Đây là một ứng dụng có giao diện sử dụng winapi. Chương trình sẽ bắt đầu từ hàm WinMain.

Graphical user interface, text, application, email

Description automatically generated

Đoạn đầu của WinMain thực hiện check quyền admin, do vậy ta phải chạy IDA bằng quyền admin để có thể debug được.

WinMain không có gì đặc biệt, chỉ cho biết function nào được sử dụng làm Procedure

Có hàm TLSCallback\_0, hàm này sẽ được gọi trước khi chương trình bắt đầu hàm main, malicious program có thể sử dụng để kiểm tra debugger hoặc làm gì đó trước khi breakpoint ở trong.

Graphical user interface, application

Description automatically generated

Hàm này có gọi NtQueryInformationProcess (debug tới đây để biết hàm nào đã được gọi) để check debugger. Nếu phát hiện debuger thì sẽ thay đổi byte thứ 10 của vùng nhớ 0x335018 thành 74h (vùng nhớ này đóng vai trò quan trọng trong việc giải lấy flag). Bypass bằng cách patch jump qua.

Graphical user interface, text, application

Description automatically generated

Quay lại Procedure của chương trình, khi bấm login thì chương trình sẽ get text đã nhập ở editbox, sau đó gọi sub\_401B40 với tham số là string đã nhập ở editbox:

Text

Description automatically generated

Đầu tiên input check phải lớn hơn 38 kí tự.

Hàm sub\_401FD0 nhận vào v23 và 1 byte tại một vùng nhớ khác. Vùng nhớ này chính là vùng nhớ mà TLS\_Callback đã sửa lúc trước. Nếu TLSCallback phát hiện ra debugger thì byte truyền vào sub\_401FD0 sẽ là 0x74, còn không thì là 0x75 (debug thử). Byte này được sử dụng làm key để suffle, thực hiện một logic gì đó và ghi kết quả vào v23 (đại khái tham số thứ 2 khác thì v23 sẽ khác và kết quả của flag sẽ sai).

Graphical user interface, text, application

Description automatically generated

Dword\_A32CB là 1 mảng gồm 38 dword, mỗi dword có giá trị từ 1-7 tương ứng với 7 case của switch

Dword\_A33F8 là 1 mảng gồm 38 dword, mỗi dword là index tới 1 char của userInput

Xem kĩ hơn switch-case: nhận thấy trong mỗi case đều thực hiện một số kĩ thuật anti debug, sau đó gọi tới hàm encrypt (tên tự đặt) với 3 tham số:

1. 0 hoặc 1 tùy kết quả check debugger
2. mảng v25 đã suffle và encrypt ở trên
3. tham số thứ 3 không rõ là gì :)

Kết quả của hàm encrypt trả về sẽ xor với user input và so sánh với giá trị tham chiếu, và return true nếu chính xác (như vậy ta cần tìm chính xác giá trị trả về của encrypt sau đó xor ngược lại với giá trị tham chiếu là tìm được user input). Sau cùng, hàm sub\_401B40 sẽ return true nếu password người dùng nhập vào là đúng.

Phân tích từng case:

Graphical user interface, text, application

Description automatically generated

Case 1: Anti-debug bằng PEB NtGlobalFlag, nếu đang bị debug thì NtGlobalFlag sẽ bằng 0x70. Chương trình check giá trị này có bằng 0x70 hay không bằng cách AND nó với 0x70 rồi so sánh với 0x70 (x AND x = x), nếu đúng thì tham số vào hàm encrypt sẽ là 1. Như vậy, khi viết script thì ta pass 0 vào.

Case 2, 3: Hàm sub\_A1600 được gọi trước. Trong hàm này thực hiện encrypt và so sánh với user input như đã nói ở trên. Kĩ thuật anti-debug được sử dụng ở đây đó là sử dụng Heap flags: <https://anti-reversing.com/Downloads/Anti-Reversing/The_Ultimate_Anti-Reversing_Reference.pdf> (đề cập ở trang 13). 2 trường hợp check trong nhiều phiên bản Windows khác nhau. Nếu có debugger thì sẽ pass 0 vào hàm encrypt.

Case 4: sử dụng HeapWalk để xem cuối Heap có xuất hiện dãy 0xABABABAB 2 lần trong cuối của Heap Block ([See this link for information](https://anti-debug.checkpoint.com/techniques/debug-flags.html)), nếu có thì sẽ có debugger và tham số của encrypt sẽ là 0.

Case 5: sử dụng CreateToolhelp32Snapshot và Process32Next để enum qua các process. Nếu có debugger thì pass 0 vào hàm encrypt sẽ là 0.

Case 6: Call BlockInput 2 lần. Như bình thường thì lần gọi thứ nhất sẽ trả về true, lần gọi thứ 2 sẽ trả về false. Nếu cả 2 lần gọi BlockInput đều trả về false có nghĩa là có thể process bị hook.(see this [link](https://anti-debug.checkpoint.com/techniques/interactive.html#blockinput)), để cả 2 BlockInput trả về giá trị không cùng nhau thì tham số vào hàm encrypt phải là 0.

Case 7: sử dụng NtQueryInformationProcess để query ra ProcessDebugFlag (see this [link](https://www.apriorit.com/dev-blog/367-anti-reverse-engineering-protection-techniques-to-use-before-releasing-software)), nếu query ra 0 có nghĩa là process bị debug. Ở đây nếu bị debug thì v20 sẽ có giá trị 0, v14 là 1 và v15 là 0, như vậy encrypt nhận 0 nếu bị debug.

Graphical user interface, text, application, email

Description automatically generated

Sau khi tổng hợp 7 case, ta cần viết 1 đoạn script mô phỏng hàm encrypt và truyền vào 0 hoặc 1 tương ứng với 7 case. Đơn giản nhất là copy pseudo code của ida ném vào visual studio. File source keygen được đính kèm cùng thư mục với write up này.

Kết quả: key: I\_10v3-y0U\_\_wh3n Y0u=c411..M3 Senor1t4 (bài hát này khá hay :) )  
 flag: vcstraining{Th3\_U1tiM4t3\_ant1\_D3Bu9\_ref3r3ncE}

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated