2020第二届网鼎杯 青龙组部分writeup

原创

L.o.W 于 2020-05-11 08:38:00 发布 1547 收藏 2

分类专栏： CTF WriteUp 文章标签： 逆向 CTF 网鼎杯

目录

Reverse
jocker
signal
bang

Reverse

jocker

前面是一个假的flag，直接跳过omg函数
下面这段代码把0x401500-0x4015ba都亦或了0x41（也就是说在运行时修改了text段的encrypt函数），从而得到真正的encrypt函数

```
00401807   8B45 F4          mov eax,dword ptr ss:[ebp-0xC]
0040180A   05 00154000      add eax,jocker.00401500
0040180F   8B55 F4          mov edx,dword ptr ss:[ebp-0xC]
00401812   81C2 00154000    add edx,jocker.00401500
00401818   0FB612           movzx edx,byte ptr ds:[edx]
0040181B   83F2 41          xor edx,0x41
0040181E   8810             mov byte ptr ds:[eax],dl
00401820   8345 F4 01       add dword ptr ss:[ebp-0xC],0x1
00401824   817D F4 BA0000   cmp dword ptr ss:[ebp-0xC],0xBA
0040182B ^ 7E DA            jle short jocker.00401807
```

这段代码会把输入与字符串`hahahaha_do_you_find_me?`亦或，如果等于0x403040的那一串值，就通过

```
0040152E   8B55 E4          mov edx,dword ptr ss:[ebp-0x1C]
00401531   8B45 08          mov eax,dword ptr ss:[ebp+0x8]
00401534   01D0             add eax,edx
00401536   0FB610           movzx edx,byte ptr ds:[eax]
00401539   8B45 E4          mov eax,dword ptr ss:[ebp-0x1C]
0040153C   05 12404000      add eax,jocker.00404012          ASCII "hahahaha_do_you_find_me?"
00401541   0FB600           movzx eax,byte ptr ds:[eax]
00401544   31D0             xor eax,edx
00401546   0FBED0           movsx edx,al
00401549   8B45 E4          mov eax,dword ptr ss:[ebp-0x1C]
0040154C   8B4485 94        mov eax,dword ptr ss:[ebp+eax*4-0x6C]
00401550   39C2             cmp edx,eax
00401552   74 1F            je short jocker.00401573
00401554   C70424 0040400   mov dword ptr ss:[esp],jocker.00404000  ASCII "wrong ~~"
0040155B   E8 E0130000      call <jmp.&msvcrt.puts>
00401560   C745 E0 000000   mov dword ptr ss:[ebp-0x20],0x0
00401567   C70424 0000000   mov dword ptr ss:[esp],0x0
0040156E   E8 AD130000      call <jmp.&msvcrt.exit>
00401573   8345 E4 01       add dword ptr ss:[ebp-0x1C],0x1
00401577   837D E4 12       cmp dword ptr ss:[ebp-0x1C],0x12
0040157B ^ 7E B1            jle short jocker.0040152E
```

这串是0x403040:[0x0e, 0x0d, 9, 6, 0x13, 5, 0x58, 0x56, 0x3e, 6, 0x0c, 0x3c, 0x1f, 0x57, 0x14, 0x6b, 0x57, 0x59, 0x0d]
长度为19
我们求出前19个为flag{d07abccf8a410c
然后我们进入final函数，发现压进去5个byte，猜测为剩下的flag

```
040159A   55                push ebp
040159B   89E5              mov ebp,esp
040159D   83EC 28           sub esp,0x28
04015A0   C645 EB 25        mov byte ptr ss:[ebp-0x15],0x25
04015A4   C645 EC 74        mov byte ptr ss:[ebp-0x14],0x74
04015A8   C645 ED 70        mov byte ptr ss:[ebp-0x13],0x70
04015AC   C645 EE 26        mov byte ptr ss:[ebp-0x12],0x26
04015B0   C645 EF 3A        mov byte ptr ss:[ebp-0x11],0x3A
04015B4   C70424 0000000    mov dword ptr ss:[esp],0x0
```

