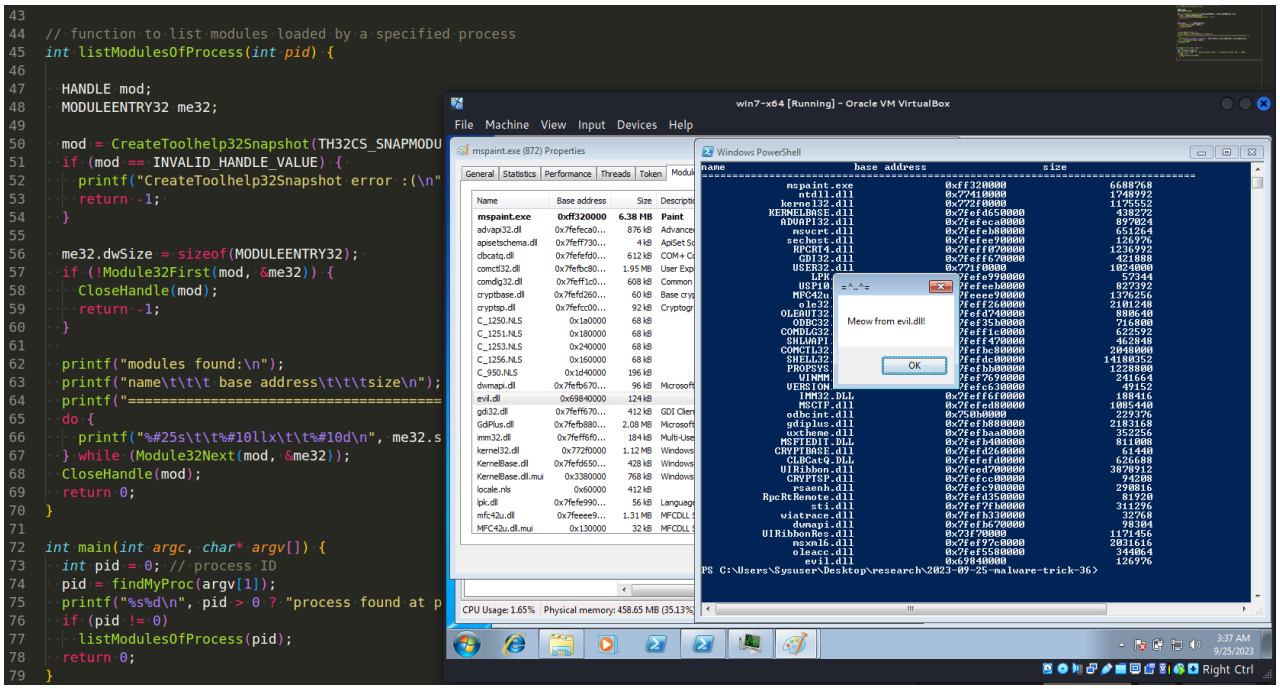
Malware development trick - part 36: Enumerate process modules. Simple C++ example.

