

BUUCTF CREAKRTF

原创

一夜通宵程序员 于 2021-05-20 22:29:33 发布 54 收藏

文章标签: 安全

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) 版权协议, 转载请附上原文出处链接和本声明。

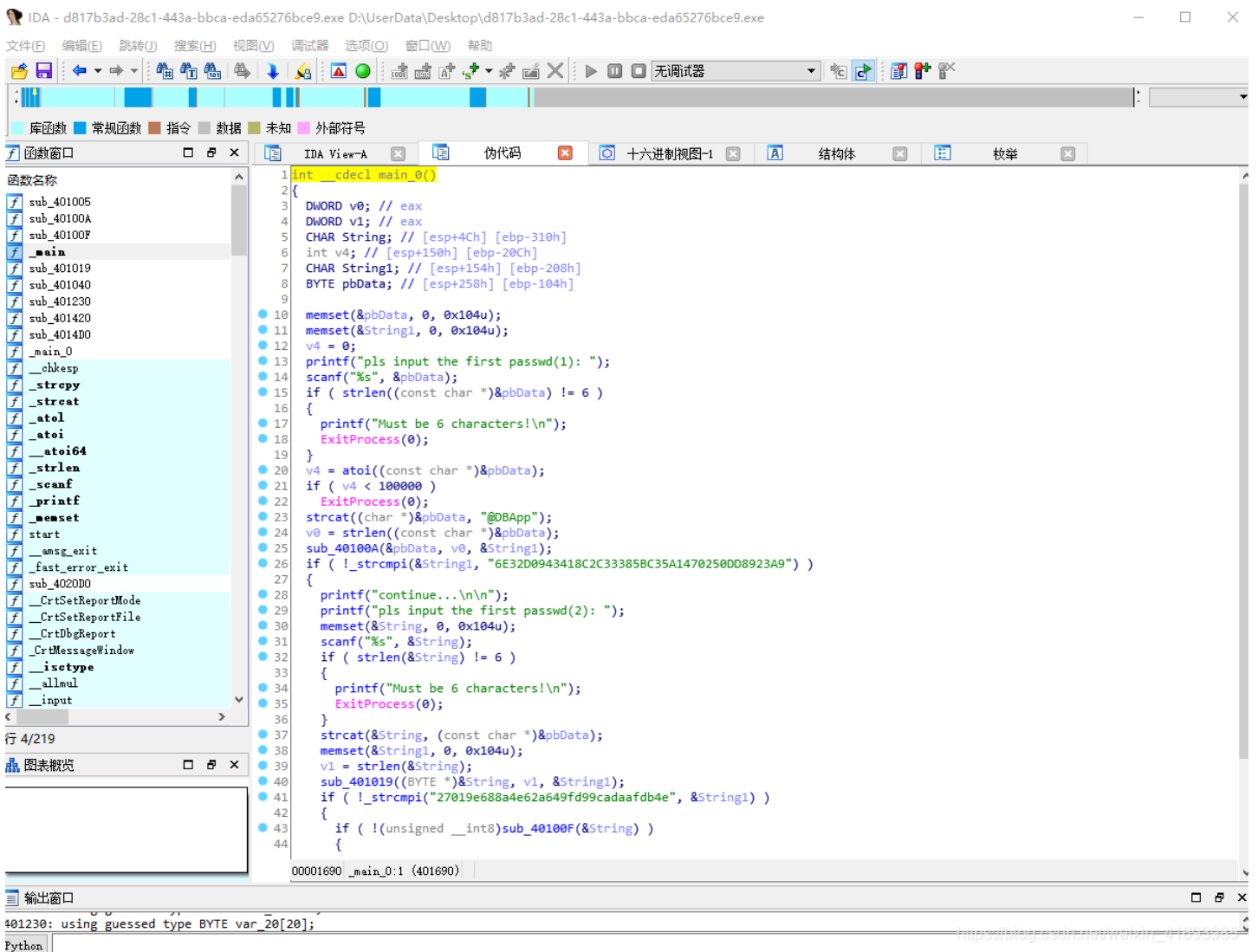
本文链接: https://blog.csdn.net/weixin_41693985/article/details/117092582

