Міністерство освіти і науки України

Національний технічний університет України

"Київський політехнічний інститут імені Ігоря Сікорського"

Фізико-технічний інститут

АНАЛIЗ БIНАРНИХ ВРАЗЛИВОСТЕЙ

Лабораторна робота

Робота №5

Виконав студент гр. ФБ-31мп:

Шевченко С. Ш.

Київ – 2024

Тема: Вразливостi на рiвнi ядра ОС.

Мета роботи: Отримати навички експлуатацiї вразливостей на рiвнi ядра ОС.

Варіант: 1 (18 у списку групи)

# Виконання завдань:

Розробiть експлоїт для вразливостi HackSysExtremeVulnerableDriver, за варiантом:

1. Write NULL

## 1) Підготовка:

Створимо віртуальну машину в VirtualBox з образу [Windows 7 x32 Ultimate SP1](https://www.techworm.net/2022/01/download-windows-7-iso-ultimate-professional-edition.html)

Встановимо [Opera](https://blogs.opera.com/desktop/2023/03/opera-95-0-4635-80-windows-7-windows-8-update/) (тому що дефолтний браузер IE не працює)

Встановимо оновлення [KB3063858](https://www.microsoft.com/en-us/download/details.aspx?id=47409)

Встановимо [Python 2.7.18](https://www.python.org/downloads/release/python-2718/) (Додамо рядок "C:\Python27;C:\Python27\Scripts" в системну змінну оточення PATH)

Встановимо [MinGW-w64 builds](https://github.com/niXman/mingw-builds-binaries)

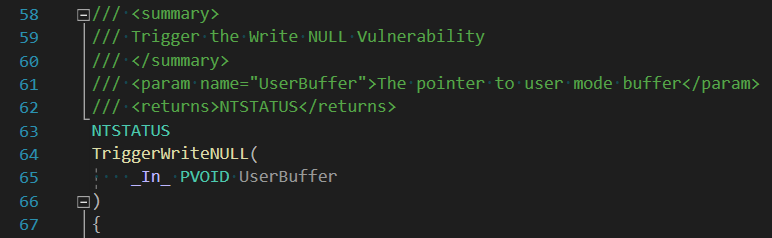
Завантажимо [HEVD v3.00](https://github.com/hacksysteam/HackSysExtremeVulnerableDriver/releases) (завантажуємо і сам [реліз](https://github.com/hacksysteam/HackSysExtremeVulnerableDriver/releases/download/v3.00/HEVD.3.00.zip), і [вихідний код](https://github.com/hacksysteam/HackSysExtremeVulnerableDriver/archive/refs/tags/v3.00.zip))

Завантажимо [OSRLOADER v3.0](http://www.osronline.com/article.cfm%5Earticle=157.htm)

Експлойт створено на основі [PoC-експлойту](https://github.com/w4fz5uck5/3XPL01t5/tree/master/OSEE_Training/HEVD_exploits/windowsx86/%5BHEVD%5D-WriteNULL), а вразливість детально розглянуто в [статті](https://wafzsucks.medium.com/hacksys-extreme-vulnerable-driver-arbitrary-write-null-new-solution-7d45bfe6d116).

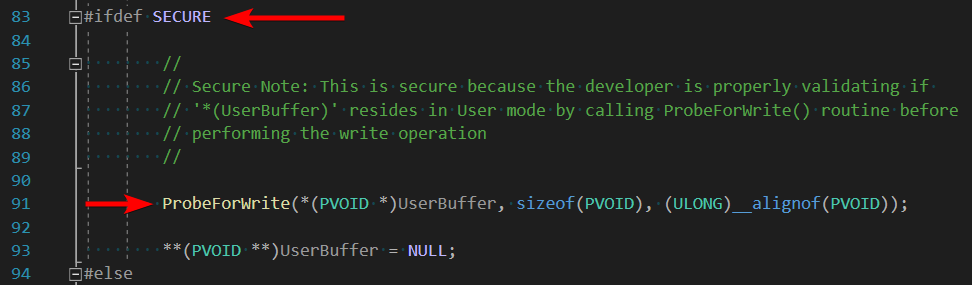
## 2) Дослідження вразливості Write NULL:

У вихідному коді HEVD v3.00 є файл HackSysExtremeVulnerableDriver-3.00\Driver\HEVD\WriteNULL.c. Даний файл містить фукнцію TriggerWriteNULL():

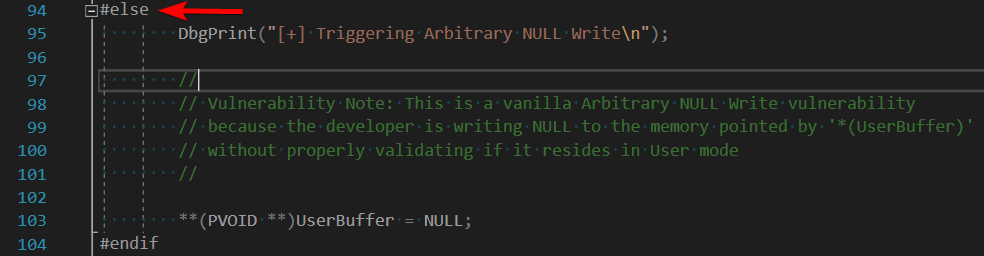


Ця функція занулює буфер, вказівник на який передається в якості аргументу.

В безпечній реалізації цієї функції (лебл SECURE), маємо виклик ф. ProbeForWrite(), що перевіряє чи є буфер доступним для запису. Фактично, ф. ProbeForWrite() перевіряє, чи розміщено буфер в адресному просторі користувача (ring3). Якщо перевірку пройдено, то буфер занулюється:



А ось у вразливій реалізації, перевірка ProbeForWrite() відсутня. Отже, буфер буде занулено навіть якщо він знаходиться в адресному просторі ядра (ring0):

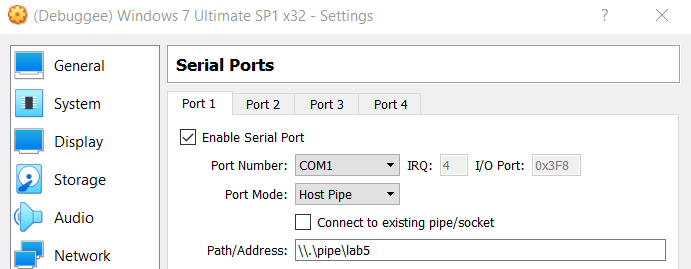


## 3) Демонстрація вразливості в WinDbg:

Хост ОС: Windows 10 x64 192.168.1.10

Цільова ОС: Windows 7 x32 192.168.1.17

В налаштуваннях VirtualBox для Windows 7 створимо pipe: \\.\pipe\lab5

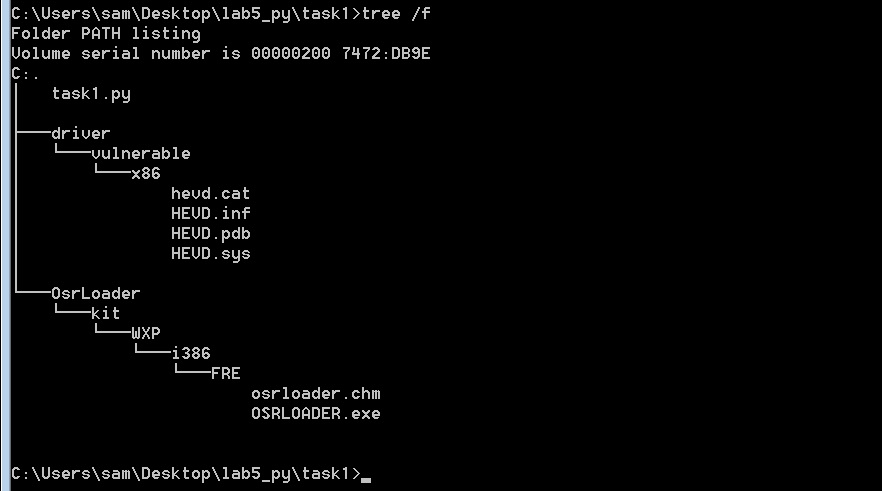


## 3.1) У Windows 7:

Перейдемо в робочу папку та виведемо дерево папок/файлів:

**$ cd C:\Users\sam\Desktop\lab5\_py\task1**

**$ tree /f**

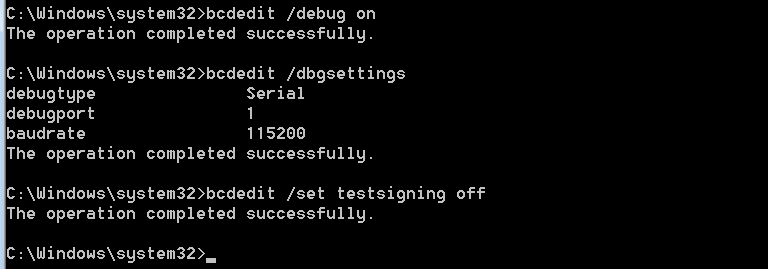


Відкриємо командний рядок від імені адміністратора та включимо дебаг ядра:

**$ bcdedit /debug on**

**$ bcdedit /dbgsettings**

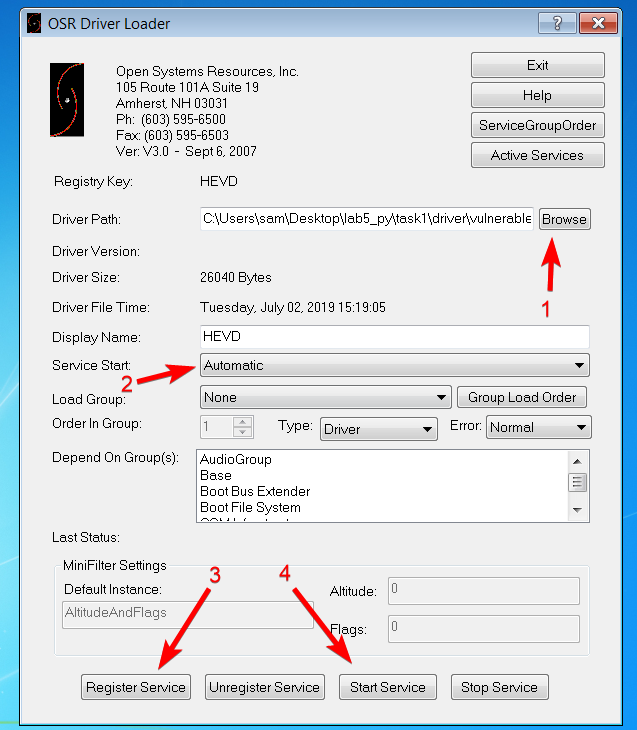
**$ bcdedit /set testsigning off**



Завантажимо вразливу реалізацію HEVD за допомогою OSRLOADER:

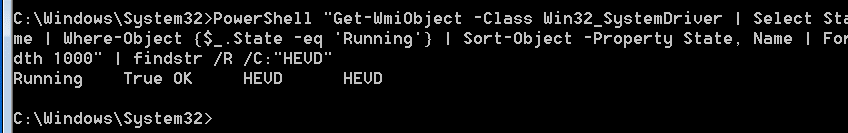
C:\Users\sam\Desktop\lab5\_py\task1\OsrLoader\kit\WXP\i386\FRE\OSRLOADER.exe

Driver Path: C:\Users\sam\Desktop\lab5\_py\task1\driver\vulnerable\x86\HEVD.sys



Переконаємося, що драйвер завантажено:

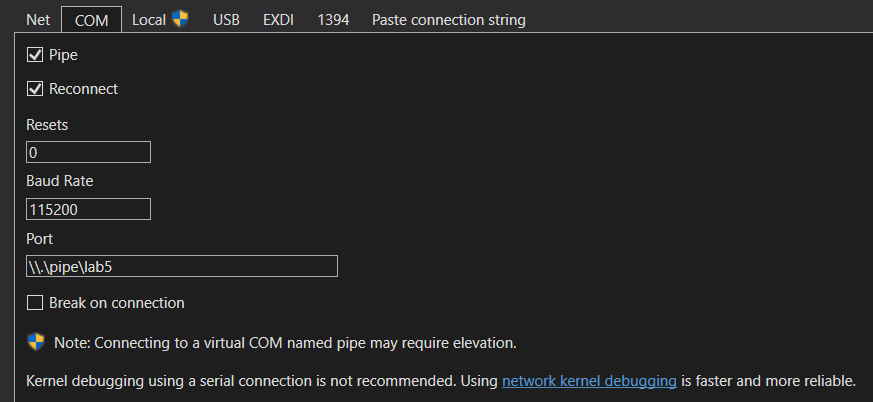
**$ PowerShell "Get-WmiObject -Class Win32\_SystemDriver | Select State, Started, Status, Name, DisplayName | Where-Object {$\_.State -eq 'Running'} | Sort-Object -Property State, Name | Format-Table -Autosize | Out-String -Width 1000" | findstr /R /C:"HEVD"**



