BUU easyre CrackRTF

原创

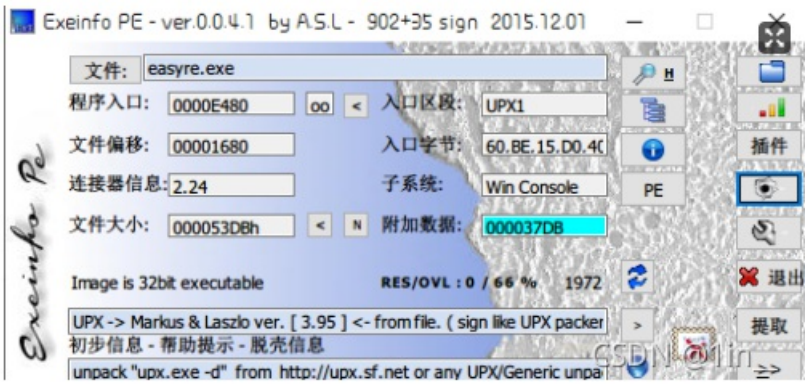
1in__ 于 2021-09-06 15:31:46 发布 27 收藏

分类专栏： SLsec 文章标签： python github c语言

本文链接：https://blog.csdn.net/duodu0/article/details/120136362

[ACTF新生赛2020]easyre

简单的UPX脱壳



获得信息：

32位

upx加密

代码分析

```
** int __cdecl main(int argc, const char **argv, const char **envp)**
**{**
**  char v4; // [esp+12h] [ebp-2Eh]**
**  char v5; // [esp+13h] [ebp-2Dh]**
**  char v6; // [esp+14h] [ebp-2Ch]**
**  char v7; // [esp+15h] [ebp-2Bh]**
**  char v8; // [esp+16h] [ebp-2Ah]**
**  char v9; // [esp+17h] [ebp-29h]**
**  char v10; // [esp+18h] [ebp-28h]**
**  char v11; // [esp+19h] [ebp-27h]**
**  char v12; // [esp+1Ah] [ebp-26h]**
**  char v13; // [esp+1Bh] [ebp-25h]**
**  char v14; // [esp+1Ch] [ebp-24h]**
**  char v15; // [esp+1Dh] [ebp-23h]**
**  int v16; // [esp+1Eh] [ebp-22h]**
**  int v17; // [esp+22h] [ebp-1Eh]**
**  int v18; // [esp+26h] [ebp-1Ah]**
**  __int16 v19; // [esp+2Ah] [ebp-16h]**
**  char v20; // [esp+2Ch] [ebp-14h]**
**  char v21; // [esp+2Dh] [ebp-13h]**
**  char v22; // [esp+2Eh] [ebp-12h]**
**  int v23; // [esp+2Fh] [ebp-11h]**
**  int v24; // [esp+33h] [ebp-Dh]**
**  int v25; // [esp+37h] [ebp-9h]**
**  char v26; // [esp+3Bh] [ebp-5h]**
**  int i; // [esp+3Ch] [ebp-4h]**
**  __main();**
**  v4 = 42;**
**  v5 = 70;**
**  v6 = 39;**
**  v7 = 34;**
**  v8 = 78;**
**  v9 = 44;**
**  v10 = 34;**
**  v11 = 40;**
**  v12 = 73;**
**  v13 = 63;**
**  v14 = 43;**
**  v15 = 64;**
**  printf("Please input:");**
**  scanf("%s", &v19);**
**  if ( (_BYTE)v19 != 65 || HIBYTE(v19) != 67 || v20 != 84 || v21 != 70 || v22 != 123 || v26 != 125 )**
**    return 0;**
**  v16 = v23;**
**  v17 = v24;**
**  v18 = v25;**
**  for ( i = 0; i <= 11; ++i )**
**  {**
**    if ( *(&v4 + i) != _data_start__[*((char *)&v16 + i) - 1] )**
**      return 0;**
**  }**
**  printf("You are correct!");**
**  return 0;**
**}**
** **
```

