به نام خدا

دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)
دانشکده مهندسی کامپیوتر

برنامه‌نویسی وب

پروژه ۱

سید محمد حجازی حسینی

۹۷۳۳۰۲۰

بهار ۱۴۰۱

سوال)

سوال ۱)

باید سرور را روی localhost بالا بیاوریم. برای مثال کتابخانه‌ی flask در python یک سرور HTTP راه اندازی می‌کند که روی localhost (127.0.0.1) یا (0.0.0.0) اجرا می‌شود. localhost در واقع یک آدرس loopback در network adapter ما است. اگر روی (0.0.0.0) برای سرور یک پورت مشخص کنیم مثلا 8081؛ برای اینکه از طریق global هم به آن دسترسی داشته باشند، باید آن را روی router و سیستم خود باز کنیم و با استفاده از IP ای که سرویس دهنده‌ی اینترنت به ما داده است، به آن port دسترسی داشته باشیم. اما اگر port را باز نکرده باشیم، نمی‌توان به آن دسترسی داشت.

سوال ۲)

فرآیند TLS Termination در یک load balancer در جلوی سرورها انجام می‌شود. درخواست‌ها از بیرون به load balancer فرستاده می‌شوند. این درخواست‌ها به صورت encrypted هستند که برای پردازش روی سرور باید decrypted شوند. TLS Termination این کار را در load balancer انجام می‌دهد تا کار سرورها سبک شود و دیگر نیازی به اشغال منابع سرورها برای decryption نباشد. همچنین کنترل TLS connection های مختلف نیز به عهده‌ی load balancer خواهد بود.

گزارش)

ابتدا در docker یک container برای httpbin بالا می‌آوریم:

```
mohammad@ubuntu:~/Desktop/jwt/server$ docker run -p 80:80 kennethreitz/httpbin
[2022-03-26 14:13:23 +0000] [1] [INFO] Starting gunicorn 19.9.0
[2022-03-26 14:13:23 +0000] [1] [INFO] Listening at: http://0.0.0.0:80 (1)
[2022-03-26 14:13:23 +0000] [1] [INFO] Using worker: gevent
[2022-03-26 14:13:23 +0000] [9] [INFO] Booting worker with pid: 9
```

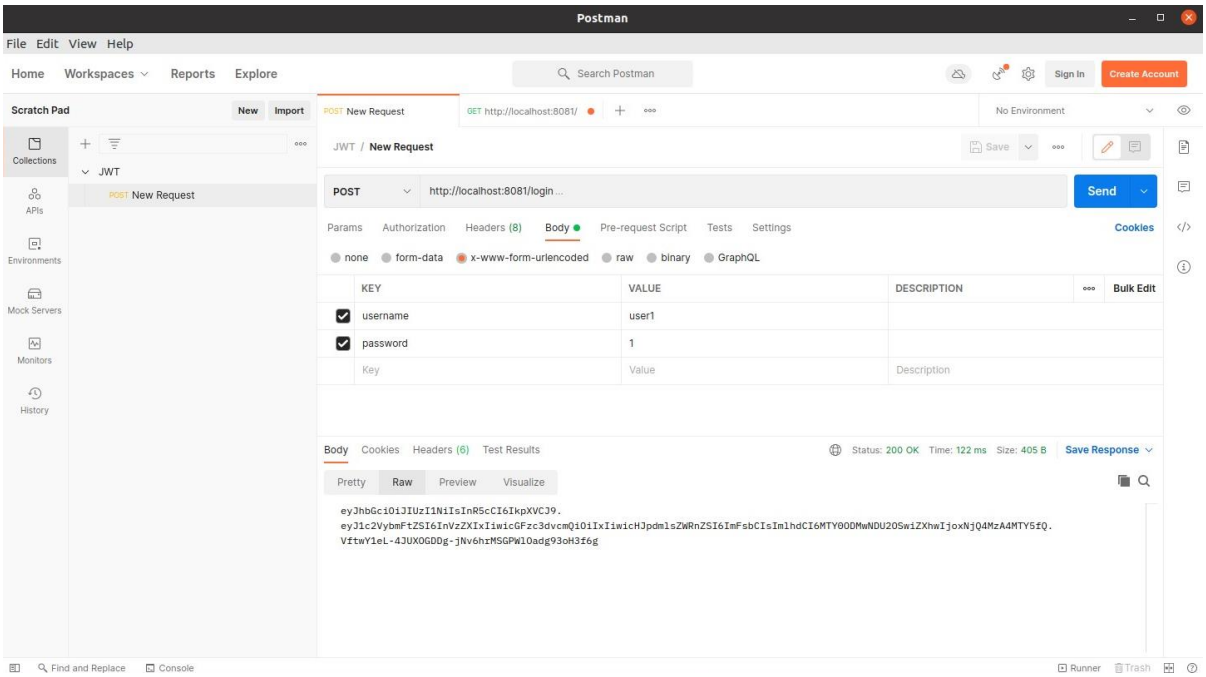
سپس کد سرور برای JWT reverse proxy را اجرا می‌کنیم. سرور proxy روی port 8081 کار گوش می‌کند:

```
mohammad@ubuntu:~/Desktop/jwt/server$ node server.js
LISTENING ON PORT 8081
```

