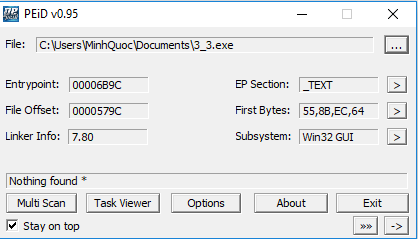
1. **Công cụ hỗ trợ**

Peid, OllyDbg

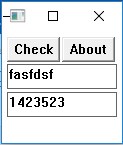
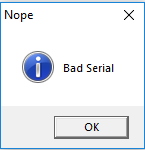
* Peid: công cụ để kiểm tra tệp thực thi đã bị pack chưa.
* OllyDbg: công cụ Debug.

1. **Thao tác**

Đầu tiên, dùng **Peid** để kiểm tra tệp cần thực thi.

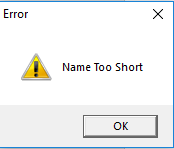


Kết quả là Nothing found \*, 1.1.exe không bị pack.

Trước tiên chạy thử file 3\_3.exe.

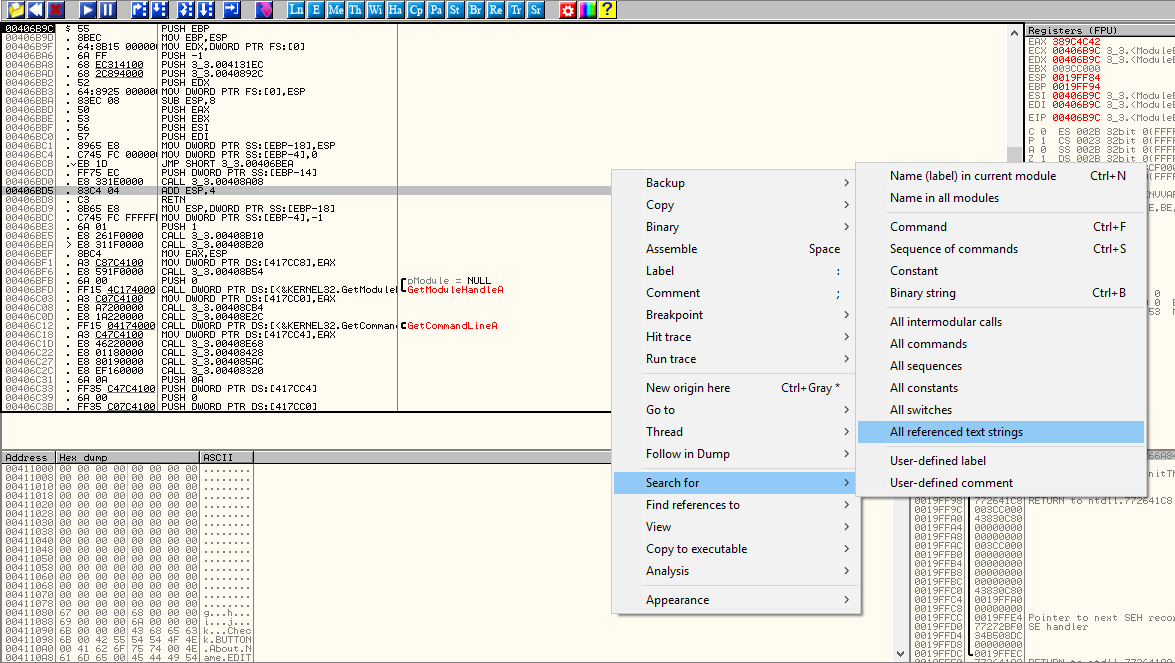
Ta thử mới Name và Serial thì kết quả hiện thông báo Bad Serial như vậy Name và Serial sai.

Ta thử tiếp với Name chỉ dưới 5 ký tự cho kết quả Name too short vậy Name phải trên 4 ký tự.

