آموزش های R4NDOM -فارسی

قسمت دهم-مراحل پچ کردن

در این آموزش ما سعی میکنیم مراحل Patch کردن در یک فایل باینری را توضیح دهیم ،که کمی طولانی هم خواهد بود ،فایل مورد استفاده همان Crackme6 در قسمت قل خواهد بود ،به طور کلی این Crackme سخت نیست ،بلکه هدف ما آنالیز دقیق تر این مثال خواهد بود

مراحل کرکینگ

اگر بخواهیم این مراحل را دسته بندی کنیم ، چهار قسمت خواهد بود:

مرحله اول –**LAME**

کوتاه شده ی (Localized Assembly Manipulation and Enhancing) است ،اگر بخواهیم این روش را تعریف کنیم به این شکل خواهد بود که در آموزش های قبلی ما بسیار از این روش استفاده کرده ایم این همان روش است که ما سعی میکنیم روتین مقاسه را پیدا کرده و شرط صحت مقایسه را با دستوراتی مانند Nopیا JMP و... تعویض کنیم ، البته منکر آن نمیشوم که این روش کار نکرده ☺ اما باید حقیقتی را بگویم ،این روش خیلی هم کار آمد نیست ،یعنی همیشه نمیتوان برآن تکیه کرد ،بسیاری از برنامه ها هستند که به همین راحتی کرک نمیشوند ،از آن گذشته این روش مشکلاتی هم دارد ،از جمله:

۱-خیلی از برنامه بیشتر از یک بار شرط ریجستر شدن را در برنامه چک میکنند ،که این مقایسه در قسمت های مختلف برنامه گنجانده شده است ،من به شخصه برنامه ایی را دیده ام که حدود ۱۹ بار مقایسه ایی مبنی بر ریجستر شدن برنامه را انجام داده

۲- بسیار از برنامه از حقه های مخصوصی استفاده میکنند ،به این صورت که مقایسه ها و پرش ها را از درون یک Dll فراخوانی میکنند

۳-گاهی اوقات ممکن است شما قسمت های زیادی از برنامه را پچ کنید ،قبول کنید این کار قشنگ و حرفه ایی نیست

مرحله ی دوم –**NOOB**

کوتاه شده ی Not Only Obvious Breakpoints است ،در این روش ما مقداری عمیق تر به کد نگاه میکنیم و در حقیقت از متد LAM شانس بهتری برای کشف روتین مقایسه کدها داریم ، در این روش شانس ما بسار زیاد است چرا که ما با رفتن به درون فراخوانی ها میتوانیم قبل از اینکه روتین مورد نظر اجرا شود ، علت آن را بیابیم، و میتوان کل روتین را از کار انداخت چرا که ممکن است این چک در چند جای دیگر هم باشد ،اما با این تفاسیر این متد هم دارای مشکلات خاص خودش است،از جمله:

۱-گاهی اوقات ممکن است که یک روتین چیزی بیشتر از یک مقایسه داشته داشته باشد ،فرض کنید یک تابع عمومی داریم که صحت رشته ی وارد شده را با برگرداندن مقدار TRUE/FALSE مشخص کند اما مثلا اگر چند روتین دیگر از این مقادیر استفاده کنند چه؟

۲- این روش مستلزم مهارت و تجربه بیشتری است ،که البته در این آموزش از این روش استفاده شده است

مرحله ی سوم –SKILLED

کوتاه شده Some Knowledge In Lower Level Engineered Data است . این روش مانند روش NOOB است با این تفاوت که از آن روش درک ما از کد بیشتر است است ،که این روش مزایای زیادی دارد ازجمله آنالیز کامل جهت درک تمامی ترفند های به کار رفته در روتین مانند (ذخیره سازی متغییر ها در حافظه برای بازیابی های بعدی) ،این روش بسیار قدرتمند است و به مرور زمان شما را قهار خواهد کرد ،فهم کامل کد های اسمبلی ،کار با پشته،ثبات ها،فهم ساختار سیستم عامل ،مباحثی هستند که به شما کمک میکنند

مرحله ی چهارم – SK1LL$

کوتاه شده ی Serial Keygenning In Low-level Languages, Stupid است،بدین معنی است که شما نه تنها برنامه را کرک کرده اید بلکه توانسته اید الگوریتم ایجاد کد را نیز بیابید و آن را باز تولید کنید که یک کاربر دیگر بتواند با استفاده از Keygen تولید شده یک کد مناسب به برنامه بدهد

بازنگری برنامه از دید مرحله ۲ (NOOB)

بیایید برنامه ی ضمیمه شده را درون olly باز کنید و یک BP بروی تابع GetDlgItemTextA بگذارید و برنامه را اجرا کنید و یک کد دلخواه وارد کنید .من "1212121212 را وارد میکنم :

```
00401258  >  6A 0C          PUSH 0C                                  Count = C (12.)
0040125A  .  68 5D304000    PUSH Crackme6.0040305D                   Buffer = Crackme6.0040305D
0040125F  .  6A 6B          PUSH 6B                                  ControlID = 6B (107.)
00401261  .  FF75 08        PUSH DWORD PTR [EBP+8]                   hWnd
00401264  .  E8 EF020000    CALL <JMP.&user32.GetDlgItemTextA>       GetDlgItemTextA
00401269  .  83F8 0B        CMP EAX,0B
0040126C  .  72 10          JB SHORT Crackme6.0040127E
0040126E  .  68 00304000    PUSH Crackme6.00403000                   Text = "ACCESS DENIED!"
00401273  .  FF35 80304000  PUSH DWORD PTR [403080]                  hWnd = 00010574 (class='Edit',parent=0001
00401279  .  E8 FE020000    CALL <JMP.&user32.SetWindowTextA>        SetWindowTextA
0040127E  >  85C0           TEST EAX,EAX
00401280  .  75 10          JNZ SHORT Crackme6.00401292
00401282  .  68 00304000    PUSH Crackme6.00403000                   Text = "ACCESS DENIED!"
00401287  .  FF35 80304000  PUSH DWORD PTR [403080]                  hWnd = 00010574 (class='Edit',parent=0001
0040128D  .  E8 EA020000    CALL <JMP.&user32.SetWindowTextA>        SetWindowTextA
00401292  >  50             PUSH EAX
00401293  .  68 5D304000    PUSH Crackme6.0040305D
```

حال اطلاعات مربوط به نحوه ی عملکرد این تابع را میبینیم :

Win32 Programmer's Reference

File Edit Bookmark Options Help

Contents Index Back << >>

GetDlgItemText Quick Info Overview Group

The **GetDlgItemText** function retrieves the title or text associated with a control in a dialog box.

```
UINT GetDlgItemText(
  HWND hDlg,          // handle of dialog box
  int nIDDlgItem,     // identifier of control
  LPTSTR lpString     // address of buffer for text
,
  int nMaxCount       // maximum size of string
);
```

Parameters

hDlg
Identifies the dialog box that contains the control.

nIDDlgItem
Specifies the identifier of the control whose title or text is to be retrieved.

lpString
Points to the buffer to receive the title or text.

nMaxCount
Specifies the maximum length, in characters, of the string to be copied to the buffer pointed to by *lpString*. If the length of the string exceeds the limit, the string is truncated.

Return Values

If the function succeeds, the return value specifies the number of characters copied to the buffer, not including the terminating null character.

If the function fails, the return value is zero.

Remarks

The **GetDlgItemText** function sends a WM_GETTEXT message to the control.

See Also

GetDlgItemInt, SetDlgItemInt, SetDlgItemText, WM_GETTEXT

همانطور که میبینید آرگومانهای این تابع مشخص شده اما مهمترین عضو آن برای ما (lpString) است که درواقع نمایشگر محل ذخیره ی پسورد است ، مقدار بازگشتی که شامل طول رشته است در ثبات EAX قرار میگیرد:

Return Values

If the function succeeds, the return value specifies the number of characters copied to the buffer, not including the terminating null character.

If the function fails, the return value is zero.

اگر به آدرس 40125A نگاه کنید محل اشاره به بافر رشته را میبینید یعنی آدرس :40305D (Olly) در قسمت توضیحات آن را Buffer معرفی کرده ،البته حدس زده) :

```
00401258  >  6A 0C        PUSH 0C                                  Count = C (12.)
0040125A  .  68 5D304000  PUSH Crackme6.0040305D                   Buffer = Crackme6.0040305D
0040125F  .  6A 6B        PUSH 6B                                  ControlID = 6B (107.)
00401261  .  FF75 08      PUSH DWORD PTR [EBP+8]                   hWnd
00401264  .  E8 EF020000  CALL <JMP.&user32.GetDlgItemTextA>       GetDlgItemTextA
00401269  .  83F8 0B      CMP EAX,0B
```

این یعنی متن درون دیالوگ باکس (پسورد وارد شده) درون این buffer در آدرس 40305D قرار میگیردو همینطور طول رشته ایی که وارد کردیم هم درون ثبات EAX قرار خواهد گرفت،یعنی طول رمز عبوری که وارد کردیم "1212121212"درون ثبات EAX قرار خواهد گرفت،حال اگر به دو خط بعدی نگاه کنید متوجه خواهید شد که مقدار این خط 0B (در مد هگزا)است که (11 در مد دسیمال است) ، که طول رشته ایی که ما وارد کرده ایم (A درمد هگزا و 10در مد دسیمال است) که به معنای این است که طول رشته ما کمتر است:

طول رشته ی وارد شده ی ما در ثبات EAX قرار گرفت :

```
EAX 0000000A
ECX 76226F42
```

همانطور که در تصویر زیر میبینید طول رمز واقعی ۱۱ کارکتر است ،اما طول رمز وارد شده ی ما ۱۰ کاراکتر است ،یعنی در خط 401269 مقایسه ایی صورت گرفته :CMP EAX,0B که (۱۱)0B طول رمز و مقدار EAX (طول رمز ما درون آن است) برابر با (۱۰) A است. و نتیجه ی این مقایسه در خط 40126C منجر به پرش میشود(بپر اگر مقدار EAX از 11 کمتر است):

```
0040125F  .  6A 6B        PUSH 6B                                  ControlID = 6B (107.)
00401261  .  FF75 08      PUSH DWORD PTR [EBP+8]                   hWnd
00401264  .  E8 EF020000  CALL <JMP.&user32.GetDlgItemTextA>       GetDlgItemTextA
00401269  .  83F8 0B      CMP EAX,0B        طول رشته 11
0040126C  .  72 10        JB SHORT Crackme6.0040127E
0040126E  .  68 00304000  PUSH Crackme6.00403000                   Text = "ACCESS DENIED!"
00401273  .  FF35 80304000 PUSH DWORD PTR [403080]                 hWnd = 00010574 (class='Edit',parent=0001056C
00401279  .  E8 FE020000  CALL <JMP.&user32.SetWindowTextA>        SetWindowTextA
0040127E  >  85C0         TEST EAX,EAX
00401280  .  75 10        JNZ SHORT Crackme6.00401292
```

بله ،پرش به سمت Bad Boy صورت میگیرد اما ما چیزی را فهمیدیم و آن این است که طول رمز ورودی باید 11 کاراکتر باشد،به خط 401280 نگاه کنید :

```
00401261  .  FF75 08        PUSH DWORD PTR [EBP+8]                  hWnd
00401264  .  E8 EF020000    CALL <JMP.&user32.GetDlgItemTextA>      GetDlgItemTextA
00401269  .  83F8 0B        CMP EAX,0B
0040126C  v  72 10          JB SHORT Crackme6.0040127E
0040126E  .  68 00304000    PUSH Crackme6.00403000                  Text = "ACCESS DENIED!"
00401273  .  FF35 80304000  PUSH DWORD PTR [403080]                 hWnd = 00010574 (class='Edit',parent=0001056C)
00401279  .  E8 FE020000    CALL <JMP.&user32.SetWindowTextA>       SetWindowTextA
0040127E  >  85C0           TEST EAX,EAX
00401280  .  75 10          JNZ SHORT Crackme6.00401292
00401282  .  68 00304000    PUSH Crackme6.00403000                  Text = "ACCESS DENIED!"
00401287  .  FF35 80304000  PUSH DWORD PTR [403080]                 hWnd = 00010574 (class='Edit',parent=0001056C)
0040128D  .  E8 EA020000    CALL <JMP.&user32.SetWindowTextA>       SetWindowTextA
00401292  >  50             PUSH EAX
00401293  .  68 5D304000    PUSH Crackme6.0040305D                  ASCII "1212121212"
00401298  .  E8 84010000    CALL Crackme6.00401421
```

در خط 40127E مقایسه ایی صورت میگیرد مقدار ثبات EAX = ۱۰ است (طول رمز ورودی) و این دستور آن را با صفر مقایسه میکند یعنی اگر eax برابر 0 است آنگاه پرش نکن در غیر اینصورت بپر.به کجا ،معلوم است به bad Boy 2 ☺

خیلی خب ما تا اینجا دو نکته یاد گرفتیم یکی اینکه طول رمز ۱۱ رقمی یا بیشتر باشد و دوم اینکه کادر پسورد نباید خالی باشد

بعد از اینکه پرش از خط 401280 صورت گرفت ، حال به خط 401292 دقت کنید PUSH EAX (طول رمز را درون پشته میگذارد) و در خط بعدی (بافر حاوی رمز ذخیره شده) درون پشته قرار میگیرد:

```
0012FAD8  0040305D  Arg1 = 0040305D ASCII "1212121212"
0012FADC  0000000A  Arg2 = 0000000A
0012FAE0  0012FB0C
0012FAE4  7623C4E7  RETURN to user32.7623C4E7
0012FAE8  004505FA
0012FAEC  00000111
```

پسورد

طول پسورد

همانطور که میبینید طول رمز ورودی در آدرس 12ADC و همینطور مقدار آدرس 12FAD8 از آدرس 4030D5 ، Push شده است بسیار خوب ما میدانیم که رمز ما در آدرس حافظه ی 4030D5 ذخیره میشود (بسیار مهم است) ،بعد از اینکه این دو آرگومان در تابع قرار گرفت و در پشته Pushشدند ، نوبت فراخوانی روتین اصلی است ،در خط 40129D میبینیم که این دستور بروی پرچم Z اثر گذاشته و آن را با مقدار 0 ارزش گذاری کرده، بدین معنی که اگر مقدار EAX برابر با صفر است ،مقدار پرچم z=1 خواهد شد و درنتیجه پرش انجام نخواهد شد و ما مستقیم به سمت Good Boy خواهیم رفت ، تمام این اما و اگر ها درون روتینی در آدرس 401298است، درواقع مقدار بازگشتی درون ثبات EAX قرار گرفته ،حال اجازه دهید درون این روتین شویم یعنی آدرس (401298) با فشردن کلید F7 درون این روتین میشویم :

```
00401421  $  55             PUSH EBP
00401422  .  8BEC           MOV EBP,ESP
00401424  .  51             PUSH ECX
00401425  .  52             PUSH EDX
00401426  .  33C9           XOR ECX,ECX
00401428  .  33D2           XOR EDX,EDX
0040142A  .  8B45 08        MOV EAX,DWORD PTR [EBP+8]
0040142D  >  813401 674523  XOR DWORD PTR [ECX+EAX],1234567
00401434  .  802401 0E      AND BYTE PTR [ECX+EAX],0E
00401438  .  83C1 04        ADD ECX,4
0040143B  .  83F9 08        CMP ECX,8
0040143E  .^ 75 ED          JNZ SHORT Crackme6.0040142D
00401440  .  33C9           XOR ECX,ECX
00401442  >  8A1401         MOV DL,BYTE PTR [ECX+EAX]
00401445  .  0050 08        ADD BYTE PTR [EAX+8],DL
00401448  .  41             INC ECX
00401449  .  3B4D 0C        CMP ECX,DWORD PTR [EBP+C]
0040144C  .^ 75 F4          JNZ SHORT Crackme6.00401442
0040144E  .  33C9           XOR ECX,ECX
00401450  >  813401 DEBC9A  XOR DWORD PTR [ECX+EAX],89ABCDE
00401457  .  802401 0E      AND BYTE PTR [ECX+EAX],0E
0040145B  .  83C1 04        ADD ECX,4
0040145E  .  83F9 08        CMP ECX,8
00401461  .^ 75 ED          JNZ SHORT Crackme6.00401450
00401463  .  33C9           XOR ECX,ECX
00401465  >  8A1401         MOV DL,BYTE PTR [ECX+EAX]
00401468  .  0050 09        ADD BYTE PTR [EAX+9],DL
0040146B  .  41             INC ECX
0040146C  .  3B4D 0C        CMP ECX,DWORD PTR [EBP+C]
0040146F  .^ 75 F4          JNZ SHORT Crackme6.00401465
00401471  .  8A50 09        MOV DL,BYTE PTR [EAX+9]
00401474  .  8A70 08        MOV DH,BYTE PTR [EAX+8]
00401477  .  66:81FA DE42   CMP DX,42DE
0040147C  .v 0F85 8E000000  JNZ Crackme6.00401510
00401482  .  B9 09000000    MOV ECX,9
00401487  >  8A1401         MOV DL,BYTE PTR [ECX+EAX]
```

نترسید ☺ بیایید به انتهای روتین برویم (RETN) جایی که مقدار ثبات EAX برابر با ۱ میشود :

```
004014E6  .v 75 1F          JNZ SHORT Crackme6.00401507
004014E8  .  80F9 8D        CMP CL,8D
004014EB  .v 75 23          JNZ SHORT Crackme6.00401510
004014ED  .  8078 05 BF     CMP BYTE PTR DS:[EAX+5],0BF
004014F1  .v 75 14          JNZ SHORT Crackme6.00401507
004014F3  .  25 FFFF0000    AND EAX,0FFFF
004014F8  .  66:33C0        XOR AX,AX
004014FB  .v EB 18          JMP SHORT Crackme6.00401515
004014FD  >  8AD1           MOV DL,CL
004014FF  .  32CA           XOR CL,DL
00401501  .  B0 01          MOV AL,1
00401503  .  04 20          ADD AL,20
00401505  .  8AC8           MOV CL,AL
00401507  >  66:B9 9138     MOV CX,3891
0040150B  .  66:81F1 AD0F   XOR CX,0FAD
00401510  >  B8 01000000    MOV EAX,1
00401515  >  5A             POP EDX
00401516  .  59             POP ECX
00401517  .  C9             LEAVE
00401518  .  C2 0800        RETN 8
```

If we jump here...

