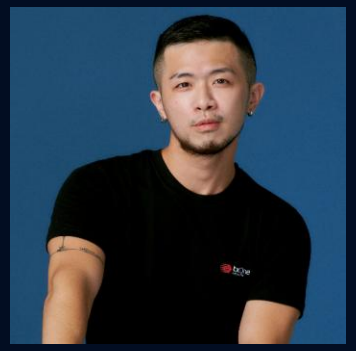
內核遊戲防護神仙大戰

從拆解到接管現代 Hyper-V 架構戰勝內核外掛威脅

Shenghao Ma, Team Lead
YaoDe Cai, Threat Researcher
PSIRT and Threat Research, TXOne Networks Inc.

April 22, 2025 @CYBERSEC 2025

TXOne Threat Researchers from TW



Team Lead, PSIRT and Threat Research at TXOne Networks Inc.

- 馬聖豪（@aaaddress1）目前為 TXOne Networks Inc. 產品資安事件應變暨威脅研究團隊 Team Lead 專研逆向工程分析超過十五年經驗並熱愛與涉略 漏洞技巧、Symbolic Execution、AI/ML、自然語言處理、編譯器實務 與 作業系統原理 等領域。
- 曾任 Black Hat USA、Black Hat MEA、DEFCON、CODEBLUE、S4、SECTOR、HITB、VXCON、HITCON、AVTOKYO、ROOTCON、CYBERSEC 等各個國內外年會講者與授課培訓，並著有全球熱銷中英資安書籍《Windows APT Warfare：惡意程式前線戰術指南》

Threat Researcher, PSIRT and Threat Research at TXOne Networks Inc.

- 蔡耀德（Legbone）目前為 TXOne Networks Inc. 產品資安事件應變暨威脅研究團隊 資安威脅研究員 對於 Windows 產品安全與防護破解技術有超過十年的經驗。
- 專注於系統核心安全、驅動攻擊/設計原理、 與多款知名商業反外掛防護的 Hyper-V 實現技術都有深度研究。他也曾分享研究與擔任多個國內知名研討會 的講者與教育訓練講師，如 CYBERSEC、HITCON、SITCON，並在第二十六屆資訊安全會議發表論文《針對未知程式動 態行為攔截之沙箱系統實作》

Outline

- 當今遊戲反作弊引擎與資安解決方案對於內核攻擊的偵測難題
- 微軟 Hyper-V 設計架構與其介面與系統內核界接存在的隱憂
- Riot Vanguard 反作弊引擎採用上述架構設計轉為偵測防線 (Demo)
- 結語：解決方案可能的使用策略

Exploiting Windows' vulnerabilities with Hyper-V: A Hacker's swiss army knife

> 利用 Hyper-V 攻擊 Windows 漏洞：駭客的瑞士軍刀

In-depth analysis on Valorant's Guarded Regions

> 深入逆向分析特戰英豪的防守禁域

Xyrem :: 18 min read (3833 words)

ERNW
providing security.

Virtual Secure Mode: Protections of Communication Interfaces

Aleksandar Milenkoski
amilenkoski@ernw.de

This work is part of the *Windows Insight* series. This series aims to assist efforts on analysing inner working principles, functionalities, and properties of the Microsoft Windows operating system. For general inquiries contact Aleksandar Milenkoski (amilenkoski@ernw.de) or Dominik Phillips (dphillips@ernw.de). For inquiries on this work contact the corresponding author (✉).

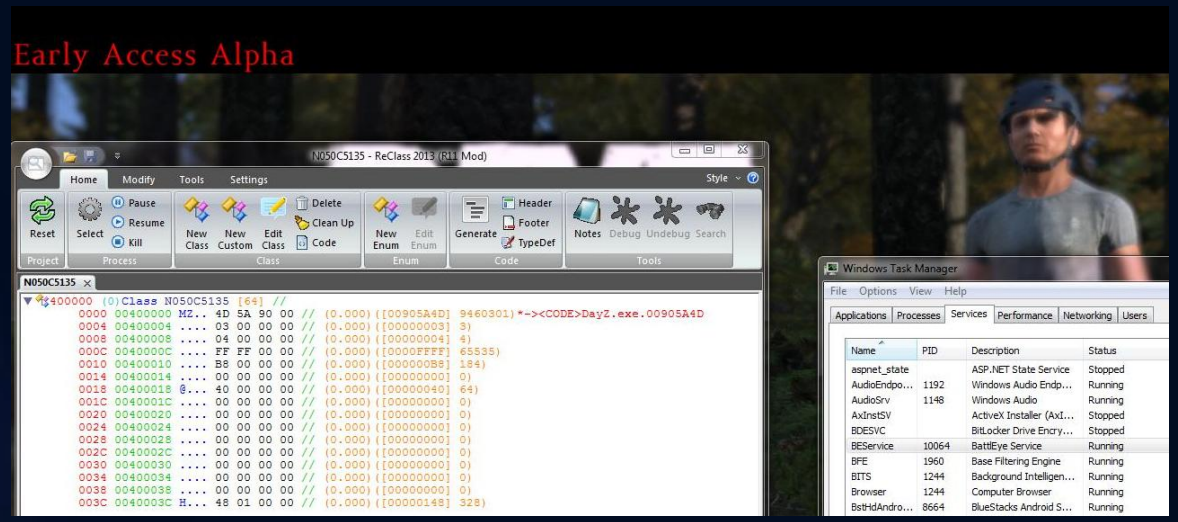
The content of this work has been created in the course of the project named 'Studie zu Systemaufbau, Protokollierung, Härtung und Sicherheitsfunktionen in Windows 10 (SiSyPHuS Win10)' (ger.) - 'Study of system design, logging, hardening, and security functions in Windows 10' (eng.). This project has been contracted by the German Federal Office for Information Security (ger., Bundesamt für Sicherheit in der Informationstechnik - BSI).

txOne networks

Kernel Abuse:
DKOM Is All You Need

OpenProcess

- (Douggem's Game Hacking 2015) ObRegisterCallbacks and countermeasures - Anti-Cheat BattleEye & EasyAntiCheat
- (XPN's InfoSec 2017) Windows Anti-Debug techniques - OpenProcess filtering
  - Cylance, AVG, Sophos, etc.
  - *"Hopefully this gives you a good introduction to this anti-debugging method used by quite a number of security products. If you found this interesting, there are plenty of Bug Bounties available, including a nice one for AVG on BugCrowd here, Cylance, Sophos... (although I wouldn't try raising the above as a vulnerability, DKOM will likely be out of scope"*



Windows Anti-Debug techniques - OpenProcess filtering

Posted on 2017-12-13 Tagged in reversing

This week I took a break from SYSTEM chasing to review some anti-debugging techniques. With quite a few Bug Bounty programs available relying on client-side applications, I thought I'd share one of the techniques used by numerous security products (and apparently game anti-cheat engines) to stop you from debugging core components, and just how we can go about bypassing this.

ObRegisterCallback internals

With our kernel debugger connection established, we will start with :`nt!ProcessType`

