[XCTF-Reverse] 41-44

原创

石氏是时试 已于 2022-03-21 16:29:03 修改 41 收藏
分类专栏： CTF reverse 文章标签： reverse
于 2022-03-21 16:22:46 首次发布

41，SHCTF-2017_crackme

用北极星加的壳，去壳以后就一眼明白了，就是个异或

网上讲了些手工去壳的方法，没学会。直接用的工具。

```
  printf("Please Input Flag:");
  gets_s(&Buf, 0x2Cu);
  if ( strlen(&Buf) == 42 )
  {
    v4 = 0;
    while ( (*(&Buf + v4) ^ byte_402130[v4 % 16]) == dword_402150[v4] )
    {
      if ( ++v4 >= 42 )
      {
        printf("right!\n");
        goto LABEL_8;
      }
    }
    printf("error!\n");
LABEL_8:
    result = 0;
  }
```

```
#exeinfope 查壳：nsPack ver.3.x-4.1 reg by North Star  [ Size Of Code = 00 Kb - FIX IT ! NSTD ! ]

a = [0x12,    4,    8, 0x14, 0x24, 0x5C, 0x4A, 0x3D, 0x56, 0x0A,
     0x10, 0x67,    0, 0x41,    0,    1, 0x46, 0x5A, 0x44, 0x42,
     0x6E, 0x0C, 0x44, 0x72, 0x0C, 0x0D, 0x40, 0x3E, 0x4B, 0x5F,
      0x2,    1, 0x4C, 0x5E, 0x5B, 0x17, 0x6E, 0x0C, 0x16, 0x68, 0x5B, 0x12]
b = b'this_is_not_flag'

print(''.join([chr(b[i%16]^a[i]) for i in range(42)]))
#flag{59b8ed8f-af22-11e7-bb4a-3cf862d1ee75}
```

42，梅津美治郎

ida打开，没有啥保护很容易就过一第关，但第2关入口看不到，有个特殊的函数

```
strcpy(v8, "r0b0RUlez!");
dword_40AD94 = (int)&v9;
dword_40ADA0 = (int)&v49;
dword_40AD8C = (char *)&v40;
dword_40AD90 = (char *)&v31;
dword_40AD98 = (int)&v28;
lpProcName = (LPCSTR)&v17;
lpModuleName = (LPCSTR)&v24;
dword_40ADA4 = (char *)&v10;
sub_401500(0);
v3 = lpProcName;
v4 = GetModuleHandleA(lpModuleName);
v5 = (void (__stdcall *)(HMODULE, LPCSTR))GetProcAddress(v4, v3);
v5((HMODULE)1, (LPCSTR)sub_40157F);     //<----- 第一关通过后会进入sub_4015ea然后调用sub_40157f进入第2关
puts(dword_40AD8C);
scanf("%20s", &v7);
if ( !strcmp(&v7, v8) )
{
  puts("You passed level1!");
  sub_4015EA(0);
}
```

第2关只是个简单的异或，在比较前下断点，看内存

```
ebp+c:0x28f7b4->0x28fdcc->'u1nnf2lg'
```

得到第2个串，然后还原（两段用下划线连上，这个没有提示）

```
a1 = b'r0b0RUlez!'
a2 = b'u1nnf2lg'
a2 = [i^2 for i in a2]
print(a1 +b'_'+ bytes(a2))
#b'r0b0RUlez!_w3lld0ne'
#flag{r0b0RUlez!_w3lld0ne}
```

43，2019“嘉韦思杯”上海市网络安全邀请赛_76号

这个跟前边41动态调试的类似，只是不用调试。

一开始不能反编译，但大概能看明白，一直调用到sub_8048580 检查函数

这个函数是个大switch v2作序号，从0到N，序号和字符对应的就是那个字符。

两个小坑，第1个出现的10表示回车，这里的序号是13说明前边处理13个字符（逻辑上不大成立）

```
       case 10:                                 // 回车结束 13
         return v2 == 13 && v28 != 0;           // 总共13，12在这里return
```

第2个小坑就是，既然总共有13个那到第13个就该返回了，第13个字符有两种情况一种是返回的一种是不返回继续的，应该选返回的那个

```
        case 54:      //这个看上去正确，但没有返回，不选
          if ( v2 != 12 || !v35 )
            return 0;
          v2 = 13;
          continue;
......
        case 107:
          return v2 == 12 && v14 != 0;    //最后一位选这个
```

最后把这些连一起

```
a = bytes([48, 57, 118, 100, 102, 55, 119, 101, 102, 105, 106, 98, 107])
print(a)
#09vdf7wefijbk
#flag{09vdf7wefijbk}
```

44，Hack-you-2014_APK逆向2

这个题有点怪，这不是个apk是个.net的exe文件

用ILSpy打开有3个小函数。主程序就是把一串的字母逐个从第2个串里找偏移然后16进制打印出来

```
private static void Main(string[] args)
{
 string hostname = "127.0.0.1";
 int port = 31337;
 TcpClient tcpClient = new TcpClient();
 try
 {
  Console.WriteLine("Connecting...");
  tcpClient.Connect(hostname, port);
 }
 catch (Exception)
 {
  Console.WriteLine("Cannot connect!\nFail!");
  return;
 }
 Socket client = tcpClient.Client;
 string text = "Super Secret Key";
 string text2 = read();
 client.Send(Encoding.ASCII.GetBytes("CTF{"));
 string text3 = text;
 foreach (char x in text3)
 {
  client.Send(Encoding.ASCII.GetBytes(search(x, text2)));
 }
 client.Send(Encoding.ASCII.GetBytes("}"));
 client.Close();
 tcpClient.Close();
 Console.WriteLine("Success!");
}
```

第2个是这个偏移怎么计算

```
private static string search(char x, string text)
{
 int length = text.Length;
 for (int i = 0; i < length; i++)
 {
  if (x == text[i])
  {
   int num = i * 1337 % 256;
   return Convert.ToString(num, 16).PadLeft(2, '0');
  }
 }
 return "??";
}
```

关键是这第3个串，调用的read但并不是平常用的read而是自己写的一个，把程序本身读进来当字符串2

```
private static string search(char x, string text)
{
 int length = text.Length;
 for (int i = 0; i < length; i++)
 {
  if (x == text[i])
  {
   int num = i * 1337 % 256;
   return Convert.ToString(num, 16).PadLeft(2, '0');
  }
 }
 return "??";
}
```

看上去就行了

```
a = b"Super Secret Key"
text = open('4122e391e1574335907f8e2c4f438d0e.exe', 'rb').read()
b = ''.join([hex(text.index(i) * 1337 % 256)[2:].rjust(2, '0') for i in a])
print('CTF{'+b+'}')
#CTF{7eb67b0bb4427e0b43b40b6042a00b8e}
#CTF{7eb67b0bb4427e0b43b40b6042670b55}  <----？？
```

但运行结果与网上搜的差两个字符，不清楚为什么。