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

BÁO CÁO THỰC HÀNH

Môn học: Cơ chế hoạt động của mã độc

Kỳ báo cáo: Bài tập 03

Tên chủ đề: KỸ THUẬT CHỐNG DỊCH NGƯỢC CỦA MÃ ĐỘC

GVHD: Phan Thế Duy

Ngày báo cáo: 22/07/2020

Nhóm MSN

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT230.K21.ANTT

STT	Họ và tên	MSSV	Email
1	Đoàn Nhật Minh	17520744	17520744@gm.uit.edu.vn
2	Hà Vũ Minh Ngọc	17520808	17520808@gm.uit.edu.vn
3	Nguyễn Hoàng Sơn	17520987	17520987@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:[1]

STT	Công việc	Kết quả tự đánh giá
1	Chức năng a: MessageBox	100%
2	Chức năng b: Anti-VM	100%
3	Chức năng c: Anti-Debugger	100%
4	Chức năng d: Packer/Decryptor	100%
5	…	…

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

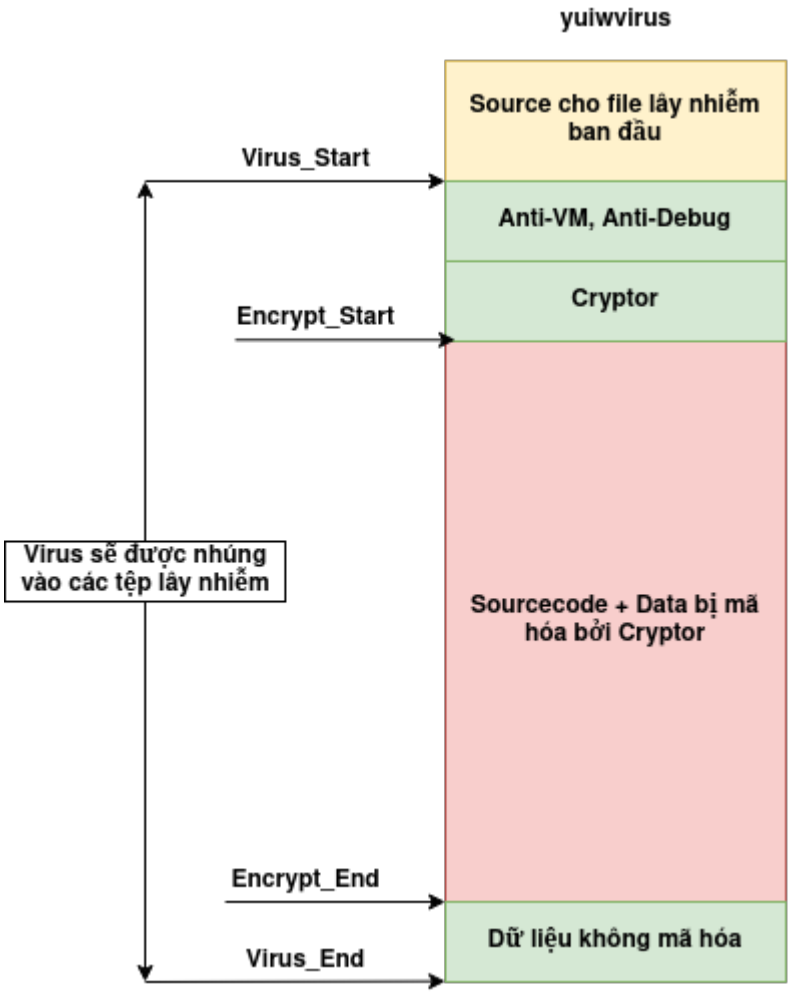
I. Tổng quan:

- Tạo một chương trình (dùng ngôn ngữ MASM) vừa là Virus vừa để lây nhiễm cho các tệp ban đầu.

→ Các file bị lây nhiễm virus có khả năng lây nhiễm lên các file khác trong cùng thư mục chứa khi được thực thi.

- Kết hợp với các kỹ thuật Anti-VM, Anti-Debugger và Decryptor để chống dịch ngược.