EAX will equal 1

بله اینجا eax برابر با ۱ میشود که در نتیجه ما را به قسمت BAD BOY میرد ،همانطور که در کنار این دستور میبینید فلشی به این قسمت ارجاع داده شده که در آدرس 4014FB است:

```
004014E8  .  80F9 8D        CMP CL,8D
004014EB  .v 75 23          JNZ SHORT Crackme6.00401510
004014ED  .  8078 05 BF     CMP BYTE PTR [EAX+5],0BF
004014F1  .v 75 14          JNZ SHORT Crackme6.00401507
004014F3  .  25 FFFF0000    AND EAX,0FFFF
004014F8  .  66:33C0        XOR AX,AX
004014FB  .v EB 18          JMP SHORT Crackme6.00401515
004014FD  >  8AD1           MOV DL,CL
004014FF  .  32CA           XOR CL,DL
00401501  .  B0 01          MOV AL,1
00401503  .  04 20          ADD AL,20
00401505  .  8AC8           MOV CL,AL
00401507  >  66:B9 9138     MOV CX,3891
0040150B  .  66:81F1 AD0F   XOR CX,0FAD
00401510  >  B8 01000000    MOV EAX,1
00401515  >  5A             POP EDX
00401516  .  59             POP ECX
00401517  .  C9             LEAVE
00401518  .  C2 0800        RET 8
0040151B  $  55             PUSH EBP
0040151C  .  8BEC           MOV EBP,ESP
```

برابر ۱ صفر شده

برابر با ۱ شده

همانطور که مشاهده میکنید در خط 4014F8 مقدار ثبات ax برابر با صفر میشود، ودر خط بعد پرشی به آدرس 401515انجام میدهد که نتیجه این میشود که از قسمت Bad Boy عبور میکنیم چون همواره مقدار صفر را درون ثبات EAX جای میدهد،حال به آدرس 401510 نگاه کنید میبیند که پرشی در این آدرس صورت میگیرد :

```
0040147C  .v 0F85 8E000000   JNZ Crackme6.0040151
00401482  .  B9 09000000      MOV ECX,9
00401487  >  8A1401           MOV DL,BYTE PTR DS:
0040148A  .  321401           XOR DL,BYTE PTR DS:
0040148D  .  49               DEC ECX
0040148E  .  67:E3 02         JCXZ SHORT Crackme6
00401491  .^ EB F4            JMP SHORT Crackme6.
00401493  >  66:8B48 08       MOV CX,WORD PTR DS:[
00401497  .  66:81F1 EEEE     XOR CX,0EEEE
0040149C  .  66:81F9 AC30     CMP CX,30AC
004014A1  .v 75 64            JNZ SHORT Crackme6.
004014A3  .  8A08             MOV CL,BYTE PTR DS:[
004014A5  .  8A68 01          MOV CH,BYTE PTR DS:[
004014A8  .  66:81C1 9235     ADD CX,3592
004014AD  .  66:81F9 9AE5     CMP CX,0E59A
004014B2  .v 75 49            JNZ SHORT Crackme6.
004014B4  .  8138 08B0817A    CMP DWORD PTR DS:[EA
004014BA  .v 75 4B            JNZ SHORT Crackme6.
004014BC  .  66:33C9          XOR CX,CX
004014BF  >  80F2 0A          XOR DL,0A
004014C2  .  83C1 04          ADD ECX,4
004014C5  .  83F9 0C          CMP ECX,0C
004014C8  .^ 7E F5            JLE SHORT Crackme6.
004014CA  .  8178 04 02BF8D38 CMP DWORD PTR DS:[EA
004014D1  .v 75 3D            JNZ SHORT Crackme6.
004014D3  .  8ACA             MOV CL,DL
004014D5  .  32D1             XOR DL,CL
004014D7  .  8AD1             MOV DL,CL
004014D9  .  32CA             XOR CL,DL
004014DB  .  8A48 05          MOV CL,BYTE PTR DS:[
004014DE  .  8A50 06          MOV DL,BYTE PTR DS:[
004014E1  .  86D1             XCHG CL,DL
004014E3  .  80FA BF          CMP DL,0BF
004014E6  .v 75 1F            JNZ SHORT Crackme6.
004014E8  .  80F9 8D          CMP CL,8D
004014EB  .v 75 23            JNZ SHORT Crackme6.
004014ED  .  8078 05 BF       CMP BYTE PTR DS:[EAX
004014F1  .v 75 14            JNZ SHORT Crackme6.
004014F3  .  25 FFFF0000      AND EAX,0FFFF
004014F8  .  66:33C0          XOR AX,AX
004014FB  .v EB 18            JMP SHORT Crackme6.
004014FD  >  8AD1             MOV DL,CL
004014FF  .  32CA             XOR CL,DL
00401501  .  B0 01            MOV AL,1
00401503  .  04 20            ADD AL,20
00401505  .  8AC8             MOV CL,AL
00401507  >  66:B9 9138       MOV CX,3891
0040150B  .  66:81F1 AD0F     XOR CX,0FAD
00401510  >  B8 01000000      MOV EAX,1
00401515  >  5A               POP EDX
00401516     59               POP ECX
```

این آدرس 40147C ما را مستقیم به سمت Bad Boy میبرد،بسیار خوب ما الان درباره ی این روتین چیزهایی میدانیم و برای پچ کردن هم راهای زیادی است،به نظر شما برای پچ بهترین گزینه کدام است که این روتین همیشه مقدار صفر را برگداند ؟

۱- تغییر دستور MOV EAX,1 به MOV EAX,0 ؟

۲- تغییر دستور JNZ در آدرس 40147C به NOP و تغییر آدرس 4014D1 به NOP ؟

چه را هایی دیگری وجود دارد؟بله را های دیگری هم قطعا هستند که کنکاش را به عهده ی شما میگذارم ،این بررسی از منظر "مرحله ی دوم –NOOB" بود

ادامه ی کند و کاو از دیدگاه مرحله ی ۳-SKILLED

شاید از خود بپرسید که چه لزومی دارد که ادامه بدهیم !! ما که موفق شدیم !! البته من درک میکنم که شما مبتدی هستید،اما فرض کنید که یک برنامه داریم که در آن واحد دارای نسخه های "Private"، "Corporate"، "Enterprise" است که به طور معمول نیاز باشد درون روتین ها شویم ☺ ودلیل دوم اینکه درک شما را از کل برنامه بالا میبرد،حال اخم نکنید و بیایید ادامه دهیم ☺ :

```
00401421  $  55               PUSH EBP
00401422  .  8BEC             MOV EBP,ESP
00401424  .  51               PUSH ECX
00401426  .  52               PUSH EDX
00401427  .  33C9             XOR ECX,ECX
00401428  .  33D2             XOR EDX,EDX
0040142A  .  8B45 08          MOV EAX,[ARG.1]
0040142D  >  813401 67452301  XOR DWORD PTR DS:[ECX+EAX],1234567
00401434  .  802401 0E        AND BYTE PTR DS:[ECX+EAX],0E
00401438  .  83C1 04          ADD ECX,4
0040143B  .  83F9 08          CMP ECX,8
0040143E  .^ 75 ED            JNZ SHORT Crackme6.0040142D
00401440  .  33C9             XOR ECX,ECX
00401442  >  8A1401           MOV DL,BYTE PTR DS:[ECX+EAX]
```

Move password into EAX

EAX is 1st digit of pass

ECX=0

and pass will be XOR'd with 1234567

در ابتدای روتین ثبات ها در فضای پشته گذاشته شدند و فضایی برای متغییر های محلی ساخته شد،مقادیری که درون ثبات های بودند ،درون پشته قرار گرفتند,EDX ECX

به خط 40142D نگاه کنید، این خط آرگومانهای محلی را درون پشته قرار میدهد ،اگر به پنجره ی ریجستر ها نگاه کنید میبینید که محتوای ثبات EAX آدرس 40305D است ،که همان آدرس پسورد ماست (یادتان هست؟) و در خط بعدی داریم:

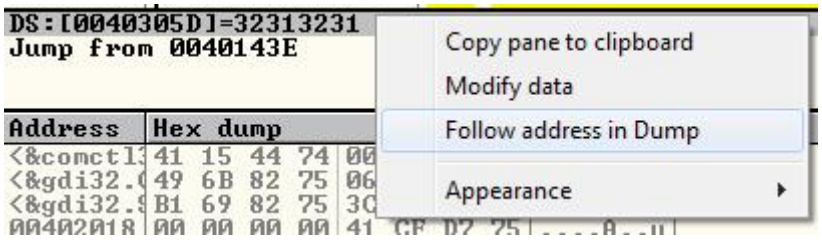
XOR DWORD PTR DS:[ECX+EAX], 1234567

این خط مقدار ثبات ECX (الان صفر است) را به ابتدای آدرس EAX (همان پسورد خودمان که در آدرس 40305D ذخیره شده است) را اضافه میکند، و سپس مقدار DWORD (۴بایتی) را در آدرس 40305D (همان پسورد خودمان)را با مقدار 123456 در مد هگزا XOR میکند،از آنجا که مقدار ثبات ECXبرابر با صفر است ،پس چیزی به پسورد ما اضافه نمیکند،پس داستان به این شکل است که ۴ رقم از پسورد ما را با مقدار 123456 در مد هگزا را XORمیکند و نتیجه ی این عملیات را در همان جای پسورد مینویسد اگر بروی خط 40142D توقف کنیم میتوانیم این عملیات را ببینیم،به پنجره ی Info در بالای پنجره ی دامپ نگاه کنید:

```
004014D1  .v 75 3D
DS:[0040305D]=32313231
Jump from 0040143E
```

