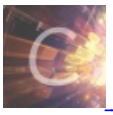
看雪CTF2017第五题 独行孤客CrackMe的writeup

原创

anhkgg 于 2017-06-14 09:11:11 发布 1174 收藏

分类专栏： 原创 文章标签： ctf 逆向 看雪 crackme

本文链接：https://blog.csdn.net/angelxf/article/details/73198392



题目入口：http://ctf.pediy.com/game-fight-35.htm，可下载相关文件

> 本题需要在XP系统运行，因为驱动只支持xp

00. 先看驱动

驱动不大，才20多个函数。

从入口开始分析。

1. 创建设备

```
.text:000107D5 68 58 13 01 00                                  push    offset aDeviceVm
.text:000107DA 8D 45 F4                                        lea     eax, [ebp+Destin
.text:000107DD 33 FF                                           xor     edi, edi
.text:000107DF 50                                              push    eax
.text:000107E0 89 7D FC                                        mov     [ebp+DeviceObjec
.text:000107E3 FF D6                                           call    esi ; RtlInitUni
```

用来与应用层通信

2. IRP_MJ_FUNCTION

主要有三个，read/write/ioctl。

```
.text:00010870 C7 46 44 A8 05 01 00                            mov     dword ptr [esi+4
.text:00010877 C7 46 48 1C 06 01 00                            mov     dword ptr [esi+4
.text:0001087E C7 46 70 1A 07 01 00                            mov     dword ptr [esi+7
```

先看f_DrvWrite_1061C，通过irp获取到上层传入的数据，然后通过104b6获取某个输出存入全局变量g_READCC（根据read的分析，可以知道长度为4的4字节数组）。

```
.text:00010669 57                                                       push    edi
.text:0001066A FF 75 0C                                                 push    [ebp+Irp]
.text:0001066D 53                                                       push    ebx
.text:0001066E E8 11 0C 00 00                                           call    memcpy
.text:00010673 83 C4 18                                                 add     esp, 18h
.text:00010676 83 3D D8 14 01 00 00                                     cmp     dword ptr is_cle
.text:0001067D 74 15                                                    jz      short loc_10694
.text:0001067F 68 C8 14 01 00                                           push    offset g_READCC
.text:00010684 53                                                       push    ebx
.text:00010685 E8 2C FE FF FF                                           call    f_GetMd5_104B6
.text:0001068A C7 05 DC 14 01 00 01 00 00 00                            mov     is_write, 1
```

进入104b6内部，key是个16字节数组，初始化0。然后将上面传下的数据拷贝到key中，长度需要小于16。然后将key进行一下变换。
key[0] ++(反调试标志为1，后面再说)，其他key[i] += i

```
.text:000104F5 56                                                       push    esi
.text:000104F6 51                                                       push    ecx
.text:000104F7 8D 45 EC                                                 lea     eax, [ebp+key]
.text:000104FA 50                                                       push    eax
.text:000104FB E8 84 0D 00 00                                           call    memcpy
...
text:00010505 39 05 D8 14 01 00                                        cmp     dword ptr is_clea
.text:0001050B 74 03                                                    jz      short loc_10510
.text:0001050D FE 45 EC                                                 inc     [ebp+key]
.text:00010510
.text:00010510                                                  loc_10510:
.text:00010510 3B F0                                                    cmp     esi, eax
.text:00010512 7E 09                                                    jle     short loc_1051D
.text:00010514
.text:00010514                                                  loc_10514:
.text:00010514 00 44 05 EC                                              add     [ebp+eax+key], a
.text:00010518 40                                                       inc     eax
.text:00010519 3B C6                                                    cmp     eax, esi
.text:0001051B 7C F7                                                    jl      short loc_10514
```

接着通过下面三个函数对key进行计算，输出结果

```
f_Md5_Init_108B2((MD5OBJ *)&v5);
f_Md5_j_11124((MD5OBJ *)&v5, key, strlen(key));
f_Md5_hexdigest((int)&v5, md5);
```

进入108b2一看就猜测是md5计算，f_Md5_hexdigest将计算结果(32字节字符)保存到md5字段中输出，设置计算标志。也就是大致确认write是计算md5，然后保存到g_READCC

```
MD5OBJ *__stdcall f_Md5_Init_108B2(MD5OBJ *a1)
{
  MD5OBJ *result; // eax@1

  result = a1;
  a1->len8 = 0;
  a1->unk_4 = 0;
  a1->s1 = 0x67452301;
  a1->s2 = 0xEFCDAB89;
  a1->s3 = 0x98BADCFE;
  a1->s4 = 0x10325476;
  return result;
}
```