-Cấu trúc như sau:



-Sourcecode: yuiwvirus.asm

-Exec file: yuiwvirus.exe

-Có thể tải các tệp liên quan tại: https://drive.google.com/drive/folders/1PuJYY42s-Fgo4-iEh07-JlgLg_O3XlsE?usp=sharing

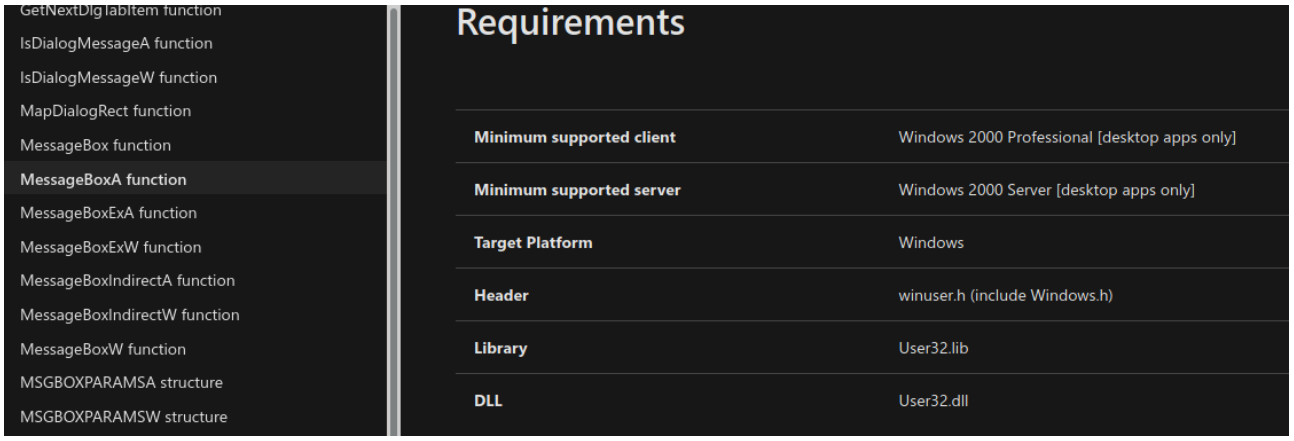
II. Chèn shellcode hiển thị thông điệp trong Cửa sổ cảnh báo trên Windows:

```
;Hien thi MessageBox theo yeu cau
push    0
lea     esi, [ebp + msgTitle]
push    esi
lea     esi, [ebp + msgContent]
push    esi
push    0
call    [ebp + offset _MessageBoxA]
```


-Dùng Gọi API MessageBox để thực hiện:

```
int MessageBoxA(
  HWND   hWnd,
  LPCSTR lpText,
  LPCSTR lpCaption,
  UINT   uType
);
```

-Các API được sử dụng được Virus tìm bằng cách tìm địa chỉ của Kernel32.dll sau đó lấy các địa chỉ của RVAExport, Export, AddressTableVA, NameTableVA, OrdinalTableVA, Counter để tìm các API tương ứng.



-Do MessageBox thuộc User32.lib, nên dùng API LoadLibrary của Kernel32.dll để load DLL này, sau đó dùng API GetProcAddress để lấy offset của API MessageBoxA.

III. Phát hiện máy ảo – Anti-VM:

- Sử dụng kỹ thuật kiểm tra lệnh cpuid

```
    ;Anti-VM
    xor     eax, eax
    inc     eax
    cpuid
    bt      ecx, 1Fh
    jb      GotoOEP
```

→ Do với máy vật lý thì bit thứ 31 của ecx luôn bằng 0 còn máy ảo sẽ là 1

→ Nếu phát hiện máy ảo sẽ thực hiện trả về chương trình gốc của tệp.

IV. Phát hiện máy ảo – Anti-Debugger:

- Sử dụng kỹ thuật NtGlobalFlags

```
    ;Anti-Debug
    ASSUME  FS:Nothing
    mov     eax, fs:[030h]
    ASSUME  FS:ERROR
    cmp     dword ptr [eax + 068h], 070h
    jz      GotoOEP
```

→ Nếu phát hiện trình debug sẽ thực hiện trả về chương trình gốc của tệp.

→ Với Anti-VM và Anti-Debug trên nếu phát hiện máy ảo/trình gỡ lỗi sẽ nhảy tới GotoOEP

```
GotoOEP:
    mov     eax, dword ptr [ebp + offset RecoveryOEP]
    call    eax
```

→ Tại đây gọi offset RecoveryOEP = ImageBase + OEP gốc của tệp ban đầu chưa bị lây nhiễm

→ Trả về chức năng của tệp.