cocomelonc.github.io/malware/2023/09/25/malware-trick-36.html

September 25, 2023

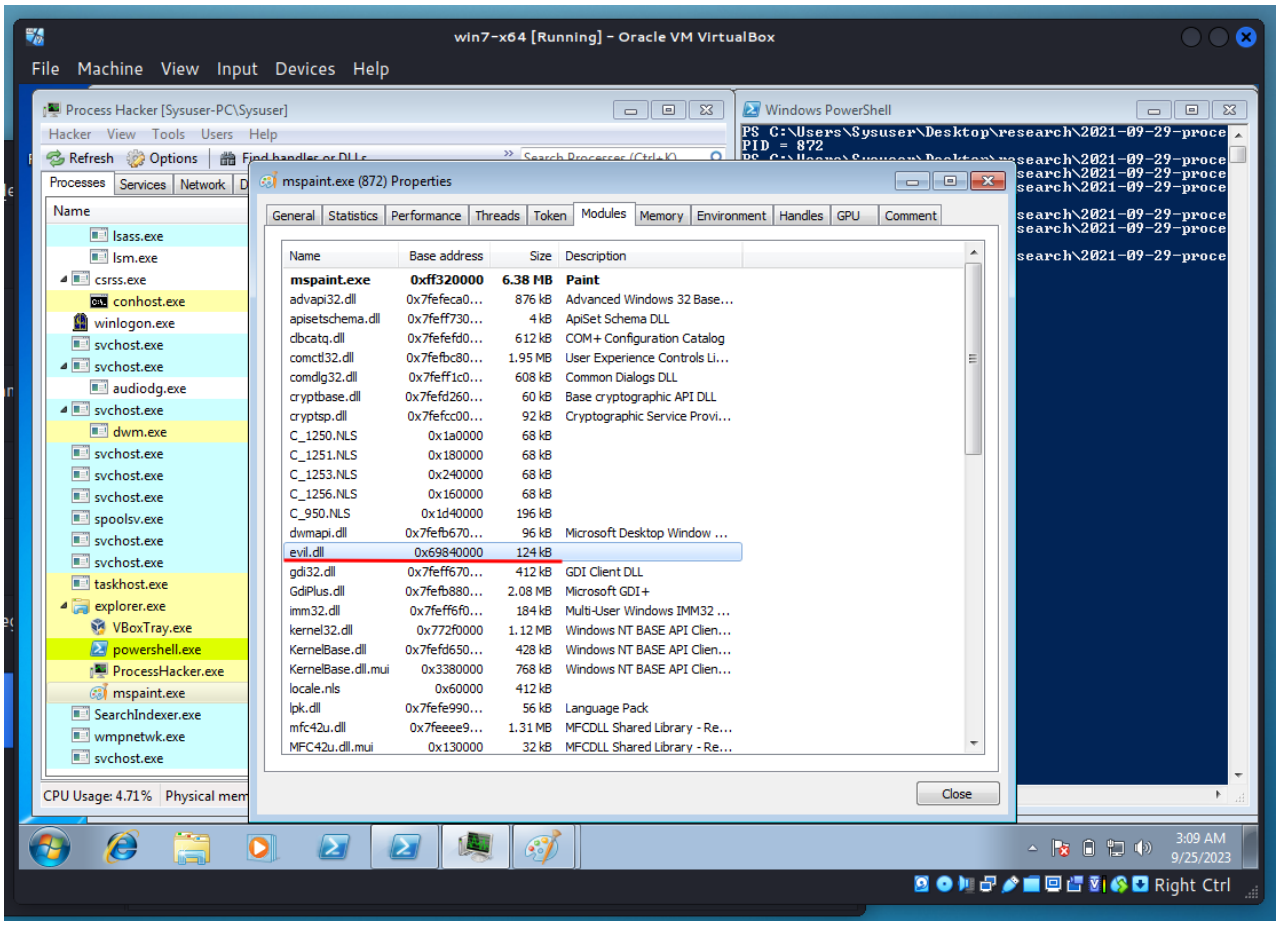
3 minute read

Hello, cybersecurity enthusiasts and white hackers!



Today, this post is the result of my own research on another popular malware development trick: get list of modules of target process.

Let's say we created successfully DLL injection to process. How to check if DLL in list of modules of our process?



practical example

First of all, we just use one of the methods to find target process PID. For example I used this one:

```
typedef NTSTATUS (NTAPI * fNtGetNextProcess)(
  _In_ HANDLE ph,
  _In_ ACCESS_MASK DesiredAccess,
  _In_ ULONG HandleAttributes,
  _In_ ULONG Flags,
  _Out_ PHANDLE Newph
);

int findMyProc(const char * procname) {
  int pid = 0;
  HANDLE current = NULL;
  char procName[MAX_PATH];

  // resolve function address
  fNtGetNextProcess myNtGetNextProcess = (fNtGetNextProcess)
GetProcAddress(GetModuleHandle("ntdll.dll"), "NtGetNextProcess");

  // loop through all processes
  while (!myNtGetNextProcess(current, MAXIMUM_ALLOWED, 0, 0, &current)) {
    GetProcessImageFileNameA(current, procName, MAX_PATH);
    if (lstrcmpiA(procname, PathFindFileName((LPCSTR) procName)) == 0) {
      pid = GetProcessId(current);
      break;
    }
  }

  return pid;
}
```

Then, just use `Module32First` and `Module32Next` functions from Windows API.

```
// function to list modules loaded by a specified process
int listModulesOfProcess(int pid) {

  HANDLE mod;
  MODULEENTRY32 me32;

  mod = CreateToolhelp32Snapshot(TH32CS_SNAPMODULE | TH32CS_SNAPMODULE32, pid);
  if (mod == INVALID_HANDLE_VALUE) {
    printf("CreateToolhelp32Snapshot error :(\n");
    return -1;
  }

  me32.dwSize = sizeof(MODULEENTRY32);
  if (!Module32First(mod, &me32)) {
    CloseHandle(mod);
    return -1;
  }

  printf("modules found:\n");
  printf("name\t\t\t base address\t\t\tsize\n");

printf("================================================================================
====\n");
  do {
    printf("%#25s\t\t%#10llx\t\t%#10d\n", me32.szModule, me32.modBaseAddr,
me32.modBaseSize);
  } while (Module32Next(mod, &me32));
  CloseHandle(mod);
  return 0;
}
```

As you can see, the code is a bit similar to the PID search logic with `CreateToolHelp32Snapshot`, `Process32First` and `Process32Next`.

So, the full source code is looks like this (`hack.c`):

```
/*
 * hack.c - get the list of modules of the process. C++ implementation
 * @cocomelonc
 * https://cocomelonc.github.io/malware/2023/09/25/malware-tricks-36.html
*/
#include <windows.h>
#include <stdio.h>
#include <winternl.h>
#include <tlhelp32.h>
#include <shlwapi.h>
#include <psapi.h>

#pragma comment(lib, "ntdll.lib")
#pragma comment(lib, "shlwapi.lib")

typedef NTSTATUS (NTAPI * fNtGetNextProcess)(
  _In_ HANDLE ph,
  _In_ ACCESS_MASK DesiredAccess,
  _In_ ULONG HandleAttributes,
  _In_ ULONG Flags,
  _Out_ PHANDLE Newph
);

int findMyProc(const char * procname) {
  int pid = 0;
  HANDLE current = NULL;
  char procName[MAX_PATH];

  // resolve function address
  fNtGetNextProcess myNtGetNextProcess = (fNtGetNextProcess)
GetProcAddress(GetModuleHandle("ntdll.dll"), "NtGetNextProcess");

  // loop through all processes
  while (!myNtGetNextProcess(current, MAXIMUM_ALLOWED, 0, 0, &current)) {
    GetProcessImageFileNameA(current, procName, MAX_PATH);
    if (lstrcmpiA(procname, PathFindFileName((LPCSTR) procName)) == 0) {
      pid = GetProcessId(current);
      break;
    }
  }

  return pid;
}

// function to list modules loaded by a specified process
int listModulesOfProcess(int pid) {

  HANDLE mod;
  MODULEENTRY32 me32;

  mod = CreateToolhelp32Snapshot(TH32CS_SNAPMODULE | TH32CS_SNAPMODULE32, pid);
  if (mod == INVALID_HANDLE_VALUE) {
```

```
    printf("CreateToolhelp32Snapshot error :(\n");
    return -1;
  }

  me32.dwSize = sizeof(MODULEENTRY32);
  if (!Module32First(mod, &me32)) {
    CloseHandle(mod);
    return -1;
  }

  printf("modules found:\n");
  printf("name\t\t\t base address\t\t\tsize\n");

printf("================================================================================
====\n");
  do {
    printf("%#25s\t\t%#10llx\t\t%#10d\n", me32.szModule, me32.modBaseAddr,
me32.modBaseSize);
  } while (Module32Next(mod, &me32));
  CloseHandle(mod);
  return 0;
}

int main(int argc, char* argv[]) {
  int pid = 0; // process ID
  pid = findMyProc(argv[1]);
  printf("%s%d\n", pid > 0 ? "process found at pid = " : "process not found. pid = ",
pid);
  if (pid != 0)
    listModulesOfProcess(pid);
  return 0;
}
```