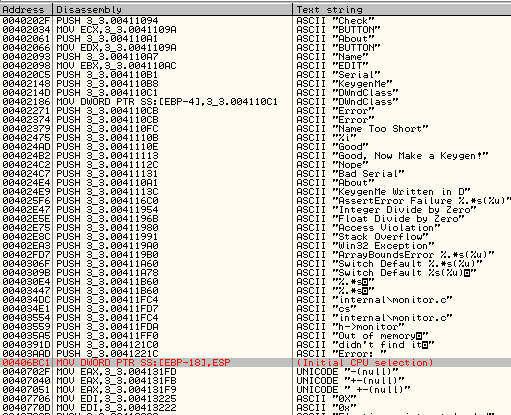


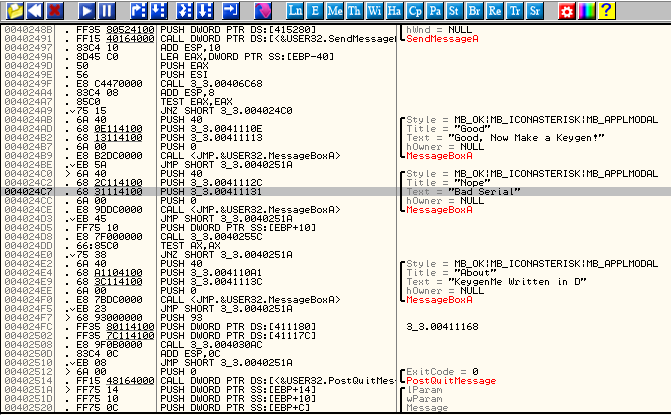
Sau khi có được các thông báo trên ta tiến hành phân tích mã hợp ngữ bằng OllyDbg.

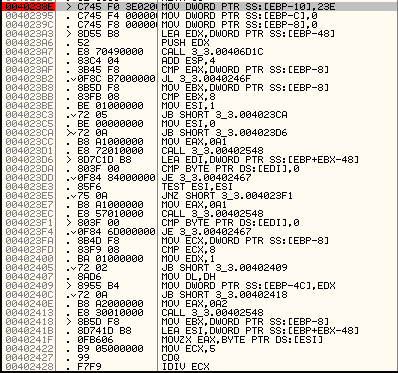
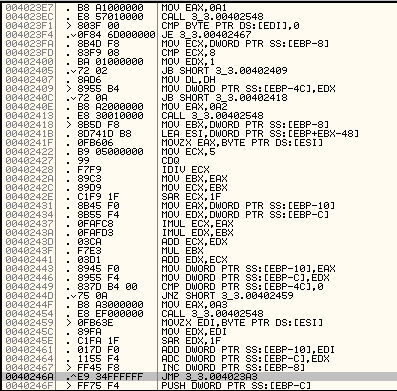
Chạy **OllyDbg** và mở 3\_3.exe. Nhấn chuột phải tại **Diassembler Window -> Search for -> All referenced text strings.**

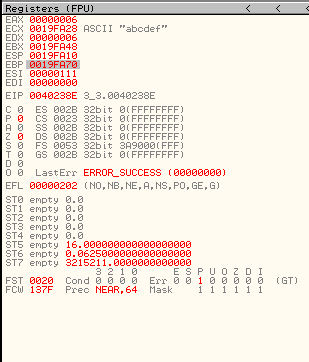
****

Cửa sổ xuất hiện, ta sẽ thấy có nhiều chuỗi chúng ta cần chú ý đây có thể là manh mối để ta tìm key.



Ta nhấn đúp chuột tại dòng **“Bad Serial”** ta sẽ nhảy đến đoạn code phát sinh nó.

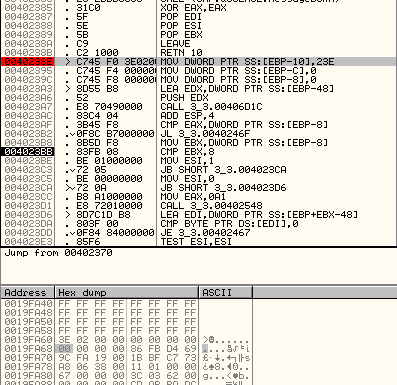
Ta đoán được đoạn phát sinh code sẽ nằm bên dưới câu lệnh”**Name Too Short**” tại địa chỉ **0040238E** đến địa chỉ **0040246A** , đặt **break point** tại địa chỉ trên tiến hành F9 để chạy với **Name là abcdefg** và **Serial là 123456** và F7 để chạy từng bước.

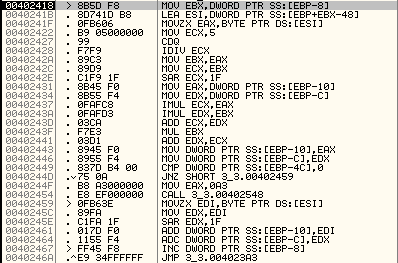
Ta nhận thấy số ký tự trong chuỗi chỉ còn là 6. Vậy hàm phát sinh code chỉ giới hạn trong 6 ký tự.

Đoạn lệnh đầu tiên là lưu giá trị 23E vào địa chỉ EBP-10 với EBP=0019FA70.



Và tiếp theo là lưu địa chỉ EBP-8 vào EBX dể xét điều kiện.



