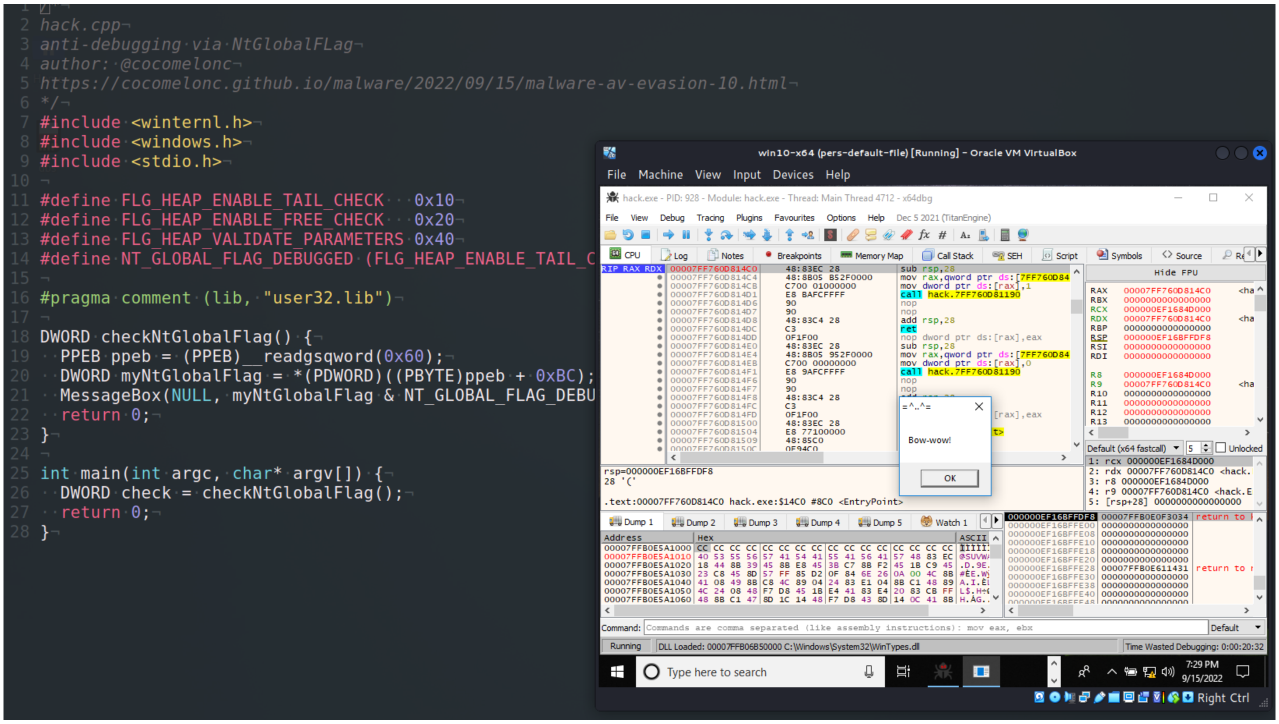
Malware AV/VM evasion - part 10: anti-debugging. NtGlobalFlag. Simple C++ example.

cocomelonc.github.io/malware/2022/09/15/malware-av-evasion-10.html

September 15, 2022

1 minute read

Hello, cybersecurity enthusiasts and white hackers!



```cpp
/*
hack.cpp
anti-debugging via NtGlobalFLag
author: @cocomelonc
https://cocomelonc.github.io/malware/2022/09/15/malware-av-evasion-10.html
*/
#include <winternl.h>
#include <windows.h>
#include <stdio.h>

#define FLG_HEAP_ENABLE_TAIL_CHECK   0x10
#define FLG_HEAP_ENABLE_FREE_CHECK   0x20
#define FLG_HEAP_VALIDATE_PARAMETERS 0x40
#define NT_GLOBAL_FLAG_DEBUGGED (FLG_HEAP_ENABLE_TAIL_C

#pragma comment (lib, "user32.lib")

DWORD checkNtGlobalFlag() {
  PPEB ppeb = (PPEB)__readgsqword(0x60);
  DWORD myNtGlobalFlag = *(PDWORD)((PBYTE)ppeb + 0xBC);
  MessageBox(NULL, myNtGlobalFlag & NT_GLOBAL_FLAG_DEBU
  return 0;
}

int main(int argc, char* argv[]) {
  DWORD check = checkNtGlobalFlag();
  return 0;
}
```