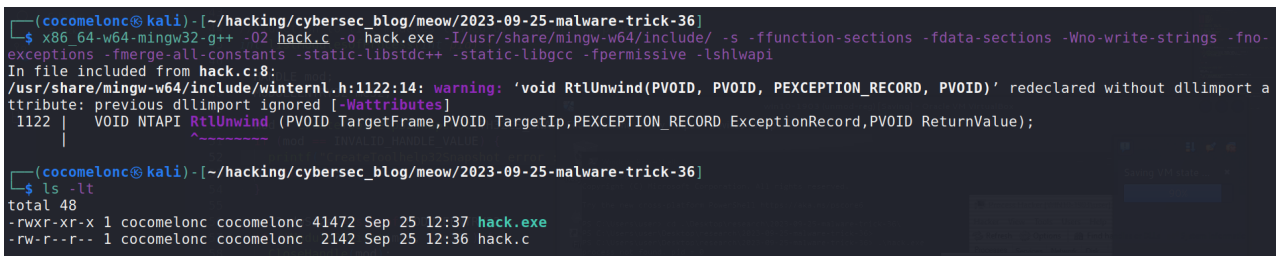
You can use this code to check if a DLL is in the list of modules of the target process.

demo

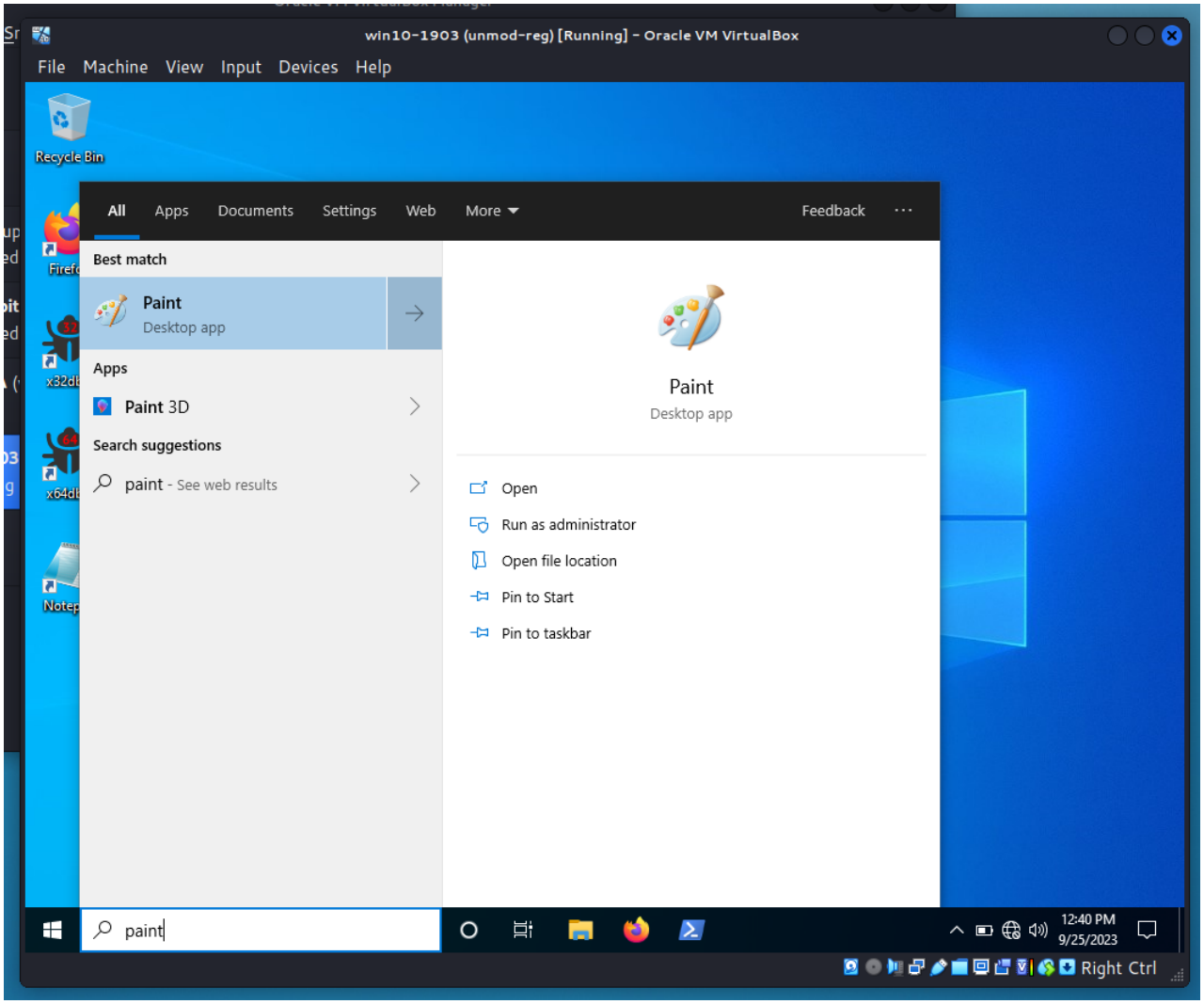
Let's go to see this logic in action.