版权

BUUCTF CREAKRTF

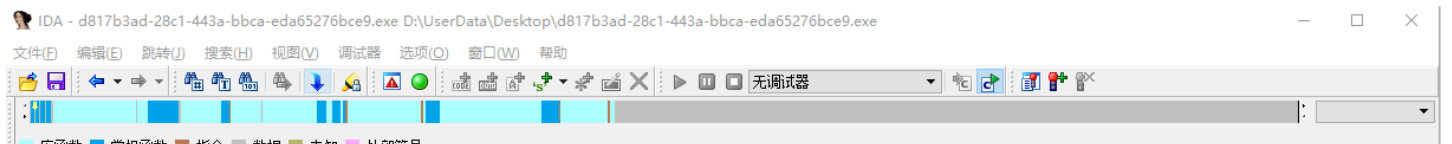
拿到程序, 首先查壳, 无壳, 是32位程序

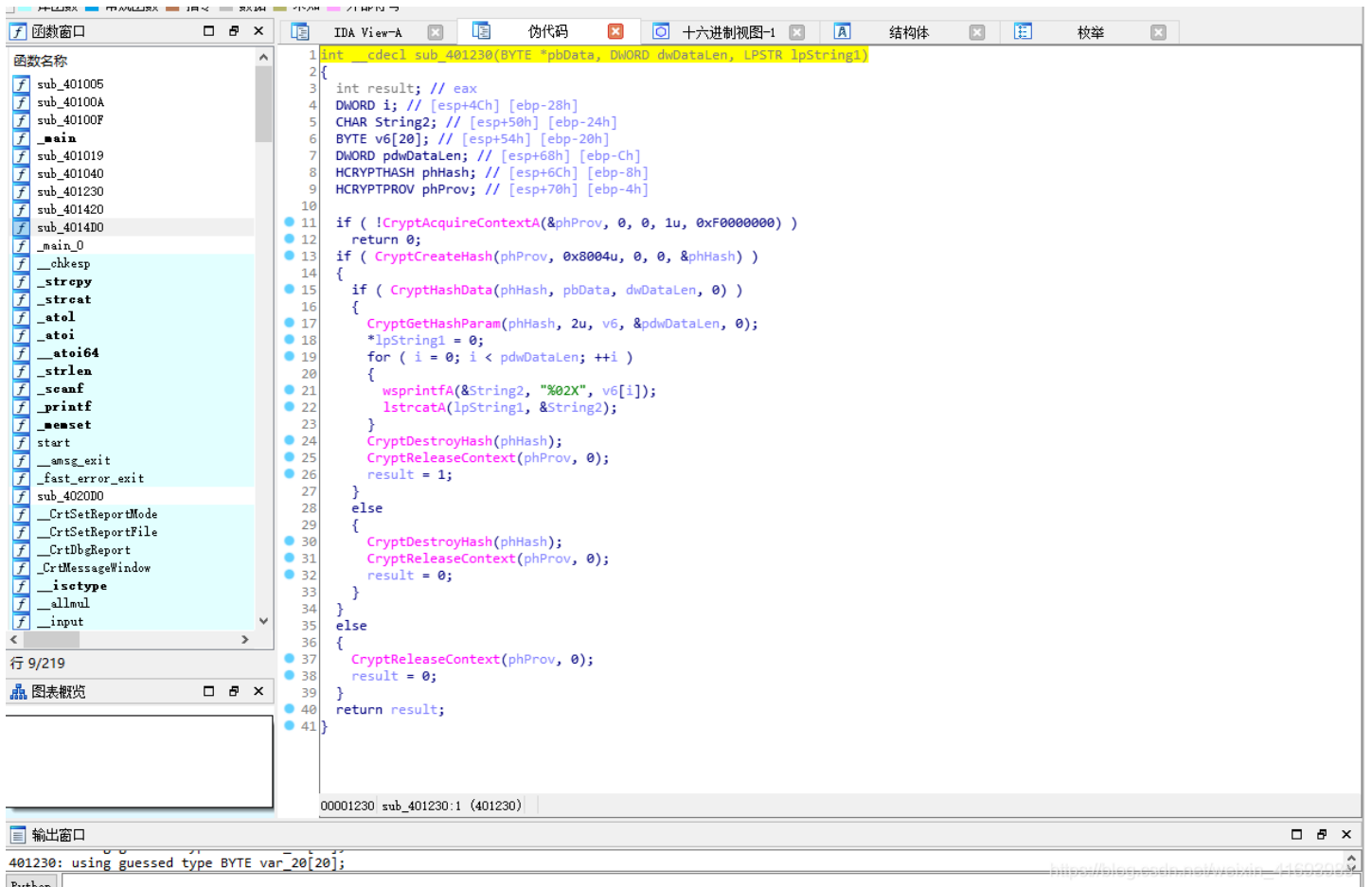
拖入IDA, 进main函数



分析代码, 首先是要我们输入一串字符且长度为6, 再往下分析可知, atoi是把字符串变为整数型, 所有可知, 我们输入的字符是大于100000小于999999的

ok, 继续往下走, 可以看到strcat把我们输入的字符串和@DBApp拼接, 然后在下一个函数进行加密得到的字符串就是"6E32D0943418C2C33385BC35A1470250DD8923A9", 进入加密函数





直接百度搜索CryptAcquireContextA了解这是一个加密函数，而8004u是标识符，可知是SHA1，好像是不可逆加密，不确定，所以直接暴力破解，脚本如下：

```

import hashlib

s = "@DBApp"
flag = ""

for i in range(100000, 999999):
    b=str(i)+s
    c=hashlib.sha1(b.encode())
    d=c.hexdigest()
    if d == "6e32d0943418c2c33385bc35a1470250dd8923a9":
        print(str(i)+s)

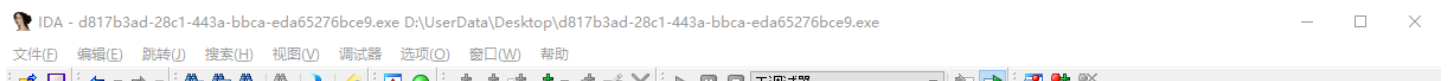
```

https://blog.csdn.net/weixin_41693985

输出：

123321@DBApp

下面的代码和上面的就差不多了，区别在于下面的代码不能进行爆破破解，所以只能进入加密函数里面去看看



函数名称

- sub_401005
- sub_40100A
- sub_40100F
- main
- sub_401019
- sub_401040
- sub_401230
- sub_401420
- sub_4014D0
- main_0
- __chkesp
- __strcpy
- __streat
- __atol
- __atoi
- __atoi64
- __strlen
- __scanf
- __printf
- __memset
- start
- __amsg_exit
- __fast_error_exit
- sub_4020D0
- __CrtSetReportMode
- __CrtSetReportFile
- __CrtDbgReport
- __CrtMessageBox
- __isctype
- __allmul
- __input

行 9/219

输出窗口

401040: using guessed type BYTE var_1C[16];

```

