# [转自“看雪论坛”]RtlAdjustPrivliege (http://bbs.pediy.com/showthread.php?t=76552)

转载

lyclowlevel 于 2010-10-12 17:26:00 发布 8668 收藏
分类专栏： win32非界面开发 文章标签： token null query c 汇编 windows


win32非界面开发 专栏收录该内容
22 篇文章 0 订阅
订阅专栏

前言：

今天逆向一个非常实用的函数RtlAdjustPrivliege

这个函数封装在NtDll.dll中（在所有DLL加载之前加载），被微软严格保密，就是说你在MSDN上查不到关于他的任何信息。

先来看看这个函数的定义(Winehq给出)：

引用:

```
NTSTATUS RtlAdjustPrivilege
(
 ULONG    Privilege,
 BOOLEAN  Enable,
 BOOLEAN  CurrentThread,
 PBOOLEAN Enabled
)
```

参数的含义：

引用:

```
Privilege  [In] Privilege index to change.
// 所需要的权限名称，可以到MSDN查找关于Process Token & Privilege内容可以查到
Enable  [In] If TRUE, then enable the privilege otherwise disable.
// 如果为True 就是打开相应权限，如果为False 则是关闭相应权限
CurrentThread  [In] If TRUE, then enable in calling thread, otherwise process.
// 如果为True 则仅提升当前线程权限，否则提升整个进程的权限
Enabled  [Out] Whether privilege was previously enabled or disabled.
//  输出原来相应权限的状态（打开 | 关闭）
```

很多人大概没有听说过他的大名，但是相信有很多人见过进程提权的过程
拷一段我写的提权上来吧
:

```
BOOL ImproveProcPriv()
{
    HANDLE token;
    //提升权限
    if(!OpenProcessToken(GetCurrentProcess(),TOKEN_ADJUST_PRIVILEGES,&token))
    {
        MessageBox(NULL,"打开进程令牌失败...","错误",MB_ICONSTOP);
        return FALSE;
    }
    TOKEN_PRIVILEGES tkp;
    tkp.PrivilegeCount = 1;
    ::LookupPrivilegeValue(NULL,SE_DEBUG_NAME,&tkp.Privileges[0].Luid);  //  获得 SE_DEBUG_NAME 特权
    tkp.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;
    if(!AdjustTokenPrivileges(token,FALSE,&tkp,sizeof(tkp),NULL,NULL))
    {
        MessageBox(NULL,"调整令牌权限失败...","错误",MB_ICONSTOP);
        return FALSE;
    }
    CloseHandle(token);
    return TRUE;
}
```

看看吧，这个提权快要累死了...
但是 如果有这个函数就不一样了，你可以只用一个函数就实现这个功能，甚至功能远多于上面的代码...
通过恰当的IDE设置和必要的Defination，上面这个函数的功能你完全可以通过一行代码来实现。

代码:

```
RtlAdjustPrivilege(SE_DEBUG_NAME,1,0,NULL);
```

正文：
下面我们看一下这个函数是怎么运行的，顺便学习下强大的IDA
IDA 载入ntdll.dll （我这里载入的是 WinDBG自动下载的 Symbol里面的英文版本 可能不同的Windows版本略有不同）

先把函数的原型给输入IDA 方便一下阅读，然后开始阅读汇编代码了（党和国家考验我们的时候到了）。
看看Graph View 真的是很牛啊...
看看函数最开头...