win10-x64 (pers-default-file) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
hack.exe - PID: 928 - Module: hack.exe - Thread: Main Thread 4712 - x64dbg
File View Debug Tracing Plugins Favourites Options Help
CPU
Log
Notes
Breakpoints
Memory Map
Call Stack
SEH
Script
Symbols
Source
Dump 1
Dump 2
Dump 3
Dump 4
Dump 5
Watch 1
Hide FPU
Default (x64 fastcall)
Unlocked
Command:
Default
Running
Type here to search
Right Ctrl

```
=^..^=
Bow-wow!
OK
```

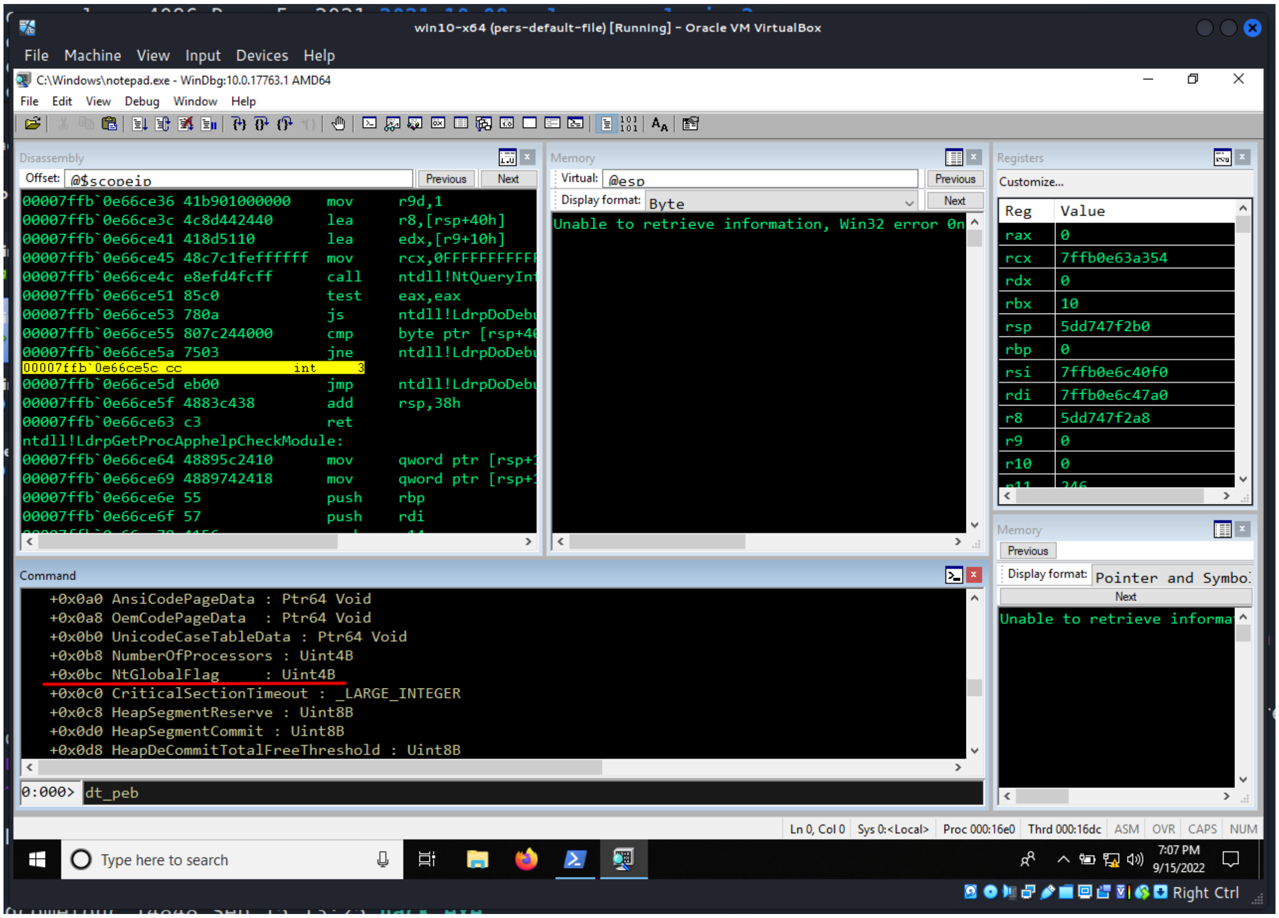
This post is the result of my own research into interesting anti-debugging trick: checking `NtGlobalFlag`.