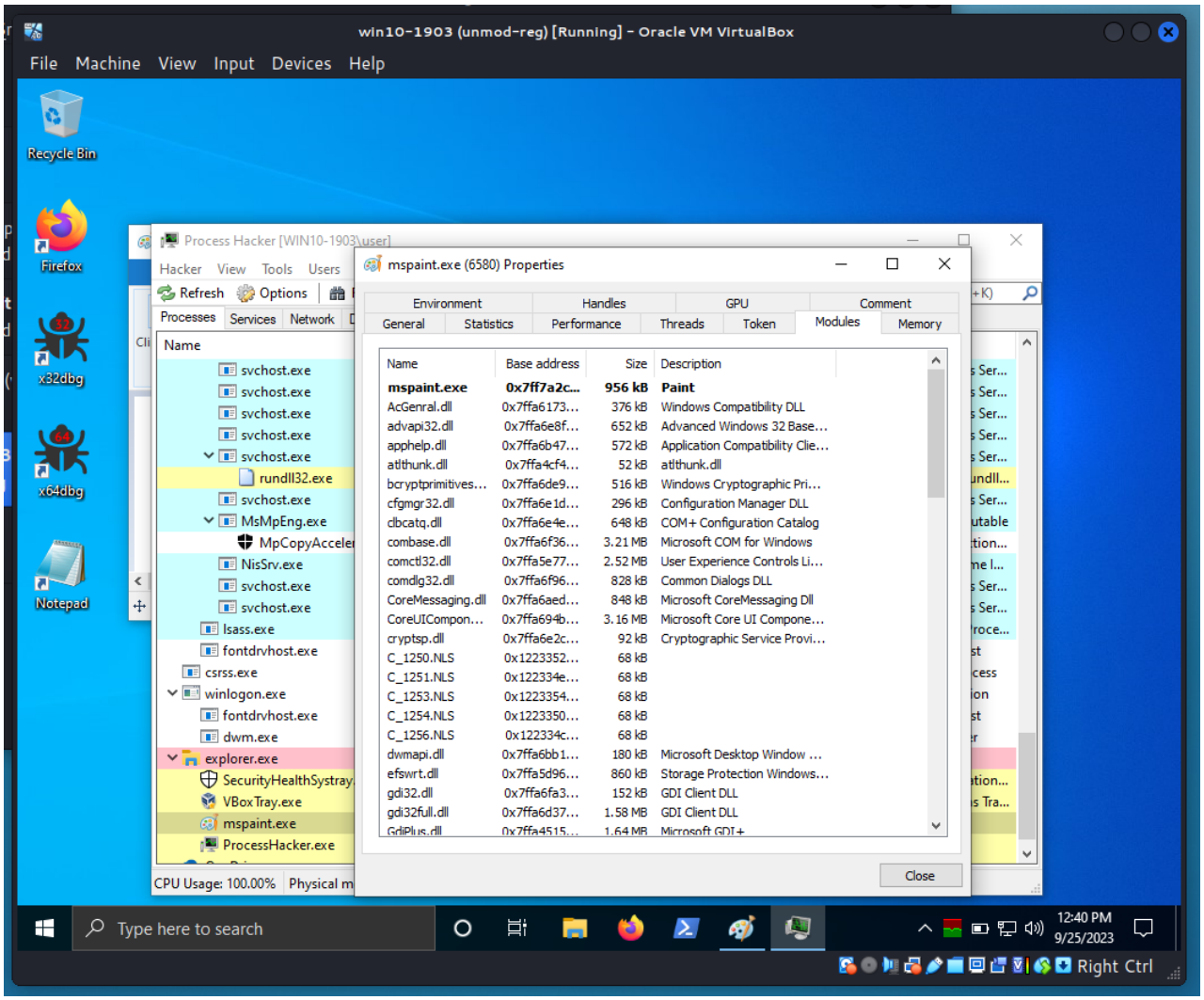
Compile it:

```
x86_64-w64-mingw32-g++ -O2 hack.c -o hack.exe -I/usr/share/mingw-w64/include/ -s -
ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions -fmerge-all-
constants -static-libstdc++ -static-libgcc -fpermissive -lshlwapi
```