*****Dòng lệnh phát sinh key***

***Phân tích các dòng lệnh:***

Độ dài chuỗi bắt buộc nằm trong khoảng từ [5-6] bất kể chuỗi dài hay ngắn đều chỉ lấy tối đa 6 ký tự hoặc nếu chuỗi nhỏ hơn 6 và lớn hơn 4 sẽ lấy độ dài là 5 tương đương 5 ký tự

Lấy từng ký tự lưu vào EAX tại đia chỉ **00402418.**

Gán ECX=5 (**00402422**).

Chia EAX cho ECX giá trị lưu vào EAX (**00402428**).

EAX sẽ được gán cho EBX và ECX(**0040242A** và **0040242C**).

Sau đó EAX lưu giá trị tại vị trí mà 23E được tức là EBP-10 (**00402431**).

Nhân EAX cho EBX kết quả lưu vào EAX(**0040243F**).

Lưu EAX vào địa chỉ EBP-10(**00402443**).

Gán ESI =0019FA28 ASCII”...” chính là các ký tự vào EDI .



Gán EDI vào EDX.

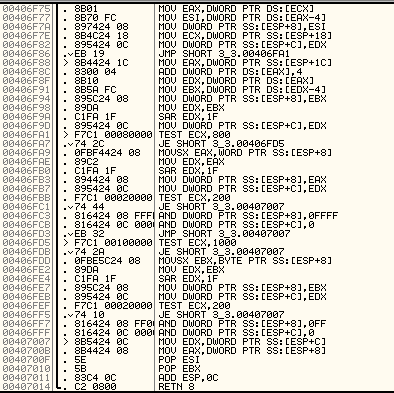
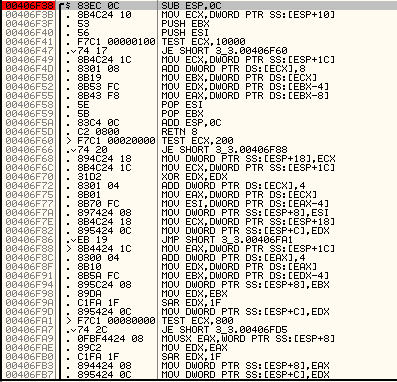
Cộng địa chỉ EBP-10 với EDI(**00402461**).

Do thanh ghi chỉ chứa tối đa 8 bit do đó nếu vượt quá 8 bit thì sẽ lấy phần chưa tràn để lặp hàm

Tăng 1 đơn vị tại địa chỉ EBP-8 (0040246A), nếu bằng 6 thì sẽ dừng với câu lệnh sau.

Lệnh gọi 3\_3.00406D1C trả về số lượng ký tự trong chuỗi(độ dài) và so sánh với địa chỉ EBP-8 nếu bằng 6 sẽ gọi lệnh 3\_3.0040246F dể thoát còn không bằng sẽ tiếp tục vòng lặp.

***Dòng lệnh kiểm tra âm dương(00406F38-00407014)***



Ta quan tâm đến các dòng lệnh sau.



Gán giá trị tại dịa chỉ mà EAX=0019F9DC vào EDX=0019FA0C (**00406F8F**).

Gán giá trị tại dịa chỉ mà EDX-4 vào EBX(**00406F91**).

Gán EBX vào EDX.

Dịch phải EDX 31 lần.

Gán giá trị EDX vào địa chỉ ESP+C(**0019F914**).

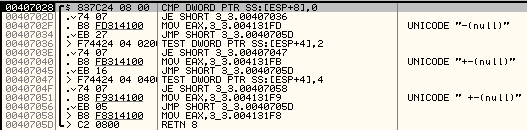
Gán lại giá trị tại địa chỉ ESP+C vào EDX.

Gán giá trị tại địa chỉ ESP+8 vào EAX.

Kiểm tra EAX có bằng 0 sau nó nhảy đến **3\_3.00407627**.



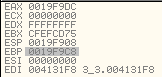
Gọi lệnh **3\_3.00407028**.

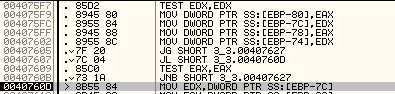


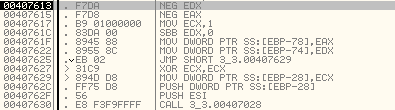
Nếu giá trị tại địa chỉ [ESP+8]=0 thì nhảy lệnh 3\_3.00407036 thực hiện kiểm tra EDX có bằng 2 và 4 không vì Shift Right chỉ cho kết quả là 1 hoặc 0 nên câu lệnh trên sẽ RETN 8 và tiếp tục các đoạn lệnh tiếp theo.

**Như vậy giá trị chuỗi key được tạo ra không thay đổi.**

Nếu EDX=FFFFFFFF

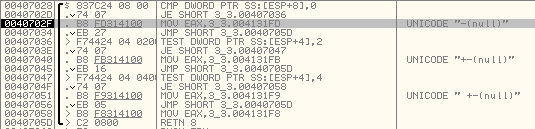


Kiểm tra EDX có bằng 0 và nhảy lệnh 3\_3.0040760D.