由于剩下的flag中最后一个一定是}，我们用}(0x7d)和0x3a亦或，得到0x47
然后用0x47和剩下的4个亦或即可得到flag
最终Exp:

```
'''fake flag
s = [0x66, 0x6b, 0x63, 0x64, 0x7f, 0x61, 0x67, 0x64, 0x3b, 0x56, 0x6b, 0x61, 0x7b, 0x26, 0x3b, 0x50, 0x63, 0x5f,
 0x4d, 0x5a, 0x71, 0x0c, 0x37, 0x66]
flag = ''
for i in range(24):
 if i % 2 == 1:
  s[i] += i
 else:
  s[i] ^= i

 flag += chr(s[i])

print flag
'''
part1 = [0x0e, 0x0d, 9, 6, 0x13, 5, 0x58, 0x56, 0x3e, 6, 0x0c, 0x3c, 0x1f, 0x57, 0x14, 0x6b, 0x57, 0x59, 0x0d, 0
x25, 0x74, 0x70, 0x26, 0x3a]
part1_s = 'hahahaha_do_you_find_me?'
flag = ''
for i in range(19):
 flag += chr(part1[i]^ord(part1_s[i]))
guess = ord('}') ^ 0x3a
for i in range(19, 24):
 flag += chr(part1[i]^guess)

print flag
```

signal

这题有点类似虚拟机吧，反正就是操作码+操作数，F5还是很清晰的

人肉翻译结果如下：

> 0A 00 00 00 read

> 04 00 00 00 v5 = 0x10^input[0]
> 10 00 00 00 pass
> 08 00 00 00 input[0] = v5
> 03 00 00 00 v5 = input[0] - 5
> 05 00 00 00 pass
> 01 00 00 00 check[0]=v5

> 04 00 00 00 v5 = 0x20^input[1]
> 20 00 00 00 pass
> 08 00 00 00 input[1]=v5
> 05 00 00 00 v5 = 3 * input[1]
> 03 00 00 00 pass
> 01 00 00 00 check[1]=v5

> 03 00 00 00 v5 = input[2] - 2
> 02 00 00 00 pass
> 08 00 00 00 input[2] = v5
> 0B 00 00 00 v5 = input[2] - 1
> 01 00 00 00 check[2] = v5

```
0C 00 00 00 v5 = input[3] + 1
08 00 00 00 input[3] = v5
04 00 00 00 v5 = 4 ^ input[3];
04 00 00 00
01 00 00 00 check[3] = v5
```

```
05 00 00 00 v5 = 3 * input[4]
03 00 00 00
08 00 00 00 input[4] = v5
03 00 00 00 v5 = input[4] - 0x21
21 00 00 00
01 00 00 00 check[4] = v5
```

```
0B 00 00 00 v5 = input[5] - 1
08 00 00 00 input[5] = v5
0B 00 00 00 v5 = input[5] - 1
01 00 00 00 check[5] = v5
```

```
04 00 00 00 v5 = 9 ^input[6]
09 00 00 00
08 00 00 00 input[6] = v5
03 00 00 00 v5 = input[6] - 0x20
20 00 00 00
01 00 00 00 check[6] = input[6]
```

```
02 00 00 00 v5 = 0x51 + input[7]
51 00 00 00
08 00 00 00 input[7] = v5
04 00 00 00 input[7] ^= 0x24
24 00 00 00
01 00 00 00 check[7] = input[7]
```

```
0C 00 00 00 v5 = input[8] + 1
08 00 00 00
0B 00 00 00
01 00 00 00 same
```

```
05 00 00 00 v5 = input[9] * 2
02 00 00 00
08 00 00 00
02 00 00 00 input[9] + 0x25
25 00 00 00
01 00 00 00
```

```
02 00 00 00 input[10] + 0x36
36 00 00 00
08 00 00 00
04 00 00 00 input[10] ^ 0x41
41 00 00 00
01 00 00 00
```

```
02 00 00 00 v5 = input[11] + 0x20
20 00 00 00
08 00 00 00
05 00 00 00 input[11] * 1
01 00 00 00
01 00 00 00
```

```
05 00 00 00 v5 = input[12] * 3
03 00 00 00
08 00 00 00
02 00 00 00 input[12] + 0x25
25 00 00 00
01 00 00 00
```

```
04 00 00 00 v5 = input[13] ^ 9
09 00 00 00
08 00 00 00
03 00 00 00 input[13] - 0x20
20 00 00 00
01 00 00 00
```

```
02 00 00 00 v5 = input[14] + 0x41
41 00 00 00
08 00 00 00
0C 00 00 00 input[14] + 1
01 00 00 00
```

