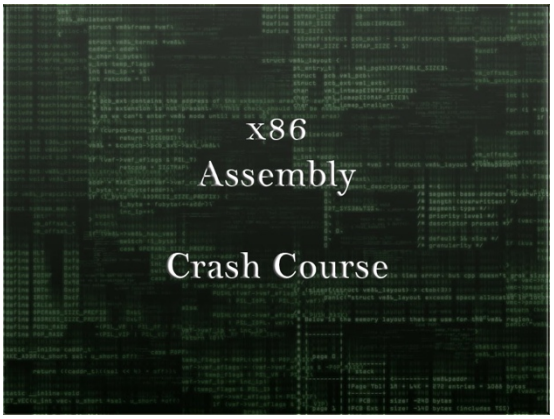
BÁO CÁO THỰC HÀNH



CHỦ ĐỀ: HỢP NGỮ X86
CRACK PHẦN MỀM ĐƠN GIẢN

Tên: Đỗ Thành Nhơn - 1512387
Nguyễn Quốc Huy - 1512203
Châu Hoàng Long - 1512292
Lớp: 15CNTN
GVHD: Nguyễn Thanh Quân

I. Tổng quan đồ án

1. Yêu cầu
   - Cơ bản: với mỗi crackme sinh viên phải chỉ ra đoạn phát sinh key, giải thích ý nghĩa và đưa ra một key tương ứng với user name minh họa.
   - Nâng cao: viết chương trình keygen để phát sinh khóa từ username người dùng nhập vào.

2. Đề bài
   (1512387 + 1512203 + 1512292) % 3 + 1 = 1.

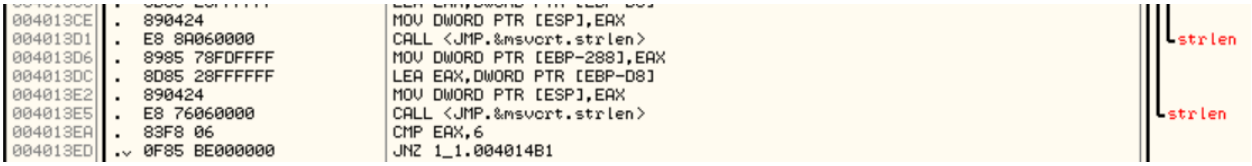
3. Mức độ hoàn thành đồ án.
   100% hoàn thành các yêu cầu.

II. Giải pháp

1. Câu 1_1.exe
   - Good boy

   - Bad boy

   - Thuật toán kiểm tra key



Key do người dùng nhập vào đầu tiên sẽ được so sánh độ dài với 6. Nếu độ dài khác 6 thì sẽ đi đến bad boy và kết thúc. Ngược lại độ dài bằng 6 sẽ tiến hành biến đổi và kiểm tra key.

```
004013F3  .  0FB685 28FFFFFF     MOVZX EAX,BYTE PTR [EBP-D8]
004013FA  .  34 34               XOR AL,34
004013FC  .  0FBEC0              MOVSX EAX,AL
004013FF  .  8985 74FDFFFF       MOV DWORD PTR [EBP-28C],EAX
00401405  .  0FB685 29FFFFFF     MOVZX EAX,BYTE PTR [EBP-D7]
0040140C  .  34 78               XOR AL,78
0040140E  .  0FBEC0              MOVSX EAX,AL
00401411  .  8985 70FDFFFF       MOV DWORD PTR [EBP-290],EAX
00401417  .  0FB685 2AFFFFFF     MOVZX EAX,BYTE PTR [EBP-D6]
0040141E  .  34 12               XOR AL,12
00401420  .  0FBEC0              MOVSX EAX,AL
00401423  .  8985 6CFDFFFF       MOV DWORD PTR [EBP-294],EAX
00401429  .  0FBE85 2BFFFFFF     MOVSX EAX,BYTE PTR [EBP-D5]
00401430  .  35 FE000000         XOR EAX,0FE
00401435  .  8985 68FDFFFF       MOV DWORD PTR [EBP-298],EAX
0040143B  .  0FBE85 2CFFFFFF     MOVSX EAX,BYTE PTR [EBP-D4]
00401442  .  35 DB000000         XOR EAX,0DB
00401447  .  8985 64FDFFFF       MOV DWORD PTR [EBP-29C],EAX
0040144D  .  0FB685 2DFFFFFF     MOVZX EAX,BYTE PTR [EBP-D3]
00401454  .  34 78               XOR AL,78
00401456  .  0FBEC0              MOVSX EAX,AL
```

```
Registers (FPU)                                <
EAX 0060FCC0 ASCII "4D11628EBE1D"
ECX 11BCED56
EDX 0060FD90 ASCII "551A719ABE1E"
EBX 00004000
ESP 0060FC50
EBP 0060FF38
ESI 00401220 1_1.<ModuleEntryPoint>
EDI 00401220 1_1.<ModuleEntryPoint>

EIP 004014C4 1_1.004014C4
```