نتیجه ی عبارت EAX+ECX برابر با 40305D است ، و 31 32 31 32 همان رمز ورودی (کد اسکی 1212)هست (ایندین را یادتان هست؟ ☺)

حالا بروی همین خط در پنجره ی INFO راست کلیک کنید :"DS:[0040305D]=32313231"



داریم:

Address	Hex dump	ASCII
0040305D	31 32 31 32 31 32 31 32	12121212
00403065	31 32 21 00 56 60 7A 7D	12!.V`z}
0040306D	2F 66 61 7F 7A 7B 2F 7F	/fa.z{/.
00403075	63 6A 6E 7C 6A 21 00 00	cjn|j!..
0040307D	00 40 00 0E 05 02 00 03	.@.....
00403085	00 00 00 00 00 00 00 00	.....
0040308D	00 00 00 00 00 00 00 00	.....
00403095	00 00 00 00 00 00 00 00	.....
0040309D	00 00 00 00 00 00 00 00	.....
004030A5	00 00 00 00 00 00 00 00	.....
004030AD	00 00 00 00 00 00 00 00	.....
004030B5	00 00 00 00 00 00 00 00	.....
004030BD	00 00 00 00 00 00 00 00	.....
004030C5	00 00 00 00 00 00 00 00	.....

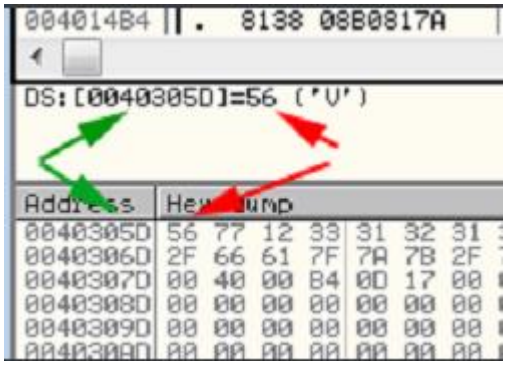
همانطور که میبینید اولین 10بایت پسورد وارد شده ی ما است که در اینجا ذخیره شده،حالا قرار است که چهار بایت اول از پسورد ما یعنی (31 32 31 32) با (123456) ، XOR شود و نتیجه در همان آدرس ذخیره شود ،بسیار خب با فشردن کلید F8 دستور فوق را اجرا کرده که نتیجه را در پنجره ی دامپ میبینیم :

Address	Hex dump	ASCII
0040305D	56 77 12 33 31 32 31 32	Vw.31212
00403065	31 32 21 00 56 60 7A 7D	12!.V`z}
0040306D	2F 66 61 7F 7A 7B 2F 7F	/fa.z{/.
00403075	63 6A 6E 7C 6A 21 00 00	cjn|j!..
0040307D	00 40 00 0E 05 02 00 03	.@.....
00403085	00 00 00 00 00 00 00 00	.....
0040308D	00 00 00 00 00 00 00 00	

همانطور که مشاهده میکنید نتیجه ی عملیات در همان آدرس Overwrite شد ،حال خط بعدی را اجرا کنید:

AND BYTE PTR DS:[ECX+EAX] , 0E

ما میدانیم که نتیجه ی ECX +EAX برابر است با آدرس پسورد سابق مادر (40305D)،حال 1 بایت از آدرس (40305D) را با مقدار 0E (جمع) AND میکند ،ونتیجه را در همان محل درج میشود،یعنی اولین رقم از رمز سابق ما با 0E جمع میشود و نتیجه در همان محل ثبت میشود:



تصویر بالا به ما میگوید که مقدار اولین بایت در آدرس 40305Dبرابر با 56 با کد اسکی " V" است ،حال اگر یکبار کلید F8را بفشارید پنجره ی دامپ اینگونه خواهد بود :

Address	Hex dump	ASCII
0040305D	06 77 12 33 31 32 31 32	.w.31212
00403065	31 32 21 00 56 60 7A 7D	12!.V`z}
0040306D	2F 66 61 7F 7A 7B 2F 7F	/fa.z{/.
00403075	63 6A 6E 7C 6A 21 00 00	cjn|j!..
0040307D	00 40 00 0E 05 02 00 03	.@.....

همانطور که در تصویر بالا میبینید نتیجه ی عملیات and کردن 56 با 0E برابر با 06 شده است ،حال به خط بعدی توجه کنید که ثبات ECX افزایش داشته (آماده شدن برای واکشی چهار بایت بعدی پسورد ما)و در خط بعدی با مقدار 8 مقایسه میشود که این معنی است که این حلقه دو بار اجرا خواهد شد:

پس بعد از اینکه ۸ بایت ما در عملیات XOR و AND قرار گرفت سپس از حلقه بیرون می آییم،بعد از فشردن کلید F8 به دستور العمل JNZ میرسیم، دوباره به ابتدای حلقه ارجاع داده میشویم و عملیات بروی چهار بایت بعدی هم صورت میگیرد:

```
Address   Hex dump                          ASCII
0040305D  06 77 12 33 06 77 12 33           .w.3.w.3
00403065  31 32 21 00 56 60 7A 7D           12!.V`z}
0040306D  2F 66 61 7F 7A 7B 2F 7F           /fa.z{/.
00403075  63 6A 6E 7C 6A 21 00 00           cjn|j!..
0040307D  00 40 00 0E 05 02 00 03           .@......
```

حال به مقدار ثبات ECX توجه کنید که دارای مقدار 8 است ،بدین معنی که 8 بایت درون حلقه شده است ،حالا در خط 40143B مقدار ثبات ECXبا 8 مقایسه میشود ، ومشخص است که پرش صورت نمیگیرد:

```
Jump is NOT taken
0040142D=Crackme6.0040142D
```

حالا ما در خط 401440 هستیم ،و ثبات ECX برابر با صفر شده است :

```
0040142D  >  813401 67452301  XOR DWORD PTR DS:[ECX+EAX],1234567
00401434  .  802401 0E        AND BYTE PTR DS:[ECX+EAX],0E
00401438  .  83C1 04          ADD ECX,4
0040143B  .  83F9 08          CMP ECX,8
0040143E  .^ 75 ED            JNZ SHORT Crackme6.0040142D
00401440  .  33C9             XOR ECX,ECX        ECX = 0
00401442  >  8A1401           MOV DL,BYTE PTR DS:[ECX+EAX]
00401445  .  0050 08          ADD BYTE PTR DS:[EAX+8],DL
00401448  .  41               INC ECX
00401449  .  3B4D 0C          CMP ECX,[ARG.2]
0040144C  .^ 75 F4            JNZ SHORT Crackme6.00401442
0040144E  .  33C9             XOR ECX,ECX
```

حال به دستورالعمل بعدی نگاه کنید :

```
Mov into DL                ECX = 0                EAX = beg. of password
0040143E  .^ 75 ED         JNZ SHORT Crackme6.0040142D
00401440  .  33C9          XOR ECX,ECX
00401442  >  8A1401        MOV DL,BYTE PTR DS:[ECX+EAX]
00401445  .  0050 08       ADD BYTE PTR DS:[EAX+8],DL
00401448  .  41            INC ECX
00401449  .  3B4D 0C       CMP ECX,[ARG.2]
0040144C  .^ 75 F4         JNZ SHORT C
0040144E  .  33C9          XOR ECX,ECX
So DL will be 1st digit of password
```

در این خط ابتدا مقدار DL با اولین بایت از پسورد سابق ما قرار میگیرد یعنی (6) ،در همین حین مقدار ECX که دوباره صفر است ومقدار ثبات EAXهم برابر با آدرس پسورد سابق ما است یعنی 40305D :

```
Registers (FPU)
EAX 0040305D Crackme6.0040305D
ECX 00000000
EDX 00000006
EBX 00000000
ESP 0012FAC8
EBP 0012FAD0
ESI 00000111
EDI 0012FB5C
```

و در خط بعدی مقدار DL که (6)است را با مقدار (EAX+8)جمع میکند و نتیجه را در همان جا اضافه میکند:

```
0040305D  06 77 12 33  06 77 12 33  .w.3.w.3
00403065  31 32 00 00  56 60 7A 7D  12..V`z}
0040306D  2F 66 61 7F  7A 7B 2F 7F  /fa.z{/.
00403075  63 6A 6E 7C  6A 21 00 00  cjn|j!..
0040307D  00 40 00 10  05 02 00 02  .@......
```

و پس از اجرای آن :

```
Address   Hex dump                  ASCII
0040305D  06 77 12 33  06 77 12 33  .w.3.w.3
00403065  37 32 00 00  56 60 7A 7D  72..V`z}
0040306D  2F 66 61 7F  7A 7B 2F 7F  /fa.z{/.
00403075  63 6A 6E 7C  6A 21 00 00  cjn|j!..
```