This is just another way how malware can detect that it is running in a debugger.

NtGlobalFlag

During debugging, the system sets the `FLG_HEAP_ENABLE_TAIL_CHECK` (`0x10`), `FLG_HEAP_ENABLE_FREE_CHECK` (`0x20`) and `FLG_HEAP_VALIDATE_PARAMETERS` (`0x40`) flags in the `NtGlobalFlag` field, which is located in the `PEB` structure.

The `NtGlobalFlag` has the value `0x68` offset on `32-bit` Windows, the value of `0xbc` on `64-bit` Windows and both of them are set to `0`:



win10-x64 (pers-default-file) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
C:\Windows\notepad.exe - WinDbg:10.0.17763.1 AMD64
File Edit View Debug Window Help
Disassembly
Offset: @$scopeip
Previous
Next
Memory
Virtual: @esp
Display format: Byte
Registers
Customize...
Command
Display format: Pointer and Symbol
Ln 0, Col 0 Sys 0:<Local> Proc 000:16e0 Thrd 000:16dc ASM OVR CAPS NUM
Type here to search
7:07 PM
9/15/2022
Right Ctrl

```
00007ffb`0e66ce36 41b901000000    mov     r9d,1
00007ffb`0e66ce3c 4c8d442440      lea     r8,[rsp+40h]
00007ffb`0e66ce41 418d5110        lea     edx,[r9+10h]
00007ffb`0e66ce45 48c7c1feffffff  mov     rcx,0FFFFFFFFFFF
00007ffb`0e66ce4c e8efd4fcff      call    ntdll!NtQueryInt
00007ffb`0e66ce51 85c0            test    eax,eax
00007ffb`0e66ce53 780a            js      ntdll!LdrpDoDebu
00007ffb`0e66ce55 807c244000      cmp     byte ptr [rsp+40
00007ffb`0e66ce5a 7503            jne     ntdll!LdrpDoDebu
00007ffb`0e66ce5c cc              int     3
00007ffb`0e66ce5d eb00            jmp     ntdll!LdrpDoDebu
00007ffb`0e66ce5f 4883c438        add     rsp,38h
00007ffb`0e66ce63 c3              ret
ntdll!LdrpGetProcApphelpCheckModule:
00007ffb`0e66ce64 48895c2410      mov     qword ptr [rsp+1
00007ffb`0e66ce69 4889742418      mov     qword ptr [rsp+1
00007ffb`0e66ce6e 55              push    rbp
00007ffb`0e66ce6f 57              push    rdi
```

```
Unable to retrieve information, Win32 error 0n
```

Reg	Value
rax	0
rcx	7ffb0e63a354
rdx	0
rbx	10
rsp	5dd747f2b0
rbp	0
rsi	7ffb0e6c40f0
rdi	7ffb0e6c47a0
r8	5dd747f2a8
r9	0
r10	0

```
   +0x0a0 AnsiCodePageData : Ptr64 Void
   +0x0a8 OemCodePageData  : Ptr64 Void
   +0x0b0 UnicodeCaseTableData : Ptr64 Void
   +0x0b8 NumberOfProcessors : Uint4B
   +0x0bc NtGlobalFlag     : Uint4B
   +0x0c0 CriticalSectionTimeout : _LARGE_INTEGER
   +0x0c8 HeapSegmentReserve : Uint8B
   +0x0d0 HeapSegmentCommit : Uint8B
   +0x0d8 HeapDeCommitTotalFreeThreshold : Uint8B
0:000> dt_peb
```

```
Unable to retrieve informa
```