Sau khi nhập thử key là abcdef, đặt break point ở lệnh

```
004014C4 |·  E8 87050000 |CALL <JMP.&msvcrt.strcmp>          |└MSVCRT.strcmp
```

Ta được key cứng của bài toán là 4D11628EBE1D.

Quy luật của key:

0x4D XOR 0x34 => 0x79 => ‘y’

0x11 XOR 0x78 => 0x69 => ‘i’

0x62 XOR 0x12 => 0x70 => ‘p’

0x8E XOR 0x0FE => 0x70 => ‘p’

0xBE XOR 0x0DB => 0x65 => ‘e’

0x1D XOR 0x78 => 0x65 =>’e’

Vậy bài này là key cứng “yippee” nên không có keygen.

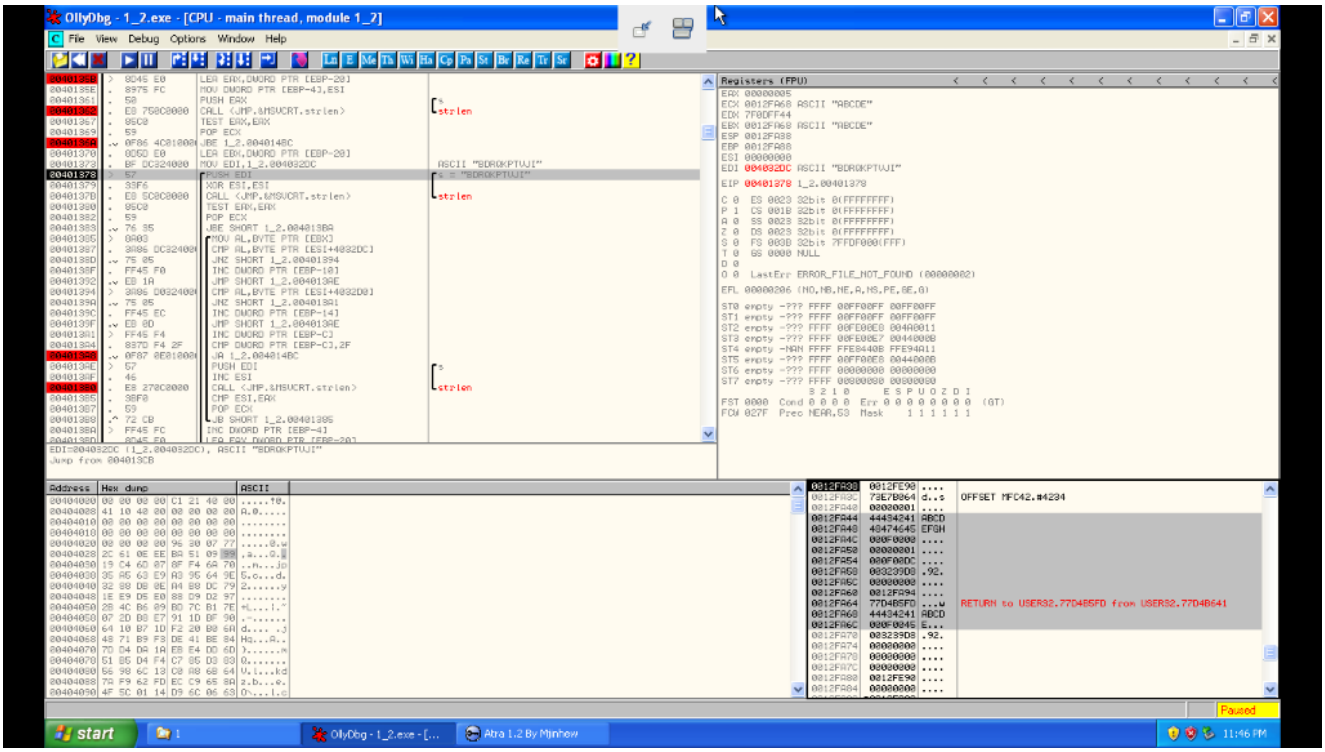
2. Câu 1_2.exe
    - Good boy

```
004014AA    .  85C0          TEST EAX,EAX
004014AC    .  6A 00         PUSH 0
004014AE    .  68 C0444000   PUSH 1_2.004044C0                ASCII "Atra"
004014B3    .v 75 0E         JNZ SHORT 1_2.004014C3
004014B5    .  68 9C444000   PUSH 1_2.0040449C                ASCII "The entered serial number is VALiD!"
004014BA    .v EB 0C         JMP SHORT 1_2.004014C8
004014BC    >  6A 00         PUSH 0
```