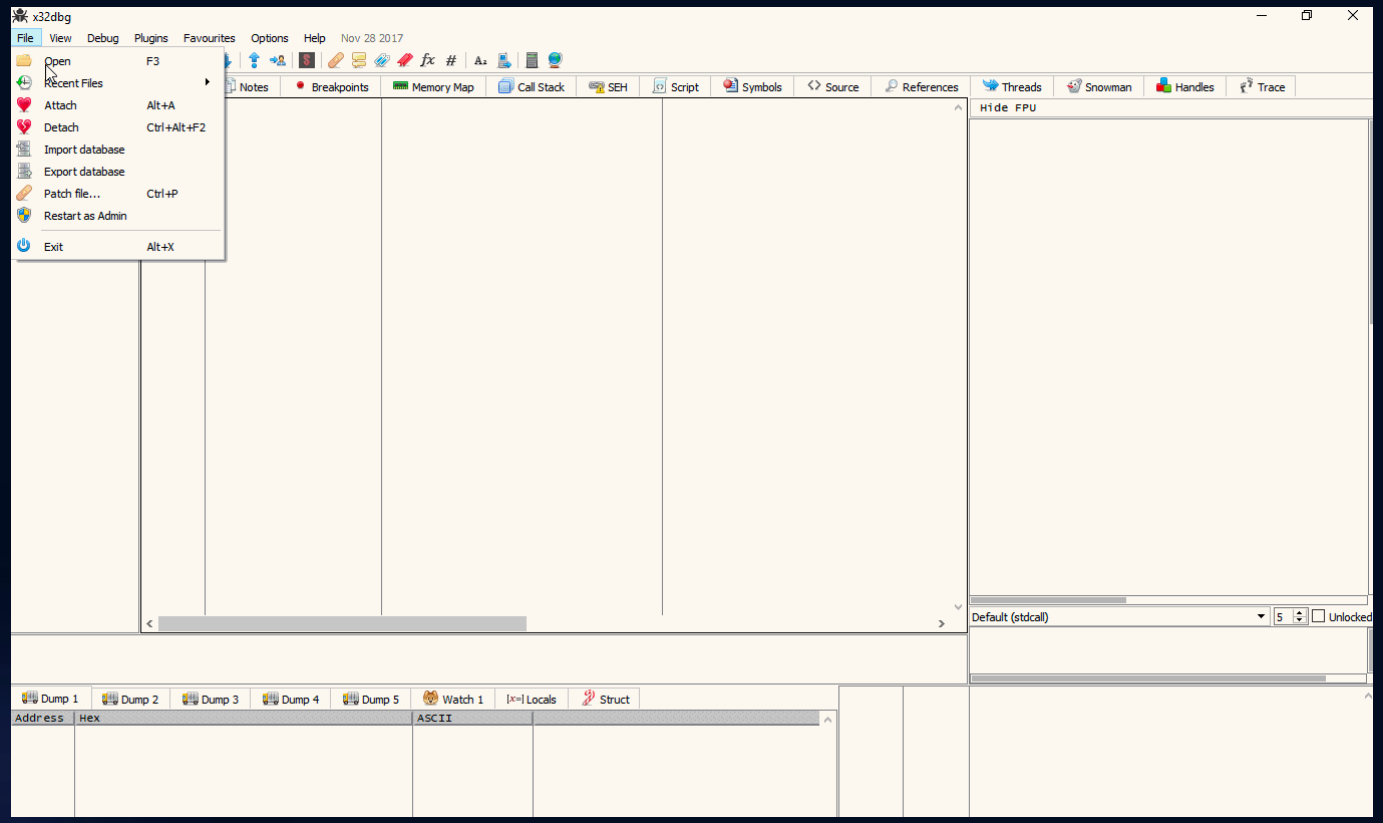
```
kd> dt nt!_OBJECT_TYPE poi(nt!PsProcessType)
+0x000 TypeList         : _LIST_ENTRY [ 0xffffcb82`dee6cf20 - 0xffffcb82`dee6cf20 ]
+0x010 Name             : _UNICODE_STRING "Process"
+0x020 DefaultObject    : (null)
+0x028 Index            : 0x7 ''
+0x02c TotalNumberOfObjects : 0x26
+0x030 TotalNumberOfHandles : 0xe8
+0x034 HighWaterNumberOfObjects : 0x26
+0x038 HighWaterNumberOfHandles : 0xea
+0x040 TypeInfo         : _OBJECT_TYPE_INITIALIZER
+0x0b8 TypeLock         : _EX_PUSH_LOCK
+0x0c0 Key              : 0x636f7250
+0x0c8 CallbackList     : _LIST_ENTRY [ 0xffffa002`d31bacd0 - 0xffffa002`d35d2450 ]
```

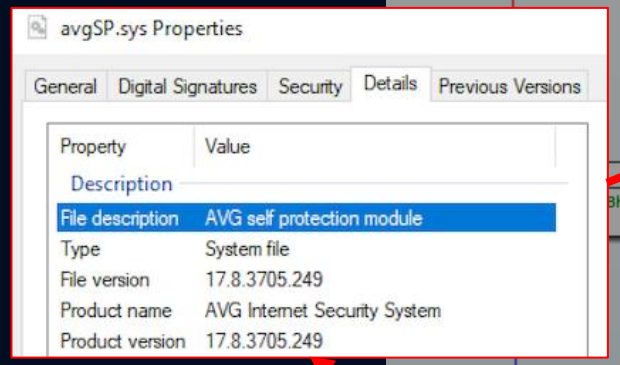
This symbol provides a pointer to a object, which defines the "Process" type and, of particular interest to us, a property. This property defines a list of callbacks which have been registered by `ObRegisterCallbacks`, and subsequently, each is called by the kernel when an attempt is made to acquire a process handle. Knowing this, we can traverse the list to find any registered handlers which may be interfering with our calls._`OBJECT_TYPECallbackListOpenProcess`

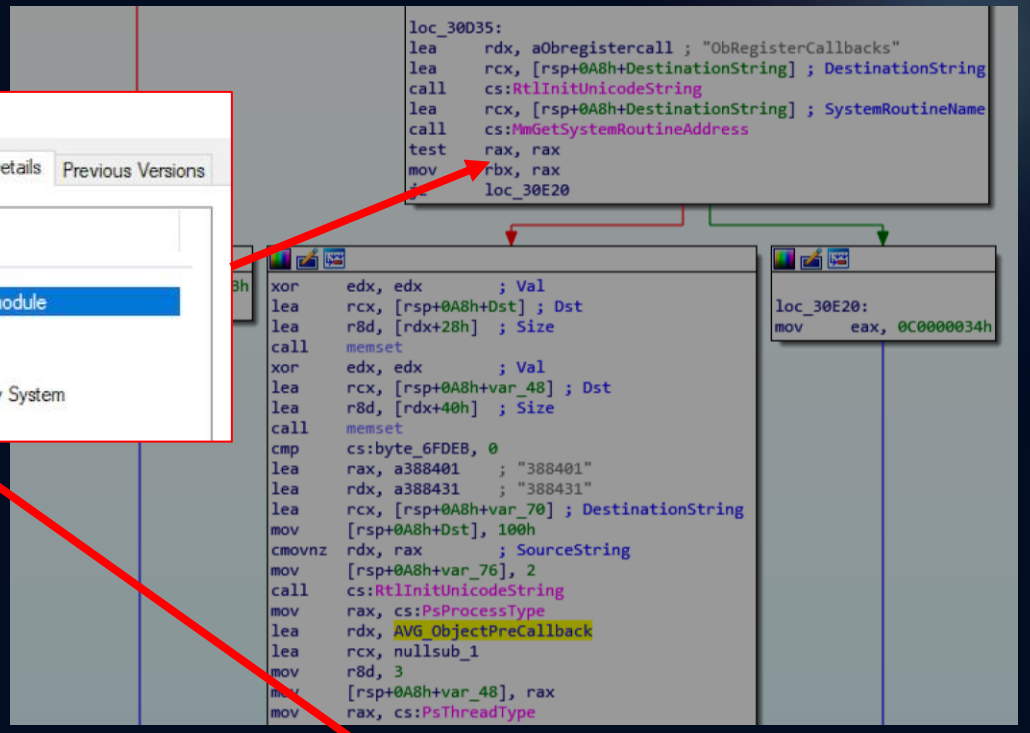
OpenProcess Filter

Once we have identified the correct address, we can NULL out the pointer to disable the handler using:

```
eq 0xffffa002`d31bacf8 0
```








```
Browse full module list
start             end                 module name
fffff805`ea000000 fffff805`ea071000   avgSP      (no symbols)
    Loaded symbol image file: avgSP.sys
    Image path: avgSP.sys
    Image name: avgSP.sys
    Browse all global symbols  functions  data
    Timestamp:        Tue Nov 14 11:23:32 2017 (5A0AD234)
    CheckSum:         00072070
    ImageSize:        00071000
    Translations:     0000.04b0 0000.04e4 0409.04b0 0409.04e4

Browse full module list
start             end                 module name
fffff805`e9760000 fffff805`e97ae000   WdFilter   (no symbols)
    Loaded symbol image file: WdFilter.sys
    Image path: WdFilter.sys
    Image name: WdFilter.sys
    Browse all global symbols  functions  data
    Timestamp:        ***** Invalid (F9B582DD)
    CheckSum:         0004E39B
    ImageSize:        0004E000
    Translations:     0000.04b0 0000.04e4 0409.04b0 0409.04e4
```