practical example

Simple PoC code for anti-debugging:

```
/*
hack.cpp
anti-debugging via NtGlobalFLag
author: @cocomelonc
https://cocomelonc.github.io/malware/2022/09/15/malware-av-evasion-10.html
*/
#include <winternl.h>
#include <windows.h>
#include <stdio.h>

#define FLG_HEAP_ENABLE_TAIL_CHECK   0x10
#define FLG_HEAP_ENABLE_FREE_CHECK   0x20
#define FLG_HEAP_VALIDATE_PARAMETERS 0x40
#define NT_GLOBAL_FLAG_DEBUGGED (FLG_HEAP_ENABLE_TAIL_CHECK |
FLG_HEAP_ENABLE_FREE_CHECK | FLG_HEAP_VALIDATE_PARAMETERS)

#pragma comment (lib, "user32.lib")

DWORD checkNtGlobalFlag() {
  PPEB ppeb = (PPEB)__readgsqword(0x60);
  DWORD myNtGlobalFlag = *(PDWORD)((PBYTE)ppeb + 0xBC);
  MessageBox(NULL, myNtGlobalFlag & NT_GLOBAL_FLAG_DEBUGGED ? "Bow-wow!" : "Meow-
meow!", "=^..^=", MB_OK);
  return 0;
}

int main(int argc, char* argv[]) {
  DWORD check = checkNtGlobalFlag();
  return 0;
}
```

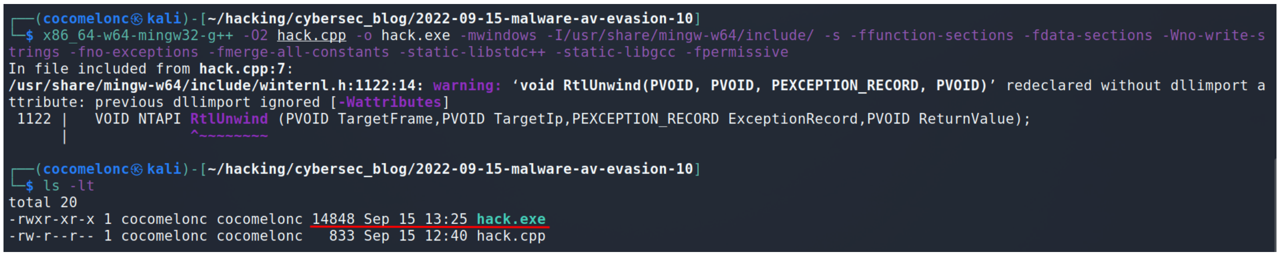
As you can see, the logic is pretty simple, we just check a combination of flags.