接着看f_DrvRead_105A8，看刚才的计算标志是否为0，为0就初始化g_READCC一段值（不知道作者意图，迷惑cracker？），如果计算标志是1，就直接返回计算的结果，然后该值返回到用户空间。也就是如果通过write计算了md5，这里就是获取md5计算结果。

```
.text:000105AD
 if ( !is_write )
  {
    i = 3;
    do
    {
      g_READCC[i] = 3 * i - 'd';
      ++i;
    }
    while ( i < 16 );
    g_READCC[0] = 0xCBu;
    g_READCC[1] = 0xAAu;
    g_READCC[2] = 0xDEu;
    g_READCC[3] = 0xB0u;
  }
  //返回数据
   *(_DWORD *)&MasterIrp->Type = *(_DWORD *)g_READCC;
  v4 = (int)&MasterIrp->MdlAddress;
  *(_DWORD *)v4 = *(_DWORD *)&g_READCC[4];
  v4 += 4;
  *(_DWORD *)v4 = *(_DWORD *)&g_READCC[8];
  *(_DWORD *)(v4 + 4) = *(_DWORD *)&g_READCC[12];
```

最后看f_DrvControl_1071A，支持多个命令号，但只有222004h有用。设置反调试标志为1，然后进入10486看看

```
.text:00010734 2D 04 20 22 00                                        sub     eax, 222004h
.text:00010739 8B 4E 0C                                              mov     ecx, [esi+0Ch]
.text:0001073C 74 2C                                                 jz      short loc_1076A
...
.text:0001076A                                          loc_1076A:
.text:0001076A C7 05 D8 14 01 00 01 00 00 00                         mov     dword ptr is_cle
.text:00010774 FF 15 80 13 01 00                                     call    ds:IoGetCurrentP
.text:0001077A A3 E0 14 01 00                                        mov     eproc, eax
.text:0001077F E8 02 FD FF FF                                        call    f_ClearDebugPort
```

枚举进程找到当前进程的eprocess(其实没必要枚举把)，置eprocess->DebugPort = NULL，让应用层调试器失效，达到反跳试效果。

```
result = IoGetCurrentProcess();
  v1 = result;
  while ( result != (PEPROCESS)eproc )
  {
    result = (PEPROCESS)(*((_DWORD *)result + 0x22) - 0x88);// eproc->ActiveProcessLinks.Flink
    if ( result == v1 )
      return result;
  }
  *((_DWORD *)result + 0x2F) = 0;               // eproc->DebugPort = 0
```

这里猜想一下，如果破解者通过应用层patch，不发送222004h命令来解除反跳试的话，那么这里的反跳试标志就是0，然后在write中计算md5时，对key[0]就不会做++操作，那么上层就会获取到一个错误的值，从而影响破解。

3. k掉驱动反调试

首先想到的是将驱动文件patch，也就是DebugPort置零的指令nop掉

```
.text:000104A9 83 A0 BC 00 00 00 00                                    and     dword ptr [eax+0
```

通过reshacker将驱动资源导出来，然后hex编辑工具修改104A9的内容(文件内存对齐一样)为7个NOP，然后再将patch驱动文件导入到exe中。

会提示驱动加载失败，可能有校验，不再细跟。

没办法，为了让od能够调试，我写了个简单驱动，在本驱动加载时，将104A进行patch，通过反跳试。

01. 再看CrackMe

既然知道有驱动了，先找找释放和加载驱动的代码，通过 FindResourceA和CreateService即可定位（不再详述），注意到的是，驱动加载成功会设置一个标志，用于后面验证的判断

```
v5 = f_CreaetSrv_401AA0(ServiceName, &Buffer);// vmxdrv
 v1->is_drv_run = v5;
```

然后再找和驱动通信的代码，通过DeviceIoControl找到调用222004命令好的代码。通过创建一个线程，循环调用该接口来清零DebugPort

```
while ( 1 )
 {
   v0 = CreateFileA(FileName, 0xC0000000, 0, 0, 3u, 0x80u, 0);
   if ( v0 == (HANDLE)-1 )
     break;
   DeviceIoControl(v0, 0x222004u, 0, 0, &OutBuffer, 0x100u, &BytesReturned, 0);
   CloseHandle(v0);
   Sleep(0xBB8u);
 }
```

