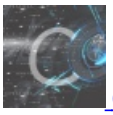
MOCTF新春欢乐赛-逆向writeup

原创

Jaken1223 于 2018-02-15 12:20:16 发布 789 收藏

分类专栏： CTFWritup

本文链接：https://blog.csdn.net/qq_35495684/article/details/79327954



1.easyre（50）

<给入门小伙子看的哈>
这是一个linux的运行文件。
IDA打开：

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char v4; // [sp+0h] [bp-70h]@1
  int v5; // [sp+6Ch] [bp-4h]@1

  v5 = 0;
  printf("input your flag:  ", argv, envp);
  __isoc99_scanf("%s", &v4);
  v5 = getflag(&v4);
  if ( !v5 )
    puts("The flag has been told you");
  return 0;
}
```


知道关键代码在getflag函数。
双击进入getflag

```
s1 = a1;
*(_QWORD *)dest = 0x7B6674636F6DLL;
memset(&v4, 0, 0x58uLL);
v5 = 0;
strcat(dest, acc);
strcat(dest, add);
*(_WORD *)&dest[strlen(dest)] = 0x2D;
strcat(dest, abb);
*(_WORD *)&dest[strlen(dest)] = 0x7D;
if ( !strcmp(s1, dest) )
{
  printf("great~", dest);
```


IDA会把字符串转为16进制。
现在你需要将这些转回来。对着16进制按R

```
*(_QWORD *)dest = '{ftcom';
memset(&v4, 0, 'X');
v5 = 0;
strcat(dest, acc);
strcat(dest, add);
*(_WORD *)&dest[strlen(dest)] = '-';
strcat(dest, abb);
*(_WORD *)&dest[strlen(dest)] = '}';
if ( !strcmp(s1, dest) )
{
  printf("great~", dest);
  result = 1LL;
}
```


现在代码的思路就很清楚了。
创建一个字符串=“}moctf”
strcat拼接acc，add，‘-’，abb
最后加上‘}’
可以双击acc看到acc的字符串。
然后拼接起来就是flag了。

2.CrackMe1(100)

先看程序运行现象：
flag一直是8位十六进制，而且一直在变，如果你C语言编程经验比较充足的话<错误经验-->
很快就能猜到这是地址。这里就get到提示，flag在这个地址所在的数据里。
<当然直接IDA看代码也是可以的>
再看代码：

```
 v13 = "2410488";
 v12 = 0;
 v11 = 0;
 i = 0;
 v7 = operator new(0x1Cu);
 v9 = v7;
 v8 = strlen(v13);
 for ( i = 0; i < v8; ++i )
 {
   v0 = 2 * v13[i] - 96;
   v9[i] = ((((BYTE4(v0) & 3) + (signed int)v0) >> 2) + 3) % 10;
 }
 v1 = v9;
 sub_401285((int)&unk_47BE90, "flag:");
 v2 = (void *)sub_401041(v1);
 v3 = sub_4011E0(v2, (int)sub_4010CD);
 v4 = (void *)sub_401285(v3, "æ€Žä¹ˆä¸€ç›´åœ¨å ¨???");
 sub_4011E0(v4, (int)sub_4010CD);
```

可以看出核心在for循环里，翻译for的加密部分，可以变成一个py

```
a="2410488"
for i in range(0,len(a)):
    v0=2*ord(a[i])-96
    k=((((v0 & 3)+ v0) >> 2) + 3) % 10
    print k
```

这里有一个小坑，96 代表的是‘0’*2，很明显v0为一个数字的两倍，(v0 & 3)其实是多余的，通常为0，但是当a[i]为奇数时，就会=2，导致得出错误的flag。
IDA反编译也是会出错的兄弟。
把(v0 & 3)去掉或者去掉它的括号，就是正确答案了。
下面是正确的py：

```
a="2410488"
for i in range(0,len(a)):
    v0=2*ord(a[i])-96
    k=(((v0 & 3+ v0) >> 2) + 3) % 10
    print k
```

有兴趣的可以探究下其中的原因~。.
IDA后面的代码其实就是输出这个加密完数据的地址，也可以不用看懂，可以把加密完的加上moctf{}后尝试提交一下。
就。。。通过了,不愧是100的简单题。

3.我的VIP呢？（150，apk出题人不是我，代写）

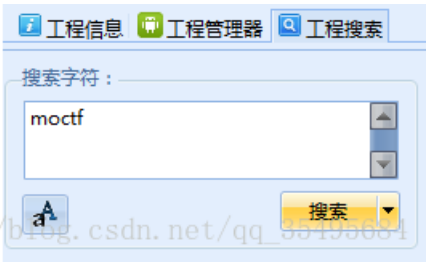
用Android Killer反编译。
点击MainActivity.smali，右键查看源码。