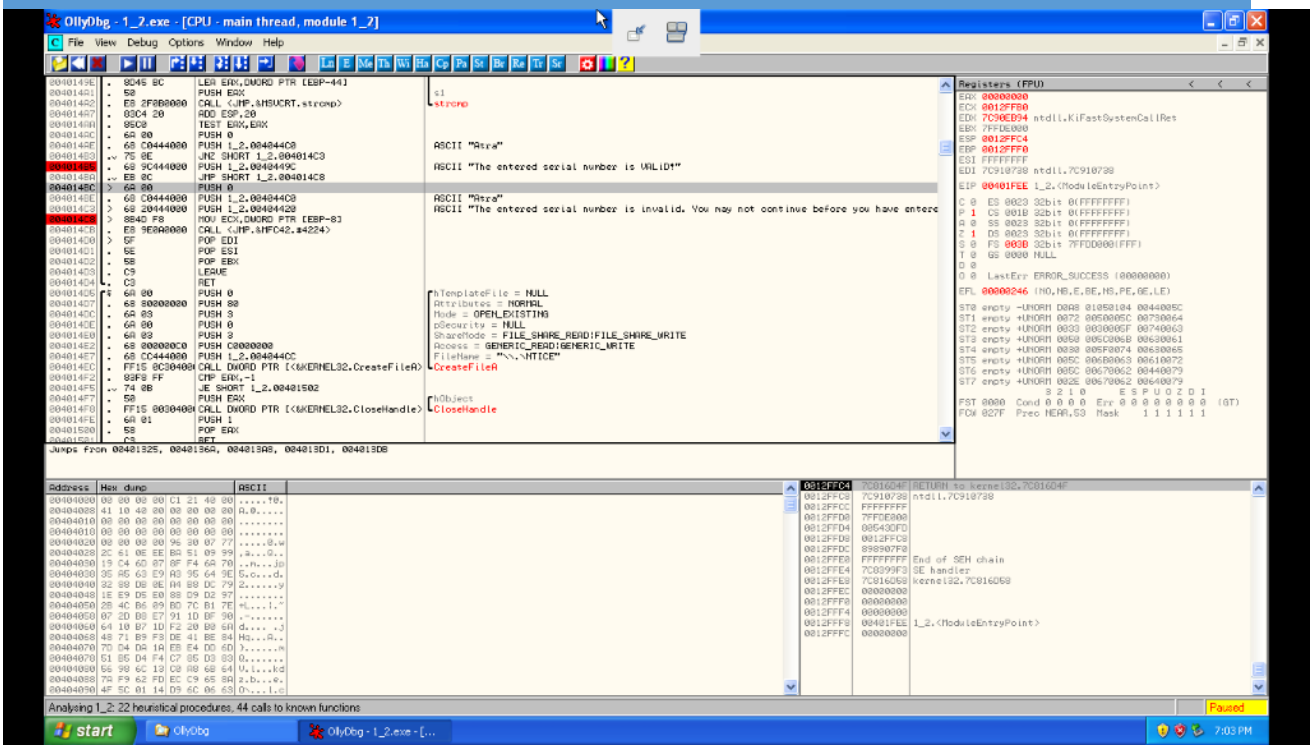
- Bad boy

```
004014BC    >  6A 00         PUSH 0
004014BE    .  68 C0444000   PUSH 1_2.004044C0                ASCII "Atra"
004014C3    >  68 20444000   PUSH 1_2.00404420                ASCII "The entered serial number is invalid. You may not continue before you have entered an valid serial number. Please re-enter."
004014C8    >  8B4D F8       MOV ECX,DWORD PTR [EBP-8]
004014CB    .  E8 9E0A0000   CALL <JMP.&MFC42.#4224>
004014D0    >  5F            POP EDI
004014D1    .  5E            POP ESI
```

Đánh dấu các vị trí liên quan tới xử lý chuỗi (có các dòng như strcmp, strlen, ...) thì ta sẽ tìm ra được địa chỉ lưu chuỗi serial number mà ta vừa nhập. Ví dụ khi nhập serial number là ABCDE ABCDEFGH thì ta thấy chuỗi serial thứ nhất lưu tại địa chỉ 0012FA68, còn chuỗi serial thứ hai lưu tại địa chỉ 0012FA44.