> For simplicity, I have only considered 64-bit Windows

demo

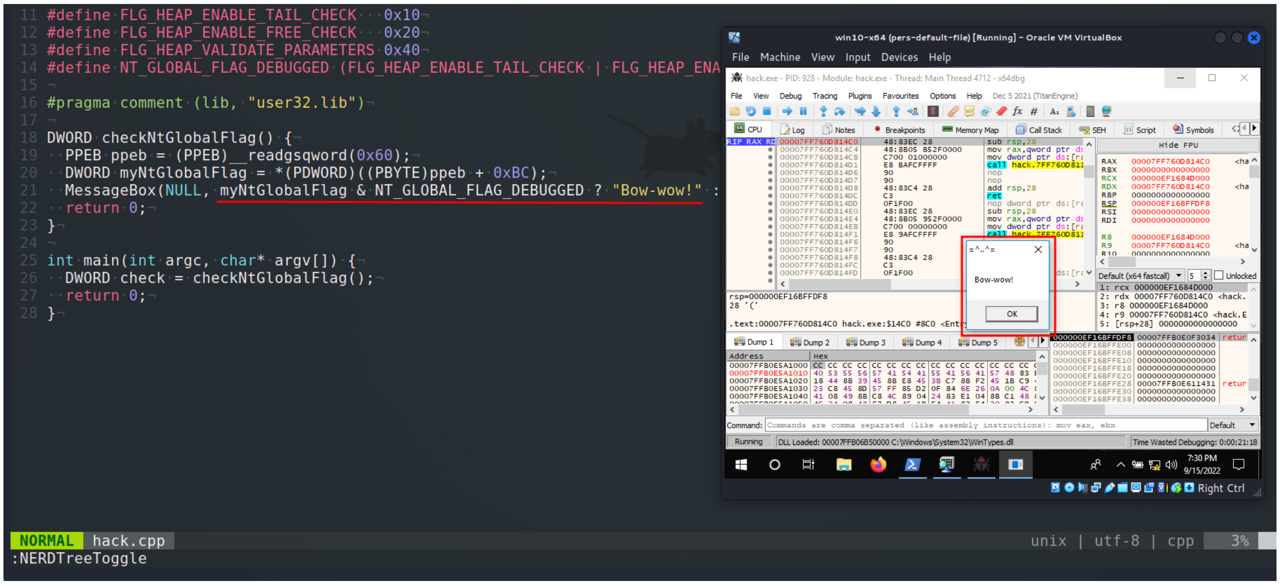
Let’s go to see everything in action. Compile:

```
x86_64-w64-mingw32-g++ -O2 hack.cpp -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
```



```
┌──(cocomelonc㉿kali)-[~/hacking/cybersec_blog/2022-09-15-malware-av-evasion-10]
└─$ x86_64-w64-mingw32-g++ -O2 hack.cpp -o hack.exe -mwindows -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-s
trings -fno-exceptions -fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive
In file included from hack.cpp:7:
/usr/share/mingw-w64/include/winternl.h:1122:14: warning: ‘void RtlUnwind(PVOID, PVOID, PEXCEPTION_RECORD, PVOID)’ redeclared without dllimport a
ttribute: previous dllimport ignored [-Wattributes]
 1122 |   VOID NTAPI RtlUnwind (PVOID TargetFrame,PVOID TargetIp,PEXCEPTION_RECORD ExceptionRecord,PVOID ReturnValue);
      |              ^~~~~~~~~

┌──(cocomelonc㉿kali)-[~/hacking/cybersec_blog/2022-09-15-malware-av-evasion-10]
└─$ ls -lt
total 20
-rwxr-xr-x 1 cocomelonc cocomelonc 14848 Sep 15 13:25 hack.exe
-rw-r--r-- 1 cocomelonc cocomelonc   833 Sep 15 12:40 hack.cpp
```

Run it via `x64dbg` debugger:

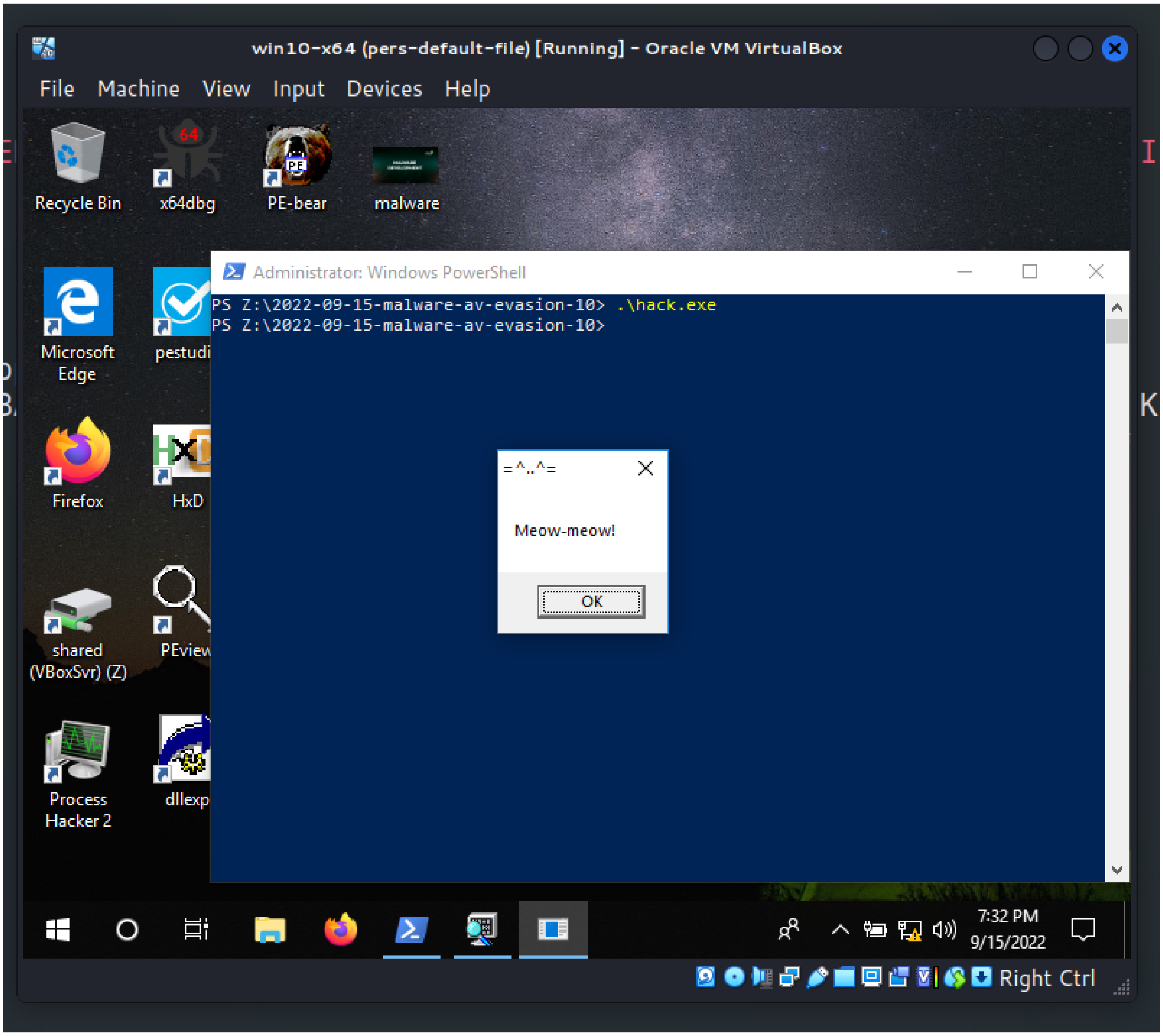


```
#define FLG_HEAP_ENABLE_TAIL_CHECK   0x10
#define FLG_HEAP_ENABLE_FREE_CHECK   0x20
#define FLG_HEAP_VALIDATE_PARAMETERS 0x40
#define NT_GLOBAL_FLAG_DEBUGGED (FLG_HEAP_ENABLE_TAIL_CHECK | FLG_HEAP_ENA

#pragma comment (lib, "user32.lib")

DWORD checkNtGlobalFlag() {
  PPEB ppeb = (PPEB)__readgsqword(0x60);
  DWORD myNtGlobalFlag = *(PDWORD)((PBYTE)ppeb + 0xBC);
  MessageBox(NULL, myNtGlobalFlag & NT_GLOBAL_FLAG_DEBUGGED ? "Bow-wow!" :
  return 0;
}

int main(int argc, char* argv[]) {
  DWORD check = checkNtGlobalFlag();
  return 0;
}
```

```
NORMAL  hack.cpp                                   unix | utf-8 | cpp   3%
:NERDTreeToggle
```

win10-x64 (pers-default-file) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
hack.exe - PID: 928 - Module: hack.exe - Thread: Main Thread 4712 - x64dbg
=^..^=
Bow-wow!
OK
7:30 PM
9/15/2022
Right Ctrl