找到关键语句

```
for ( i = 0; i <= 11; ++i )
{
  if ( *(&v4 + i) != _data_start__[*((char *)&v16 + i) - 1] )
    return 0;
}
```

点进_data_start__ 查看字符串

```
7E 7D 7C 7B 7A 79 78 77  76 75 74 73 72 71 70 6F  ~}|{zyxwvutsrqpo
6E 6D 6C 6B 6A 69 68 67  66 65 64 63 62 61 60 5F  nmlkjihgfedcba`_
5E 5D 5C 5B 5A 59 58 57  56 55 54 53 52 51 50 4F  ^]\[ZYXWVUTSRQPO
4E 4D 4C 4B 4A 49 48 47  46 45 44 43 42 41 40 3F  NMLKJIHGFEDCBA@?
3E 3D 3C 3B 3A 39 38 37  36 35 34 33 32 31 30 2F  >=<;:9876543210/
2E 2D 2C 2B 2A 29 28 27  26 25 24 23 20 21 22 00  .-,+*)('&%$#·!".
```

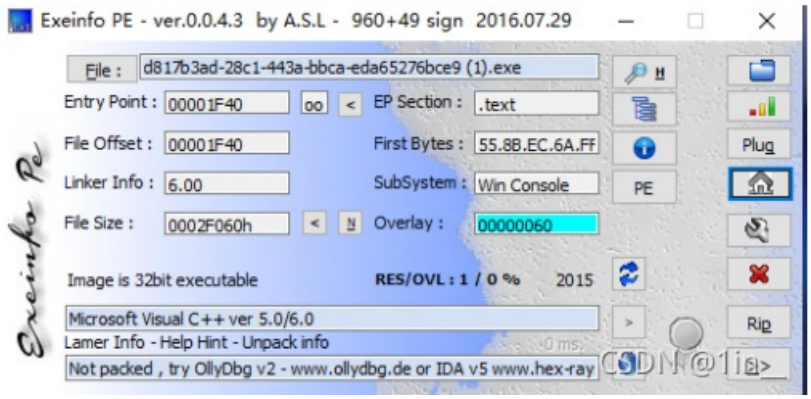
编写脚本拿到flag

```
key = '~}|{zyxwvutsrqponmlkjihgfedcba`_^][ZYXWVUTSRQPONMLKJIHGFEDCBA@?>=<;:9876543210/.-,+*)('&%$# !"' **#'一定要加**
encrypt = [42,70,39,34,78,44,34,40,73,63,43,64]
x = []
flag = ''
for i in encrypt:
x.append(key.find(chr(i))+1)
for i in x:
flag += chr(i)
print(flag)
```