در خط بعد داریم :

```
00401442  >  8A1401        MOV DL,BYTE PTR [ECX+EAX]
00401445  .  0050 08       ADD BYTE PTR [EAX+8],DL
00401448  .  41            INC ECX
00401449  .  3B4D 0C       CMP ECX,DWORD PTR [EBP+C]
0040144C  .^ 75 F4         JNZ SHORT Crackme6.00401442
0040144E  .  33C9          XOR ECX,ECX
```

ecx=ecx+1

آرگومان دوم تابع،که حاوی طول پسورد است

همانطور که در تصویر بالا مشخص است ابتدا یک واحد به ثبات ECX اضافه میشود و در خط بعد طول پسورد با مقدار همین ثبات مقایسه میشود. طول پسورد وارد شده ی ما برابر با 10 بایت است:

```
00401442  >  8A1401        MOV DL,BYTE PTR [ECX+EAX]
00401445  .  0050 08       ADD BYTE PTR [EAX+8],DL
00401448  .  41            INC ECX
00401449  .  3B4D 0C       CMP ECX,DWORD PTR [EBP+C]
0040144C  .^ 75 F4         JNZ SHORT Crackme6.00401442
0040144E  .  33C9          XOR ECX,ECX
00401450  >  813401 DEBC9A XOR DWORD PTR [ECX+EAX],89ABCDE
00401457  .  802401 0E     AND BYTE PTR [ECX+EAX],0E
0040145B  .  83C1 04       ADD ECX,4
0040145E  .  83F9 08       CMP ECX,8
00401461  .^ 75 ED         JNZ SHORT Crackme6.00401450
00401463  .  33C9          XOR ECX,ECX
00401465  >  8A1401        MOV DL,BYTE PTR [ECX+EAX]
00401468  .  0050 09       ADD BYTE PTR [EAX+9],DL
0040146B  .  41            INC ECX
0040146C  .  3B4D 0C       CMP ECX,DWORD PTR [EBP+C]
0040146F  .^ 75 F4         JNZ SHORT Crackme6.00401465
00401471  .  8A50 09       MOV DL,BYTE PTR [EAX+9]
00401474  .  8A70 08       MOV DH,BYTE PTR [EAX+8]
00401477  .  66:81FA DE42  CMP DX,42DE
0040147C  .v 0F85 8E000000 JNZ Crackme6.00401510
00401482  .  B9 09000000   MOV ECX,9
00401487  >  8A1401        MOV DL,BYTE PTR [ECX+EAX]
0040148A  .  321401        XOR DL,BYTE PTR [ECX+EAX]
0040148D  .  49            DEC ECX
0040148E  .  67:E3 02      JCXZ SHORT Crackme6.00401493
00401491  .^ EB F4         JMP SHORT Crackme6.00401487
00401493  >  66:8B48 08    MOV CX,WORD PTR [EAX+8]
00401497  .  66:81F1 EEEE  XOR CX,0EEEE
0040149C  .  66:81F9 AC30  CMP CX,30AC
004014A1  .v 75 64         JNZ SHORT Crackme6.00401507
004014A3  .  8A08          MOV CL,BYTE PTR [EAX]
004014A5  .  8A68 01       MOV CH,BYTE PTR [EAX+1]
004014A8  .  66:81C1 9235  ADD CX,3592
004014AD  .  66:81F9 9AE5  CMP CX,0E59A
004014B2  .v 75 49         JNZ SHORT Crackme6.004014FD
004014B4  .  8138 08B0817A CMP DWORD PTR [EAX],7A81B008
004014BA  .v 75 4B         JNZ SHORT Crackme6.00401507
004014BC  .  66:33C9       XOR CX,CX
Stack SS:[0012FADC]=0000000A
ECX=00000001

Address   Hex dump                  ASCII
0012FADC  0A 00 00 00  0C FB 12 00  ........
0012FAE4  E7 C4 A2 76  84 04 02 00  ...v....
0012FAEC  11 01 00 00  69 00 00 00  ....i...
```

سپس این حلقه مجدد تکرار میشود ،و دوباره به خط 401442 ارجاع داده میشود ،ومقدار دو ثبات ECXو اولین بایت (پسورد) EAX جمع شده و درون DL قرار میگیرد یعنی "77"

نکته:ثبات ECXدر هر بار اجرا یک بایت اضافه میشود و با مقدار EAX جمع میشود که نتیجه این میشود :EAX+1

و در خط بعدی مقدار DL که (77)است را با مقدار (EAX+8)جمع میکند و نتیجه را در همان جا اضافه میکند.

سپس دوباره یک واحد به ECX اضافه و با طول رمز مقایسه میشود این روند تا رسیدن به طول پسورد ده رقمی ادامه پیدا میکند یعنی تا موقعی که ECX=10 باشد.

```
Address   Hex dump                          ASCII
0040305D  06 77 12 33 06 77 12 33           .w.3.w.3
00403065  9C 32 00 00 56 60 7A 7D           .2..V`z}
0040306D  2F 66 61 7F 7A 7B 2F 7F           /fa.z{/.
00403075  63 6A 6E 7C 6H 21 00 03           cjn|h!..
0040307D  00 40 00 CA 04 01 00 04           .@......
00403085  00 00 00 00 00 00 00 00           ........
0040308D  00 00 00 00 00 00 00 00           ........
00403095  00 00 00 00 00 00 00 00           ........
0040309D  00 00 00 00 00 00 00 00           ........
004030A5  00 00 00 00 00 00 00 00           ........
```

نتیجه ی عملیات بروی رمز ۱۰ رقمی

حالا ما واقعا میفهمیم که چرا پسورد باید ۱۱ رقمی باشد،بعد از حلقه ی دوم ،در خط بعد دوباره مقدار ECXبرابر با صفر میشود،این حلقه هم مشابه قبلی است با این تفاوت و مقدار xor شده در این حلقه 89ABCDE است :

```
00401450  >  813401 DEBC9(  XOR DWORD PTR [ECX+EAX],89ABCDE
00401457  .  802401 0E     AND BYTE PTR [ECX+EAX],0E
0040145B  .  83C1 04       ADD ECX,4
0040145E  .  83F9 08       CMP ECX,8
00401461  .^ 75 ED         JNZ SHORT Crackme6.00401450
```

```
.   33C9          XOR ECX,ECX
>   813401 DEBC9( XOR DWORD PTR [ECX+EAX],89ABCDE
.   802401 0E     AND BYTE PTR [ECX+EAX],0E
.   83C1 04       ADD ECX,4
.   83F9 08       CMP ECX,8
.^  75 ED         JNZ SHORT Crackme6.00401450
.   33C9          XOR ECX,ECX
>   8A1401        MOV DL,BYTE PTR [ECX+EAX]
.   0050 09       ADD BYTE PTR [EAX+9],DL
.   41            INC ECX
.   3B4D 0C       CMP ECX,DWORD PTR [EBP+C]
.^  75 F4         JNZ SHORT Crackme6.00401465
.   8A50 09       MOV DL,BYTE PTR [EAX+9]
.   8A70 08       MOV DH,BYTE PTR [EAX+8]
.   66:81FA DE42  CMP DX,42DE
.v  0F85 8E00000( JNZ Crackme6.00401510
.   B9 09000000   MOV ECX,9
>   8A1401        MOV DL,BYTE PTR [ECX+EAX]
.   321401        XOR DL,BYTE PTR [ECX+EAX]
.   49            DEC ECX
.   67:E3 02      JCXZ SHORT Crackme6.00401493
.^  EB F4         JMP SHORT Crackme6.00401487
>   66:8B48 08    MOV CX,WORD PTR [EAX+8]
.   66:81F1 EEEE  XOR CX,0EEEE
.   66:81F9 AC30  CMP CX,30AC
.v  75 64         JNZ SHORT Crackme6.00401507
.   8A08          MOV CL,BYTE PTR [EAX]
.   8A68 01       MOV CH,BYTE PTR [EAX+1]
.   66:81C1 9235  ADD CX,3592
.   66:81F9 9AE5  CMP CX,0E59A
.v  75 49         JNZ SHORT Crackme6.004014FD
.   8138 08B0817( CMP DWORD PTR [EAX],7A81B008
.v  75 4B         JNZ SHORT Crackme6.00401507
.   66:33C9       XOR CX,CX
>   80F2 0A       XOR DL,0A
.   83C1 04       ADD ECX,4
.   83F9 0C       CMP ECX,0C
.^  7E F5         JLE SHORT Crackme6.004014BF
.   8178 04 02BF( CMP DWORD PTR [EAX+4],388DBF02
.   75 3D         JNZ SHORT Crackme6.00401510
.   8A0A          MOV CL,DL
.   32D1          XOR DL,CL
.   8AD1          MOV DL,CL
.   32CA          XOR CL,DL
.   8A48 05       MOV CL,BYTE PTR [EAX+5]
.   8A50 06       MOV DL,BYTE PTR [EAX+6]
.   86D1          XCHG CL,DL
.   80FA BF       CMP DL,0BF
.v  75 1F         JNZ SHORT Crackme6.00401507
.   80F9 8D       CMP CL,8D
.v  75 23         JNZ SHORT Crackme6.00401510
.   8078 05 BF    CMP BYTE PTR [EAX+5],0BF
.v  75 14         JNZ SHORT Crackme6.00401507
.   25 FFFF0000   AND EAX,0FFFF
.   66:33C0       XOR AX,AX
.v  EB 18         JMP SHORT Crackme6.00401515
>   8AD1          MOV DL,CL
.   32CA          XOR CL,DL
.   B0 01         MOV AL,1
.   04 20         ADD AL,20
.   8AC8          MOV CL,AL
>   66:B9 9138    MOV CX,3891
.   66:81F1 0D0F  XOR CX,0F0D
>   B8 01000000   MOV EAX,1
>   5A            POP EDX
.   59            POP ECX
```