```
package com.example.crackme;
```

```
import android.app.Activity;
import android.app.AlertDialog.Builder;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity
  extends Activity
  implements View.OnClickListener
{
  private TextView a;
  private Button b;
  private Button c;

  private int a()
  {
    return getSharedPreferences("data", 0).getInt("points", 0);
  }

  public void onClick(View paramView)
  {
    switch (paramView.getId())
    {
    default:
      return;
    case 2131296336:
      int i = a();
      paramView = new AlertDialog.Builder(this);
      paramView.setTitle("激活vip");
      if (i < 30)
      {
        paramView.setMessage("目前你拥有积分：" + i + ",你的积分不够啊，当然，你不一定要用积分");
        paramView.setNegativeButton("好吧", new a(this));
      }
      for (;;)
      {
        paramView.show();
        return;
        paramView.setMessage("成为vip用户需要消耗30点积分哦，决定了吗？");
        paramView.setPositiveButton("升级", new b(this));
        paramView.setNegativeButton("退出", new c(this));
      }
    }
    startActivity(new Intent(this, VipFunction.class));
  }

  protected void onCreate(Bundle paramBundle)
  {
    super.onCreate(paramBundle);
    paramBundle = getSharedPreferences("share", 0);
    boolean bool = paramBundle.getBoolean("isFirstRun", true);
    paramBundle = paramBundle.edit();
    if (bool)
    {
```

```
        SharedPreferences.Editor localEditor = getSharedPreferences("data", 0).edit();
        localEditor.putInt("points", 0);
        localEditor.commit();
        paramBundle.putBoolean("isFirstRun", false);
        paramBundle.commit();
      }
      setContentView(2130903065);
      this.a = ((TextView)findViewById(2131296335));
      this.b = ((Button)findViewById(2131296336));
      this.c = ((Button)findViewById(2131296337));
      this.b.setOnClickListener(this);
      this.c.setOnClickListener(this);
      int i = getSharedPreferences("data", 0).getInt("points", 0);
      if (i > 0) {
        this.a.setText("普通用户");
      }
      if (i < 0)
      {
        this.c.setVisibility(0);
        this.a.setText("vip用户");
        this.b.setVisibility(4);
      }
    }
  }
```

代码大致的意思是要你是VIP账户才可以进行操作，而一进来的你只有普通用户的权限。
出题人的想法应该是要解题人静态修改判断代码，让自己变成VIP，然后弹出VIP界面，得到flag。
但是界面就在res文件夹的layout里，所以可以直接找到含有flag的vipfunction.xml。
或者更简单的直接搜索moctf字符串



就可以找到flag了。
这可能是出题人的一个失误，你可试下改判断的，学习下咯~

4.哇有毒吧（200）

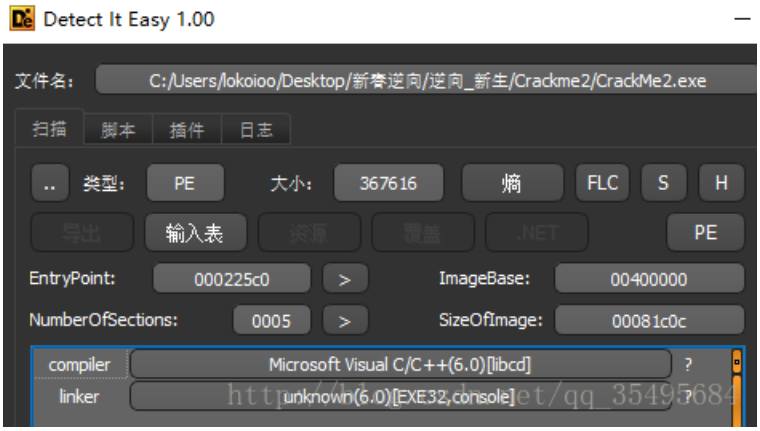
哇，这个是真的有毒，纯脑洞题，就看你大不大胆。
老规矩，先用Android Killer反编译，发现反编译失败。
就把apk文件加上zip后缀，解压。
用Android 逆向助手把classes.dex文件转为.jar文件就可以看软件的逻辑了
大致如下：