Để mở cửa sổ thông báo nhập sai serial number (để chạy vào bad boy), lệnh 004014BC 6A 00 PUSH 0 cần được thực hiện. Mà có 5 lệnh nhảy đến lệnh trên là 00401325, 0040136A, 004013A8, 004013D1, 004013DB



Tạo breakpoint tại các vị trí trên, chạy thử chương trình và nhập đại một key nào đó (ví dụ ABCDE ABCDEFGH), chương trình nhảy qua 0040136A 1 lần, nhảy qua 004013A8 48 lần.

Nhập đại một key khác (ví dụ 12345 12345678), chương trình nhảy qua 0040136A 1 lần, nhảy qua 004013A8 45 lần.

Xét đoạn lệnh 0040136A:

00401367 85C0 TEST EAX,EAX

0040136A 0F86 4C010000 JBE 1_2.004014BC

Ta có JBE nhảy khi CF = 1 hoặc ZF = 1. Còn lệnh TEST đưa CF = 0, còn ZF = 1 nếu EAX AND EAX = 0 (có nghĩa là EAX = 0) và ZF = 0 khi ngược lại. Do đó, ZF = 1 khi EAX = 0, hay nói cách khác, JBE chỉ nhảy khi EAX = 0.

Xét đoạn lệnh 004013D1:

004013CD 837D F8 03 CMP DWORD PTR [EBP-10],3

004013D1 0F85 E5000000 JNZ 1_2.004014BC

Theo các lệnh ở trên thì đoạn lệnh này so sánh [EBP-10] với 3, với [EBP-10] là số lượng ký tự trong chuỗi serial thứ nhát mà xuất hiện trong chuỗi "BDRQKPTVJI"

Xét đoạn lệnh 004013DB:

004013D7 837D EC 02 CMP DWORD PTR [EBP-14],2

004013DB 0F85 DB000000 JNZ 1_2.004014BC

Theo các lệnh ở trên thì đoạn lệnh này so sánh [EBP-10] với 2, với [EBP-10] là số lượng ký tự trong chuỗi serial thứ nhất mà xuất hiện trong chuỗi "0123456789"

Suy ra: Chuỗi serial thứ nhất là hợp lệ khi có 3 ký tự trong chuỗi "BDRQKPTVJI" và 2 ký tự trong chuỗi "0123456789"

Giờ xét tới chuỗi serial thứ hai: Qua các lệnh trong đoạn từ lệnh

004013E1 8D45 E0 LEA,DWORD PTR [EBP=20]

tới lệnh

004014BD 0068 C0 ADD BYTE PTR [EAX=40],CH

thì ta hiểu cách sinh serial number thứ hai như sau:

- Gán giá trị tại các địa chỉ 0012F9BC, 0012F9C0, 0012F9C4, 0012F9C8 thành 0x64752301, 0xEFCDAB89, 0x98BADCFE, 0x10325476
- Thực hiện một loạt các phép biến đổi giá trị các biến trên, kết hợp với các dữ liệu cố định trong vùng nhớ từ 0012F9D4 tới 0012FA13 (5 byte đầu của vùng nhớ này là chuỗi serial thứ nhất, byte kế tiếp mang giá trị 0x80, byte thứ 56 mang gía trị 0x28, các bayte còn lại của vùng nhớ mang giá trị 0x0)
- Sau khi thực hiện, chép kêt quả của 4 vùng nhớ trên lần lượt vào 0012FA5C, 0012FA60, 0012FA64, 0012FA68
- Lấy xỏr của 4 giá trị tại 4 vùng nhớ vừa được chép vào
- Chuyển giá trị này sang dạng chuỗi (ở hệ hexa)
- Thực hiện tiếp các lệnh biến đổi từng byte của chuỗi (shift left, xor, ...) để tạo thành chuỗi kết quả (serial number thứ hai) (có sử dụng các dữ liệu trong vùng nhớ cố định).