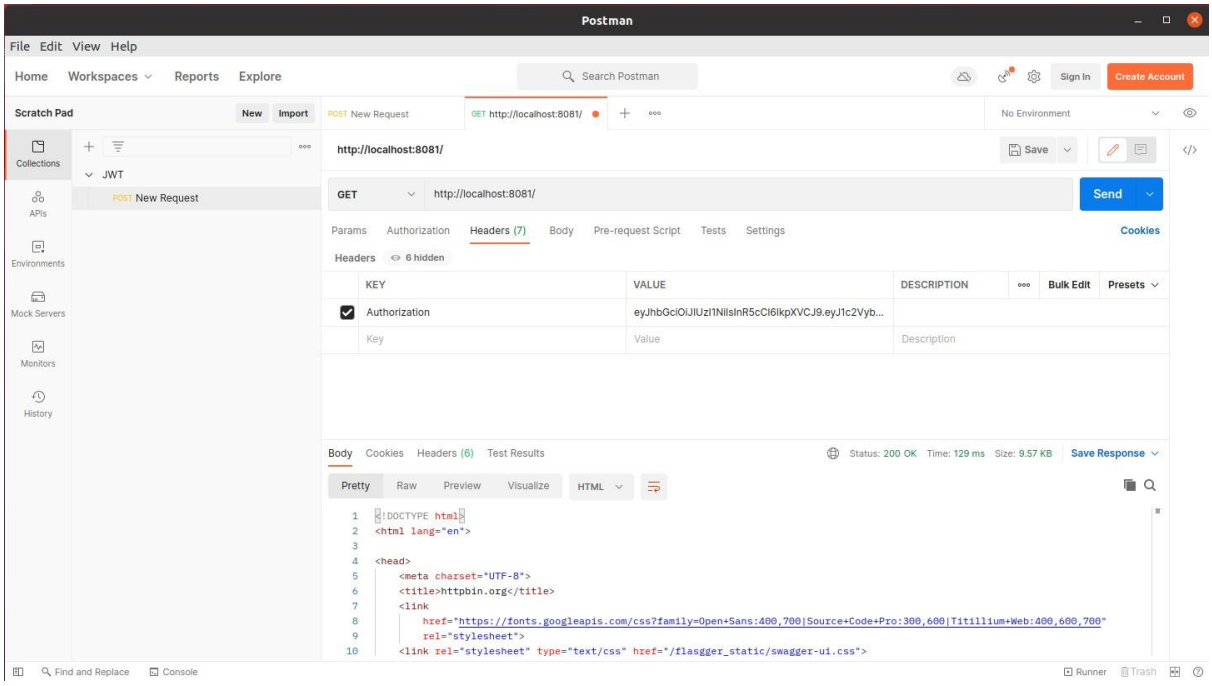
سپس با استفاده postman آن را تست می‌کنیم.