```
EditText Pass;

public void check(String paramString1, String paramString2)
{
  if ((paramString1.equals("MQLSY_s")) && (paramString2.equals("66666")))
  {
    Toast.makeText(this, "bW9jdGZ7dGhlX0NURl9JU18/fQ==", 0).show();
    return;
  }
  if ((paramString1.equals("mqlsys")) && (paramString2.equals("23333")))
  {
    Toast.makeText(this, "bW9jdGZ7ZmFsc2U/fQ==", 0).show();
    return;
  }
  if ((paramString1.equals("")) && (paramString2.equals("")))
  {
    Toast.makeText(this, "哇，这你都敢尝试，厉害厉害", 0).show();
    return;
  }
  if ((paramString1.equals("MQL")) && (paramString2.equals("2018")))
  {
    Toast.makeText(this, "bW9jdGZ7dGhpc19pc24ndF9mbGFnfQ==", 0).show();
    return;
  }
  if ((paramString1.equals("admin")) && (paramString2.equals("admin")))
  {
    Toast.makeText(this, "登录成功", 0).show();
    return;
  }
  if ((paramString1.equals("MQL")) && (paramString2.equals("666")))
  {
    Toast.makeText(this, "bW9jdGZ7dHJ1ZT99", 0).show();
    return;
```


题目的意思是根据你输入的用户名和密码的不同会输出不同的base64密文。。。
随便尝试了下，bW9jdGZ7dHJ1ZT99 ->moctf{true?}
试了下没通过，就在想应该不会这么简单吧flag发出来，就再分析其他的文件无果。。。
最后在询问一位大佬后才知道flag就是第一个的base64：bW9jdGZ7dGhlX0NURl9JU18/fQ== 的解密。
哇~这题真难到我了~~。*
（果然脑洞题就是不一样~，脑子有点死板了，要改下~~）

4.CrackMe2（200）

先看下软件的状态属性



可以看出这个是用VC6.0编写的32位程序。
尝试运行下，没有什么提示。
用32的IDA打开<如果用64位的也可以打开但是不能F5分析>，根据程序的运行用关键字搜索找到关键函数的位置。
按F5查看，大致如下。

```
v8 = "10<1<>;?8:%w!##&#q./,x(,((";
v3 = (char *)operator new(0x64u);
v7 = v3;
v2 = (char *)operator new(0x64u);
v6 = v2;
printf("请输入flag: ");
sub_40103C((int)&dword_47AF00, (int)v7);
v5 = strlen(v7);
for ( i = 0; i < v5; ++i )
  v6[i] = sub_4010CD(v7[i]);
v6[i] = 0;
if ( !strcmp(v6, v8) )
{
  sub_40117C((int)&unk_47AE70, "Your get the flag,but it ....");
  sub_401131(sub_40107D);
}
else
{
  sub_40117C((int)&unk_47AE70, "Not it..");
}
```

大致意思是根据输入的flag进行加密后与保存的密文“10<1<>;?8:%w!##&#q./,x(,((”比对，如果一样就输出get flag…

要满足v6=v8，v8是固定的。v6是加密的。
v7是你输入的字符串，所以解题的关键在for循环中的sub_4010CD函数了。
进去跟下：

```
char __cdecl sub_401570(char a1)
{
  return dword_475DC0++ ^ a1;
}
```

对dword_475DC0++后与输入的a1异或。
双击查看dword_475DC0的内存。

```
.data:00475DBF                 db    0
.data:00475DC0 dword_475DC0    dd 6
.data:00475DC0
```

是6
那么加密的代码的逻辑就很清晰了。

```
lm=6
key="........"  //length is 26
for i in range(0,26):
    lm=lm+1
    c=chr((ord(key[i]))^lm)
    print c
```

逆向来大概是这样：