Về chương trình keygen: Chương trình đọc dữ liệu từ file fixed_data.txt chứa dữ liệu đã qua xử lý về vùng nhớ cố định (dữ liệu thô trong file memory.txt được xử lý bởi chương trình có mã nguồn preproc.cpp), sau đó sinh ra tất cả các key có thể có và lưu trong file output.txt. Ngoài ra chương trình còn cung cấp giao diện cho phép người dùng nhập vào serial number thứ nhất và tính toán ra serial number thứ hai.

Một ví dụ về key: BBB12 - 26E06908

3. Câu 1_3.exe
   - Good boy

```
0040134D  $  6A 30        PUSH 30                                  Style = MB_OK|MB_ICONEXCLAMATION|MB_APPLMODAL
0040134F  .  68 29214000  PUSH 1_3.00402129                        Title = "Good work!"
00401354  .  68 34214000  PUSH 1_3.00402134                        Text = "Great work, mate!♪Now try the next CrackMe!"
00401359  .  FF75 08      PUSH DWORD PTR [EBP+8]                   hOwner
0040135C  .  E8 D9000000  CALL <JMP.&USER32.MessageBoxA>           MessageBoxA
00401361  L. C3           RET
```

   - Bad boy

```
00401362  $  6A 00        PUSH 0                                   BeepType = MB_OK
00401364  .  E8 AD000000  CALL <JMP.&USER32.MessageBeep>           MessageBeep
00401369  .  6A 30        PUSH 30                                  Style = MB_OK|MB_ICONEXCLAMATION|MB_APPLMODAL
0040136B  .  68 60214000  PUSH 1_3.00402160                        Title = "No luck!"
00401370  .  68 69214000  PUSH 1_3.00402169                        Text = "No luck there, mate!"
00401375  .  FF75 08      PUSH DWORD PTR [EBP+8]                   hOwner
00401378  .  E8 BD000000  CALL <JMP.&USER32.MessageBoxA>           MessageBoxA
0040137D  L. C3           RET
```

```
004013AD  .  6A 30        PUSH 30                                  Style = MB_OK|MB_ICONEXCLAMATION|MB_APPLMODAL
004013AF  .  68 60214000  PUSH 1_3.00402160                        Title = "No luck!"
004013B4  .  68 69214000  PUSH 1_3.00402169                        Text = "No luck there, mate!"
004013B9  .  FF75 08      PUSH DWORD PTR [EBP+8]                   hOwner
004013BC  .  E8 79000000  CALL <JMP.&USER32.MessageBoxA>           MessageBoxA
004013C1  L> C3           RET
```

   - Đặt break point và nhập thử abc – 123 chạy debug ta thấy được nơi lưu trữ user name và password trong quá trình kiểm tra:

```
Registers (FPU)
EAX 00000001
ECX 179469E9
EDX 00000000
EBX 00401128 1_3.WndProc
ESP 0019FDD4
EBP 0019FDE8
ESI 0040218E ASCII "abc"
EDI 00000000

EIP 00401382 1_3.00401382
```

```
Registers (FPU)
EAX 00000000
ECX 179469E9
EDX 00000000
EBX 00000000
ESP 0019FDCC
EBP 0019FDE8
ESI 0040217E ASCII "123"
EDI 00000000

EIP 004013E2 1_3.004013E2
```

Quy luật của user name:

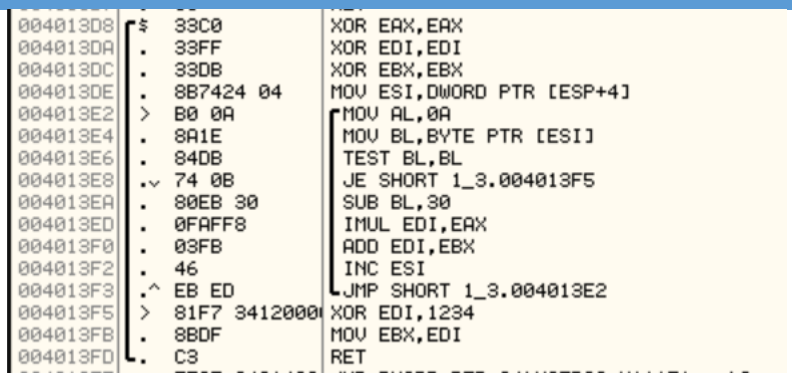
```
00401383  >  8A06          MOV AL,BYTE PTR [ESI]
00401385  .  84C0          TEST AL,AL
00401387  .v 74 13         JE SHORT 1_3.0040139C
00401389  .  3C 41         CMP AL,41
0040138B  .v 72 1F         JB SHORT 1_3.004013AC
0040138D  .  3C 5A         CMP AL,5A
0040138F  .v 73 03         JNB SHORT 1_3.00401394
00401391  .  46            INC ESI
00401392  .^ EB EF         JMP SHORT 1_3.00401383
00401394  >  E8 39000000   CALL 1_3.004013D2
00401399  .  46            INC ESI
0040139A  .^ EB E7         JMP SHORT 1_3.00401383
0040139C  >  5E            POP ESI
0040139D  .  E8 20000000   CALL 1_3.004013C2
004013A2  .  81F7 7856000( XOR EDI,5678
004013A8  .  8BC7          MOV EAX,EDI
004013AA  .v EB 15         JMP SHORT 1_3.004013C1
004013AC  >  5E            POP ESI
004013AD  .  6A 30         PUSH 30
004013AF  .  68 60214000   PUSH 1_3.00402160
004013B4  .  68 69214000   PUSH 1_3.00402169
004013B9  .  FF75 08       PUSH DWORD PTR [EBP+8]
004013BC  .  E8 79000000   CALL <JMP.&USER32.MessageBoxA>
004013C1  L> C3            RET
004013C2  r$ 33FF          XOR EDI,EDI
004013C4  .  33DB          XOR EBX,EBX
004013C6  >  8A1E          MOV BL,BYTE PTR [ESI]
004013C8  .  84DB          TEST BL,BL
004013CA  .v 74 05         JE SHORT 1_3.004013D1
004013CC  .  03FB          ADD EDI,EBX
004013CE  .  46            INC ESI
004013CF  .^ EB F5         JMP SHORT 1_3.004013C6
004013D1  L> C3            RET
004013D2  r$ 2C 20         SUB AL,20
004013D4  .  8806          MOV BYTE PTR [ESI],AL
004013D6  L. C3            RET
004013D7     C3            RET
```

Ta xét quy luật của user name thông qua đoạn mã giả sau:

```
string s : username
n : s.legnth()
res = 0x00
for (i = 0; i < n; ++i){
    if (s[i] < 0x41)        // 0x41 => 65
        => return false
    if (s[i] >= 0x5A)       // 0x5A => 90
        s[i] = s[i] - 0x20  // 0x20 => 32
    res = res + s[i]
}

res = res ^ 0x5678      // 0x5678 => 22136
```

Quy luật của password:

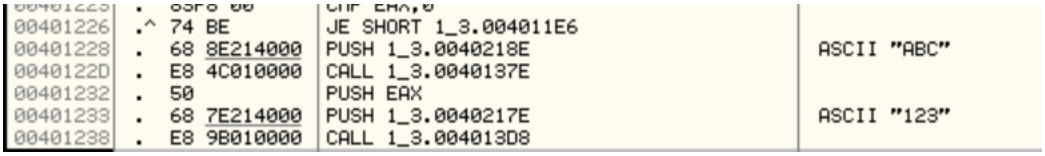


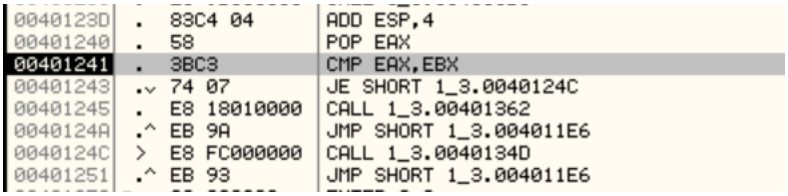
Ta xét quy luật của password thông qua đoạn mã giả sau:

```
string s //password
n = s.legnth()
res = 0x00
for (i = 0; i < n; ++i){
    s[i] = s[i] - 0x30  //0x30 => 48
    res = res + s[i]
}

res = res ^ 0x1234  // 0x1234 => 4660
```

Một cặp user name và password đúng sẽ thoả mãn kết quả của 2 phép biến đổi trên bằng nhau.





Bài này có keygen. Sau đây là 1 ví dụ:

User name: abcde

Password: 17667