操作完之后，操作码7就是check了，根据check的值和上面翻译的操作来逆向，exp如下：

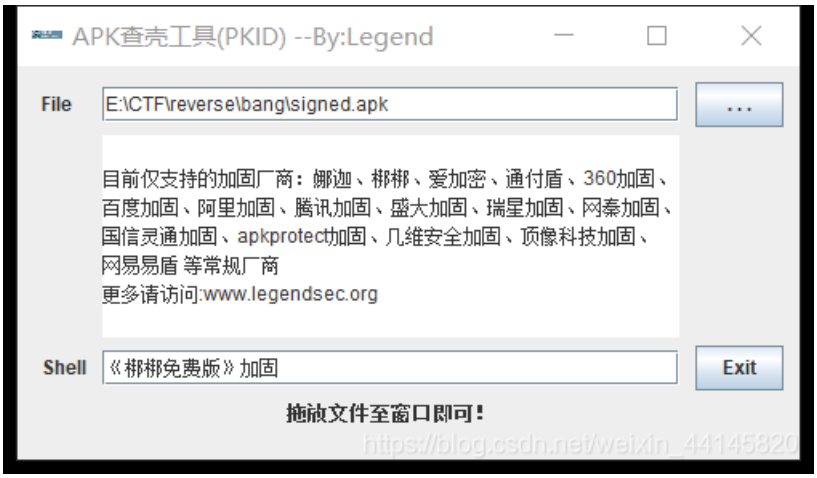
```
res = [0x22,0x3f,0x34,0x32,0x72,0x33,0x18,0xFFFFFFA7,0x31,0xFFFFFFF1,0x28,0xFFFFFF84,0xFFFFFFC1,0x1e,0x7a]
flag = 'flag{'
flag += chr((res[0]+5)^0x10)
flag += chr((res[1]/3)^0x20)
flag += chr(res[2]+3)
flag += chr((res[3]^4)-1)
flag += chr((res[4]+0x21)/3)
flag += chr(res[5]+2)
flag += chr((res[6]+0x20)^9)
flag += chr(((res[7]&0xff)^0x24)-0x51)
flag += chr((res[8]))
flag += chr(((res[9]&0xff)-0x25)/2)
flag += chr((res[10]^0x41)-0x36)
flag += chr(((res[11]&0xff)-0x20))
flag += chr(((res[12]&0xff)-0x25)/3)
flag += chr((res[13]+0x20)^9)
flag += chr((res[14]-0x42))
flag += '}'
print flag
```

至于提取操作码和操作数，这里给一个IDA python脚本供大家参考

```
addr = 0x403040
offset = 0x403208-addr
ans = []
for i in range(offset/4):
 print(hex(get_wide_dword(addr+i*4)))
```

bang

使用APK-Scan查壳，发现为梆梆加固



战队的大佬介绍了frida-dexdump的一个脱壳工具，非常方便使用，接下来我们就使用这个工具进行脱壳
首先我们打开模拟器（我这里用的是雷电）

adb shell连接上去

```
E:\CTF\tool\dnplayer2>adb shell
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
```

使用adb push 把frida-server文件放到/data/local/tmp目录下，server下载地址

```
E:\CTF\tool\dnplayer2>adb.exe push frida-server-x86 /data/local/tmp
5304 KB/s (26114852 bytes in 4.808s)
```

> 这里有个值得注意的地方，server一定要和你的模拟器或者手机架构一致，电脑上的模拟器（雷电什么的）一般是x86架构，而手机一般是arm的
> 如果server不对可能会出现：
> frida.NotSupportedError: unable to inject library into process without libc