IV. Áp dụng Packer/Decryptor:

-Sử dụng một bộ mã hóa giải mã với thuật toán mã hóa đối xứng đơn giản: **E = P ⊕ K**

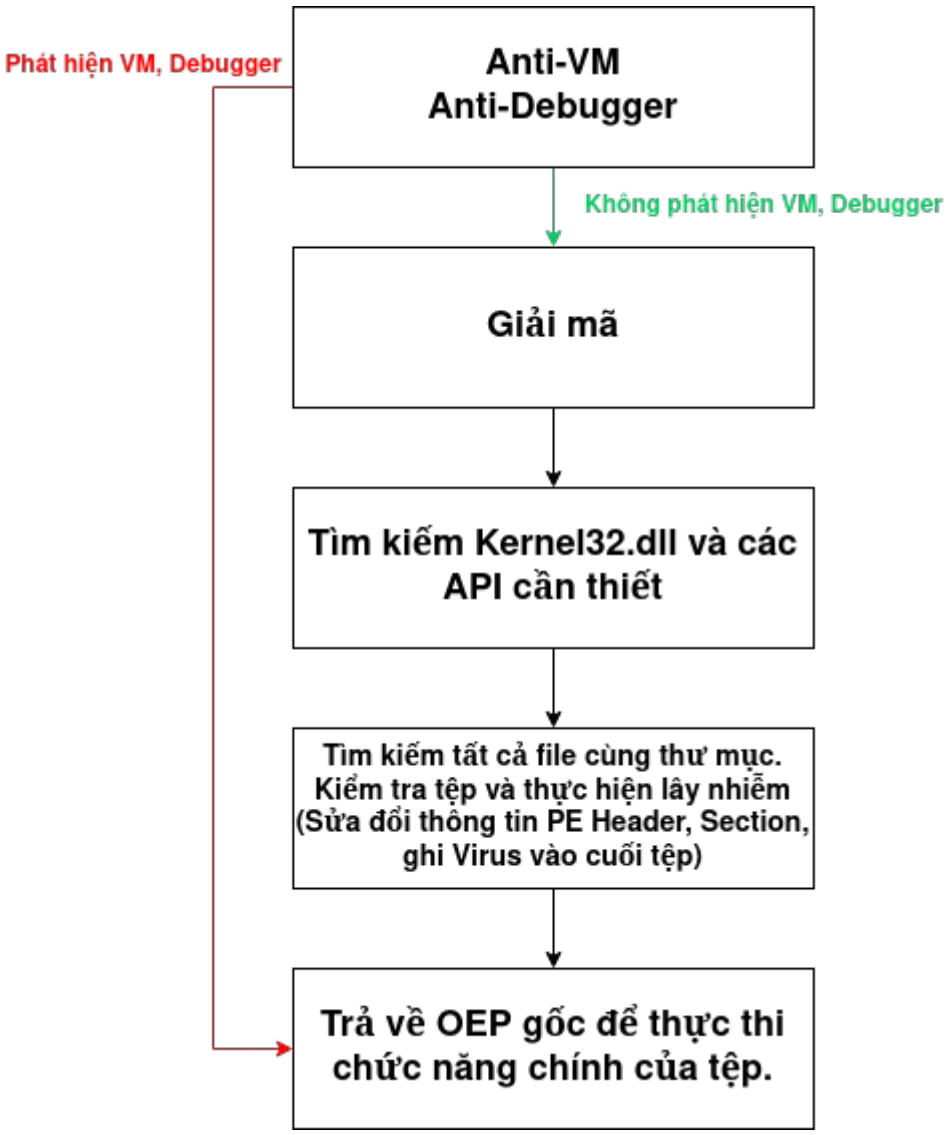
```
Cryptor:
    xor     eax, eax
    mov     eax, 012345678h; Init Key = 56h XOR 78h = 2eh
    mov     ecx, dword ptr [ebp + offset CryptSize]

Cryptor_Loop:
    mov     dl, byte ptr [esi]
    xor     byte ptr [esi], al
    xor     byte ptr [esi], ah
    mov     al, dl
    mov     ah, byte ptr [esi]
    add     esi, 1
    loop    Cryptor_Loop
    ret
```

