سری آموزش های R4NDOM-فارسی

قسمت 16- بخش اول-روش های مقابله با پیام های ویندوز

سلام، بعد از غلبه بر دو ویروس یکی برای کامپیوترم ودیگری برای خودم (اشاره به کسالت شخصی) ،این آموزش را تقدیم میکنم ،که شامل سه بخش خواهد بود ،فایل تمرین با نام Crackme12 نوشته شده توسط Detten را در بخش اول یعنی Windows messagingرا بررسی خواهیم کرد،در بخش دوم درباره مبحث self-modifying code بحث خواهیم کرد،وهمینطورهمین برنامه را کرک خواهیم کرد و در بخش سوم حمله ی bruteforce به همین فایل باینری را جهت بدست آوردن پسورد ، خواهیم آموخت ،این سه مبحث ممکن است کمی سخت به نظر برسد (مترجم :نگران نباشید سعی میکنم در صورت گنگ بودن توضیحات کمکی اضافه کنم ☺)

ناگفته نماند که یادگیری این سه بخش گام بسیار مهمی در ادامه ی مبحث باینری کرکینگ برای شماست واینکه اگر سوالی دارید حتما واکیدا توصیه میکنم بپرسید (در فروم)

مثله همیشه همه ی فایل های مورد نیاز ضمیمه ی آموزش شده،در قسمت اول سه فایل ضمیمه شده با نام های : crackme12.exe و Windows Messages.txt و همین PDF .

بدون هیچگونه معطلی شروع میکنیم

آشنایی با پیام های ویندوزی

در این قسمت ما در باره ی پیام های ویندوزی و رویه هایی که آنها را هدایت میکنند صحبت میکنیم برخلاف برنامه های تحت داس برنامه های تحت ویندوز از event-driven استفاده میکنند ،برنامه نویسی Event-Drivenبه سبک خاصی از برنامه نویسی می‌گویند که در آن جریان اجرای برنامه توسطEvent ها تعیین می‌شود. در هنگام تعریف هر Event برنامه نویس موظف است برای آن Event یک-Event Handlerیا Event-Callback نیز تعریف کند تا در هنگام اتفاق افتادن آن Event صدا زده شوند و وظیفه خود را انجام دهد برای مثال آب درون کتری را درنظر بگیرید ، اگر آب درون کتری جوش بیاید ، Event جوش آمدن آب درون کتری اتفاق میافتد و وقتی آب جوش آمد ما زیر گاز را خاموش میکنم ، که درواقع Event-Callback را صدا زده ایم.درواقع windowsیک سیستم عامل رویدادگرا یا دقیقتر پیام گرا است

حال یک پیام (Message) چیست ؟

پیام اطلاعاتی است درباره تغییر در رابط کاربر (GUI) نظیر شروع حرکت یک پنجره یا فشار یک کلید بر روی صفحه کلید بوسیله کاربر،بستن یک پنجره،حرکت موس و .. از نقطه نظر برنامه نویسی، یک پیام مقداری صحیح Integer بدون علامت (UNIT) است که برای راحت خواندن با یک پارامتر که با _WM شروع می شود، مانند WM_LBUTTONDOW ارائه می شود. پیام WM_LBUTTONDOW به این معنی است که کلید سمت چپ ماوس فشار داده شده است. پیامWM_LBUTTONUP وقتی فرستاده می شود که کلید سمت چپ ماوس رها شده است. حتماً تا کنون تعاریفی به این شکل را در برنامه های مختلف دیده اید: :

```
1  #define WM_INITDIALOG                    0x0110
2  #define WM_COMMAND                       0x0111
3
4  #define WM_LBUTTONDOWN                   0x0201
5
```

لیست پیام ها در ضمیمه آموزش موجود است

از پیامها (Messages) برای برقراری ارتباط بین هر آنچه در ویندوز موجود است ، در پائین ترین سطح ممکن استفاده میشود . اگر میخواهید که یک پنجره یا یک کنترل (که خود نوعی پنجره است) در ویندوز ، کار خاصی انجام دهد ، باید پیامی برای آن ارسال کنید .به همین شکل اگر پنجره دیگری بخواهد که شما (بعنوان یک پنجره) کاری را انجام دهید ، منظورش را بصورت یک پیام (Message) برای شما میفرستد . اگر رویدادی اتفاق بیفتد (مثلاً کاربر کلمه ای را تایپ کند ، موس حرکت کند و یا دکمه (Button) ای فشار داده شود (سیستم برای پنجره ها پیام را ارسال میکند . اگر شما هم یکی از آن پنجره ها باشید ، باید پیام رسیده را هندل کنید و مطابق با آن عمل کنید).