این پرش رو هم نمیخواهیم

و پس از اتمام این حلقه داریم :

```
Jump is NOT taken
00401450=Crackme6.00401450

Address   Hex dump                          ASCII
0040305D  08 CB 88 3B 08 CB 88 3B  ...;...;
00403065  9C 32 00 00 56 60 7A 7D  .2..V`z}
0040306D  2F 66 61 7F 7A 7B 2F 7F  /fa.z{/.
00403075  63 6A 6E 7C 6A 21 00 00  cjn|j!..
0040307D  00 40 00 CA 04 01 00 05  .@......
```

Xor by 89ABCDE : نتیجه ی

و وارد حلقه ی سوم میشویم در ابتدا طبق معمول ECX=0 وداریم :

```
00401461  .^ 75 ED          JNZ SHORT Crackme6.00401450
00401463  .  33C9           XOR ECX,ECX
00401465  >  8A1401         MOV DL,BYTE PTR DS:[ECX+EAX]
00401468  .  0050 09        ADD BYTE PTR DS:[EAX+9],DL
0040146B  .  41             INC ECX
0040146C  .  3B4D 0C        CMP ECX,[ARG.2]
0040146F  .^ 75 F4          JNZ SHORT Crackme6.00401465
00401471  .  8A50 09        MOV DL,BYTE PTR DS:[EAX+9]
00401474  .  8A70 08        MOV DH,BYTE PTR DS:[EAX+8]
00401477  .  66:81FA DE42   CMP DX,42DE
0040147C  .v 0F85 8E000000  JNZ Crackme6.00401510
```

This time It's +9

همه چیز مانند قبل است با این تفاوت که این بار نتیجه درون EAX+9 ذخیره میشود :

```
0040305D  08 CB 88 3B 08 CB 88 3B  ...;...;
00403065  9C F4 00 00 56 60 7A 7D  ....V`z}
0040306D  2F 66 61 7F 7A 7B 2F 7F  /fa.z{/.
00403075  63 6A 6E 7C 6A 21 00 00  cjn|j!..
0040307D  00 40 00 CA 04 01 00 05  .@......
00403085  00 00 00 00 00 00 00 00  ........
```

```
Address   Hex dump                          ASCII
0040305D  08 CB 88 3B 08 CB 88 3B  ...;...;
00403065  9C F4 00 00 [illegible] 7D  ....V`z}
0040306D  2F 66 61 [illegible]  /fa.z{/.
00403075  63 6A 6E 7C [illegible] 00  cjn|j!..
0040307D  00 40 00 CA 04 01 00 05  .@......
00403085  00 00 00 00 00 00 00 00  ........
0040308D  00 [illegible] 00 00 00 00  ........
00403095  [illegible] 00 00 00  ........
0040309D  [illegible] 00 00 00 00  ........
004030A5  00 00 00 00 00 00 00 00  ........
```

درون DL قرار میگیرد

درون DH قرار میگیرد

جمع بندی مراحل انجام شده :

۱-هر ۴ بایت پسورد وارد شده ی ما (در دو مرحله) با مقدار 1234567، XOR شد و نتیجه در جای خودش قرار گرفت

۲- اولین رقم پسورد XORشده با مقدار 0E، And شد و در جای خودش قرار گرفت

۳-مقادیر بایت های ذخیر شده مشخص شد EAX+8

۴-دوباره هر ۴ بایت پسورد وارد شده ی ما (در دو مرحله) با مقدار 9ABCDEF، XOR شد و نتیجه در جای خودش قرار گرفت

۵- ذخیره شدن مقادیر بایت ها در جایگاه EAX+9

خب ما الگوریتم را بدست آوردیم و فهمیدیم این برنامه با پسورد چکار میکند ،نتیجه این شد که دو ارزش 9C و F4 که یکی در EAX+8 دیگری در EAX+9 واقع در DL,DH قرار دارد ، به ثبات EDX نگاه کنید :

```
EDX 00009CF4
```

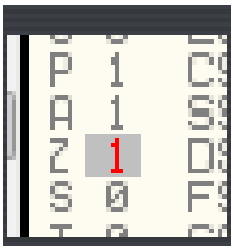
حال این مقدار (9CF4) با مقدار(42DE) مقایسه میشود :

```
00401474  .  8A70 08        MOV DH,BYTE PTR [EAX+8]
00401477  .  66:81FA DE42   CMP DX,42DE
0040147C  .v 0F85 8E000000  JNZ Crackme6.00401510
00401482  .  B9 09000000    MOV ECX,9
00401487  >  8A1401         MOV DL,BYTE PTR [ECX+EAX]
0040148A  .  321401         XOR DL,BYTE PTR [ECX+EAX]
```

شاید بپرسید چرا 42DE ،خب این در حقیقت یک عدد است که با الگوریتمی کد شده ، از آنجایی که ما پسورد درست را وارد نکرده ایم در خط 40147C پرش به صوی MOV EAX,1 صورت خواهد پذیرفت :

```
00401474  .  8A70 08        MOV DH,BYTE PTR DS:[EAX+8]
00401477  .  66:81FA DE42   CMP DX,42DE
0040147C  .v 0F85 8E000000  JNZ Crackme6.00401510
00401482  .  B9 09000000    MOV ECX,9
00401487  >  8A1401         MOV DL,BYTE PTR DS:[ECX+EAX]
0040148A  .  321401         XOR DL,BYTE PTR DS:[ECX+EAX]
0040148D  .  49             DEC ECX
0040148E  .  67:E3 02       JCXZ SHORT Crackme6.00401493
00401491  .^ EB F4          JMP SHORT Crackme6.00401487
00401493  >  66:8B48 08     MOV CX,WORD PTR DS:[EAX+8]
00401497  .  66:81F1 EEEE   XOR CX,0EEEE
0040149C  .  66:81F9 AC30   CMP CX,30AC
004014A1  .v 75 64          JNZ SHORT Crackme6.00401507
004014A3  .  8A08           MOV CL,BYTE PTR DS:[EAX]
004014A5  .  8A68 01        MOV CH,BYTE PTR DS:[EAX+1]
004014A8  .  66:81C1 9235   ADD CX,3592
004014AD  .  66:81F9 9AE5   CMP CX,0E59A
004014B2  .v 75 49          JNZ SHORT Crackme6.004014FD
004014B4  .  8138 08B0817A  CMP DWORD PTR DS:[EAX],7A81B008
004014BA  .v 75 4B          JNZ SHORT Crackme6.00401507
004014BC  .  66:33C9        XOR CX,CX
004014BF  >  80F2 0A        XOR DL,0A
004014C2  .  83C1 04        ADD ECX,4
004014C5  .  83F9 0C        CMP ECX,0C
004014C8  .^ 7E F5          JLE SHORT Crackme6.004014BF
004014CA  .  8178 04 02BF8D38 CMP DWORD PTR DS:[EAX+4],388DBF02
004014D1  .v 75 3D          JNZ SHORT Crackme6.00401510
004014D3  .  8ACA           MOV CL,DL
004014D5  .  32D1           XOR DL,CL
004014D7  .  8AD1           MOV DL,CL
004014D9  .  32CA           XOR CL,DL
004014DB  .  8A48 05        MOV CL,BYTE PTR DS:[EAX+5]
```

بسیار خب با تغییر پرچم Z کار خود را ادامه میدهیم :



در خط بعد مقدار 9 درون ECX قرار گرفت (۹ رقم از باقر) و در خط بعد مقدار ECX که همان مقدار F4 هست درون DL دوباره قرار میگیرد و با خودش در خط بعدی XOR میشود که نتیجه 00 خواهد شد ،در خط بعد یک واحد از ECX کم میشود

دستورالعمل پرش شرطی JCXZ به این منظور به کار می‌رود که اگر محتوای ثبات CX صفر باشد به مقصدی که برای آن تعیین می‌کند پرش کند.این دستور همانند دستورالعمل‌های پرشی دیگر به دو بایت کد هدف نیاز دارد و بر روی فلگ‌ها هیچ اثری نمی‌گذارد.