→ Tôi viết cho nó rắc rối chút nhưng bản chất vẫn là E = P ⊕ K

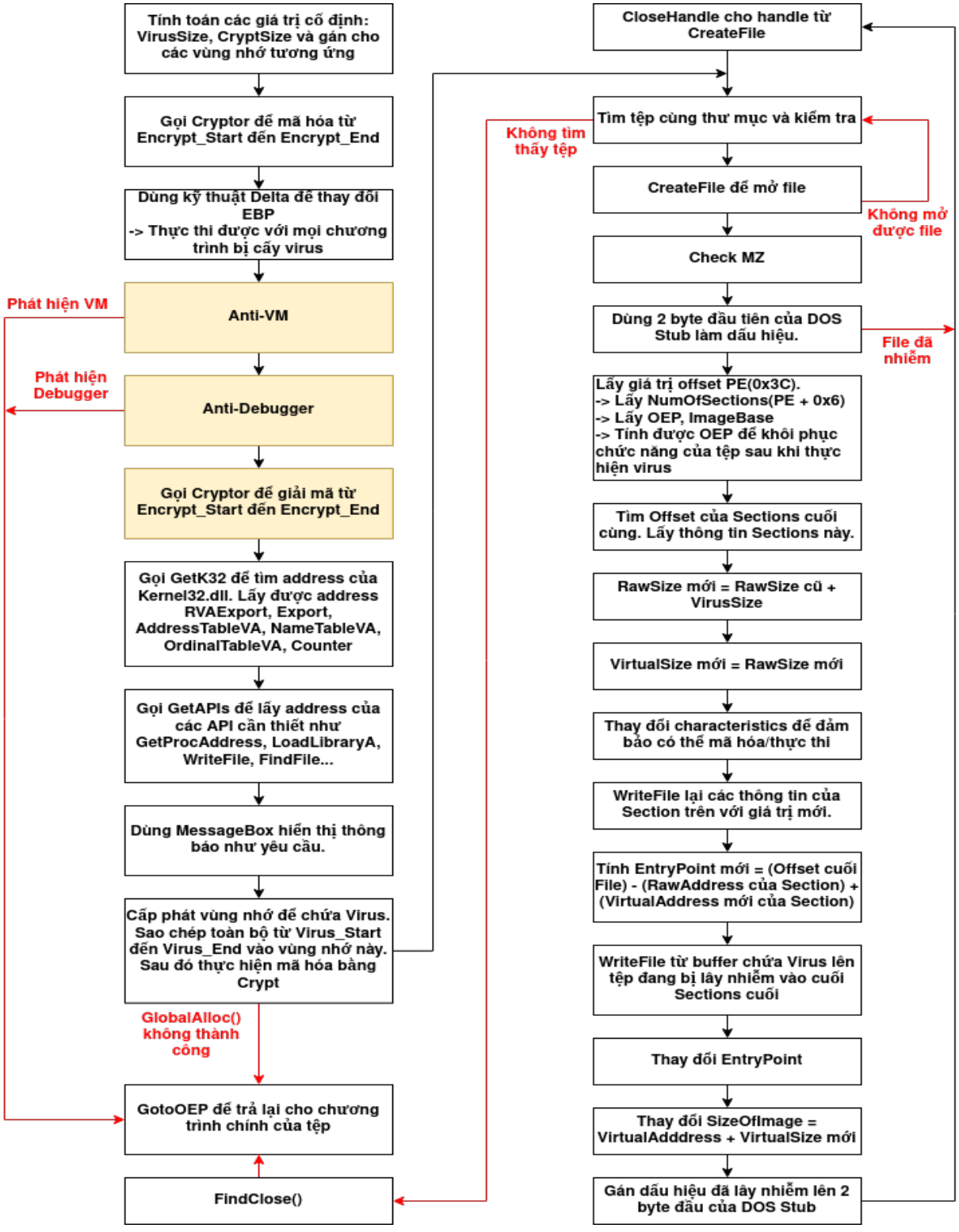
(Ei = Pi ⊕ Pi-1 ⊕ Ei-1 = Pi ⊕ Pi-1 ⊕ (Pi-1 ⊕ CONST) = Pi ⊕ K)

V. Quy trình virus thực hiện:

-Tôi có sơ đồ tổng quan như sau:



-Sơ đồ chi tiết:



-Quá trình thực thi kiểm tra Virus trên được thể hiện qua video sau: https://youtu.be/NnM_H4orrIM

Hết