1 int cdecl sub_401040(BYTE *pbData, DWORD dwDataLen, LPSTR lpString1)
2 {
3     int result; // eax
4     DWORD i; // [esp+4Ch] [ebp-24h]
5     CHAR String2; // [esp+50h] [ebp-20h]
6     BYTE v6[16]; // [esp+54h] [ebp-1Ch]
7     DWORD pdwDataLen; // [esp+64h] [ebp-Ch]
8     HCRYPTHASH phHash; // [esp+68h] [ebp-8h]
9     HCRYPTPROV phProv; // [esp+6Ch] [ebp-4h]
10
11     if ( !CryptAcquireContextA(&phProv, 0, 0, 1u, 0xF0000000) )
12         return 0;
13     if ( CryptCreateHash(phProv, 0x8003u, 0, 0, &phHash) )
14     {
15         if ( CryptHashData(phHash, pbData, dwDataLen, 0) )
16         {
17             CryptGetHashParam(phHash, 2u, v6, &pdwDataLen, 0);
18             *lpString1 = 0;
19             for ( i = 0; i < pdwDataLen; ++i )
20             {
21                 wsprintfA(&String2, "%02X", v6[i]);
22                 lstrcatA(lpString1, &String2);
23             }
24             CryptDestroyHash(phHash);
25             CryptReleaseContext(phProv, 0);
26             result = 1;
27         }
28         else
29         {
30             CryptDestroyHash(phHash);
31             CryptReleaseContext(phProv, 0);
32             result = 0;
33         }
34     }
35     else
36     {
37         CryptReleaseContext(phProv, 0);
38         result = 0;
39     }
40     return result;
41 }