chmod 755，然后启动server

```
root@aosp:/ # cd /data/local/tmp
cd /data/local/tmp
root@aosp:/data/local/tmp # ls -l
ls -l
-rw-rw-rw- root     root     26114852 2020-05-11 19:39 frida-server-x86
-rwxr-xr-x root     root     16245824 2020-05-10 20:42 frida-server32
drwxr-xr-x root     root              2020-05-11 18:25 re.frida.server
root@aosp:/data/local/tmp # ./frida-server-x86
./frida-server-x86
/system/bin/sh: ./frida-server-x86: can't execute: Permission denied
126|root@aosp:/data/local/tmp # chmod 755 frida-server-x86
chmod 755 frida-server-x86
root@aosp:/data/local/tmp # ./frida-server-x86
./frida-server-x86
WARNING: linker: ./frida-server-x86: unused DT entry: type 0x6ffffffe arg 0x1d24
```


adb forwad 设置转发，然后frida-ps查看启动的进程

```
E:\CTF\tool\dnplayer2>adb.exe forward tcp:27042 tcp:27042

E:\CTF\tool\dnplayer2>frida-ps -U
  PID  Name
----  -----------------------------------
 1402  adbd
 1902  android.process.acore
 2045  android.process.media
 2261  com.android.defcontainer
 1779  com.android.emu.coreservice
 1737  com.android.emu.inputservice
 1972  com.android.flysilkworm
 2093  com.android.flysilkworm:PushProcess
 2039  com.android.flysilkworm:keepLife
 2173  com.android.keychain
 1838  com.android.launcher3
 1803  com.android.phone
 2193  com.android.providers.calendar
 2360  com.android.settings
 2303  com.android.settings:superuser
 1672  com.android.systemui
 2383  com.example.how_debug
 2327  com.svox.pico
 1391  debuggerd
 1393  drmserver
 2546  frida-server-x86
 1385  healthd
    1  init
```


frida安装步骤

```
pip3 install frida
pip3 install frida-tools
```

用adb install signed.apk安装，运行app

然后使用工具把真正的dex文件导出，地址：https://github.com/hluwa/FRIDA-DEXDump

```
E:\CTF\tool\FRIDA-DEXDump-master>python main.py
Traceback (most recent call last):
  File "main.py", line 56, in <module>
    target = device.get_frontmost_application()
  File "C:\Users\ASUS\AppData\Roaming\Python\Python37\site-packages\frida\core.py", line 26, in wrapper
    return f(*args, **kwargs)
  File "C:\Users\ASUS\AppData\Roaming\Python\Python37\site-packages\frida\core.py", line 93, in get_frontmost_applicatio
n
    return self._impl.get_frontmost_application()
frida.NotSupportedError: unable to inject library into process without libc

E:\CTF\tool\FRIDA-DEXDump-master>python main.py
05-11/19:42:34 INFO [DEXDump]: found target [2383] com.example.how_debug
[DEXDump]: DexSize=0x1d60b8, SavePath=E:\CTF\tool\FRIDA-DEXDump-master/com.example.how_debug/0xa53cd000.dex
```


然后就是dex2jar转为jar文件

```
E:\CTF\tool\dex2jar-2.0>d2j-dex2jar.bat 0xa53cd000.dex
dex2jar 0xa53cd000.dex -> .\0xa53cd000-dex2jar.jar
Detail Error Information in File .\0xa53cd000-error.zip
Please report this file to http://code.google.com/p/dex2jar/issues/entry if possible.
```

最后使用jd-gui打开，发现flag

```
protected void onCreate(Bundle paramBundle)
{
  super.onCreate(paramBundle);
  setContentView(2131296284);
  mContext = getApplicationContext();
  jinit();
  Log.i("tag", "protect MainActivity onCreate");
  ((TextView)findViewById(2131165322)).setText(stringFromJNI1());
  paramBundle = (EditText)findViewById(2131165239);
  EditText localEditText = (EditText)findViewById(2131165238);
  ((Button)findViewById(2131165218)).setOnClickListener(new View.OnClickListener(paramBundle, localEditText)
  {
    public void onClick(View paramView)
    {
      paramView = this.val$et1.getText().toString();
      String str = this.val$et2.getText().toString();
      if (paramView.equals(str))
      {
        MainActivity.showmsg("user is equal passwd");
        return;
      }
      if ((paramView.equals("admin") & str.equals("pass71487")))
      {
        MainActivity.showmsg("success");
        MainActivity.showmsg("flag is flag{borring_things}");
        return;
      }
      MainActivity.showmsg("wrong");
    }
```