按理说这里可以patch掉来去掉反跳试，但就会出现我前面分析提到的问题。

通过WriteFile找到调用read/write的位置，也就是计算md5和获取md5的位置。

```
.text:00401D50 ; HANDLE __thiscall f_CalcKeyMd5_401D50(void *this, char *key, size_t len)
...
.text:00401E4E                 push    ebx             ; lpOverlapped
.text:00401E4F                 push    eax             ; lpNumberOfBytesWritten
.text:00401E50                 lea     ecx, [esp+344h+Buffer] //用户输入的key相关数据
.text:00401E54                 push    esi             ; nNumberOfBytesToWrite
.text:00401E55                 push    ecx             ; lpBuffer
.text:00401E56                 push    edi             ; hFile
.text:00401E57                 call    ds:WriteFile //计算md5
.text:00401E5D                 test    eax, eax
.text:00401E5F                 jz      short loc_401ED4
.text:00401E61                 lea     edx, [esp+33Ch+NumberOfBytesRead]
.text:00401E65                 push    ebx             ; lpOverlapped
.text:00401E66                 push    edx             ; lpNumberOfBytesRead
.text:00401E67                 lea     eax, [esp+344h+keymd5]
.text:00401E6E                 push    10h             ; nNumberOfBytesToRead
.text:00401E70                 push    eax             ; lpBuffer
.text:00401E71                 push    edi             ; hFile
.text:00401E72                 call    ds:ReadFile //读取md5
```

f_CalcKeyMd5_401D50回溯一层就是输入key回车的响应函数。
这里先通过UpdateData(1)获取输入数据，然后拷贝到局部变量

```
f_UpdateData_41A4F7(1);
f_CString_copy_417D43((CString *)&key, (LPCSTR *)&v1->key);//用户输入的
```

然后输入进行小写和反转变换

```
f_CString_lwr_4182FA((CString *)&key); //小写
f_Cstring_rev_41830C((CString *)&key);        // 反转
```

判断输入长度是否为6，不是退出，清除输入，并通过IsDebuggerPresent检查是否在调试（OD直接过），是调试也退出，清理出输入。

```
  if ( *(_DWORD *)(key - 8) != 6 || IsDebuggerPresent() )
  {
    CString::operator=((CString *)&v1->unk_6c, byte_431398);
    CString::operator=((CString *)&v1->key, byte_431398);
    f_UpdateData_41A4F7(0);
  }
```

满足长度要求，再看驱动是否加载，再调用f_CalcKeyMd5_401D50计算md5. 也就是调用驱动获取md5，记为KeyMd51.

```
//.text:004017DE
 if ( v1->is_drv_run )
    {
      keymd5str = *(_DWORD *)(key - 8);
      v3 = sub_418263(&key, 0);
      f_CalcKeyMd5_401D50(v1, (char *)v3, keymd5str);
    }
```

接着下面两个函数，先调用f_GetStrMd5_401920（应用层的Md5，通过调试可以很快确认，内部也有md5特征）计算KeyMd51的Md5，记为KeyMd52，然后调用sub_415A78截取KeyMd52从第3为开始的10字符，记为KeyMd53。

```
    f_GetStrMd5_401920((char)v4, (CString *)keymd5str);// 00943950  37 63 37 36 36 65 32 61 31 63 61 30
                                                       // 00943960  62 30 65 39 31 66 39 33 35 65 64 61 61 64
                                                       //
                                                       //
                                                       //
    sub_415A78((LPCSTR *)&keymd5str_obj, (int)&v9, 2, 0xAu);// 截取2开始长度0xA的值
                                                       // 00943900  37 36 36 65 32 61 31 63 61 30 00 38 39 30
                                                       // 00943910  33 39 32 36 39 32 65 38 32 64 36 33 62 31
                                                       //
```

最后KeyMd53与888aeda4ab比较，成功提示Success^^!

```
    if ( _mbsicmp(keymd5str_obj, a888aeda4ab) ) // 888aeda4ab
    {
      CString::operator=((CString *)&v1->unk_6c, byte_431398);
      CString::operator=((CString *)&v1->key, byte_431398);
      f_UpdateData_41A4F7(0);
    }
    else
    {
      f_ShowSuccess_402030(v1);//成功提示
    }
```

总结算法：
1. KEY1 = rev(lwr(key))，key长度6，将输入转小写，逆序
2. 反调试成功时KEY1[0]+=1, 其他KEY1[i]+=i;
3. KEY2 = DrvMd5(KEY1)，驱动MD5计算
4. KEY3 = Md5(KEY2), 应用层Md5计算
5. KEY4 = KEY3[2:12]，取第3位开始的10个字符
6. KEY4 == ‘888aeda4ab’