txOne networks

BYOVD

RANSOMWARE GANGS EXPLOIT A PARAGON PARTITION MANAGER BIONTDRV.SYS DRIVER ZERO-DAY

Pierluigi Paganini March 01, 2025

```
void WriteMemoryPrimitive(HANDLE Device,
    DWORD Size, DWORD64 Address, DWORD Value) {
    RTCORE64_MEMORY_READ MemoryRead {};
    MemoryRead.Address = Address;
    MemoryRead.ReadSize = Size;
    MemoryRead.Value = Value;
    DWORD BytesReturned;
    DeviceIoControl(
        Device, RTCORE64_MEMORY_WRITE_CODE,
        &MemoryRead, sizeof(MemoryRead),
        &MemoryRead, sizeof(MemoryRead),
                    &BytesReturned,
                    nullptr);
}
```

github.com/zeze-zeze/CYBERSEC2023-BYOVD-Demo

5 月 11 日（四）| 10:15 - 10:45

潛入核心：惡意程式與驅動程式的狼狽為奸

近幾年能看到濫用有漏洞或可利用的驅動程式的案例數量上升。議程中將會分享那些曾經被惡意程式武器化的驅動程式是如何被濫用的，以及此類型攻擊的目的，最後提供驅動程式開發商與系統管理員針對此種攻擊的防禦建議。

Zeze / TeamT5 杜浦數位安全 Security Researcher

HELP NET SECURITY

Zeljka Zorz, Editor-in-Chief, Help Net Security
August 20, 2024

0-day in Windows driver exploited by North Korean hackers to deliver rootkit (CVE-2024-38193)

'Silver Fox' APT Skirts Windows Blocklist in BYOVD Attack

There's an untapped universe of exploitable drivers in the wild today. By exploiting just one of them, attackers were able to defeat security tools and infect Asian citizens with Gh0stRAT.

kaspersky

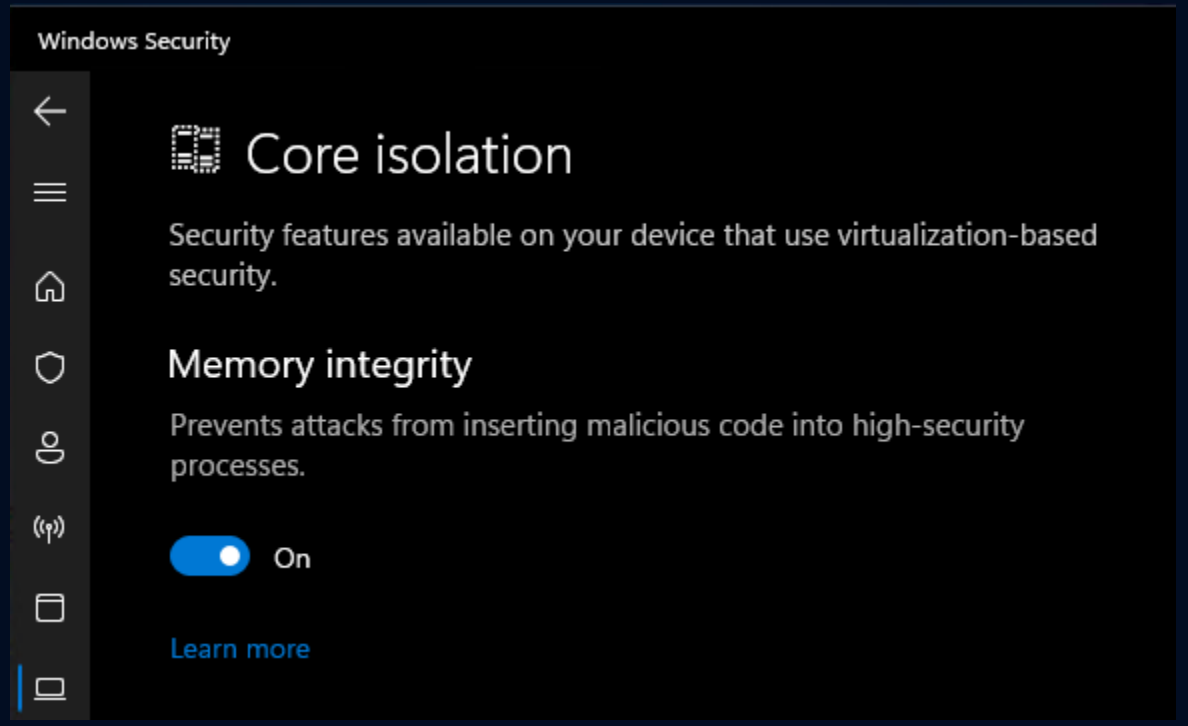
Kaspersky detects 23% increase in attacks targeting vulnerable Windows drivers

August 21, 2024

txOne networks

Virtualization-Based Security

- https://connormcgarr.github.io/hvci/

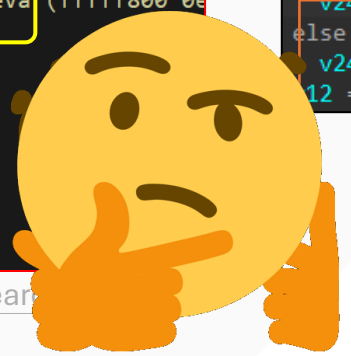


Use Hyper-V to Prevent RWX

- **VSM startup section of Windows Internals 7**
  - *"… Starts the VTL secure memory manager finally walks the NT loaded module list to establish each driver state, creating a NAR (normal address range) data structure for each one and compiling an Normal Table Entry (NTE) for every page composing the boot driver's sections. FURTHERMORE, THIS APPLIES THE CORRECT VTL 0 SLAT PROTECTION TO EACH DRIVER'S SECTIONS."*
  - Intel SLAT: CPU will follow this table when HVCI is on

```
kd> u HvcallInitiateHypercall
nt!HvcallInitiateHypercall:
fffff800`0e52eea0 4883ec28        sub     rsp,28h
fffff800`0e52eea4 488b0555e41900  mov     rax,qword ptr [nt!HvcallCodeVa (fffff800`0e
fffff800`0e52eeab ffd0            call    rax
fffff800`0e52eead 0f1f00          nop     dword ptr [rax]
fffff800`0e52eeb0 4883c428        add     rsp,28h
fffff800`0e52eeb4 c3              ret
fffff800`0e52eeb5 cc              int     3
fffff800`0e52eeb6 cc              int     3
kd> u poi(nt!HvcallCodeVa)
fffff800`0e0f9000 0f01c1          vmcall
fffff800`0e0f9003 c3              ret
```

https://msrc.microsoft.com/blog/2018/12/first-steps-in-hyper-v-resear



```
PFN = (*v4 >> 12) & 0xFFFFFFFFFFi64;
v34 = SKMI_GET_PROTECTION_FROM_VALID_PTE(v4);
v61 = v34;
if ( (SkmiFlags & 0x10) != 0 && (v34 & 2) != 0 )
{
  v36 = (PFN << 12) | SkmiProtectionToPte[v34] & 0xFFF0000000000FFFui64 | 0x200;
  if ( (*(v2 + 104) & 0x800000) != 0 && SkmiRelocateBootPage(v4, v35, 2i64) )
  {
    v36 ^= (*v4 ^ v36) & 0xFFFFFFFFF000i64;
  }
  else
  {
    if ( !SkmiClaimPhysicalPage(PFN) )
      return 0xC0000018i64;
    result = SkmiProtectSinglePage(PFN, 0x102u);
    if ( result < 0 )
      return result;
```

securekernel!SkmiProtectSinglePage

each of the boot-loaded drivers has each section (.text, etc.) protected by HVCI. This is done by iterating through each section of the boot-loaded drivers and applying the correct VTL 0 permissions.

```
__int64 __fastcall ShvlProtectContiguousPages(__int64 PFN, DWORD *a2, __int64 protectMask)
{
  char true; // r9
  __int64 var_PFN; // [rsp+30h] [rbp+8h] BYREF

  var_PFN = PFN;
  true = 1;
  return ShvlpProtectPages(&var_PFN, a2, protectMask, true);
}
```

```
if ( v14 > 0xC )
  v24 = ShvlpInitiateVariableHypercall(12, v11, 0, 0, v14, &v25);
else
  v24 = ShvlpInitiateFastHypercall(12, v11, 8 * v14 + 16, v14, &v25, 0i64, 0);
12 = v24;
```

Microsoft: Never touch my kernel code again!!!