هر پیام در ویندوز حداکثر با دو پارامتر همراه است wParam و lParam در اصل wParam 16بیتی است و lParam 32بیتی است ولی در win32 هر دو 32 بیتی هستند . لزوماً تمام پیام ها از هر دوی این پارامتر ها استفاده نمی کنند . هر پیام بصورت مجزا از این پیام ها استفاده میکند . مثلاً پیام WM_CLOSEکه برای بستن یک پنجره استفاده میشود ، از هیچیک از این پارامترها استفاده ای نمی کند،اما WM_COMMAND از هر دو استفاده میکند به این شکل:

wParam شامل دو مقدار است . کلمه باارزشتر (HIWORD) حاوی پیام اخطار و کلمه کم ارزشتر (Loword)حاوی ID ی (شناسه ی) کنترل یا منویی است که پیام را ارسال کرده است.

lParamشاملHWND (هندل پنجره) کنترلی است که پیام را ارسال کرده است و یا اینکه مقدار آن

NULLاست اگر پیام از طرف هیچ کنترلی ارسال نشده باشد.

عبارت HIWORD و LOWORD ماکروهایی هستند که توسط خود ویندوز تعریف شده اند به این صورت که برای جدا کردن دو بایت با ارزش و دو بایت کم ارزش یک کلمه 32 بیتی استفاده میشوند.

دو بایت (کلمه) با ارزشتر = x**FFFF**00000

دو بایت (کلمه) کم ارزشتر = x0000**FFFF**0

در win32 یک کلمه(Word) 16 بیتی است و DWord مقداری است 32 بیتی.

برای فرستادن یک پیام از دو تابع PostMessage و SendMessage می توان استفاده کرد:

PostMessage -پیام را داخل صف پیام ها (Message Queue) قرار میدهد و بلافاصله برمیگردد . به این معنا که وقتی شما PostMessage را فراخوانی می کنید و این تابع اجرا می شود ، ممکن است که ارسال پیام شما انجام شده باشد و یا هنوز در حال انجام شدن باشد.

-تفاوت SendMessage با قبلی هم در این است که که SendMessage تا وقتی که پیام ارسال نشود برنمیگردد . در واقع این یکی پیام را مستقیماً به پنجره مورد نظر میفرستد.

اگر ما بخواهیم که پنجره ای را ببندیم ، از PostMessage می توان به این صورت استفاده کرد:

```
PostMessage(hwnd, WM_CLOSE, 0, 0);
```

این کد دقیقاً مثل این عمل میکند که شما روی دکمه ضربدری که معمولا بالای سمت راست هر پنجره قرار دارد کلیک کنید.

همانطور که قبلاً هم گفته شد ، دو پارامتر wParam و lParam برای WM_CLOSE استفاده نمیشوند و مقدار آنها صفر است.

پیامها برای برنامه نویسان ویندوز بسیار مهم است. بیشتر کارهای که شما به عنوان برنا مه نویس ویندوز انجام می دهید شامل این موارد است: تصمیم درباره اینکه کدام پیام باید پردازش شود و از کدام پیام باید صرفنظر شود.

برای درک بهتر فرض کنید که ما یک Message Box داریم که فقط یک دکمه ی Ok دارد،ما فقط نگران کلیک کردن و یا نکردن کاربر بروی دکمه ی OK هستیم و با باقی پروسه کاری نداریم مانند اینکه اگر کاربر خارج از دکمه ی Okکلیک کرد (WM_MOUSEBUTTONDOWN) و یا اینکه Message Boxرا جابجا کند

(WM_MOVE) ،تمام این کارهایی را که نمیخواهیم ، ویندوز برای ما مدیریت وانجام میدهد.و فقط در صورت کلیک کردن بروی دکمه ی OK عملیات از طرف کد ما صورت میگیرد