```
v1 = "10<1<>;?8:%w!##&#q./,x(,(("
v8 = len(v1)
lm=6
key=""
for i in range(0,v8):
    c=chr((ord(v1[i]))^lm)
    key=key+c
    lm=lm+1
print key
```

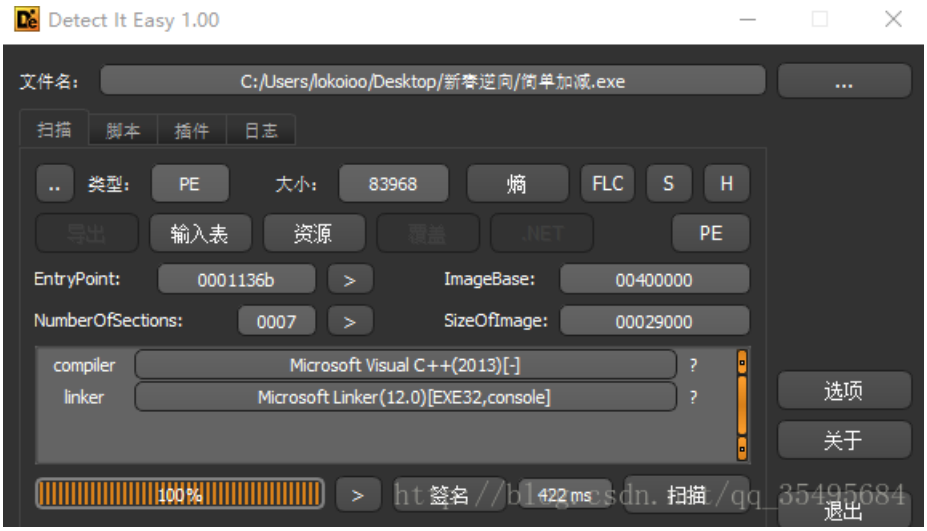
运行后可以得到以下的可视的字符串：
77486572655f30735f666c4167
验证下也是对的。

当然，通过后又有提示but it... 说明这不是最后的flag。。。
很明显是一串十六进制，把它转为aclii代码。。。
就可以获得flag了~

或者用碰撞也行可以拿到flag，从加密的代码逻辑可以看出加密方法是逐个加密的。
也就是相互之间的相关性并不大，那就可以用碰撞，有兴趣的可以自己去尝试下。

4.简单加减（300）

老规矩，先运行程序，秒退，没有什么信息。
用DIE看下软件信息。



用32位的IDA打开。
找到main，F5。

```
```

```
unsigned __int64 __cdecl main()
{
  char *v0; // eax@3
  char *v1; // eax@6
  std::basic_ostream<char,std::char_traits<char> > *v2; // eax@7
  std::basic_ostream<char,std::char_traits<char> > *v3; // eax@7
  unsigned int v4; // edx@7
  unsigned int v6; // [sp-4h] [bp-1B0h]@1
  std::basic_string<char,std::char_traits<char>,std::allocator<char> > *_Right; // [sp+Ch] [bp-1A0h]@1
  std::basic_string<char,std::char_traits<char>,std::allocator<char> > result; // [sp+14h] [bp-198h]@3
  std::basic_string<char,std::char_traits<char>,std::allocator<char> > v9; // [sp+38h] [bp-174h]@6
  std::basic_string<char,std::char_traits<char>,std::allocator<char> > v10; // [sp+5Ch] [bp-150h]@7
  unsigned int v11; // [sp+80h] [bp-12Ch]@7
  std::basic_string<char,std::char_traits<char>,std::allocator<char> > flag; // [sp+14Ch] [bp-60h]@1
  std::basic_string<char,std::char_traits<char>,std::allocator<char> > res; // [sp+170h] [bp-3Ch]@1
  int i; // [sp+194h] [bp-18h]@1
  unsigned int v15; // [sp+19Ch] [bp-10h]@1
  int v16; // [sp+1A8h] [bp-4h]@1
  int savedregs; // [sp+1ACh] [bp+0h]@1

  memset(&_Right, 0xCCu, 0x194u);
  v15 = (unsigned int)&savedregs ^ __security_cookie;
  v6 = (unsigned int)&savedregs ^ __security_cookie;
  std::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string<char,std::char_trai
    &res,
    "moctf{");
  v16 = 0;
  std::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string<char,std::char_trai
  LOBYTE(v16) = 1;
  std::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator=(&flag, 99);
  for ( i = 0; i < 7; ++i )
  {
    v0 = std::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](&flag, 0);
    std::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator=(&flag, *v0 + 3);
    _Right = std::operator+<char,std::char_traits<char>,std::allocator<char>>(&result, &res, &flag);
    std::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator=(&res, _Right);
    std::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string<char,std::char_t
  }
  std::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator=(&res, byte_420C40);
  for ( i = 0; i < 10; ++i )
  {
    v1 = std::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](&flag, 0);
    std::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator=(&flag, *v1 - 2);
    _Right = std::operator+<char,std::char_traits<char>,std::allocator<char>>(&v9, &res, &flag);
    std::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator=(&res, _Right);
    std::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string<char,std::char_t
  }
  _Right = std::operator+<char,std::char_traits<char>,std::allocator<char>>(&v10, &res, "}");
  std::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator=(&res, _Right);
  std::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string<char,std::char_tra
  std::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator=(&res, byte_420C40);
  v2 = std::operator<<<std::char_traits<char>>(std::cout, "flag is :");
  v3 = std::operator<<<char,std::char_traits<char>,std::allocator<char>>(v2, &res);
  std::basic_ostream<char,std::char_traits<char>>::operator<<(v3, std::endl<char,std::char_traits<char>
  v11 = 0;
  LOBYTE(v16) = 0;
  std::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string<char,std::char_tra
  v16 = -1;
  std::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string<char,std::char_tra
```