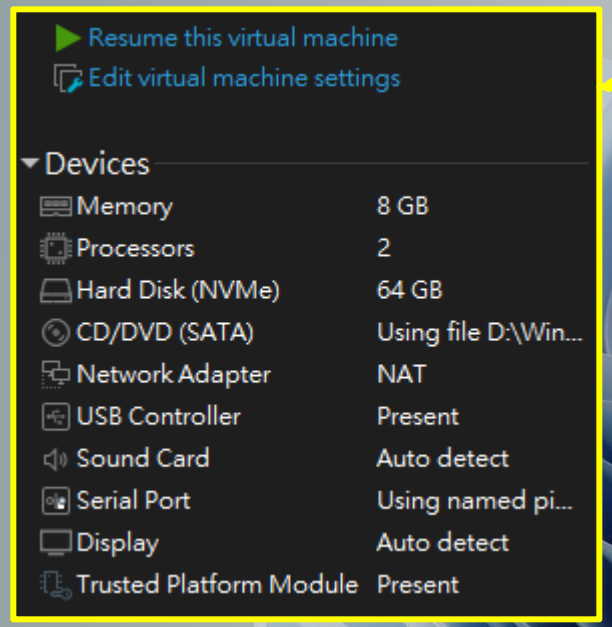
Hyper-V 101

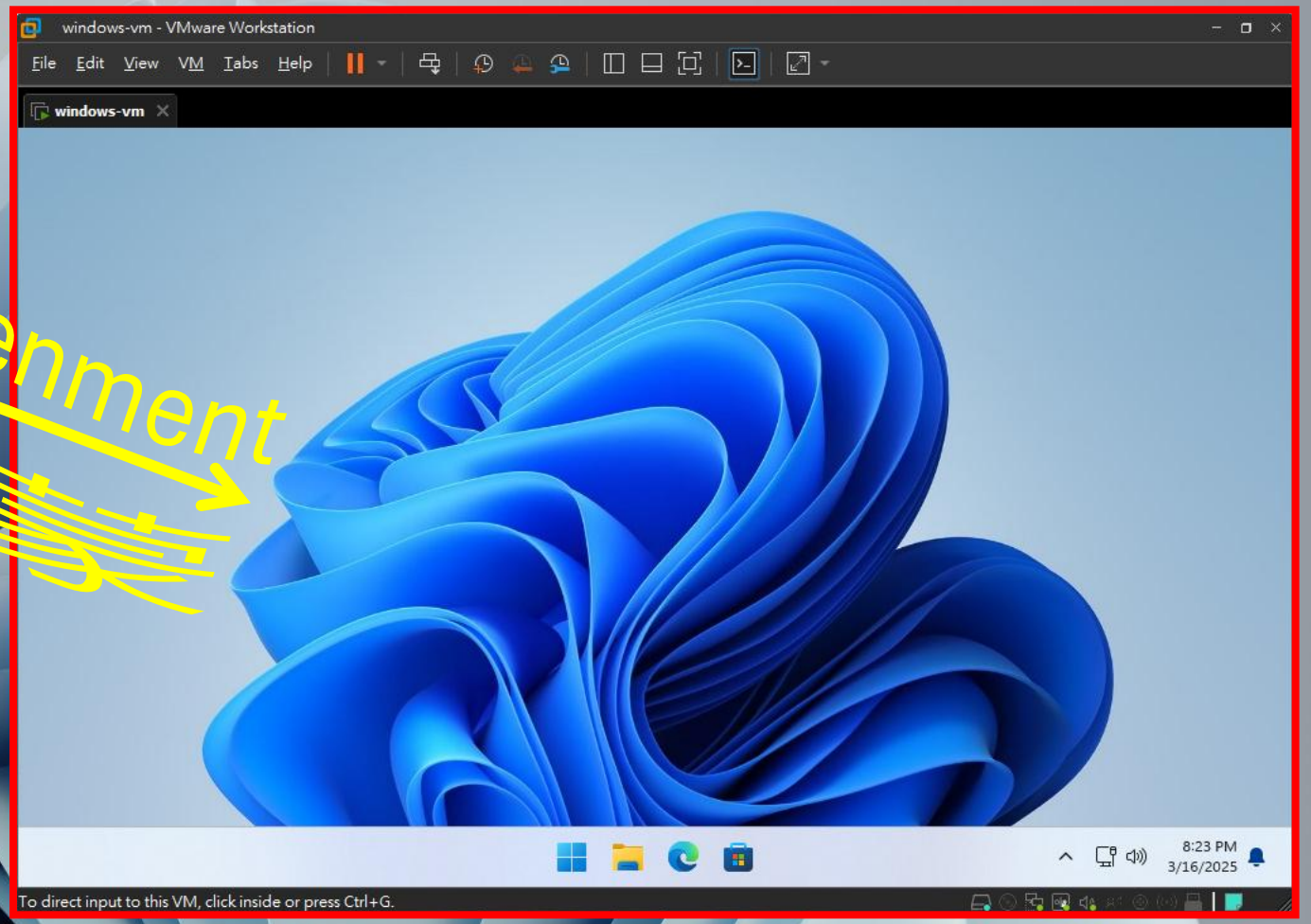
ntoskrnl (guest)

ntoskrnl (host)

Enlightenment

后文





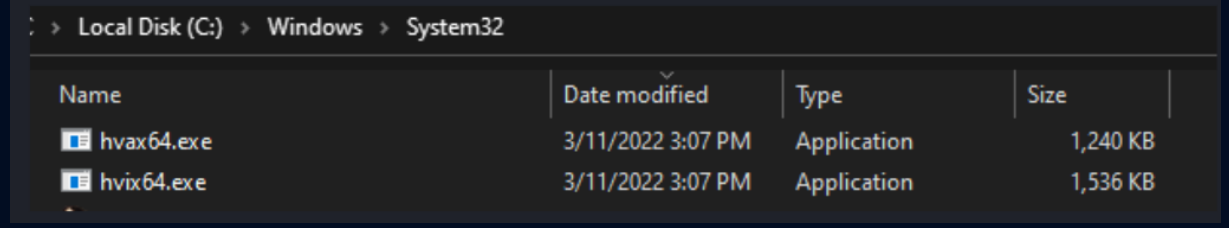
Hyper-V 合成設備

(網卡、磁碟、RAM...)

HyperCall Calling Convention

- 239 HyperCall Callbacks

The inner workings of Hyper-V can be analyzed by analyzing the modu
present inside the system root directory, where it is stored as hv
`hvax64.exe` for Intel and AMD processors, respectively.




```
CONST:FFFFF80000C00000 CONST           segment para public 'DATA' use64
CONST:FFFFF80000C00000                 assume cs:CONST
CONST:FFFFF80000C00000                 ;org 0FFFFF80000C00000h
CONST:FFFFF80000C00000 ; HVCallEntry HyperCallList[239]
CONST:FFFFF80000C00000 HyperCallList   HVCallEntry <offset sub_FFFFF8000022EE10, 0, 0, 0, 0, 0, 0, 43h>
CONST:FFFFF80000C00000                                 ; DATA XREF: sub_FFFFF80000211900+1073↑o
CONST:FFFFF80000C00000                                 ; sub_FFFFF8000028C3A8+267↑o
CONST:FFFFF80000C00000                 HVCallEntry <offset sub_FFFFF8000028E930, 1, 0, 8, 0, 0, 0, 46h>
CONST:FFFFF80000C00000                 HVCallEntry <offset sub_FFFFF800002235C0, 2, 0, 18h, 0, 0, 0, 46h>
CONST:FFFFF80000C00000                 HVCallEntry <offset sub_FFFFF80000221AD0, 3, 1, 18h, 8, 0, 0, 46h>
CONST:FFFFF80000C00000                 HVCallEntry <offset sub_FFFFF8000028F660, 4, 0, 8, 0, 20h, 0, 41h>
CONST:FFFFF80000C00000                 HVCallEntry <offset sub_FFFFF80000290640, 5, 0, 10h, 0, 0, 0, 43h>
CONST:FFFFF80000C00000                 HVCallEntry <offset sub_FFFFF8000028A560, 6, 8, 0, 0, 0, 0, 43h>
CONST:FFFFF80000C00000                 HVCallEntry <offset sub_FFFFF80000290830, 7, 0, 18h, 0, 0, 0, 43h>
CONST:FFFFF80000C00000                 HVCallEntry <offset sub_FFFFF8000021B950, 8, 0, 8, 0, 0, 0, 42h>
CONST:FFFFF80000C00000                 HVCallEntry <offset sub_FFFFF8000028F880, 9, 0, 8, 0, 0, 0, 43h>
```

```
struct HVCallEntry_t
{
    void* Callback;
    uint16_t CallCode;
    uint16_t RepCount;

    uint16_t InputSize;
    uint16_t InputSize2;

    uint16_t OutputSize;
    uint16_t OutputSize2;

    uint16_t HypercallGroupNumber;
};
```