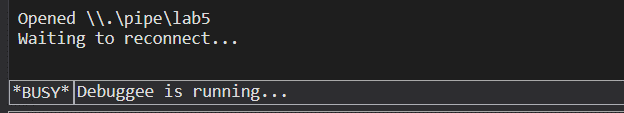
## 3.2) У Windows 10:

Встановимо [WinDbg](https://www.microsoft.com/store/productId/9PGJGD53TN86?ocid=pdpshare) (Створимо системну змінну оточення \_NT\_SYMBOL\_PATH зі значенням "SRV\*C:\Symbols\*https://msdl.microsoft.com/download/symbols")

Підключимося до ядра (File -> Attach to kernel) та вкажемо наступні параметри:

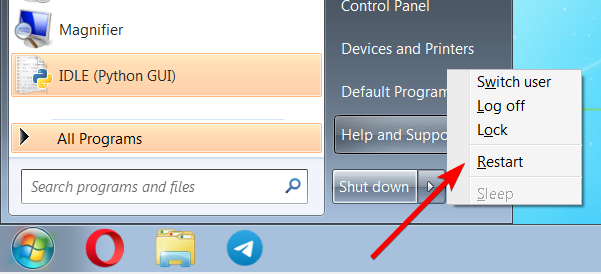


WinDbg буде очікувати перепідключення:



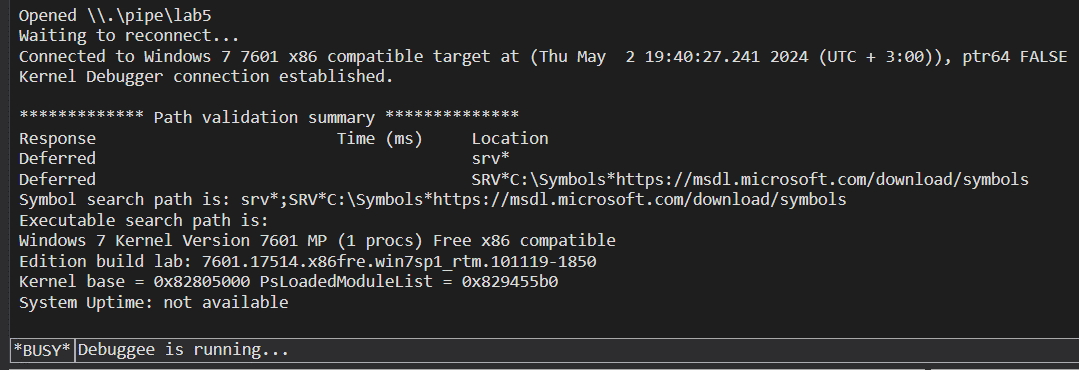
## 3.3) У Windows 7:

Перезавантажимо систему. Якщо система залагає при перезапуску, то можна її виключити (Power Off) у VirualBox і включити (Start):

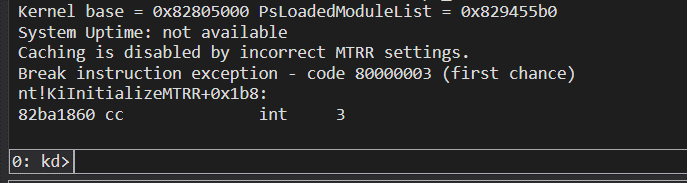


## 3.4) У Windows 10:

Після перезапуску Windows 7, WinDbg зможе підключитися до неї:

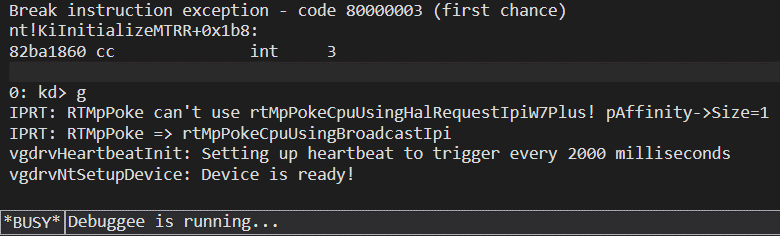


Чекаємо декілька хвилин, допоки WinDbg не зупиниться на брейкопоінті:

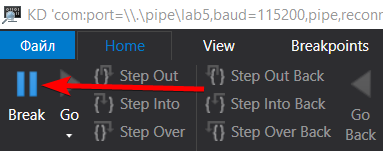


Продовжуємо виконання:

**kd> g**



Логінимося в Windows 7 та зупиняємо виконання в WinDbg:

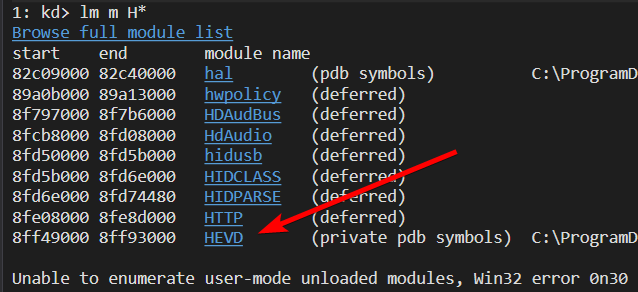


Завантажимо всі символи (дуже довгий процес, який треба виконати тільки 1 раз):

**kd> .reload /f \*.\***

Переконаємося, що драйвер HEVD є серед завантажених модулів:

**kd> lm m H\***



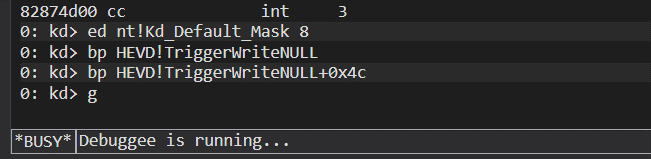
Тепер ставимо брейкпоінти на вразливу ф. TriggerWriteNULL():

**kd> ed nt!Kd\_Default\_Mask 8**

**kd> bp HEVD!TriggerWriteNULL**

**kd> bp HEVD!TriggerWriteNULL+0x4c**

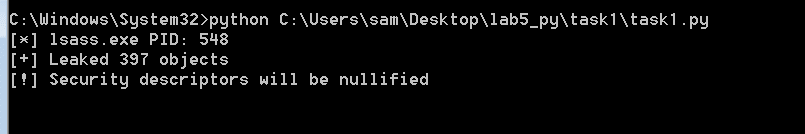
**kd> g**



## 3.5) У Windows 7:

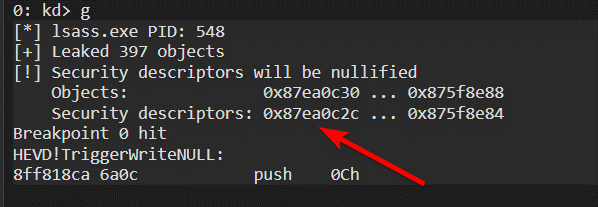
Виконаємо python2 експлойт task1.py:

**$ python C:\Users\sam\Desktop\lab5\_py\task1\task1.py**



## 3.6) У Windows 10:

WinDbg зупиниться на першому брейкпоінті:

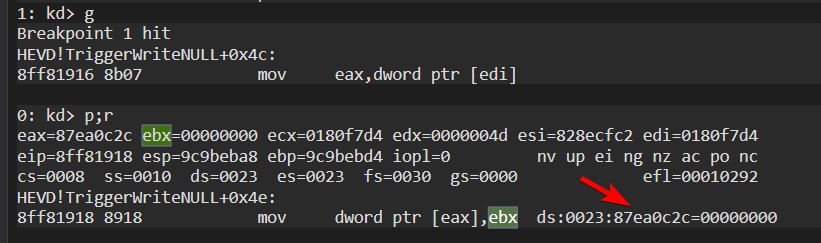


Адреса першого дескриптора захисту, який будемо зануляти: 0x87ea0c2c

Продовжуємо виконання. WinDbg зупиниться на другому брейкпоінті. Виведемо всі регістри:

**kd> g**

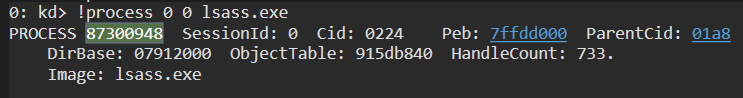
**kd> p;r**



Інструкція "mov dword ptr [eax],ebx" записує 0 за адресою 0x87ea0c2c

Виведемо інформацію про процес lssas.exe:

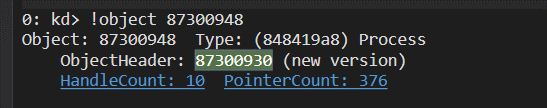
**kd> !process 0 0 lssas.exe**



Базова адреса процесу: 0x87300948

Дізнаємося адресу заголовка:

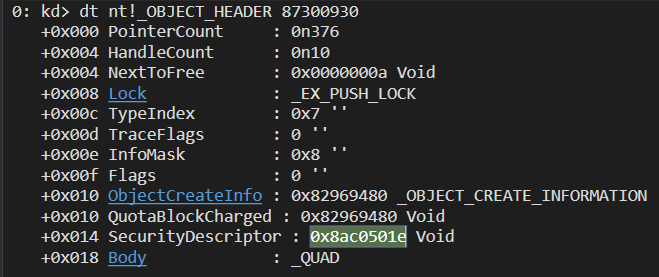
**kd> !object 87300948**



Адреса заголовка: 0x87300930

Виведемо заголовок:

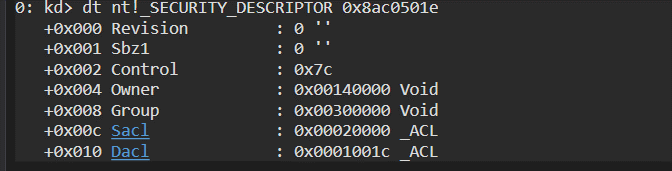
**kd> dt nt!\_OBJECT\_HEADER 87300930**



Адреса дескриптора захисту (офсет = 0x14): 0x8ac0501e

Це дійсно дескриптор захисту, що містить власника, його групу, SACL, DACL тощо:

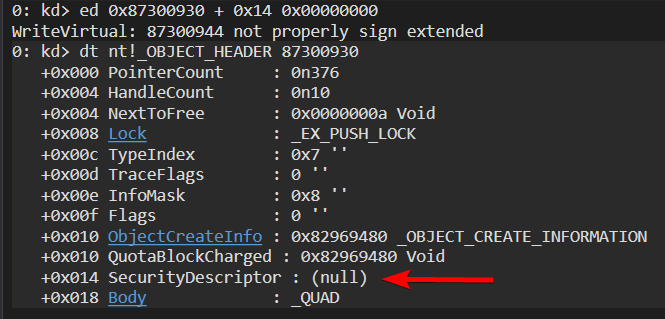
**kd> dt nt!\_SECURITY\_DESCRIPTOR 0x8ac0501e**



Земулюємо поведінку експлойту і занулимо дескриптор захисту вручну:

**kd> ed 0x87300930 + 0x14 0x00000000**

**kd> dt nt!\_OBJECT\_HEADER 87300930**



Таким чином можна занулити дескриптор захисту буд-якого об'єкту будь-якого процесу, навіть цей процес запущено від імені SYSTEM, а його адресний простір розташовано в ring0.

## 4) Суть експлойту:

**I Занулити дескриптори захисту об'єктів, які належать процесу** lsass.exe

1. Знайти PID процесу lsass.exe.

2. Знайти всі об'єкти процесу lsass.exe, адреси яких починаються з 0x87.

3. Для кожного знайденого об'єкта в пункті 2., занулити його дескриптор захисту. Це можна зробити, викликаючи TriggerWriteNULL(security\_descriptor\_ptr).

LSASS (Local Security Authority Subsystem Service) — це процес у Windows, який відповідає за виконання політики безпеки в ОС.