دوباره به ابتدای حلقه برمیگردیم ،اکنون مقدار ECX=8 و در در خط 40148A مقدار ecx+eax=8 خواهد شد و محتوای آن که الان 9C است داخل DL ریخته میشود در این هنگام ثبات EDX اینگونه است:

```
EDX 00009C9C
```

هر بار (تا انتهای حلقه که ۹ بار است) این روند ادامه پیدا میکند و به ترتیب : F4 9C 3B 88 CB 08 3B 88 CB قرار خواهند گرفت،بعد از اتمام حلقه ما در خط 401497 هستم دراین خط یک کلمه Word (دوبایتی) از محل EAX+8 یعنی F49C درون ثبات ECX قرار میگیرد :

```
0040145B  .  83C1 04      ADD ECX,4
0040145E  .  83F9 08      CMP ECX,8
00401461  .^ 75 ED        JNZ SHORT Crackme6.00401450
00401463  .  33C9         XOR ECX,ECX
00401465  >  8A1401       MOV DL,BYTE PTR [ECX+EAX]
00401468  .  0050 09      ADD BYTE PTR [EAX+9],DL
0040146B  .  41           INC ECX
0040146C  .  3B4D 0C      CMP ECX,DWORD PTR [EBP+C]
0040146F  .^ 75 F4        JNZ SHORT Crackme6.00401465
00401471  .  8A50 09      MOV DL,BYTE PTR [EAX+9]
00401474  .  8A70 08      MOV DH,BYTE PTR [EAX+8]
00401477  .  66:81FA DE42 CMP DX,42DE
0040147C  .v 0F85 8E000000 JNZ Crackme6.00401510
00401482  .  B9 09000000  MOV ECX,9
00401487  >  8A1401       MOV DL,BYTE PTR [ECX+EAX]
0040148A  .  321401       XOR DL,BYTE PTR [ECX+EAX]
0040148D  .  49           DEC ECX
0040148E  .  67:E3 02     JCXZ SHORT Crackme6.00401493
00401491  .^ EB F4        JMP SHORT Crackme6.00401487
00401493  >  66:8B48 08   MOV CX,WORD PTR [EAX+8]
00401497  .  66:81F1 EEEE XOR CX,0EEEE
0040149C  .  66:81F9 AC30 CMP CX,30AC
004014A1  .  75 64        JNZ SHORT Crackme6.00401507

Registers (FPU)
EAX 0040305D Crackme6
ECX 0000F49C
EDX 00009C00
EBX 00000000
ESP 0012FAC8
EBP 0012FAD0
ESI 00000111
EDI 0012FB5C
EIP 00401497 Crackme6
C 0  ES 0023 32bit 0
P 1  CS 001B 32bit 0
A 0  SS 0023 32bit 0
Z 1  DS 0023 32bit 0
S 0  FS 003B 32bit 7
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SU
EFL 00000246 (NO,NB,E
ST0 empty 0.0
```

در خط بعدمقدار CX با 0EEEE ،XOR میشود که نتیجه برابر با 1A74 میشود

```
0040148E  JCXZ SHORT Crackme6.00401493
00401491  JMP SHORT Crackme6.00401487
00401493  MOV CX,WORD PTR DS:[EAX+8]
00401497  .  66:81F1 EEEE  XOR CX,0EEEE
0040149C  CMP CX,30AC
004014A1  JNZ SHORT Crackme6.00401507
004014A3  MOV CL,BYTE PTR DS:[EAX]
004014A5  .  8A68 01       MOV CH,BYTE PTR DS:[EAX+1]
004014A8  .  66:81C1 9235  ADD CX,3592
004014AD  .  66:81F9 9AE5  CMP CX,0E59A
004014B2  .v 75 49         JNZ SHORT Crackme6.004014FD
004014B4  .  8138 08B0817A CMP DWORD PTR DS:[EAX],7A81B008
```

CX =F49C

XOR with 0EEEE

Is ECX =1A72

If not equal, jump

سپس مقدار CX با 30AC مقایسه میشود :

```
EAX 0040305D
ECX 00001A72
EDX 00009C00
```

و در نهایت پرش صورت میگیرد:

```
0040149C  .  66:81F9 AC30  CMP CX,30AC
004014A1  .v 75 64         JNZ SHORT Crackme6.00401507
004014A3  .  8A08          MOV CL,BYTE PTR [EAX]
004014A5  .  8A68 01       MOV CH,BYTE PTR [EAX+1]
004014A8  .  66:81C1 9235  ADD CX,3592
004014AD  .  66:81F9 9AE5  CMP CX,0E59A
004014B2  .v 75 49         JNZ SHORT Crackme6.004014FD
004014B4  .  8138 08B08176 CMP DWORD PTR [EAX],7A81B008
004014BA  .v 75 4B         JNZ SHORT Crackme6.00401507
004014BC  .  66:33C9       XOR CX,CX
004014BF  >  80F2 0A       XOR DL,0A
004014C2  .  83C1 04       ADD ECX,4
004014C5  .  83F9 0C       CMP ECX,0C
004014C8  .^ 7E F5         JLE SHORT Crackme6.004014BF
004014CA  .  8178 04 02BF8 CMP DWORD PTR [EAX+4],388DBF02
004014D1  .v 75 3D         JNZ SHORT Crackme6.00401510
004014D3  .  8ACA          MOV CL,DL
004014D5  .  32D1          XOR DL,CL
004014D7  .  8AD1          MOV DL,CL
004014D9  .  32CA          XOR CL,DL
004014DB  .  8A48 05       MOV CL,BYTE PTR [EAX+5]
004014DE  .  8A50 06       MOV DL,BYTE PTR [EAX+6]
004014E1  .  86D1          XCHG CL,DL
004014E3  .  80FA BF       CMP DL,0BF
004014E6  .v 75 1F         JNZ SHORT Crackme6.00401507
004014E8  .  80F9 8D       CMP CL,8D
004014EB  .v 75 23         JNZ SHORT Crackme6.00401510
004014ED  .  8078 05 BF    CMP BYTE PTR [EAX+5],0BF
004014F1  .v 75 14         JNZ SHORT Crackme6.00401507
004014F3  .  25 FFFF0000   AND EAX,0FFFF
004014F8  .  66:33C0       XOR AX,AX
004014FB  .v EB 18         JMP SHORT Crackme6.00401515
004014FD  >  8AD1          MOV DL,CL
004014FF  .  32CA          XOR CL,DL
00401501  .  B0 01         MOV AL,1
00401503  .  04 20         ADD AL,20
00401505  .  8AC8          MOV CL,AL
00401507  >  66:B9 9138    MOV CX,3891
0040150B  .  66:81F1 AD0F  XOR CX,0FAD
```

اساسا دومین بار است که پسورد چک میشود ،با تغییر پرچم zپرش را غیر فعال کنید و در خط بعدی میبینید که اولین بایت 08 از ادرس EAX که همان 40305D است را درون CL میگذارید و درخط بعدی دومین بایت CB را درون CH میگذارد.

و در خط بعد مقدار 3592 را با 08CB(مقدار CX) جمع میکند و نتیجه را درون CX مینویسد :

```
004014A1  .v 75 64         JNZ SHORT Crackme6.00401507
004014A3  .  8A08          MOV CL,BYTE PTR [EAX]      08
004014A5  .  8A68 01       MOV CH,BYTE PTR [EAX+1]    CB
004014A8  .  66:81C1 9235  ADD CX,3592
004014AD  .  66:81F9 9AE5  CMP CX,0E59A
004014B2  .v 75 49         JNZ SHORT Crackme6.004014FD
```

CB08+3592=1009A

```
0000009A
00009C00
```

توجه داشته باشید که منظور ثبات ۱۶ بیتی CX به کار گرفته شده در واقع ۱۶ بیت استفاده میشود در نتیجه مقدار این ثبات 009A: است

درخط 4014AD مقدار CX(009A) را با مقدار 0E59A مقایسه و درصورت مساوی نبودن پرش میکند،پس پرش صورت میگیرد:

```
004014B2  .v  75 49          JNZ SHORT Crackme6.004014FD
004014B4  .   8138 08B08176  CMP DWORD PTR [EAX],7A81B008
004014BA  .v  75 4B          JNZ SHORT Crackme6.00401507
004014BC  .   66:33C9        XOR CX,CX
004014BF  >   80F2 0A        XOR DL,0A
004014C2  .   83C1 04        ADD ECX,4
004014C5  .   83F9 0C        CMP ECX,0C
004014C8  .^  7E F5          JLE SHORT Crackme6.004014BF
004014CA  .   8178 04 02BF8  CMP DWORD PTR [EAX+4],388DBF02
004014D1  .v  75 3D          JNZ SHORT Crackme6.00401510
004014D3  .   8ACA           MOV CL,DL
004014D5  .   32D1           XOR DL,CL
004014D7  .   8AD1           MOV DL,CL
004014D9  .   32CA           XOR CL,DL
004014DB  .   8A48 05        MOV CL,BYTE PTR [EAX+5]
004014DE  .   8A50 06        MOV DL,BYTE PTR [EAX+6]
004014E1  .   86D1           XCHG CL,DL
004014E3  .   80FA BF        CMP DL,0BF
004014E6  .v  75 1F          JNZ SHORT Crackme6.00401507
004014E8  .   80F9 8D        CMP CL,8D
004014EB  .v  75 23          JNZ SHORT Crackme6.00401510
004014ED  .   8078 05 BF     CMP BYTE PTR [EAX+5],0BF
004014F1  .v  75 14          JNZ SHORT Crackme6.00401507
004014F3  .   25 FFFF0000    AND EAX,0FFFF
004014F8  .   66:33C0        XOR AX,AX
004014FB  .v  EB 18          JMP SHORT Crackme6.00401515
004014FD  >   8AD1           MOV DL,CL
004014FF  .   32CA           XOR CL,DL
```

البته این پرش نباید صورت بگیرد،برای همین از olly کمک میگریم و مقدار پرچم z را به 1 تغییر میدهیم ،همانطور که میبینید یک حلقه ی دیگر وجود دارد،در آدرس 004014B4 یک مقایسه صورت گرفته که اگر چهاربایت اول از ثبات EAXبرابر با مقدار 7A8aB008 نیست پرش به سمت bad boy صورت میگیرد،همانطور که در تصویر زیر میبینید در پنجره ی Info بالای پنجره ی دامپ ایت چهار بایت برابر است با:

```
DS:[0040305D]=3B88CB08
```