Dialog Box Procedure

Dialogproce مسئول هندل کردن تمام پیام های مربوط به دیالوگ باکس برنامه است،درواقع مسئولیت پاسخ به تمام Event ها را برعهده دارد، و مانند شکل زیر در C++ تعریف میشود :

```
INT_PTR CALLBACK DialogProc(HWND hDlg, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
  return FALSE;
}
```

ویندوز این تابع را برای هر پیام مربوط دیالوگ باکس برنامه، فراخوانی میکند،اگر مقدار بازگشتی برابر False تنظیم شود به این معنی است که ما از پیام هایی که علاقه ایی به هندل آنها نداریم چشم پوشی کرده و این کار را به ویندوز واگذار میکنیم، ویندوز به صورت پیش فرض به این پیام ها پاسخ میدهد،و اگر مقدار بازگشتی تابع برابر با True مقدار دهی شود به این معناست که کنترل را از ویندوز گرفته وخود تمام پیام ها را مدیریت میکنیم.

WindowProc Callback Function

این تابع در برنامه تعریف شده است که پیام های ارسال شده به یک پنجره را پردازش میکند و مانند شکل زیر تعریف میشود:

```
LRESULT CALLBACK WindowProc(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
```

hWnd

هندل پنجره ی شماست . همان که پیام را میپذیرد . تشخیص این مهم است که کدام پنجره باید پیام را بپذیرد . ممکن است شما چندین پنجره از یک کلاس داشته باشید که همگی از یک Window Procedure یعنی WndProc() استفاده کنند . تفاوت فقط توسط پارامتر hwnd مشخص میشود . مثلاً وقتی از پیام WM_CLOSEاستفاده میشود "WM_CLOSE همان پیامی است که وقتی کاربر روی علامت ضربدر

پنجره (معمولا بالای تمام پنجره ها ، سمت راست) را کلیک کند ، فرستاده می شود " ، فقط یک پنجره باید بسته شود و در سایر پنجره ها نباید تاثیری داشته باشد.

uMsg

شناسایی پیام واقعی

wParam و *lParam*

دو پارامتر این تابع هستند که نقش آنها بستگی به نوع پیام شناسایی شده در *uMsg* دارد

نکته: به صورت پیش فرض به محض اتمام مدیریت پنجره توسط WndProc اگر پیام جدید در لیست هندل ما نباشد این وظیفه را DefWindowProc انجام خواهد داد

صف پیامها (Message Queue) چیست ؟

فرض کنید که که ما مشغول پاسخگویی به پیام WM_PAINT هستیم و در این هنگام کاربر کلمه ای را با استفاده از کیبورد تایپ میکند . به نظر شما چه اتفاقی می افتد؟

آیا باید در کار ترسیم صفحه وقفه ای وارد شود و به سراغ کیبورد برویم و یااینکه تایپ کاربر را نادیده بگیریم ؟ همانطور که (احتمالاً) میدانید هر دوی اینها غلط است . راه حلی که در اینگونه موارد استفاده میشود استفاده از صف پیام است به این شکل که وقتی پیامی ارسال میشود ، به انتهای صف اضافه شده و به محض اینکه نوبت بررسی و اجرای آنها برسد از صف حذف میشوند . با این کار هیچ پیامی بدون بررسی حذف نمی شود.

حلقه پیام (Message Loop) چیست؟

```
while(GetMessage(&Msg, NULL, 0, 0) > 0)
{
    WNDPROC fWndProc = (WNDPROC)GetWindowLong(Msg.hwnd, GWL_WNDPROC);
    fWndProc(Msg.hwnd, Msg.message, Msg.wParam, Msg.lParam);
}
```

1-حلقه پیام ، تابع GetMessages را فراخوانی میکند و به این ترتیب پیامی از صف پیامها خوانده میشود . اگر صف پیام خالی باشد ، برنامه شما (پروسس برنامه شما) بلاک میشود. یعنی منتظر یک پیام می ماند

2-وقتی رویدادی (event) اتفاق می افتد که باعث افزوده شدن پیامی به صف پیامها می- شود (مثلاً یک کلیک موس) ، تابع GetMessages مقداری مثبت برمیگرداند که نشان دهنده وجود پیامی برای بررسی است . در ادامه ساختار Msg (Struct) که ما به آن پاس داده ایم با مقادیر مناسب پر می شود . اگر مقدار برگردانده شده صفر باشد یعنی با پیام WM_QUIT مواجه شده ایم و اگر مقدار منفی برگردانده شود یعنی خطایی اتفاق افتاده است.