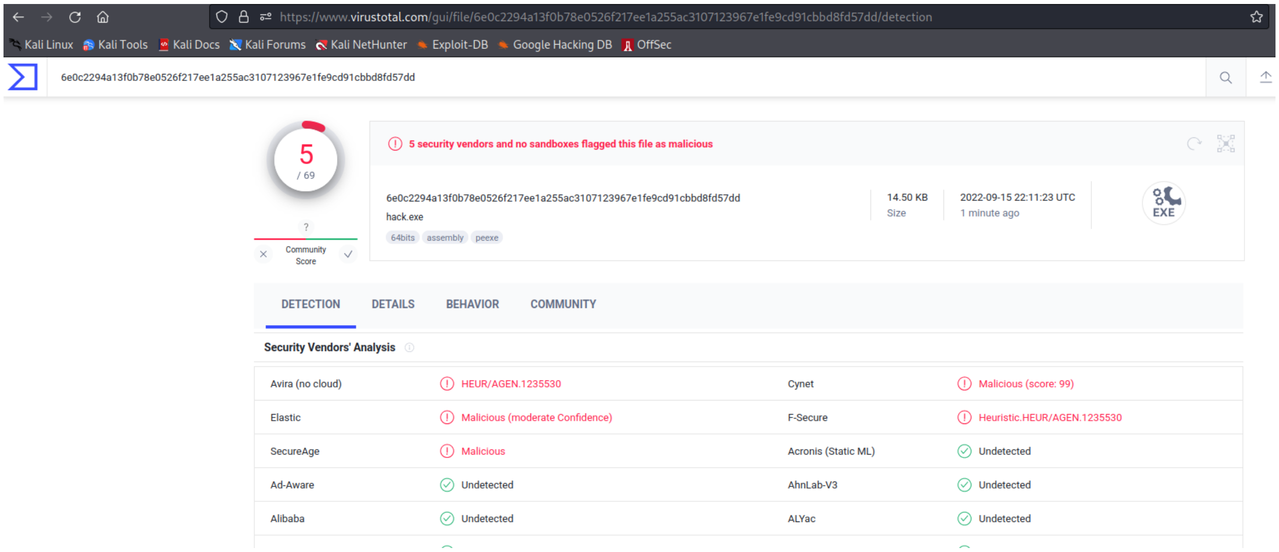
and run from cmd:



```
PS Z:\2022-09-15-malware-av-evasion-10> .\hack.exe
PS Z:\2022-09-15-malware-av-evasion-10>
```

As you can see everything is worked perfectly :)

Upload it to VirusTotal:



https://www.virustotal.com/gui/file/6e0c2294a13f0b78e0526f217ee1a255ac3107123967e1fe9cd91cbbd8fd57dd/detection
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
6e0c2294a13f0b78e0526f217ee1a255ac3107123967e1fe9cd91cbbd8fd57dd
5
/ 69
?
Community Score
EXE
DETECTION
DETAILS
BEHAVIOR
COMMUNITY

5 security vendors and no sandboxes flagged this file as malicious

6e0c2294a13f0b78e0526f217ee1a255ac3107123967e1fe9cd91cbbd8fd57dd
hack.exe
64bits assembly peexe

14.50 KB
Size

2022-09-15 22:11:23 UTC
1 minute ago

Security Vendors' Analysis

Avira (no cloud)	HEUR/AGEN.1235530	Cynet	Malicious (score: 99)
Elastic	Malicious (moderate Confidence)	F-Secure	Heuristic.HEUR/AGEN.1235530
SecureAge	Malicious	Acronis (Static ML)	Undetected
Ad-Aware	Undetected	AhnLab-V3	Undetected
Alibaba	Undetected	ALYac	Undetected

As you can see, 5 of 69 AV engines detect our PoC file as malicious.

https://www.virustotal.com/gui/file/6e0c2294a13f0b78e0526f217ee1a255ac3107123967e1fe9cd91cbbd8fd57dd/detection

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

MITRE ATT&CK: Debugger evasion
MSDN: PEB structure
x64dbg
al-khaser
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*