引用:

```
mov    edi, edi      ; 这句话是废指令
push   ebp
mov    ebp, esp
sub    esp, 30h      ; 48个字节的子过程域Auto变量
cmp    [ebp+CurrentThread], 1 ; 判断CurrentThread参数是否被指定为1
mov    eax, dword_7C97B0C8
mov    [ebp+var_4], eax
mov    eax, [ebp+Enabled]
mov    [ebp+lsEnabled], eax ; BOOL *lsEnabled = Enabled;
```

```
lea     eax, [ebp+var_28]
push    eax
jz      loc_7C93378B
```

判断是调整进程权限还是线程权限，

**CurrentThread == TRUE**

引用:

```
push    0
push    28h          ; TOKEN_ADJUST_PRIVILEGES | TOKEN_QUERY
push    0FFFFFFFEh     ; GetCurrentThread()
call    ZwOpenThreadToken
jmp     loc_7C929A7A
```

**CurrentThread == FALSE**

引用:

```
push    28h          ; TOKEN_ADJUST_PRIVILEGES | TOKEN_QUERY
push    0FFFFFFFFh     ; GetCurrentProcess()
call    NtOpenProcessToken
```

然后两个代码块同时指向这里

引用:

```
loc_7C929A7A:         ; 很明白了吧 判断进程/线程令牌是否成功被打开
test    eax, eax
jl      short loc_7C929AE4 ; 没成功则跳
```

若 执行成功

引用:

```
mov     eax, [ebp+Privilege]
mov     [ebp+dwPrivilege], eax
mov     al, [ebp+Enable]
xor     ecx, ecx       ; ecx清零
neg     al
push    esi
mov     [ebp+NewState], 1
mov     [ebp+var_C], ecx
sbb     eax, eax
and     eax, 2
mov     [ebp+var_8], eax
lea     eax, [ebp+ReturnLength] ; 实际返回长度
push    eax
lea     eax, [ebp+OldState]
push    eax          ; 旧的特权 指针
push    10h          ; sizeof(TOKEN_PRIVILEGES)
lea     eax, [ebp+NewState]
push    eax          ; 新的特权 指针
push    ecx          ; FALSE 因为上面有xor ecx,ecx
push    [ebp+TokenHandle]
call    NtAdjustPrivilegesToken ； 调用 AdjustPrivilegesToken提权
push    [ebp+TokenHandle]
mov     esi, eax       ; eax备份
call    ZwClose        ; 关闭 内核对象句柄
cmp     esi, 106h      ; 判断NtAdjustPrivilege执行情况 106h = STATUS_NOT_ALL_ASSIGNED
jz      loc_7C947DF2
```

判断是否执行成功之后，开始输出最后一个参数

引用:

```
cmp     [ebp+OldState], 0
mov     ecx, [ebp+IsEnabled]
jnz     loc_7C929E99
```

若 **OldState != 0** 则

引用:

```
mov    al, [ebp+Enable]      ; 应该很明显了 把Enable变量赋给al 也就是eax最后两位
```

若 **OldState == 0** 则

引用:

```
mov    eax, [ebp+var_18]
shr    eax, 1
and    al, 1
jmp    loc_7C929ADF
```

这个函数大致流程就是这样。

到这里差不多可以按一下传说中的F5了

```
int __stdcall RtlAdjustPrivilege(int Privilege, char Enable,
char CurrentThread, int Enabled)
{
  int result; // eax@2
  signed int AdjustResult; // esi@4
  char returnValue; // al@7
  int v7; // [sp+2Ch] [bp-4h]@1
  int IsEnabled; // [sp+4h] [bp-2Ch]@1
  int TokenHandle; // [sp+8h] [bp-28h]@2
  int dwPrivilege; // [sp+20h] [bp-10h]@4
  signed int NewState; // [sp+1Ch] [bp-14h]@4
  int v12; // [sp+24h] [bp-Ch]@4
  int v13; // [sp+28h] [bp-8h]@4
  int OldState; // [sp+Ch] [bp-24h]@4
  char ReturnLength; // [sp+0h] [bp-30h]@4
  unsigned int v16; // [sp+18h] [bp-18h]@11
  v7 = dword_7C97B0C8;
  IsEnabled = Enabled;
  if ( CurrentThread == 1 )
    result = ZwOpenThreadToken(-2, 40, 0, &TokenHandle);
  else
    result = NtOpenProcessToken(-1, 40, &TokenHandle);
  if ( result >= 0 )
  {
    dwPrivilege = Privilege;
    NewState = 1;
    v12 = 0;
    v13 = -(Enable != 0) & 2;
    AdjustResult = NtAdjustPrivilegesToken(TokenHandle, 0, &NewState, 16, &OldState, &ReturnLength);
    ZwClose(TokenHandle);
    if ( AdjustResult == 262 )
      AdjustResult = -1073741727;
    if ( AdjustResult >= 0 )
    {
      if ( OldState )
        returnValue = (v16 >> 1) & 1;
      else
        returnValue = Enable;
      *(_BYTE *)IsEnabled = returnValue;
```

```
      *(_BYTE *)pIsEnabled = fEnabled;
    }
    result = AdjustResult;
  }
  return result;
}
```