Xyrem Engineering

Menu

Exploiting Windows' vulnerabilities with Hyper-V: A Hacker's swiss army knife

利用 Hyper-V 攻擊 Windows 漏洞：駭客的瑞士軍刀

Xyrem :: 11 min read (2289 words)

#Windows #HyperV #HyperDeceit
#Hypercall #Reverse Engineering

txOne networks

(24h2) nt!HvlpTryConfigureInterface

- SecureKernelRunning
  - Enable Kernel Core Isolation Yet?
  - LoaderBlock – HvCallCodeVa shared by VM host
  - If non host but enable SecureKernel?
  - Ntoskrnl will host this symbol on it own
- nt!HvCallCodeVa writeable

```
NTSTATUS HvlpTryConfigureInterface( _LOADER_PARAMETER_BLOCK* LoaderBlock )
{
    HviGetHypervisorFeatures( &HypervisorFeatures );
    if ( !HviIsHypervisorMicrosoftCompatible( ) ||
    !HypervisorFeatures.PartitionPrivileges.AccessHypercallMsrs )
        ...

    if ( LoaderBlock ) {
        Extension = LoaderBlock->Extension;
        MappedPhysPage = Extension->HypercallCodeVa;
        SecureKernelRunning = Extension->IumEnabled != 0;
        if ( MappedPhysPage )
            goto $SetupHypercallPtrs;
    }

    __writemsr( HV_X64_MSR_GUEST_OS_ID,
        uint16_t(NtBuildNumber) | ((*(uint8_t*)&CmNtCSDVersion | 0x1040A0000) << 16)
    HypercallPhysicalPage = __readmsr( HV_X64_MSR_HYPERCALL ) | 1;
```

```
    if ( HyperVisorFeatures.PartitionPrivileges.AccessMemoryPool || SecureKernelRunning )
        ...
    else {
        if ( !LoaderBlock ) {
            PhysicalAddress = MmGetPhysicalAddress( HvlpHypercallCodeVa );
            MappedPhysPage = HvlpHypercallCodeVa;
            AllocatedPhysPage = PhysicalAddress;
            goto $SetupHypercallPtrsAndHypercallMSRAndSetHypercallPhysicalPage;
        }

        MappedPhysPage = HalpAllocateEarlyPages( LoaderBlock,
            MEMORY_CACHING_TYPE::MmCached, &AllocatedPhysPage, PAGE_EXECUTE_READ );
        if ( MappedPhysPage )
        {
$SetupHypercallPtrsAndHypercallMSRAndSetHypercallPhysicalPage:
            HypercallPhysicalPage = AllocatedPhysPage.QuadPart ^
            (LOWORD( AllocatedPhysPage.LowPart ) ^ (unsigned __int16)HypercallPhysicalPage)

$SetupHypercallPtrsAndHypercallMSR:
            __writemsr( HV_X64_MSR_HYPERCALL, HypercallPhysicalPage );

$SetupHypercallPtrs:
            HvcallCodeVa = MappedPhysPage;
            _InterlockedExchange64( &HvlpHypercallCodeVa, MappedPhysPage );
            return STATUS_SUCCESS;
        }
```

```
CFGRO:0000000140E01860 ; __int64 (__fastcall *HvcallCodeVa)(_QWORD, _QWORD, _QWORD)
CFGRO:0000000140E01860 HvcallCodeVa    dq offset HvcallpNoHypervisorPresent
```

Choose segment to jump

Name	Start	End	R	W	X	D
Pad3	0000000140D9E000	0000000140E00000	R	W	.	.
CFGRO	0000000140E00000	0000000140E03000	R	W	.	.
Pad4	0000000140E03000	0000000141000000	R	W	.	.