CrackRTF

拖进去查看信息

32位



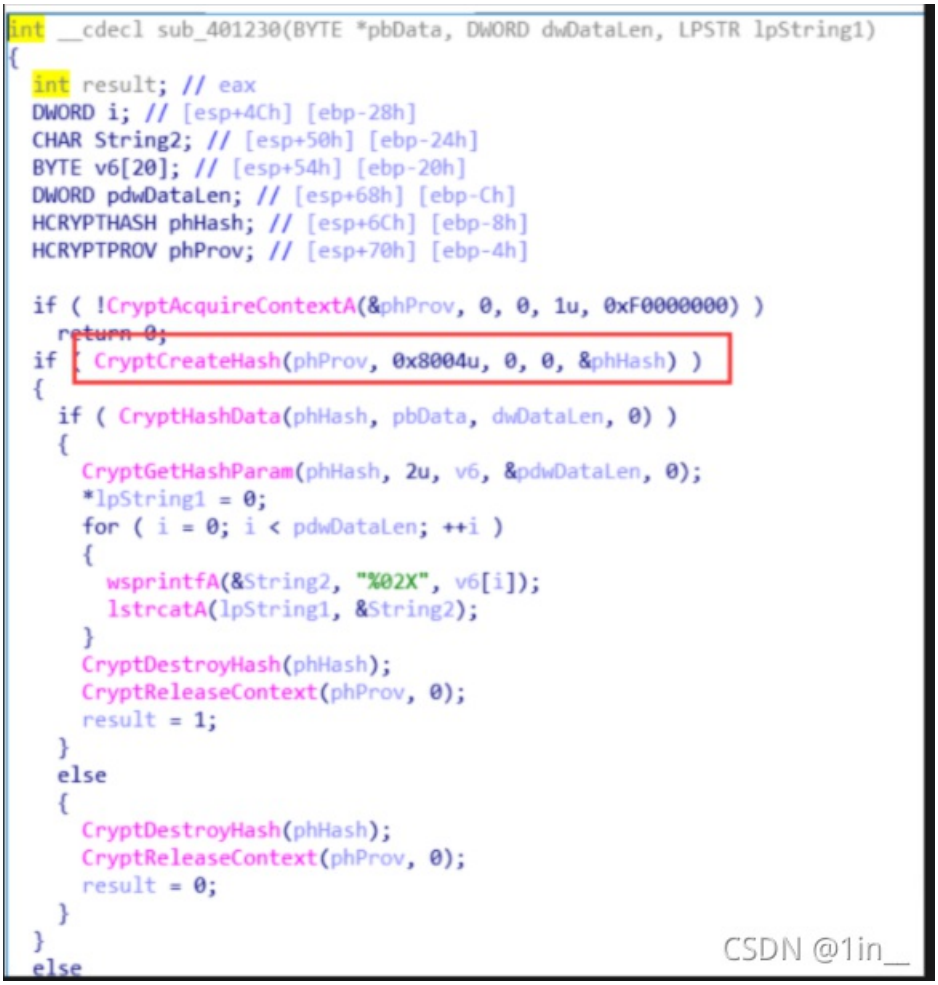
拖进IDA里面 查看主函数 F5大法

```
printf("pls input the first passwd(1): ");
scanf("%s", &pbData);
if ( strlen((const char *)&pbData) != 6 )
{
  printf("Must be 6 characters!\n");
  ExitProcess(0);
}
v4 = atoi((const char *)&pbData);
if ( v4 < 100000 )
  ExitProcess(0);
strcat((char *)&pbData, "@DBApp");
v0 = strlen((const char *)&pbData);
sub_40100A(&pbData, v0, &String1);
if ( !_strcmpi(&String1, "6E32D0943418C2C33385BC35A1470250DD8923A9") )
{
  printf("continue...\n\n");
  printf("pls input the first passwd(2): ");
  memset(&String, 0, 0x104u);
  scanf("%s", &String);
  if ( strlen(&String) != 6 )
  {
    printf("Must be 6 characters!\n");
    ExitProcess(0);
  }
  strcat(&String, (const char *)&pbData);
  memset(&String1, 0, 0x104u);
  v1 = strlen(&String);
  sub_401019((BYTE *)&String, v1, &String1);
  if ( !_strcmpi("27019e688a4e62a649fd99cadaafdb4e", &String1) )
  {
    if ( !(unsigned __int8)sub_40100F(&String) )
    {
      printf("Error!!\n");
      ExitProcess(0);
```


在第一部分里面 我们需要输入的是6个字符，并且要>100000，那么应该是数字组成的
然后连接上@DBApp
通过一个sub_40100A函数进行加密，然后与`6E32D0943418C2C33385BC35A1470250DD8923A9`进行匹配

```
printf("pls input the first passwd(1): ");
scanf("%s", &pbData);
if ( strlen((const char *)&pbData) != 6 )
{
  printf("Must be 6 characters!\n");
  ExitProcess(0);
}
v4 = atoi((const char *)&pbData);
if ( v4 < 100000 )
  ExitProcess(0);
strcat((char *)&pbData, "@DBApp");
v0 = strlen((const char *)&pbData);
sub_40100A(&pbData, v0, &String1);
if ( !_strcmpi(&String1, "6E32D0943418C2C33385BC35A1470250DD8923A9") )
```