可读性好像仍然不高，看看这个...

```
/*********************************************************************************
*  RtlAdjustPrivilege          [NTDLL.@]
*
* Enables or disables a privilege from the calling thread or process.
*
* PARAMS
*  Privilege     [I] Privilege index to change.
*  Enable        [I] If TRUE, then enable the privilege otherwise disable.
*  CurrentThread [I] If TRUE, then enable in calling thread, otherwise process.
*  Enabled       [O] Whether privilege was previously enabled or disabled.
*
* RETURNS
*  Success: STATUS_SUCCESS.
*  Failure: NTSTATUS code.
*
* SEE ALSO
*  NtAdjustPrivilegesToken, NtOpenThreadToken, NtOpenProcessToken.
*
*/
NTSTATUS WINAPI
RtlAdjustPrivilege(ULONG Privilege,
                   BOOLEAN Enable,
                   BOOLEAN CurrentThread,
                   PBOOLEAN Enabled)
{
    TOKEN_PRIVILEGES NewState;
    TOKEN_PRIVILEGES OldState;
    ULONG ReturnLength;
    HANDLE TokenHandle;
    NTSTATUS Status;
    TRACE("(%d, %s, %s, %p)/n", Privilege, Enable ? "TRUE" :
"FALSE",
        CurrentThread ? "TRUE" : "FALSE", Enabled);
    if (CurrentThread)
    {
        Status = NtOpenThreadToken(GetCurrentThread(),
                                   TOKEN_ADJUST_PRIVILEGES | TOKEN_QUERY,
                                   FALSE,
                                   &TokenHandle);
    }
```

```
    else
    {
        Status = NtOpenProcessToken(GetCurrentProcess(),
                                    TOKEN_ADJUST_PRIVILEGES | TOKEN_QUERY,
                                    &TokenHandle);
    }
    if (!NT_SUCCESS(Status))
    {
        WARN("Retrieving token handle failed (Status %x)/n", Status);
        return Status;
    }
    OldState.PrivilegeCount = 1;
    NewState.PrivilegeCount = 1;
    NewState.Privileges[0].Luid.LowPart = Privilege;
    NewState.Privileges[0].Luid.HighPart = 0;
    NewState.Privileges[0].Attributes = (Enable) ? SE_PRIVILEGE_ENABLED : 0;
    Status = NtAdjustPrivilegesToken(TokenHandle,
                                     FALSE,
                                     &NewState,
                                     sizeof(TOKEN_PRIVILEGES),
                                     &OldState,
                                     &ReturnLength);
    NtClose (TokenHandle);
    if (Status == STATUS_NOT_ALL_ASSIGNED)
    {
        TRACE("Failed to assign all privileges/n");
        return STATUS_PRIVILEGE_NOT_HELD;
    }
    if (!NT_SUCCESS(Status))
    {
        WARN("NtAdjustPrivilegesToken() failed (Status %x)/n", Status);
        return Status;
    }
    if (OldState.PrivilegeCount == 0)
        *Enabled = Enable;
    else
        *Enabled = (OldState.Privileges[0].Attributes & SE_PRIVILEGE_ENABLED);
    return STATUS_SUCCESS;
}
```