در سرور دو route وجود دارد؛ /login برای post و */ برای get است. ابتدا کاربر login می‌کند و یک JWT token را دریافت می‌کند. سپس با استفاده از آن در Authorization header می‌تواند درخواست get با path دلخواه در سرویس پشت proxy یعنی httbin ارسال کند. proxy درخواست get را برای container ارسال و پس از گرفتن جواب آن را به کاربر اولیه ارسال می‌کند.

درخواست POST در /login به این صورت است:

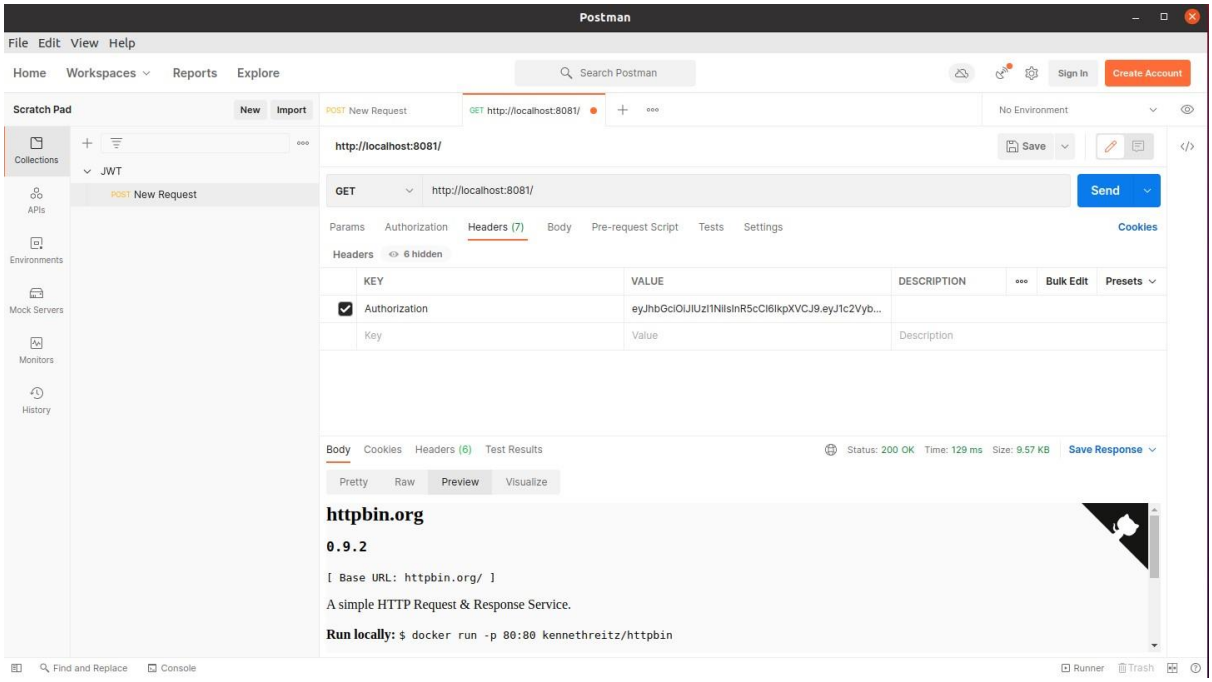


درخواست get پس از login با JWT token:



همانطور که در قسمت response body جواب را از proxy دریافت کرده‌ایم.

به صورت preview:



توضیح کد)

کد سرور reverse proxy با NodeJS نوشته شده است. منبع اصلی کد در اینجا قرار دارد که تغییراتی روی آن انجام شده است:

https://medium.com/@maison.moa/using-jwt-json-web-tokens-to-authorize-users-and-protect-api-routes-3e04a1453c3e

کد فایل server.js:

```
const express = require('express');
const app = express();
const port = 8081;

app.use(express.urlencoded({ extended: true }));
app.use(express.json());

require('./routes')(app);

app.listen(port, () => console.log(`LISTENING ON PORT ${port}`));
```

با استفاده از کتابخانه‌ی express یک API سرور را می‌سازیم. یک Object express ساخته شده و سپس با listen کار آن شروع می‌شود. خط require برای پاس دادن app به فایل index.js است که سپس به تمامی فایل‌های userRoutes.js پاس داده می‌شود. اینکار برای بهتر کنترل کردن کدها در فایل‌های مختلف و ایجاد یک hierarchy مناسب برای کد است.