(24h2) InitializeHypercallPageForGuest

- nt!HvCallCodeVa: executable shellcode to vmcall
  - Kernel shellcode payload
  - End with nop; nop; nop; nop … (\x90\x90\x90…)

```
    if ( HyperVisorFeatures.PartitionPrivileges.AccessMemoryPool || SecureKernelRunning )
        ...
    else {
        if ( !LoaderBlock ) {
            PhysicalAddress = MmGetPhysicalAddress( HvlpHypercallCodeVa );
            MappedPhysPage = HvlpHypercallCodeVa;
            AllocatedPhysPage = PhysicalAddress;
            goto $SetupHypercallPtrsAndHypercallMSRAndSetHypercallPhysicalPage;
        }

        MappedPhysPage = HalpAllocateEarlyPages( LoaderBlock,
            MEMORY_CACHING_TYPE::MmCached, &AllocatedPhysPage, PAGE_EXECUTE_READ );
        if ( MappedPhysPage )
        {
$SetupHypercallPtrsAndHypercallMSRAndSetHypercallPhysicalPage:
            HypercallPhysicalPage = AllocatedPhysPage.QuadPart ^
            (LOWORD( AllocatedPhysPage.LowPart ) ^ (unsigned __int16)HypercallPhysicalPage) & 0xFFF;

$SetupHypercallPtrsAndHypercallMSR:
            __writemsr( HV_X64_MSR_HYPERCALL, HypercallPhysicalPage );

$SetupHypercallPtrs:
            HvcallCodeVa = MappedPhysPage;
            _InterlockedExchange64( &HvlpHypercallCodeVa, MappedPhysPage );
            return STATUS_SUCCESS;
        }
```

```
NTSTATUS InitializeHypercallPageForGuest(QWORD FContext) {
  HypercallPhysicalPage = 0;
    ...
  while ( 1 ) {
    v6 = *(QWORD*)(FContext + 0x120);
    v7 = (v6 & 0x20000) != 0 ? 3 : ((v6 & 0x10000) != 0) + 1;
    if ( Idx >= v7 )
      break;

    Status = TranslateGuestToHostPA(
        *(QWORD *)(FContext + 304),
        (DWORD *)(FContext + 8816), &HypercallPhysicalPage
    );
    if ( Status ) ...

    HypercallPagePfn = HypercallPhysicalPage >> 12;
    *(QWORD*)(FContext + 0x46C0 * Idx + 0x3770) = HypercallPhysicalPage >> 12;

    MappedPage = MapHostPA(NtCurrentTeb()->NtTib.ExceptionList, HypercallPagePfn, 6);

    // Copy the vmcall and return stub to the mapped page.
    memcpy((void *)MappedPage, VmcallStubAddress, (unsigned int)Size);

    // Basic sanity check.
    if ( Size != 4096 ) // Setting the rest of the page with nops \x90.
      memset((void *)(Size + MappedPage), 0x90, 4096 - Size);
    UnmapHostVA(NtCurrentTeb()->NtTib.ExceptionList, MappedPage, 0);

    ++Idx;
  }

  return Status;
}
```

```
; HVCallEntry HyperCallList[239]
HyperCallList   HVCallEntry <offset sub_FFFFF8000022EE10, 0, 0, 0, 0, 0, 0, 43h>
                                        ; DATA XREF: sub_FFFFF80000211900+1073↑o
                                        ; sub_FFFFF8000028C3A8+267↑o
                HVCallEntry <offset sub_FFFFF8000028E930, 1, 0, 8, 0, 0, 0, 46h>
                HVCallEntry <offset sub_FFFFF800002235C0, 2, 0, 18h, 0, 0, 0, 46h>
                HVCallEntry <offset sub_FFFFF80000221AD0, 3, 1, 18h, 8, 0, 0, 46h>
                HVCallEntry <offset sub_FFFFF8000028F660, 4, 0, 8, 0, 20h, 0, 41h>
                HVCallEntry <offset sub_FFFFF80000290640, 5, 0, 10h, 0, 0, 0, 43h>
                HVCallEntry <offset sub_FFFFF8000028A560, 6, 8, 0, 0, 0, 0, 43h>
                HVCallEntry <offset sub_FFFFF80000290830, 7, 0, 18h, 0, 0, 0, 43h>
                HVCallEntry <offset sub_FFFFF8000021B950, 8, 0, 8, 0, 0, 0, 42h>
                HVCallEntry <offset sub_FFFFF8000028F880, 9, 0, 8, 0, 0, 0, 43h>
                HVCallEntry <offset sub_FFFFF80000291000, 0Ah, 0, 10h, 0, 0, 0, 43h>
                HVCallEntry <offset sub_FFFFF80000220BB0, 0Bh, 0, 10h, 0, 0, 0, 45h>
```

```
__int64 __fastcall HalpHvInitSystem(int a1)
{
  if ( a1 == 7 )
    HalpHvInitDiscard();
  return 0i64;
}
```

```
void __fastcall HalpHvInitDiscard()
{
  _HAL_INTEL_ENLIGHTENMENT_INFORMATION EnlightenmentInformation; // [rsp+20h

  memset(&EnlightenmentInformation, 0, sizeof(EnlightenmentInformation));
  if ( pHvlGetEnlightenmentInfo )
  {
    pHvlGetEnlightenmentInfo(&EnlightenmentInformation);
    if ( !EnlightenmentInformation.Reserved0
      && EnlightenmentInformation.SpinCountMask
      && ((EnlightenmentInformation.SpinCountMask + 1) & EnlightenmentInforma
    {
      HalpEnlightenment = *&EnlightenmentInformation.Enlightenments;
      dword_140C4A21C = EnlightenmentInformation.SpinCountMask;
      qword_140C4A220 = EnlightenmentInformation.LongSpinWait;
      qword_140C4A228 = EnlightenmentInformation.GetReferenceTime;
      qword_140C4A248 = EnlightenmentInformation.MapDeviceInterrupt;
```

```
void __fastcall HvlGetEnlightenmentInfo(_HAL_INTEL_ENLIGHTENMENT_INFORMATION *Enlightenment
{
  memset(EnlightenmentInformation, 0, sizeof(_HAL_INTEL_ENLIGHTENMENT_INFORMATION));
  EnlightenmentInformation->Enlightenments = HvlEnlightenments;
  EnlightenmentInformation->HypervisorConnected = HvlHypervisorConnected != 0;
  EnlightenmentInformation->SpinCountMask = HvlLongSpinCountMask;
  if ( (HvlEnlightenments & 0x40) != 0 )
    EnlightenmentInformation->LongSpinWait = HvlNotifyLongSpinWait;

  if ( (HvlEnlightenments & 0x100) != 0 )
    EnlightenmentInformation->GetReferenceTime = HvlGetReferenceTimeUsingTscPage;

  if ( (HvlEnlightenments & 0x4000) != 0 )
    EnlightenmentInformation->SyntheticClusterIpi = HvlSendSyntheticClusterIpi;

  if ( (HvlEnlightenments & 0x10000) != 0 )
  {
    EnlightenmentInformation->SetSystemSleepProperty = HvlSetSystemSleepProperty;
    EnlightenmentInformation->EnterSleepState = HvlEnterSleepState;
    EnlightenmentInformation->NotifyDebugDeviceAvailable = HvlNotifyDebugDeviceAvailable;
  }

  EnlightenmentInformation->VpStartEnabled = HvlHalVpStartEnabled;
  if ( (HvlEnlightenments & 0x8000) != 0 )
  {
    EnlightenmentInformation->StartVirtualProcessor = HvlHalStartVirtualProcessor;
    EnlightenmentInformation->GetVpIndexFromApicId = HvlHalGetVpIndexFromApicId;
  }

  if ( (HvlEnlightenments & 0x10) != 0 )
  {
    EnlightenmentInformation->EndOfInterrupt = HvlEndSystemInterrupt;
```

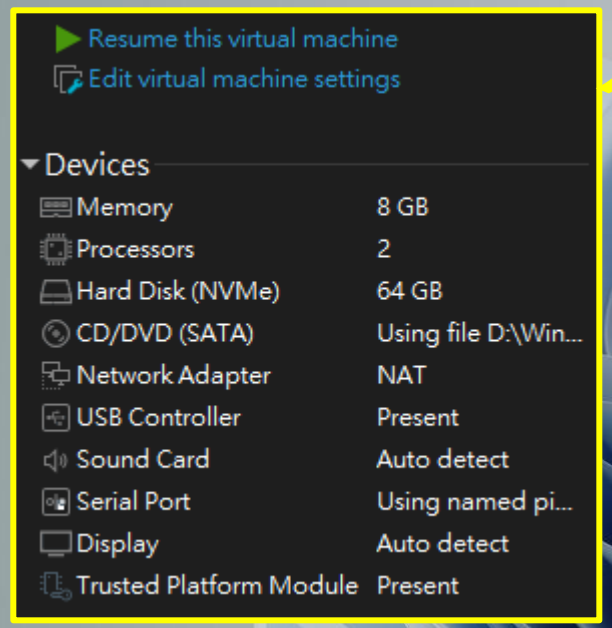
txOne networks

ntoskrnl (guest)

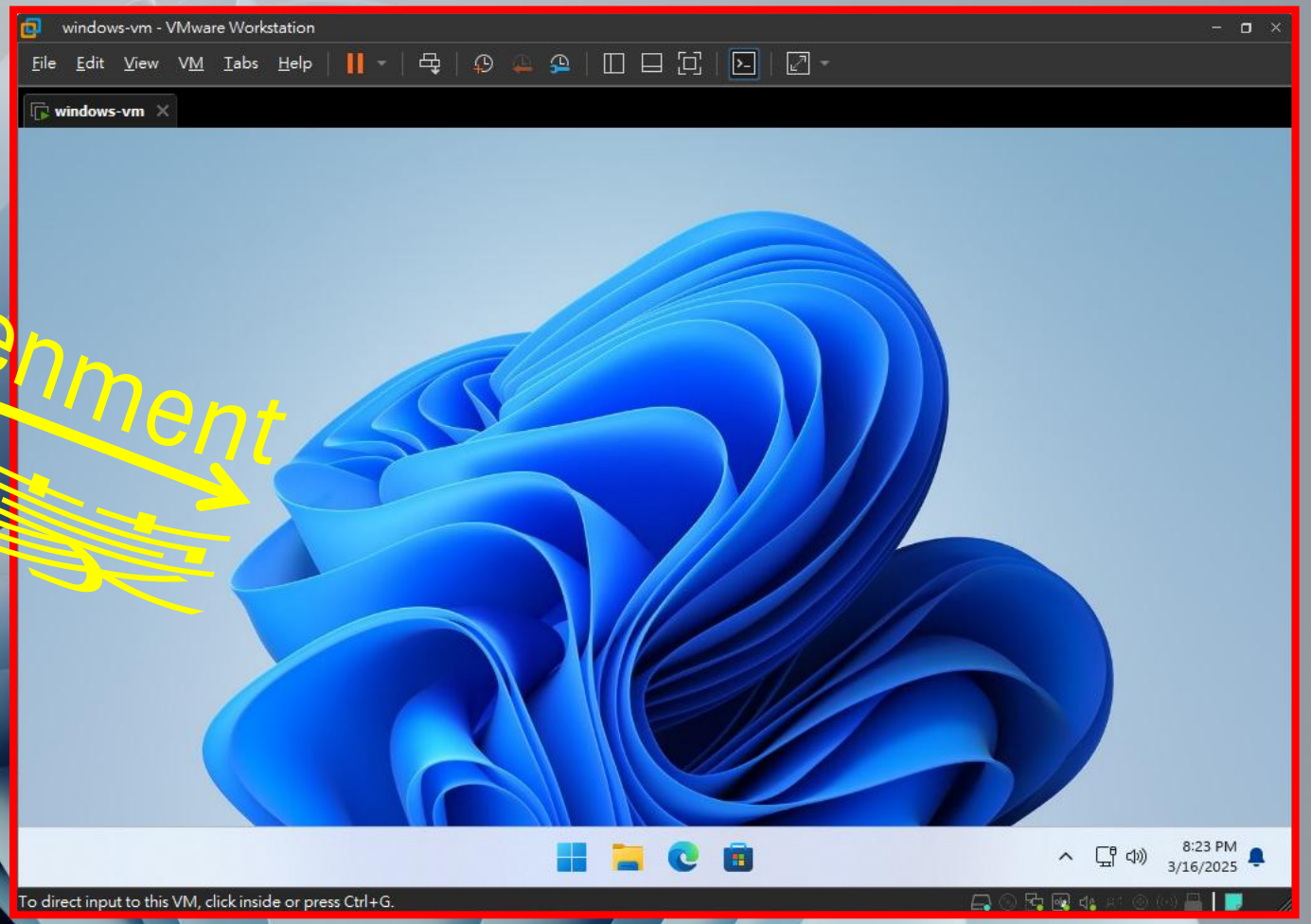
ntoskrnl (host)

Enlightenment

后文



Hyper-V 合成設備
(網卡、磁碟、RAM...)

