangi risiko modifikasi SSL oleh pihak ketiga.

4.4.3 Langkah Mitigasi

Untuk mengurangi risiko serangan SSL *hijacking* yang telah diidentifikasi dalam penelitian ini dikarnakan penulis berperan sebagai pengguna (*user*) dan penyerang (*attacker*), Salah satu poin utama adalah mitigasi serangan tidak memerlukan penggunaan VPN selama *browser* pengguna tidak dikonfigurasi untuk menggunakan *proxy mitmproxy* dan tidak menginstal sertifikat *mitmproxy*.