فایل اصلی همان userRoutes.js است که در آن مسیر post به این صورت تعریف شده است:

```
app.post('/login', (req, res, next) => {
       const { body } = req;
       const { username } = body;
       const { password } = body;

       //checking to make sure the user entered the correct username/password combo
       let found = false;
       for (let i = 0; i < users.length; i++)
       {
           user = users[i];
           if(username === user.username && password === user.password) {
               //if user log in success, generate a JWT token for the user with a secret key
               jwt.sign(user, 'privatekey', { expiresIn: '1h' },(err, token) => {
                   if(err) { console.log(err) }
                   console.log('username: ' + user.username + ' password: ' + user.password);
                   res.send(token);
               });
               found = true;
               break;
           }
       }
       if (found !== true){
           console.log('ERROR: Could not log in');
           res.sendStatus(401);
       }
   })
```

یک فایل به اسم userList.js در پوشه‌ی users نوشته شده است که به این صورت است:

```
module.exports = [
    {
        username: 'user1',
        password: '1',
        priviledge: 'all'
    },
    {
        username: 'user2',
        password: '2',
        auth: 'all'
    },
    {
        username: 'user3',
        password: '3',
        auth: 'all'
    },
];
```

در آن لیستی از افرادی با درجه‌ی authorization آن‌ها وجود دارد که در userRoutes.js استفاده می‌شود. در واقع کاربران باید در یک database ذخیره شوند اما در این پروژه هدف ایجاد خود reverse proxy بوده است؛ بنابراین این روش جایگزین انتخاب شده است. همچنین فقط یک سطح دسترسی all وجود دارد که به کل سایت httpin دسترسی خواهند داشت. اگر کاربر در این لیست نباشد یا password اشتباه بزند، سرور به آن error برمی‌گرداند و اگر درست بود JWT token را برمی‌گردانیم.

برای مسیر get به این صورت است:

```
//This is a protected route
    app.get('/*', checkToken, (req, res) => {
        //verify the JWT token generated for the user
        jwt.verify(req.token, 'privatekey', (err, authorizedData) => {
            if(err){
                //If error send Forbidden (403)
                console.log('ERROR: Could not connect to the protected route');
                res.sendStatus(403);
            } else {
                //If token is successfully verified, we can send the autorized data
                const url = 'http://0.0.0.0' + req.originalUrl.slice(1) + ':80';
                const priviledge = authorizedData.priviledge;
                if (priviledge === 'all')
                {
                    const request = new XMLHttpRequest();
                    console.log('URL: ' + url)
                    request.open("GET", url, true);
                    request.send(null);
                    request.onreadystatechange = function() {
                        if (request.readyState == 4)
                            res.send(request.responseText);
                    };
                }
                else {
                    res.json({
                        message: 'unauthorised'
                    });
                    console.log('ERROR: Connected to unauthorised route');
                }
            }
        })
    });
```

ابتدا JWT token را verify می‌کنیم. اگر privilege یا سطح دسترسی all باشد، یک request به 0.0.0.0/path:80 می‌زنیم که سرور httpbin است. خود path توسط کاربر در API مشخص می‌شود که می‌تواند همان root باشد. سپس جواب از httpbin را به کاربر برمی‌گردانیم.

به این صورت JWT Reverse Proxy را با ساده‌ترین امکانات ساخته‌ایم. نکته مهم این است که باید port 8081 را در بیرون از ماشین و router باز کنیم تا فقط از این طریق به httbin دسترسی داشته باشیم.