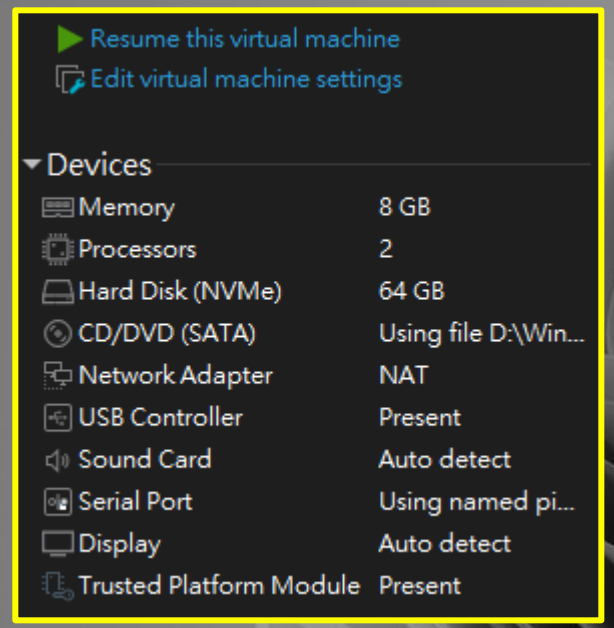


11:23
03/17

ntoskrnl (guest)

ntoskrnl (host)

Enlightenment

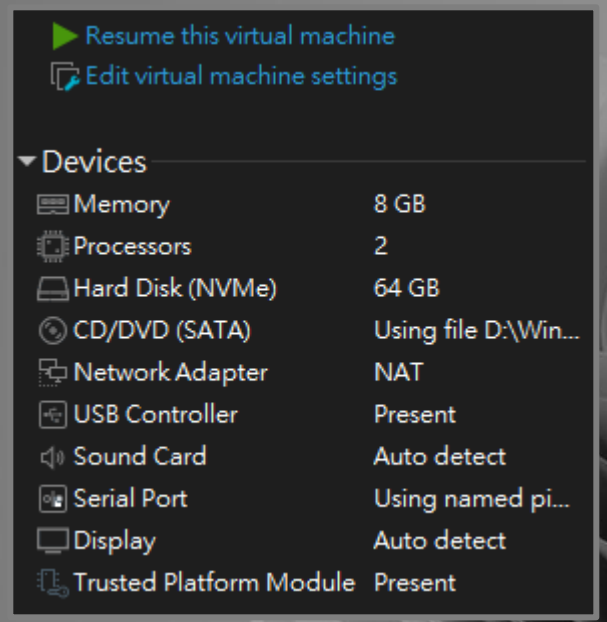


Hyper-V 合成設備
(網卡、磁碟、RAM...)

nt!HvcallCodeVa (nt!HvlGetEnlightenmentInfo[indx])


ntoskrnl (host)
Enlightenment
ntoskrnl (guest)
Resume this virtual machine
Edit virtual machine settings
Devices
Memory 8 GB
Processors 2
Hard Disk (NVMe) 64 GB
CD/DVD (SATA) Using file D:\Win...
Network Adapter NAT
USB Controller Present
Sound Card Auto detect
Serial Port Using named pi...
Display Auto detect
Trusted Platform Module Present
windows-vm - VMware Workstation
File Edit View VM Tabs Help
windows-vm
8:23 PM
3/16/2025
To direct input to this VM, click inside or press Ctrl+G.
nt!HvcallCodeVa (nt!HvlGetEnlightenmentInfo[indx])
Writable
Writable
11:23
03/17

First, we need to figure out how to intercept these hypercalls. And as discussed in the previous section, `HvcallInitiateHypercall`, and `HvcallFastExtended` are responsible for handling hypercalls, which both call `HvcallCodeVa`, which points to a `vmcall/vmmcall` as mentioned in the previous section of the post. Intercepting this would be easy as one can overwrite the hypercall stub pointer without dealing with any hindrance, such as VBS or KPP (Patchguard). Secondly, we need to set the Enlightenments for the OS to know that Hyper-V provides `x` capabilities for the guest.



✨任意 Hyper-V指令劫持 ✨

nt!HvcallCodeVa (nt!HvlGetEnlightenmentInfo[indx])

Writable

Writable


分時多工劫持?

(24h2) nt!SwapContext

- nt!SwapContext() 用於不同執行緒的上下文切換
  - 對系統內核與晶片而言，切換執行緒上下文無關乎是否同一處理序
  - 當切換執行緒時遇到「切換至不同處理序」時，將會刷新 CR3 暫存器 —— 即修改 KPROCESS.DirectBase 來切換當前執行緒允許觸碰的記憶體分頁記錄、與刷新 TLB

➢ RiotVanguard：嘿，那為何我們不劫持內核上下文切換，使其他人看不到遊戲記憶體的內核分頁就好呢？;)

  ➢ 不一定僅在切換不同處理序時，才對 CR3 做處理。而是當執行緒（即當前晶片運行身份）切換至「非遊戲處理序」時，便在 Hyper-V 層掛鉤並將遊戲自身內核分頁記錄隱藏起來。

```
if ( (HvlEnlightenments & 1) ≠ 0 )
{
  HvlSwitchVirtualAddressSpace(DirectoryTablebase);
}
else
{
  __writecr3(DirectoryTablebase);
  if ( (KiKvaShadow & 1) ≠ 0 && (DirectoryTablebase & 2) == 0 )
  {
    CR4Value = __readcr4();
    CR4Value ^= 0x80ui64;
    __writecr4(CR4Value);
    __writecr4(CR4Value ^ 0x80);
  }
```

Xyrem Engineering | Menu ▾

In-depth analysis on Valorant's Guarded Regions

深入逆向分析特戰英豪的防守禁域

Xyrem :: 18 min read (3833 words)

```
// Sanity checks to check if its executing under the game's context.
if ( PsGetThreadProcess(CurrentThread) != Data::GameProcess
  || __readcr3() != Data::GameCR3 )
  return;

bool WriteToCR3 = true;

// Copying the content of the current game cr3 to the new cloned cr3.
memmove(Data::CloneVirtCR3, Data::VirtGameCR3, 0x1000);
Data::CloneVirtCR3[Data::FreePML4Index] = Data::ShadowPML4Value;

// Loop through the whitelisted threads array, and if the current
// running thread, is inside it, then allow cr3 overwrite.
for ( int ThreadIdx = 0; ThreadIdx < Data::ThreadCount; ThreadIdx++ )
    WriteToCR3 = Data::ThreadArray[ThreadIdx] == CurrentThread;
    if ( WriteToCR3 ) break;
}

DoTask:
// Writing the cr3 to the clone cr3.
if ( WriteToCR3 ) __writecr3(Data::CloneCR3);

// Flushing TLB by toggling CR4:PGE.
if ( CanFlushTLB ) {
  uint64_t OriginalCR4 = __readcr4();
  __writecr4(OriginalCR4 ^ 0x80);
  __writecr4(OriginalCR4);
}
```