```

00001040 sub_401040:1 (401040)

一看标识符，不会，跳过向下分析下一个sub函数

函数名称

- sub_401005
- sub_40100A
- sub_40100F
- main
- sub_401019
- sub_401040
- sub_401230
- sub_401420
- sub_4014D0
- main_0
- __chkesp
- __strcpy
- __streat
- __atol
- __atoi
- __atoi64
- __strlen
- __scanf
- __printf
- __memset
- start
- __amsg_exit
- __fast_error_exit
- sub_4020D0
- __CrtSetReportMode
- __CrtSetReportFile
- __CrtDbgReport
- __CrtMessageBox
- __isctype
- __allmul

行 9/219

输出窗口

401040: using guessed type BYTE var_1C[16];

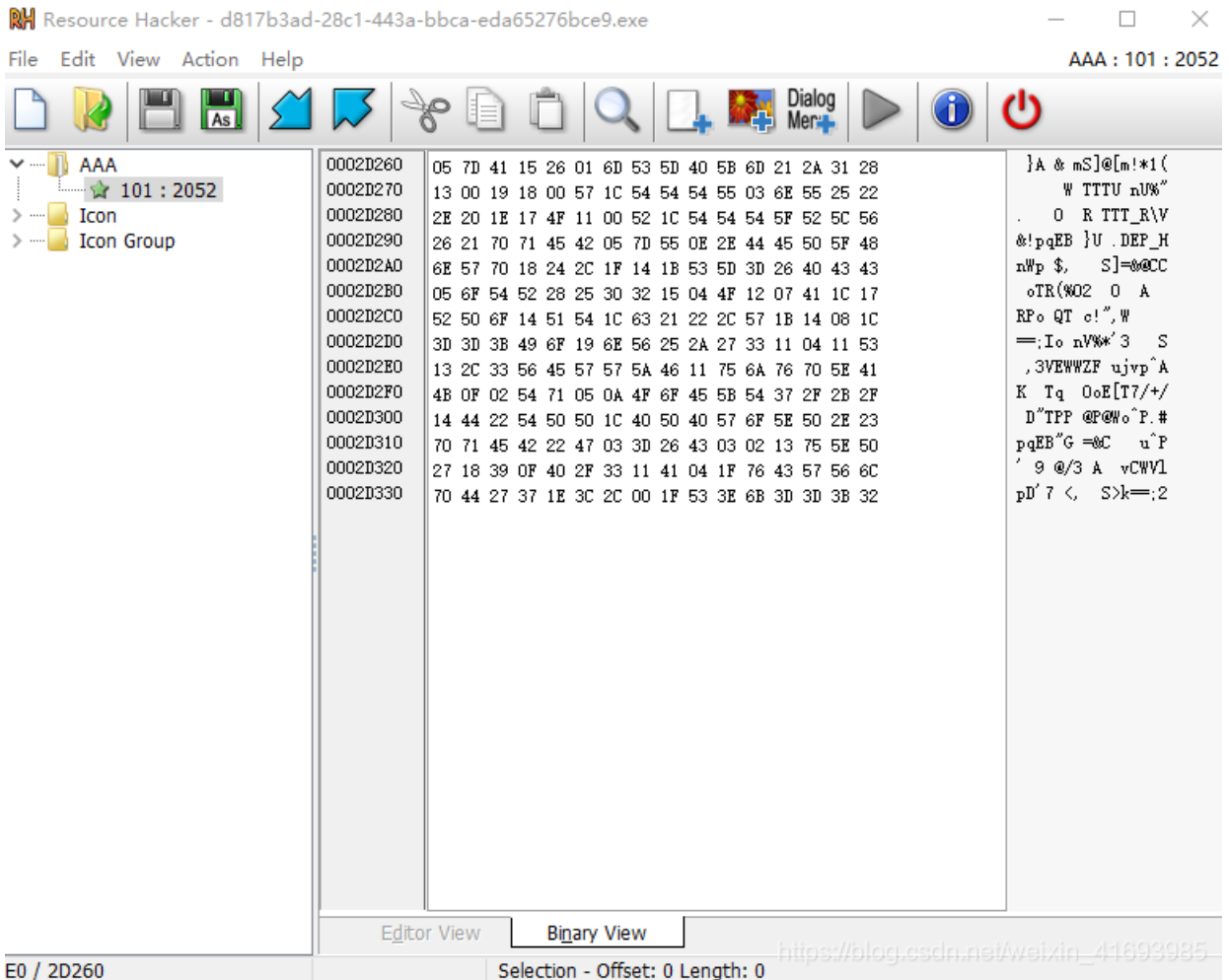
```

1 char cdecl sub_4014D0(LPCSTR lpString)
2 {
3     LPCVOID lpBuffer; // [esp+50h] [ebp-1Ch]
4     DWORD NumberOfBytesWritten; // [esp+58h] [ebp-14h]
5     DWORD NumberOfBytesToWrite; // [esp+5Ch] [ebp-10h]
6     HGLOBAL hResData; // [esp+60h] [ebp-Ch]
7     HRSRC hResInfo; // [esp+64h] [ebp-8h]
8     HANDLE hFile; // [esp+68h] [ebp-4h]
9
10    hFile = 0;
11    hResData = 0;
12    NumberOfBytesToWrite = 0;
13    NumberOfBytesWritten = 0;
14    hResInfo = FindResourceA(0, (LPCSTR)0x65, "AAA");
15    if ( !hResInfo )
16        return 0;
17    NumberOfBytesToWrite = SizeofResource(0, hResInfo);
18    hResData = LoadResource(0, hResInfo);
19    if ( !hResData )
20        return 0;
21    lpBuffer = LockResource(hResData);
22    sub_401005(lpString, (int)lpBuffer, NumberOfBytesToWrite);
23    hFile = CreateFileA("dbapp.rtf", 0x10000000u, 0, 0, 2u, 0x80u, 0);
24    if ( hFile == (HANDLE)-1 )
25        return 0;
26    if ( !WriteFile(hFile, lpBuffer, NumberOfBytesToWrite, &NumberOfBytesWritten, 0) )
27        return 0;
28    CloseHandle(hFile);
29    return 1;
30 }