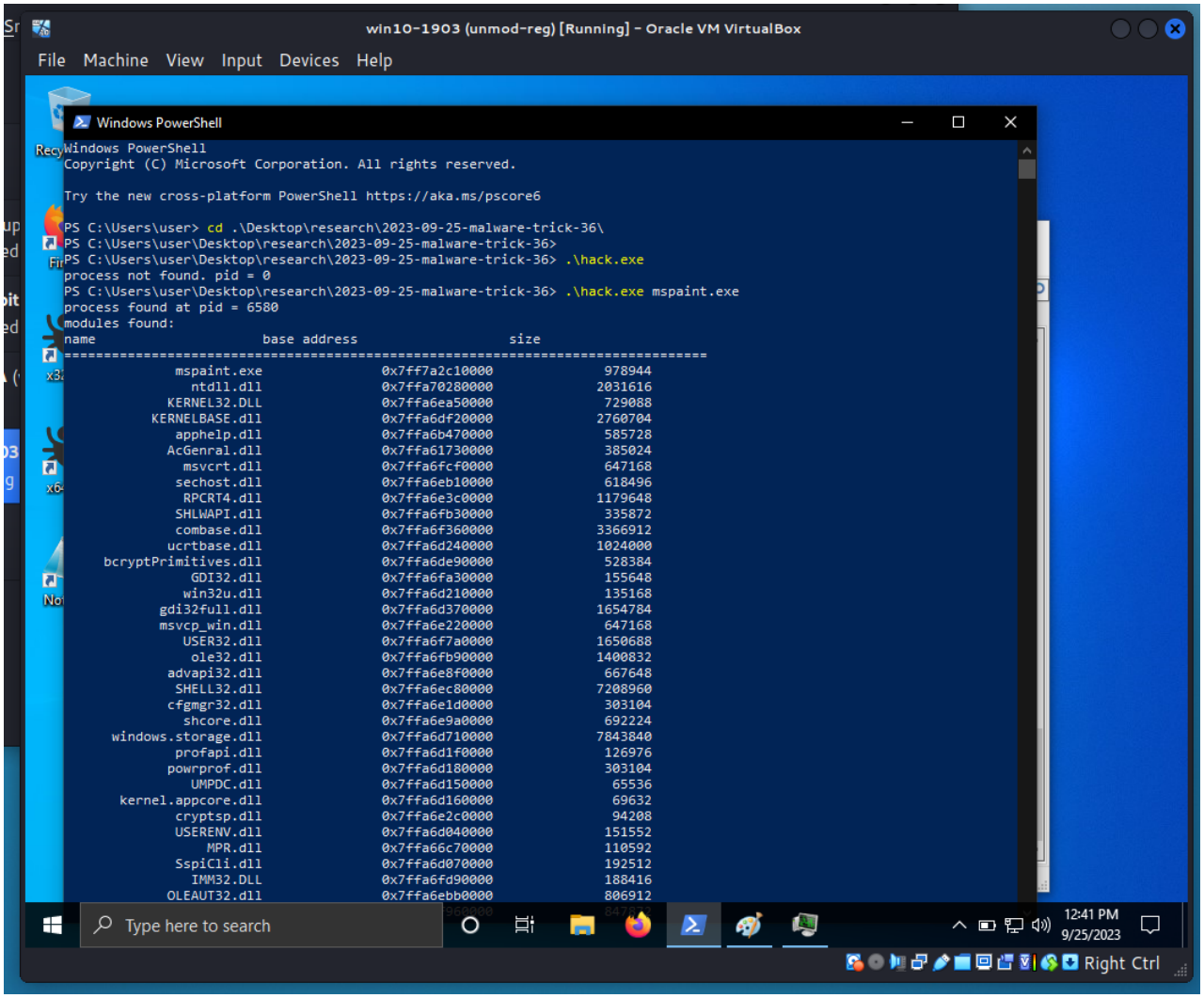
Then, open target process in the victim's machine:

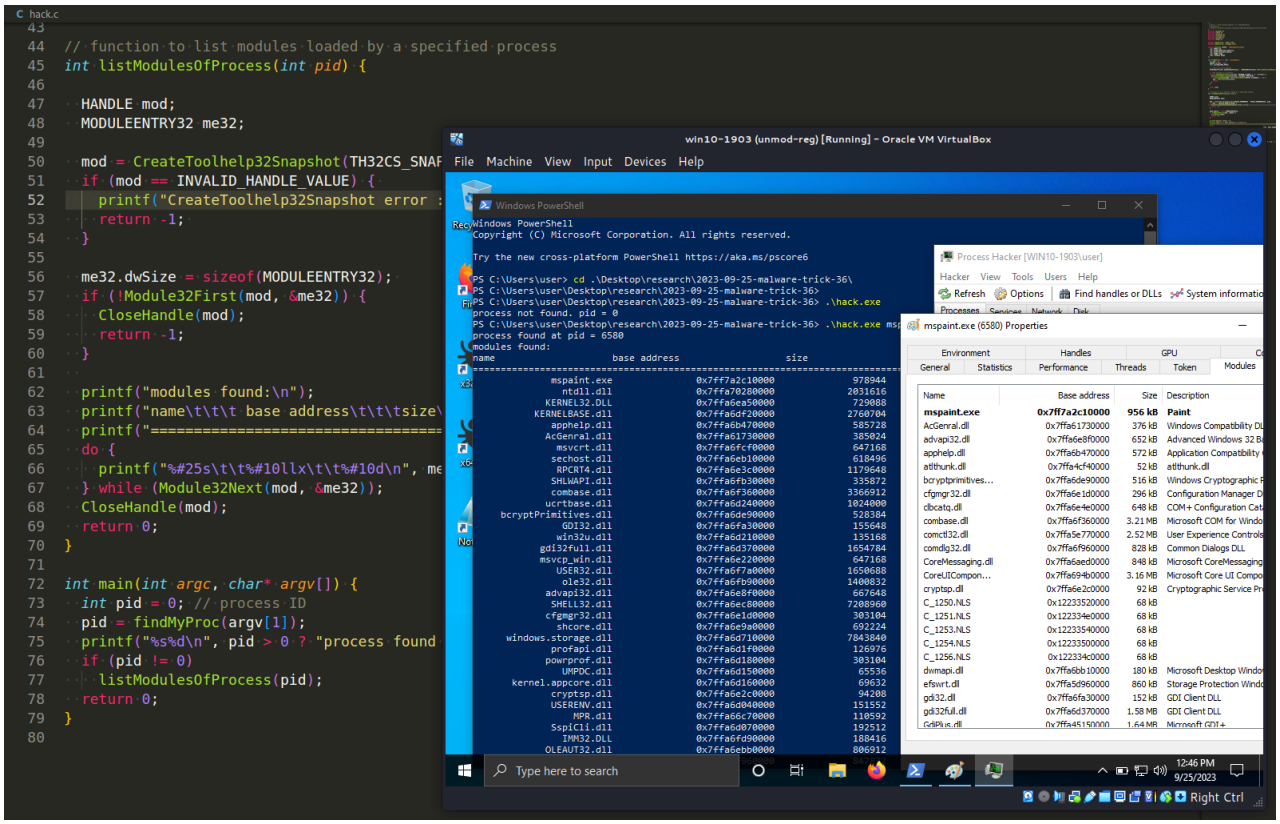


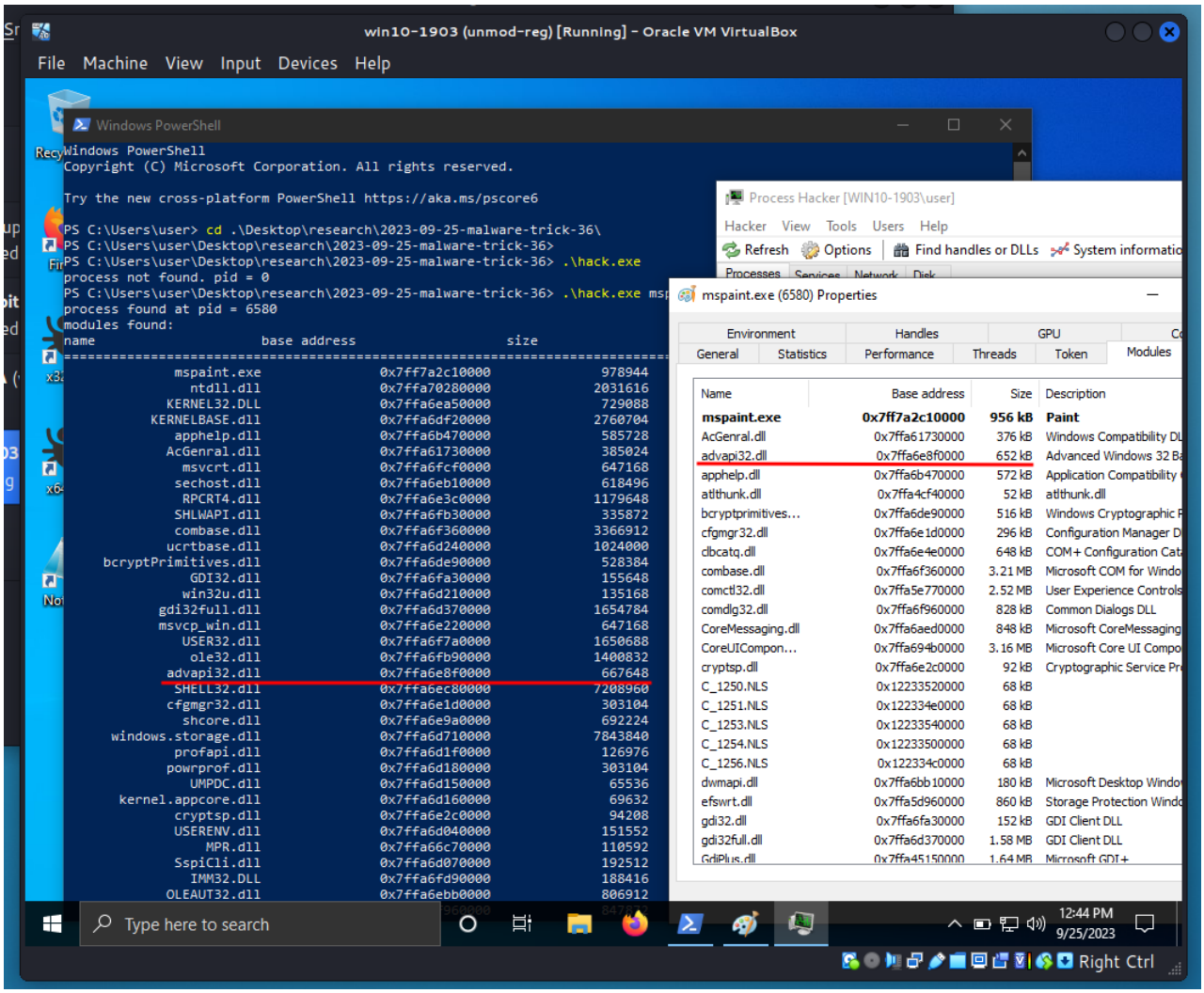


And just run our `hack.exe`:

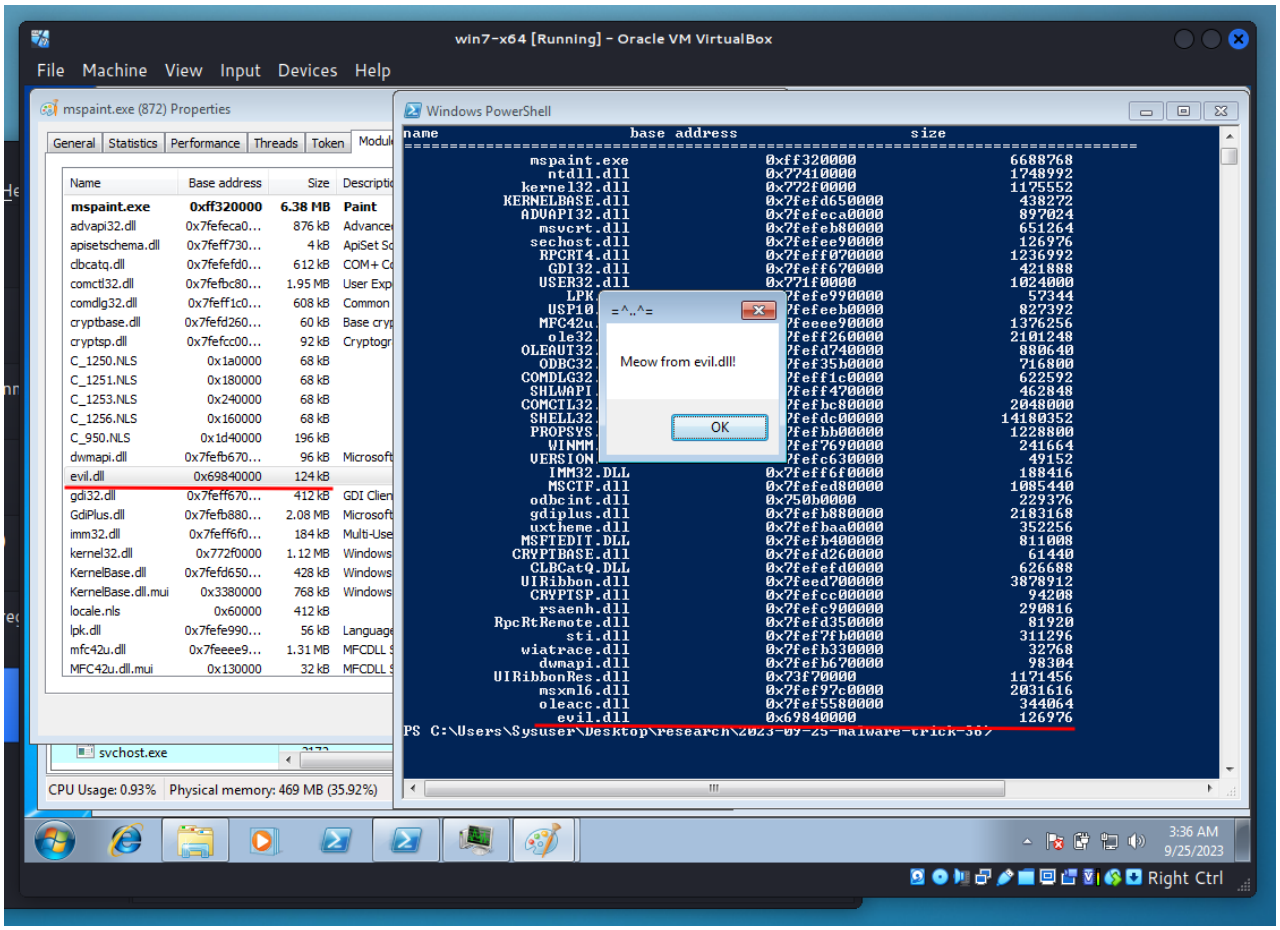
```
.\hack.exe mspaint.exe
```


win10-1903 (unmod-reg) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS C:\Users\user> cd .\Desktop\research\2023-09-25-malware-trick-36\
PS C:\Users\user\Desktop\research\2023-09-25-malware-trick-36>
PS C:\Users\user\Desktop\research\2023-09-25-malware-trick-36> .\hack.exe
process not found. pid = 0
PS C:\Users\user\Desktop\research\2023-09-25-malware-trick-36> .\hack.exe mspaint.exe
process found at pid = 6580
modules found:
name base address size
mspaint.exe 0x7ff7a2c10000 978944
ntdll.dll 0x7ffa70280000 2031616
KERNEL32.DLL 0x7ffa6ea50000 729088
KERNELBASE.dll 0x7ffa6df20000 2760704
apphelp.dll 0x7ffa6b470000 585728
AcGenral.dll 0x7ffa61730000 385024
msvcrt.dll 0x7ffa6fcf0000 647168
sechost.dll 0x7ffa6eb10000 618496
RPCRT4.dll 0x7ffa6e3c0000 1179648
SHLWAPI.dll 0x7ffa6fb30000 335872
combase.dll 0x7ffa6f360000 3366912
ucrtbase.dll 0x7ffa6d240000 1024000
bcryptPrimitives.dll 0x7ffa6de90000 528384
GDI32.dll 0x7ffa6fa30000 155648
win32u.dll 0x7ffa6d210000 135168
gdi32full.dll 0x7ffa6d370000 1654784
msvcp_win.dll 0x7ffa6e220000 647168
USER32.dll 0x7ffa6f7a0000 1650688
ole32.dll 0x7ffa6fb90000 1400832
advapi32.dll 0x7ffa6e8f0000 667648
SHELL32.dll 0x7ffa6ec80000 7208960
cfgmgr32.dll 0x7ffa6e1d0000 303104
shcore.dll 0x7ffa6e9a0000 692224
windows.storage.dll 0x7ffa6d710000 7843840
profapi.dll 0x7ffa6d1f0000 126976
powrprof.dll 0x7ffa6d180000 303104
UMPDC.dll 0x7ffa6d150000 65536
kernel.appcore.dll 0x7ffa6d160000 69632
cryptsp.dll 0x7ffa6e2c0000 94208
USERENV.dll 0x7ffa6d040000 151552
MPR.dll 0x7ffa66c70000 110592
SspiCli.dll 0x7ffa6d070000 192512
IMM32.DLL 0x7ffa6fd90000 188416
OLEAUT32.dll 0x7ffa6ebb0000 806912
Type here to search
12:41 PM
9/25/2023
Right Ctrl