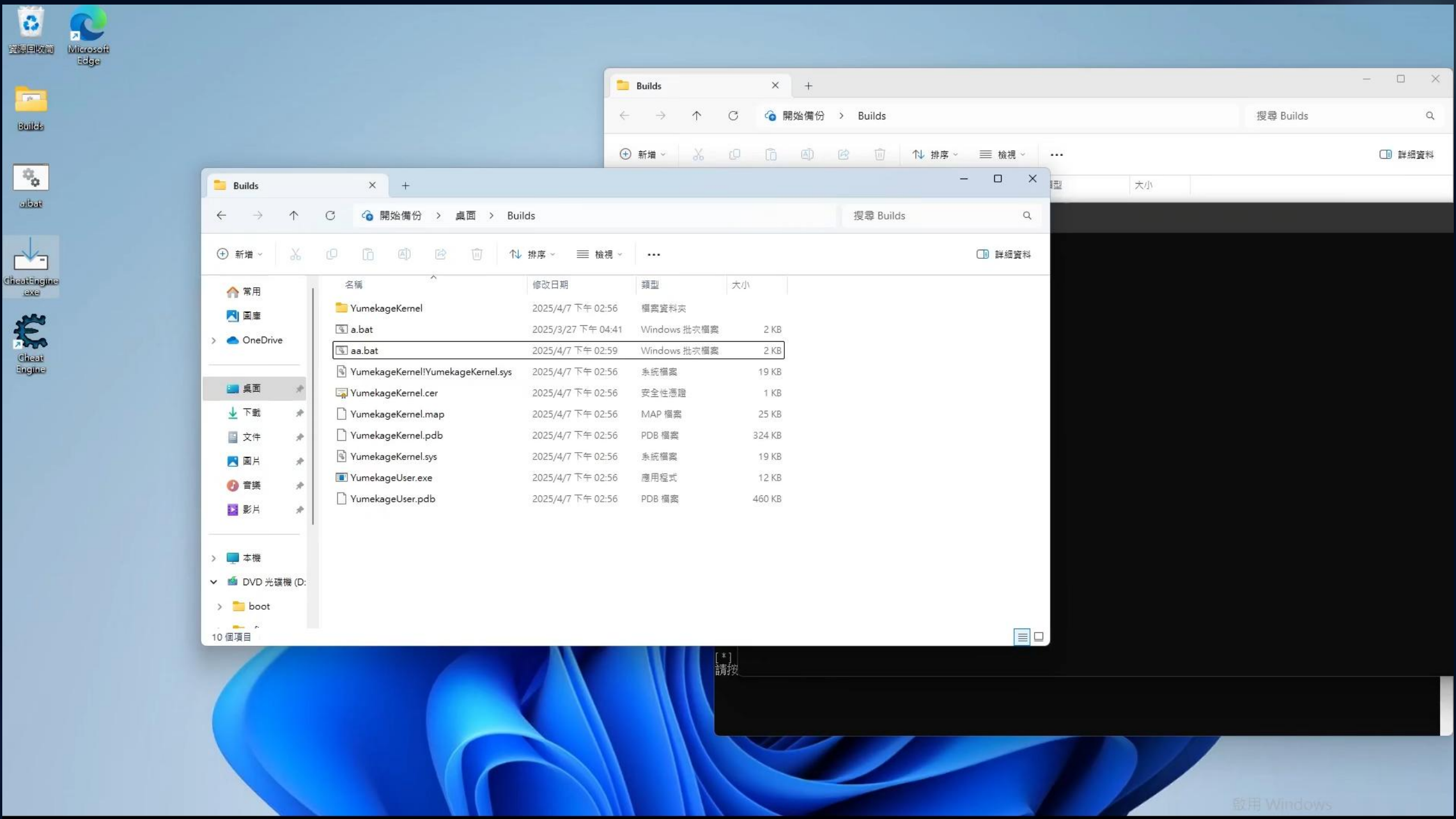
(24h2) nt!SwapContext

- nt!SwapContext() → HvlNotifyLongSpinWait
  - SwapContext 函數呼叫 HvlNotifyLongSpinWait() 作為 Hyper-V 調度延遲等待下一條指令以達到節省效能
  - 而 HvlNotifyLongSpinWait 是對 Hyper-V 的超級調用，可以使用上述方法進行攔截

```
if ( NewThread->Running )
{
  v28 = 0;
  do
  {
    if ( (++v28 & HvlLongSpinCountMask) == 0
      && (HvlEnlightenments & 0x40) != 0
      && (unsigned __int8)KiCheckVpBackingLongSpinWaitHypercall() )
    {
      HvlNotifyLongSpinWait(v28);
    }

    _mm_pause();
  }
  while ( NewThread->Running );
}

NewThread->Running = 1;
```

```
__int64 __fastcall HvlNotifyLongSpinWait(uint a1)
{
  return HvcallInitiateHypercall(0x10008i64, a1, 0i64);
}
```

```
DWORD64 __stdcall HvcallInitiateHypercall(__int64 a1, __int64 a2, __int64 a3)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  memset(v13, 0, sizeof(v13));
  if ( (BYTE4(xmmword_140D1EA90) & 0x10) != 0 )
  {
    v6 = 1;
    EtwGetKernelTraceTimestamp(v13, 0xA0000010i64);
  }
  else
  {
    v6 = 0;
  }
  v7 = HvcallCodeVa(a1, a2, a3);
  if ( v6 )
  {
    v12 = 0;
    v9 = a1;
    v10 = BYTE2(a1) & 1;
    v11 = a1 < 0;
    EtwTraceTimedEvent(0xF72, 0xA0000010, &v9, 8, 0x401A02, v13);
  }
  return v7;
```


資源回收筒
Microsoft Edge
Builds
a.bat
CheatEngine.exe
Cheat Engine
Builds
開始備份 › Builds
搜尋 Builds
新增
排序
檢視
詳細資料
大小
Builds
開始備份 › 桌面 › Builds
搜尋 Builds
新增
排序
檢視
詳細資料
常用
圖庫
OneDrive
桌面
下載
文件
圖片
音樂
影片
本機
DVD 光碟機 (D:
boot
名稱
修改日期
類型
大小
YumekageKernel 2025/4/7 下午 02:56 檔案資料夾
a.bat 2025/3/27 下午 04:41 Windows 批次檔案 2 KB
aa.bat 2025/4/7 下午 02:59 Windows 批次檔案 2 KB
YumekageKernel!YumekageKernel.sys 2025/4/7 下午 02:56 系統檔案 19 KB
YumekageKernel.cer 2025/4/7 下午 02:56 安全性憑證 1 KB
YumekageKernel.map 2025/4/7 下午 02:56 MAP 檔案 25 KB
YumekageKernel.pdb 2025/4/7 下午 02:56 PDB 檔案 324 KB
YumekageKernel.sys 2025/4/7 下午 02:56 系統檔案 19 KB
YumekageUser.exe 2025/4/7 下午 02:56 應用程式 12 KB
YumekageUser.pdb 2025/4/7 下午 02:56 PDB 檔案 460 KB
10 個項目
[*]
請按

Takeaways

- **Hyper-V 架構面存有工程設計考量的劫持機會**
  - 雖然微軟未開源，但通過對 Hyper-V 平台逆向分析得以知道 DKOM 內核任意寫的其他利用可能性
    部分前線遊戲反外掛引擎已經在落地採用於遊戲防護。

- **是一個良好的藍隊偵測新防線**
  - 雖然 nt!HvCallCodeVa 指針可寫但 KPP（Kernel Patch Protection）對其保護，
    因此 BYOD 類型的 DKOM 攻擊是無法濫用 Hyper-V 工程上弱點進行利用。
  - 對於防禦方，可以將驅動配置為 SERVICE_BOOT_START 於系統開機 Boot Loader 掛載階段劫持 Hyper-V，此時 KPP 防護
    尚未喚醒、因此不會導致藍屏 → 例如英雄聯盟關閉並恢復反外掛引擎運作，會要求強制重開機便是如此。

- Source
  - github.com/Xyrem/Yumekage
    Demo proof of concept for shadow regions, and implementation of HyperDeceit.
  - github.com/Xyrem/HyperDeceit
    HyperDeceit is the ultimate all-in-one library that emulates Hyper-V for Windows, giving you the ability to intercept and manipulate operating system tasks with ease.

Closing notes

It should be noted that the `HvcallCodeVa` **IS** protected by KPP (Patch Guard), hence it should be noted that this project must be either loaded before patch guard has fully initialized.


txOne
networks