```

000014D0 sub_4014D0:1 (4014D0)

在这里呢就用到一款工具Resource Hacker，打开AAA的文件如下：



然后查看sub_401005函数

```
1 unsigned int __cdecl sub_401420(LPCSTR lpString, int a2, int a3)
2 {
3     unsigned int result; // eax
4     unsigned int i; // [esp+4Ch] [ebp-Ch]
5     unsigned int v5; // [esp+54h] [ebp-4h]
6
7     v5 = strlenA(lpString);
8     for ( i = 0; ; ++i )
9     {
10        result = i;
11        if ( i >= a3 )
12            break;
13        *(_BYTE *)(i + a2) ^= lpString[i % v5];
14    }
15    return result;
16 }
```

https://blog.csdn.net/weixin_41693985

到这里差不多就懂了，异或嘛，逆算一下就好了，脚本如下：

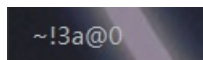
```
s = "{\\rtf1"

a = [0x05, 0x7D, 0x41, 0x15, 0x26, 0x01]

flag = ""
for i in range(0, len(s)):
    x = ord(s[i]) ^ a[i]
    flag += chr(x)
print(flag)
```

https://blog.csdn.net/weixin_41693985

结果:



将两次得到的数据输入程序就可以得到一个world文档，打开即可获得flag（我的创建的文档不知道跑哪去，我还以为我程序坏了，结果是创建到其他目录去了）