进入第一个加密函数中去看 内容如下：



```
int __cdecl sub_401230(BYTE *pbData, DWORD dwDataLen, LPSTR lpString1)
{
  int result; // eax
  DWORD i; // [esp+4Ch] [ebp-28h]
  CHAR String2; // [esp+50h] [ebp-24h]
  BYTE v6[20]; // [esp+54h] [ebp-20h]
  DWORD pdwDataLen; // [esp+68h] [ebp-Ch]
  HCRYPTHASH phHash; // [esp+6Ch] [ebp-8h]
  HCRYPTPROV phProv; // [esp+70h] [ebp-4h]

  if ( !CryptAcquireContextA(&phProv, 0, 0, 1u, 0xF0000000) )
    return 0;
  if ( CryptCreateHash(phProv, 0x8004u, 0, 0, &phHash) )
  {
    if ( CryptHashData(phHash, pbData, dwDataLen, 0) )
    {
      CryptGetHashParam(phHash, 2u, v6, &pdwDataLen, 0);
      *lpString1 = 0;
      for ( i = 0; i < pdwDataLen; ++i )
      {
        wsprintfA(&String2, "%02X", v6[i]);
        lstrcatA(lpString1, &String2);
      }
      CryptDestroyHash(phHash);
      CryptReleaseContext(phProv, 0);
      result = 1;
    }
    else
    {
      CryptDestroyHash(phHash);
      CryptReleaseContext(phProv, 0);
      result = 0;
    }
  }
  else
```


通过上网查资料 发现这是一个windows加密的加密库函数
经过发现 `6E32D0943418C2C33385BC35A1470250DD8923A9` 是40位的加密后的字符串
很有可能是sha1加密，先来爆破试一试~

爆破脚本：

```
import hashlib
flag = "@DBApp"
for i in range(100000,999999):
 s = str(i)+flag
 x = hashlib.sha1(s.encode())
 cnt = x.hexdigest()
 if "6e32d0943418c2c" in cnt:
  print(cnt)
  print(str(i)+flag)
```

得到第一次密码：`123321@DBApp`

第二次输入同理 只需要把123321@DBApp加在第二次密码的后面，并且进行加密

```
  ExitProcess(0);
strcat((char *)&pbData, "@DBApp");
v0 = strlen((const char *)&pbData);
sub_40100A(&pbData, v0, &String1);
if ( !_strcmpi(&String1, "6E32D0943418C2C33385BC35A1470250DD8923A9")
{
  printf("continue...\n\n");
  printf("pls input the first passwd(2): ");
  memset(&String, 0, 0x104u);
  scanf("%s", &String);
  if ( strlen(&String) != 6 )
  {
    printf("Must be 6 characters!\n");
    ExitProcess(0);
  }
  strcat(&String, (const char *)&pbData);
  memset(&String1, 0, 0x104u);
  v1 = strlen(&String);
  sub_401019((BYTE *)&String, v1, &String1);
  if ( !_strcmpi("27019e688a4e62a649fd99cadaafd
```


直接进入sub_401019函数进行查看：

```
int __cdecl sub_401040(BYTE *pbData, DWORD dwDataLen, LPSTR lp
{
  int result; // eax
  DWORD i; // [esp+4Ch] [ebp-24h]
  CHAR String2; // [esp+50h] [ebp-20h]
  BYTE v6[16]; // [esp+54h] [ebp-1Ch]
  DWORD pdwDataLen; // [esp+64h] [ebp-Ch]
  HCRYPTHASH phHash; // [esp+68h] [ebp-8h]
  HCRYPTPROV phProv; // [esp+6Ch] [ebp-4h]

  if ( !CryptAcquireContextA(&phProv, 0, 0, 1u, 0xF0000000) )
    return 0;
  if ( CryptCreateHash(phProv, 0x8003u, 0, 0, &phHash) )
  {
    if ( CryptHashData(phHash, pbData, dwDataLen, 0) )
    {
      CryptGetHashParam(phHash, 2u, v6, &pdwDataLen, 0);
      *lpString1 = 0;
      for ( i = 0; i < pdwDataLen; ++i )
      {
        wsprintfA(&String2, "%02X", v6[i]);
        lstrcatA(lpString1, &String2);
      }
      CryptDestroyHash(phHash);
      CryptReleaseContext(phProv, 0);
      result = 1;
```