میبینید که با 7A8AB008 برابر نیست :

```
004014B2  .v  75 49          JNZ SHORT Crackme6.004014FD
004014B4  .   8138 08B08176  CMP DWORD PTR [EAX],7A81B008
004014BA  .v  75 4B          JNZ SHORT Crackme6.00401507
004014BC  .   66:33C9        XOR CX,CX
004014BF  >   80F2 0A        XOR DL,0A
004014C2  .   83C1 04        ADD ECX,4
004014C5  .   83F9 0C        CMP ECX,0C
004014C8  .^  7E F5          JLE SHORT Crackme6.004014BF
004014CA  .   8178 04 02BF8  CMP DWORD PTR [EAX+4],388DBF02
004014D1  .v  75 3D          JNZ SHORT Crackme6.00401510
004014D3  .   8ACA           MOV CL,DL
004014D5  .   32D1           XOR DL,CL
004014D7  .   8AD1           MOV DL,CL
004014D9  .   32CA           XOR CL,DL
004014DB  .   8A48 05        MOV CL,BYTE PTR [EAX+5]
004014DE  .   8A50 06        MOV DL,BYTE PTR [EAX+6]
004014E1  .   86D1           XCHG CL,DL
004014E3  .   80FA BF        CMP DL,0BF
004014E6  .v  75 1F          JNZ SHORT Crackme6.00401507
004014E8  .   80F9 8D        CMP CL,8D
004014EB  .v  75 23          JNZ SHORT Crackme6.00401510
004014ED  .   8078 05 BF     CMP BYTE PTR [EAX+5],0BF
004014F1  .v  75 14          JNZ SHORT Crackme6.00401507
004014F3  .   25 FFFF0000    AND EAX,0FFFF
004014F8  .   66:33C0        XOR AX,AX
004014FB  .v  EB 18          JMP SHORT Crackme6.00401515
004014FD  >   8AD1           MOV DL,CL
004014FF  .   32CA           XOR CL,DL
00401501  .   B0 01          MOV AL,1
00401503  .   04 20          ADD AL,20
00401505  .   8AC8           MOV CL,AL
00401507  >   66:B9 9138     MOV CX,3891
0040150B  .   66:81F1 AD0F   XOR CX,0FAD
00401510  >   B8 01000000    MOV EAX,1
00401515  >   5A             POP EDX
00401516  .   59             POP ECX
00401517  .   C9             LEAVE
Jump is taken
```

کمکان نیازی به این پرش هم نیست با کمک Olly مقدار پرچم Z را برابر 1 قرار دهید:

```
004014BA  .v 75 4B          JNZ SHORT Crackme6.00401507
004014BC  .  66:33C9        XOR CX,CX                    CX=0
004014BF  >  80F2 0A        XOR DL,0A                    DL=0 and xoring with 0A So DL=0A
004014C2  .  83C1 04        ADD ECX,4                    ECX=0 and ecx+4=4 So ECX=4
004014C5  .  83F9 0C        CMP ECX,0C                   ecx=4 and "cmp ecx,0c" is Wrong
004014C8  .^ 7E F5          JLE SHORT Crackme6.004014BF  So Until ECX=0C (12)
004014CA  .  8178 04 02BF8  CMP DWORD PTR [EAX+4],388DBF02   CMP 388DB02 by 08BAFA4C ?
004014D1  .v 75 3D          JNZ SHORT Crackme6.00401510
004014D3  .  8ACA           MOV CL,DL
004014D5  .  32D1           XOR DL,CL
004014D7  .  8AD1           MOV DL,CL
004014D9  .  32CA           XOR CL,DL
004014DB  .  8A48 05        MOV CL,BYTE PTR [EAX+5]
004014DE  .  8A50 06        MOV DL,BYTE PTR [EAX+6]
004014E1  .  86D1           XCHG CL,DL
004014E3  .  80FA BF        CMP DL,0BF
004014E6  .v 75 1F          JNZ SHORT Crackme6.00401507
004014E8  .  80F9 8D        CMP CL,8D
004014EB  .v 75 23          JNZ SHORT Crackme6.00401510
004014ED  .  8078 05 BF     CMP BYTE PTR [EAX+5],0BF
004014F1  .v 75 14          JNZ SHORT Crackme6.00401507
004014F3  .  25 FFFF0000    AND EAX,0FFFF
004014F8  .  66:33C0        XOR AX,AX
```

همانطور که در توضیحات مشاهده میکنید ابتدا مقدار DL با XOR، 0A میشود و سپس در خط بعد مقدار ECXبا 4 جمع میشود ،وبا C مقایسه میشود و درصورت نامساوی بودن این حلقه ادامه پیدا میکند تا ECX برابر با C یا ۱۲ در مد دسیمال شود ،حالا از تمام JNZ ها عبور کنید (پرچم ها به ۱ تغییر دهید) تا به خط 4014FB برسید:

```
004014E1  .  86D1           XCHG CL,DL
004014E3  .  80FA BF        CMP DL,0BF
004014E6  .v 75 1F          JNZ SHORT Crackme6.00401507
004014E8  .  80F9 8D        CMP CL,8D
004014EB  .v 75 23          JNZ SHORT Crackme6.00401510
004014ED  .  8078 05 BF     CMP BYTE PTR DS:[EAX+5],0BF
004014F1  .v 75 14          JNZ SHORT Crackme6.00401507
004014F3  .  25 FFFF0000    AND EAX,0FFFF
004014F8  .  66:33C0        XOR AX,AX
004014FB  .v EB 18          JMP SHORT Crackme6.00401515
004014FD  >  8AD1           MOV DL,CL
004014FF  .  32CA           XOR CL,DL
00401501  .  B0 01          MOV AL,1
00401503  .  04 20          ADD AL,20
00401505  .  8AC8           MOV CL,AL
00401507  >  66:B9 9138     MOV CX,3891
0040150B  .  66:81F1 AD0F   XOR CX,0FAD
00401510  >  B8 01000000    MOV EAX,1
00401515  >  5A             POP EDX
00401516  .  59             POP ECX
00401517  .  C9             LEAVE
00401518  .  C2 0800        RETN 8
0040151B  $  55             PUSH EBP
```

به مقدار ثبات EAX در پنجره ی ریجیسترها نگاه کنید :

```
EAX 00000000
ECX 00000099
EDX 00004BAA
EBX 00000000
ESP 0012F9C8
```

بله مقدار 0 را دارد ،۴ بار کلید F8 را بفشارید تا از روتین خارج شوید :

```
00401516  .  59             POP ECX
00401517  .  C9             LEAVE
00401518  .  C2 0800        RET 8
0040151B  $  55             PUSH EBP
0040151C  .  8BEC           MOV EBP,ESP
```

وداریم ☺ :

```
- [*G.P.U* - main thread, module Crackme6]
File View Debug Plugins Options Window Help Tools BreakPoint->

0040129D  .  0BC0          OR EAX,EAX
0040129F  .v 75 1F         JNZ SHORT Crackme6.004012C0
004012A1  .  68 0F304000   PUSH Crackme6.0040300F                 Text = "ACCESS GRANTED!"
004012A6  .  FF35 80304000 PUSH DWORD PTR [403080]               hWnd = 000205B6 (class='Edit',parent=000205BC)
004012AC  .  E8 CB020000   CALL <JMP.&user32.SetWindowTextA>     SetWindowTextA
004012B1  .  6A 00         PUSH 0                                Enable = FALSE
004012B3  .  FF35 80304000 PUSH DWORD PTR [403080]               hWnd = 000205B6 (class='Edit',parent=000205BC)
004012B9  .  E8 88020000   CALL <JMP.&user32.EnableWindow>       EnableWindow
004012BE  .v EB 10         JMP SHORT Crackme6.004012D0
004012C0  > 68 00304000    PUSH Crackme6.00403000                Text = "ACCESS DENIED!"
004012C5  .  FF35 80304000 PUSH DWORD PTR [403080]               hWnd = 000205B6 (class='Edit',parent=000205BC)
004012CB  .  E8 AC020000   CALL <JMP.&user32.SetWindowTextA>     SetWindowTextA
004012D0  > 6A 0E          PUSH 0E
004012D2  .  68 00304000   PUSH Crackme6.00403000                ASCII "ACCESS DENIED!"
004012D7  .  E8 3F020000   CALL Crackme6.0040151B
004012DC  .  6A 0F         PUSH 0F
004012DE  .  68 0F304000   PUSH Crackme6.0040300F                ASCII "ACCESS GRANTED!"
004012E3  .  E8 33020000   CALL Crackme6.0040151B

Registers (FPU)
EAX 00000000
ECX 00000001
EDX 77F070B4 ntdll.KiFastSystemCallRet
EBX 00000000
ESP 0012FAE0
EBP 0012FAE0
ESI 00000111
EDI 0012FB5C

EIP 0040129D Crackme6.0040129D

C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 1  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDF000(FFF)
T 0  GS 0000 NULL
D 0
```

توجه کنید که ما پیام !ACCESS GRANTED را دریافت کرده ایم دلیلش هم این است که eax=0 است

خسته نباشید ،می فهمم احیانا خسته شدیده اید و به خودتان میگویید که وقتی که برنامه اینقدر راحت پچ میشود پس این همه کار برای چیست ،اما یادتان نرود که هدف این آموزش درک کارکرد برنامه است ،ما الان میدانیم که برنامه چگونه کار میکند ،این یعنی مهندسی معکوس ، البته اگر هنوز چیزهایی برایتان مبهم است میتوانید در سایت iranled.com آن را مطرح کنید.