Після занулення дескрипторів lsass.exe, будь хто (Anyone) буде мати права Full control на всі процеси, які запущені від імені SYSTEM. Отже, будь-який непривілейований користувач зможе робити що завгодно з іншими системними процесами, зокрема winlogon.exe.

**II Відкрити** cmd.exe

4. Знайти PID процесу winlogon.exe.

5. Заінжектити шелкод в winlogon.exe.

5.1. Записати шелкод в буфер з правами доступу RWX.

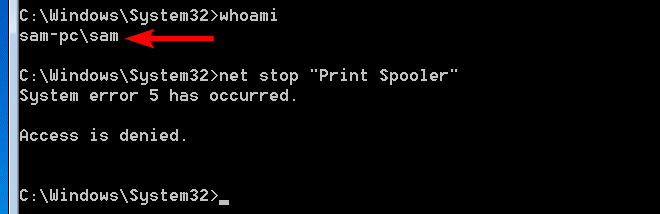
5.2. Створити потік, який буде виконувати шелкод (використати адресу буфера, створеного в пункті 5.1. в якості адреси початкової функції).

## 5) Виконання експлойту:

Звичайний користувач sam-pc\sam не може зупинити сервіс:

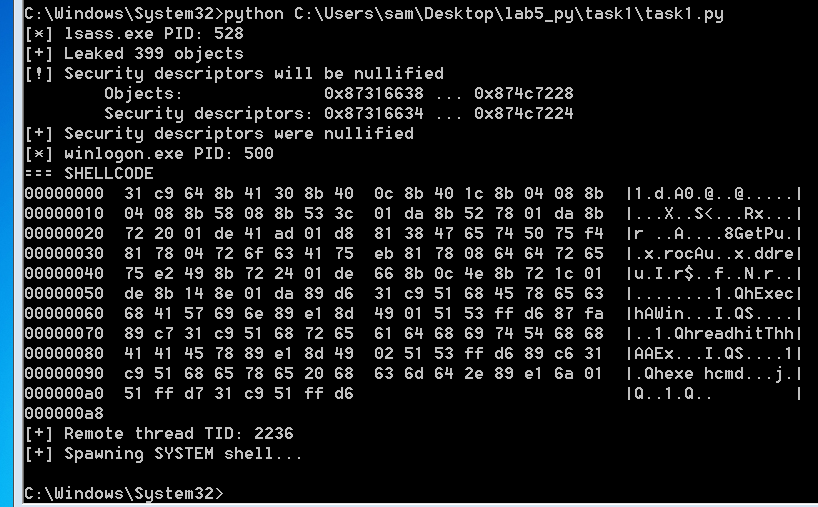
**$ whoami**

**$ net stop "Print Spooler"**

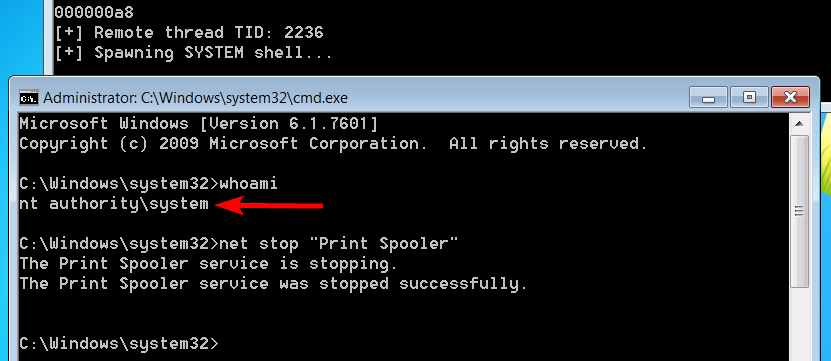


Запустимо скрипт task1.py:

**$ python C:\Users\sam\Desktop\lab5\_py\task1\task1.py**



Користувач nt authority\system може зупинити сервіс:



Отже, ескалація привілеїв пройшла успішно.

# Контрольнi питання

## 1. Якi методи обходу SMEP у Windows крiм ROP з модифiкацiєю CR4 Ви знаєте?

1. Атаки через вразливості у драйверах ядра.

2. [Використання](https://j00ru.vexillium.org/2011/06/smep-what-is-it-and-how-to-beat-it-on-windows/) NtQueueApcThreadEx().