```
    return __PAIR__(v4, v11);
}
```

你可以看到F5后的伪代码看起来明显混乱了点，没有前面那么舒适了。
这是因为引用了一些库函数。

这题的逻辑其实并不难，主要考验的是抗干扰分析的能力。
逆向最重要的是要有耐心，兄弟。。。
将伪代码中的模板类函数变成自己熟悉点的函数。
可以根据这个学习下。
http://en.cppreference.com/w/cpp/string/basic_string/operator+
找一个你觉得最舒适的编辑器，把伪代码复制进去，按逻辑一一替换下。
这是替换后的代码：

```
unsigned __int64 __cdecl main()
{
  char *v0; // eax@3
  char *v1; // eax@6
  char *v2; // eax@7
  char *v3; // eax@7
  unsigned int v4; // edx@7
  unsigned int v6; // [sp-4h] [bp-180h]@1
  string *_Right; // [sp+Ch] [bp-1A0h]@1
  string result; // [sp+14h] [bp-198h]@3
  string v9; // [sp+38h] [bp-174h]@6
  string v10; // [sp+5Ch] [bp-150h]@7
  unsigned int v11; // [sp+60h] [bp-12Ch]@7
  char flag; // [sp+14Ch] [bp-40h]@1
  char res; // [sp+170h] [bp-3Ch]@1
  int i; // [sp+194h] [bp-18h]@1
  unsigned int v15; // [sp+19Ch] [bp-10h]@1
  int v16; // [sp+1A8h] [bp-4h]@1
  int savedregs; // [sp+1ACh] [bp+0h]@1
  memset(&_Right, 0xCCu, 0x194u);

  string(&res,"moctf{");
  v16 = 0;

  v16 = 1;
  flag=99;
  for ( i = 0; i < 7; ++i )
  {
    v0 = (&flag, 0);
    &flag, *v0 + 3;
    &result= &res+&flag;
    &res=&result;
  }
  &res=byte_420C40);
  for ( i = 0; i < 10; ++i )
  {
    v1 = (&flag, 0);
    &flag, *v1 - 2;
    (&v9= &res+ &flag);

    &res=&v9
  }
  (&v10= &res+"}");
  (&res=&v10);

  (&res=byte_420C40);
  v2 = (std::cout, "flag is :");
  v3 = (v2, &res);

}
```

化成python：

```
res="moctf{"

flag=99
for i in range(0,7):
    v0 =flag
    flag=v0 + 3
    result= res+str(chr(flag))
    res=result
#res=byte_420C40

for i in range(0,10):
    v1 = flag
    flag=v1 - 2
    v9= res+str(chr(flag))
    res=v9
v10=res+"}"
res=v10;

#res=byte_420C40)

print "flag is :"
print  res
```

执行下就获得flag了。
感兴趣的可以看下byte_420C40的内存数据是多少

```
.rdata:00420C40 ; char byte_420C40[4]
.rdata:00420C40 byte_420C40     db 4 dup(0)
.rdata:00420C40
```

定义4个0字节，意为空，其实这里想起到一个混淆的坑，不知道有没有坑到人~

还有一种解法是动态调试，获得每个flag的片段，有兴趣的可以自己学习下哈~

抱歉，最近几天一直在忙(其实主要电脑都被小辈们抢走了--。还有自己在浪~)，writeup晚了一点。
总的来说新春赛的逆向还是简单的，比较适合新人的哈哈，想入门的可以对着学习下哈，大家一起共勉呦~
祝大家新春快乐!