11. 求解

由于MD5hash无法逆运算，只能爆破了，刚开始忘了题目key只能是数字和字母，结果我跑了全字符，跑了1天多….没出来，卡hi是怀疑自己

后来改成了数字字母，终于得到答案 su1987

爆破代码如下：

```
    char Seed[/*68*/36] = {
        'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't',
        '0', '1', '2', '3', '4', '5', '6', '7', '8', '9',
        };

    #define SEED_SIZE 36// 68

    typedef struct _THREAD_PARAM
    {
        int i1;
        int i2;
        int i3;
        int i2_1;
        int i2_2;
    }TPP, *PTPP;

    int g_ThreadCnt = 0;
```

```
int g_threadCnt = 0;
int g_start = 0;
long g_count = 0;

void write_file(char* sz)
{
    HANDLE hFile = CreateFileA("1.log", GENERIC_WRITE|GENERIC_READ, FILE_SHARE_READ, NULL, OPEN_ALWAYS,
    if(hFile)
    {
        SetFilePointer(hFile, 0, 0, FILE_END);
        DWORD dw = 0;
        WriteFile(hFile, sz, strlen(sz), &dw, NULL);
        CloseHandle(hFile);
        hFile = NULL;
    }
}

bool crack1(PTPP p)
{
    int i1 = p->i1;
    int i2 = p->i2;

    char sss[20] = {0};

    for(int i3=0; i3<SEED_SIZE; i3++)
    {
        for(int i4=0; i4<SEED_SIZE; i4++)
        {
            for(int i5=0; i5<SEED_SIZE; i5++)
            {
                for(int i6=0; i6<SEED_SIZE; i6++)
                {
                    char sza[7] = {Seed[i1], Seed[i2], Seed[i3], Seed[i4], Seed[i5], Seed[i5]};

                    g_count ++;

                    char sz[7] = {0};
                    //反转
                    sz[0] = Seed[i6]+1;
                    sz[1] = Seed[i5]+1;
                    sz[2] = Seed[i4]+2;
                    sz[3] = Seed[i3]+3;
                    sz[4] = Seed[i2]+4;
                    sz[5] = Seed[i1]+5;

                    FileMD5 fm;
                    char* p = (char*)fm.md5(sz, 6);
                    p = (char*)fm.md5(p, 32);
                    strncpy(sss, p+2, 10);


                    if(!stricmp(sss, "888aeda4ab"))
                    {
                        char info[1024] = {0};
                        sprintf(info, "%c%c%c%c%c%c, => %s, %s\n",
                            Seed[i1], Seed[i2], Seed[i3], Seed[i4], Seed[i5], Seed[i6],
                            sz,
                            sss
                            );
                        write_file(info);
```

```
                        int spell = GetTickCount() - g_start;
                        printf("spell time : %d s", spell/1000);

                        system("pause");

                        return true;
                    }
                }
            }
            //system("cls");
            printf("count: %ld\n", g_count);
        }

    }

    return false;
}
void crack3(PTPP p)
{
    int i1 = p->i1;
    int i2_1 = p->i2_1;
    int i2_2 = p->i2_2;

    delete[] p;

    TPP p1 = {0};
    p1.i1 = i1;
    for(int i=i2_1; i<i2_2; i++)
    {
        p1.i2 = i;
        if(crack1(&p1))
        {
            return;
        }
    }
}
void crack2(int i1, int i2_1, int i2_2)
{
    PTPP p = new TPP;//{0};
    if(p == NULL)
    {
        printf("!!!!!!!!!!!!没neicun! ! ");
        return;
    }
    memset(p, 0, sizeof(TPP));
    p->i1 = i1;
    p->i2_1 = i2_1;
    p->i2_2 = i2_2;

    HANDLE h = CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)crack3, (PVOID)p, 0, NULL);
    if(h == NULL)
    {
        printf("CreateTHREAD error [%d]\n", g_ThreadCnt);
    }
    else
    {
```

```
            g_Handles[g_ThreadCnt++] = h;
        }
    }

    void crack()
    {
        for(int i1=0; i1<SEED_SIZE; i1++)
        {
            int i2 = 0;
    #define STEP_SIZE 2
            for(i2 = 0; i2<SEED_SIZE-STEP_SIZE; i2+=STEP_SIZE)
            {
                crack2(i1, i2, i2+STEP_SIZE);
            }
            crack2(i1, i2, SEED_SIZE);
        }
    }

    int _tmain(int argc, _TCHAR* argv[])
    {
        int start = GetTickCount();
        g_start = GetTickCount();

        crack();

        WaitForMultipleObjects(g_ThreadCnt, g_Handles, TRUE, INFINITE);

        int spell = GetTickCount() - start;
        printf("spell time : %d s, thread-count: %d\n", spell, g_ThreadCnt);

        getchar();

        return 0;
    }
```

最后结果

```
    su1986, => 79;4yx，888aeda4ab
```

由于算法开始有转小写，所以其时答案中所有字母都可以是大小写选择，答案不唯一。

转载请注明出处：https://anhkgg.github.io/kxctf2017-writeup5