******

***Bù 2 EDX và EAX nhưng không lấy dấu vì vượt quá khả năng lưu trữ của thanh ghi***

Khi đó EDX sẽ có giá trị bằng 1 và gọi lệnh 3\_3.00407028.



Nếu giá trị tại địa chỉ [ESP+8]=1 thì gán EAX=004131FD UDICODE”-(null)”.

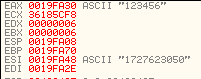
**Như vậy giá trị chuỗi key được tạo ra chính là chuỗi (“-“+”bù 2 của key ở hệ 10”).**

***Dòng lệnh so sánh Serial nhập vào và key phát sinh từ UserName***

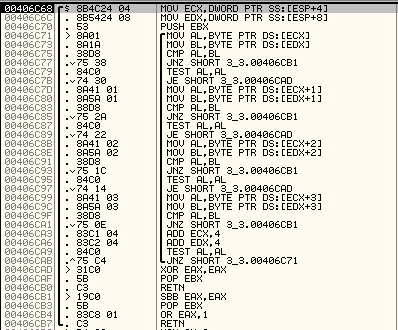
Được gọi từ địa chỉ sau



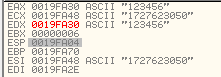
Chuỗi nhập vào và chuỗi được phát sinh từ UserName.



Đoạn code so sánh(**00406C68**-**00406CB7**)



Lưu giá trị ESP-4 vào ECX tức là key phát sinh từ UserName.

Lưu giá trị ESP+8 vào EDX tức là chuỗi được nhập vào từ bàn phím.

Gán ký tự đầu giá trị địa chỉ ECX vào AL.

Gán ký tự đầu giá trị địa chỉ EDX vào BL.

So sánh AL và BL.

Vì hai giá trị bằng nhau nên sẽ trừ nhau sẽ bằng 0 nên sẽ không nhảy lệnh JNZ .

Kiểm tra AL có bằng 0.

Gán AL bằng giá trị tiếp theo trong địa chỉ ECX.

Gán BL bằng giá trị tiếp theo trong địa chỉ EDX.

So sánh AL và BL.



Vì hai giá trị bằng nhau nên sẽ trừ nhau không bằng 0 nên sẽ nhảy lệnh JNZ .

Và thoát đoạn lệnh.

Như vậy theo như đoạn lệnh sẽ so sánh từng ký tự trong chuỗi và trong 1 vòng lặp sẽ có 4 ký tự được so sánh, sau đó quay trở lại vòng lặp và tiếp tục so sánh 4 ký tự tiếp theo.

1. **Thuật toán về phát sinh key**

Vòng for duyệt qua từng ký tự và dùng công thức.

Num=23E;

Num=Num\*(user[i]/5)+user[i].

Sau khi vòng for duyệt xong ta sẽ có key chính bằng giá trị Num.

Gán Num\_1=Num.

Dịch phải Num 31 lần.

Nếu Num=0->xuất Num\_1.

Nếu Num=1->ta cần xử lý bằng cách bù 2 Num\_1 và ghép chuỗi (“-“+”Bù 2 của Num\_1 tại hệ 10”)

1. **Ví dụ minh họa vấn đề phát sinh key**

***Với key dương***

UserName=asdasd

Ta có các vòng For

Vòng 1:

Giá trị key=2AFB

Vòng 2:

Giá trị key=3DD00

Vòng 3:

Giá trị key=4D4464

Vòng 4:

Giá trị key=5BC13CD

 Vòng 5:

Giá trị key=83E5C7DE

 Vòng 6:

Giá trị key=4DF39DBC(1307811260)

Xét âm dương:

Key = 0100 1101 1111 0011 1001 1101 1011 1100 >> 31

Key=0

**Vậy Key=1307811260**

***Với key âm***

UserName=zzzzzzzz

Ta có các vòng For

Vòng 1:

Giá trị key=364A

Vòng 2:

Giá trị key=5176A

 Vòng 3:

Giá trị key=7A326A

 Vòng 4:

Giá trị key=74BA6B

 Vòng 5:

Giá trị key=12F17A6A

 Vòng 6:

Giá trị key=C6A37A6A(3332602474‬)

Xét âm dương:

Key = 1100 0110 1010 0011 0111 1010 0110 1010 ‬>> 31

Key=1-> lấy bù 2 của key không lấy dấu là 395C 8596 sau đó đổi ra hệ 10 là 962364844 và thêm dấu trừ vào trước.

**Vậy Key=** **-962364822**