hack.c
// function to list modules loaded by a specified process
int listModulesOfProcess(int pid) {
HANDLE mod;
MODULEENTRY32 me32;
mod = CreateToolhelp32Snapshot(TH32CS_SNAP
if (mod == INVALID_HANDLE_VALUE) {
printf("CreateToolhelp32Snapshot error
return -1;
}
me32.dwSize = sizeof(MODULEENTRY32);
if (!Module32First(mod, &me32)) {
CloseHandle(mod);
return -1;
}
printf("modules found:\n");
printf("name\t\t\t base address\t\t\tsize\
printf("==========
do {
printf("%#25s\t\t%#10llx\t\t%#10d\n", me
} while (Module32Next(mod, &me32));
CloseHandle(mod);
return 0;
}
int main(int argc, char* argv[]) {
int pid = 0; // process ID
pid = findMyProc(argv[1]);
printf("%s%d\n", pid > 0 ? "process found
if (pid != 0)
listModulesOfProcess(pid);
return 0;
}
win10-1903 (unmod-reg) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS C:\Users\user> cd .\Desktop\research\2023-09-25-malware-trick-36\
PS C:\Users\user\Desktop\research\2023-09-25-malware-trick-36>
PS C:\Users\user\Desktop\research\2023-09-25-malware-trick-36> .\hack.exe
process not found. pid = 0
PS C:\Users\user\Desktop\research\2023-09-25-malware-trick-36> .\hack.exe ms
process found at pid = 6580
modules found:
name base address size
mspaint.exe 0x7ff7a2c10000 978944
ntdll.dll 0x7ffa70280000 2031616
KERNEL32.DLL 0x7ffa6ea50000 729088
KERNELBASE.dll 0x7ffa6df20000 2760704
apphelp.dll 0x7ffa6b470000 585728
AcGenral.dll 0x7ffa61730000 385024
msvcrt.dll 0x7ffa6fcf0000 647168
sechost.dll 0x7ffa6eb10000 618496
RPCRT4.dll 0x7ffa6e3c0000 1179648
SHLWAPI.dll 0x7ffa6fb30000 335872
combase.dll 0x7ffa6f360000 3366912
ucrtbase.dll 0x7ffa6d240000 1024000
bcryptPrimitives.dll 0x7ffa6de90000 528384
GDI32.dll 0x7ffa6fa30000 155648
win32u.dll 0x7ffa6d210000 135168
gdi32full.dll 0x7ffa6d370000 1654784
msvcp_win.dll 0x7ffa6e220000 647168
USER32.dll 0x7ffa6f7a0000 1650688
ole32.dll 0x7ffa6fb90000 1400832
advapi32.dll 0x7ffa6e8f0000 667648
SHELL32.dll 0x7ffa6ec80000 7208960
cfgmgr32.dll 0x7ffa6e1d0000 303104
shcore.dll 0x7ffa6e9a0000 692224
windows.storage.dll 0x7ffa6d710000 7843840
profapi.dll 0x7ffa6d1f0000 126976
powrprof.dll 0x7ffa6d180000 303104
UMPDC.dll 0x7ffa6d150000 65536
kernel.appcore.dll 0x7ffa6d160000 69632
cryptsp.dll 0x7ffa6e2c0000 94208
USERENV.dll 0x7ffa6d040000 151552
MPR.dll 0x7ffa66c70000 110592
SspiCli.dll 0x7ffa6d070000 192512
IMM32.DLL 0x7ffa6fd90000 188416
OLEAUT32.dll 0x7ffa6ebb0000 806912
Process Hacker [WIN10-1903\user]
Hacker View Tools Users Help
Refresh Options Find handles or DLLs System informatio
Processes Services Network Disk
mspaint.exe (6580) Properties
Environment Handles GPU
General Statistics Performance Threads Token Modules
Name Base address Size Description
mspaint.exe 0x7ff7a2c10000 956 kB Paint
AcGenral.dll 0x7ffa61730000 376 kB Windows Compatibility DL
advapi32.dll 0x7ffa6e8f0000 652 kB Advanced Windows 32 Ba
apphelp.dll 0x7ffa6b470000 572 kB Application Compatibility
atlthunk.dll 0x7ffa4cf40000 52 kB atlthunk.dll
bcryptprimitives... 0x7ffa6de90000 516 kB Windows Cryptographic P
cfgmgr32.dll 0x7ffa6e1d0000 296 kB Configuration Manager D
clbcatq.dll 0x7ffa6e4e0000 648 kB COM+ Configuration Cat
combase.dll 0x7ffa6f360000 3.21 MB Microsoft COM for Windo
comctl32.dll 0x7ffa5e770000 2.52 MB User Experience Controls
comdlg32.dll 0x7ffa6f960000 828 kB Common Dialogs DLL
CoreMessaging.dll 0x7ffa6aed0000 848 kB Microsoft CoreMessaging
CoreUICompon... 0x7ffa694b0000 3.16 MB Microsoft Core UI Compo
cryptsp.dll 0x7ffa6e2c0000 92 kB Cryptographic Service Pr
C_1250.NLS 0x12233520000 68 kB
C_1251.NLS 0x122334e0000 68 kB
C_1253.NLS 0x12233540000 68 kB
C_1254.NLS 0x12233500000 68 kB
C_1256.NLS 0x122334c0000 68 kB
dwmapi.dll 0x7ffa6bb10000 180 kB Microsoft Desktop Windo
efswrt.dll 0x7ffa5d960000 860 kB Storage Protection Windo
gdi32.dll 0x7ffa6fa30000 152 kB GDI Client DLL
gdi32full.dll 0x7ffa6d370000 1.58 MB GDI Client DLL
GdiPlus.dll 0x7ffa45150000 1.64 MB Microsoft GDI+
Type here to search
12:46 PM
9/25/2023
Right Ctrl



Also, check with DLL injection logic:



As you can see, everything is worked perfectly! =^..^=

Keep in mind that this code may have limitations and dependencies on specific Windows APIs. Additionally, it relies on the process name for identification, which may not be unique.

This trick is used by 4H RAT and Aria-body in the wild.

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

Find process ID by name and inject to it
Find PID via NtGetNextProcess
4H RAT
Aria-body
source code in github

> This is a practical case for educational purposes only.

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