3- پیام دریافتی (از طریق متغیر (Msg را به تابع TranslateMessage)پاس میدهیم تا یکسری پردازشهای اضافی روی پیام انجام شود.

4- سپس تابع DispatchMessage را فراخوانی میکنیم . کاری که این تابع انجام میدهد به این ترتیب است : پیام را میگیرد ، بررسی می کند که مربوط به کدام پنجره است و سپس بدنبال Window Procedureمربوط به آن پنجره می گردد . سپس آنرا فراخوانی میکند و هندل پنجره ، پیام ، wParam و lParam را بعنوان پارامتر برای آن ارسال میکند .

5- در () Window Procedure ، شما پیام و پارامترهای مربوط به آن را بررسی می کنید و هرگونه که بخواهید میتوانید با آن برخورد کنید (!) اگر شما پیامی را هندل نکنید ، تابع DefWindowProc برای آن فراخوانی می شود و عملیاتی که خود ویندوز بصورت پیش فرض در نظر گرفته است ، برای آن انجام میگیرد (اغلب هم هیچ کاری انجام نمی شود !)

6- هنگامی که کار شما و Window Procedure و DispatchMessage تمام شد ، به ابتدای حلقه برمیگردیم.

شروع

برنامه ی crackme12.exe را درون دیباگر بارگذاری کنید:

```
<ModuleEn  $  6A 00          PUSH 0                          pModule = NULL
00401002   .  E8 C5040000    CALL <JMP.&KERNEL32.GetModuleHandleA>   GetModuleHandleA
00401007   .  A3 28304000    MOV DWORD PTR [403028],EAX
0040100C   .  6A 00          PUSH 0                          lParam = NULL
0040100E   .  68 2B104000    PUSH crackme1.0040102B          DlgProc = crackme1.0040102B
00401013   .  6A 00          PUSH 0                          hOwner = NULL
00401015   .  68 20030000    PUSH 320                        pTemplate = 320
0040101A   .  FF35 28304000  PUSH DWORD PTR [403028]         hInst = NULL
00401020   .  E8 8F040000    CALL <JMP.&USER32.DialogBoxParamA>   DialogBoxParamA
00401025   .  50             PUSH EAX                        ExitCode
00401026   .  E8 9B040000    CALL <JMP.&KERNEL32.ExitProcess>   ExitProcess
0040102B   .  55             PUSH EBP
0040102C   .  8BEC           MOV EBP,ESP
0040102E   .  817D 0C 10010  CMP DWORD PTR [EBP+C],110
```

این یک برنامه نوشته استاندارد نوشته شده با Cیا Cpp است،همانطور که متوجه شدید این برنامه برای پنجره ی اصلی خود از dialog box استفاده کرده است،در ابتدا آرگومانهای مربوط به تابع DialogBoxParamA درون پشته قرار گرفته شده اند و بعد تابع DialogBoxParamA فراخوانی شده است.بیایید به جزئیات تابع DialogBoxParamA نگاهی بیندازیم :

Win32 Programmer's Reference

File Edit Bookmark Options Help

Contents Index Back << >>

DialogBoxParam Quick Info Overview Group

The **DialogBoxParam** function creates a modal dialog box from a dialog box template resource. Before displaying the dialog box, the function passes an application-defined value to the dialog box procedure as the *lParam* parameter of the WM_INITDIALOG message. An application can use this value to initialize dialog box controls.

```
int DialogBoxParam(
  HINSTANCE hInstance,        // handle to application instance
  LPCTSTR lpTemplateName,     // identifies dialog box template
  HWND hWndParent,            // handle to owner window
  DLGPROC lpDialogFunc,       // pointer to dialog box procedure
  LPARAM dwInitParam          // initialization value
);
```

Parameters

hInstance
Identifies an instance of the module whose executable file contains the dialog box template.

lpTemplateName
Identifies the dialog box template. This parameter is either the pointer to a null-terminated character string that specifies the name of the dialog box template or an integer value that specifies the resource identifier of the dialog box template. If the parameter specifies a resource identifier, its high-order word must be zero and its low-order word must contain the identifier. You can use the MAKEINTRESOURCE macro to create this value.

hWndParent
Identifies the window that owns the dialog box.

lpDialogFunc
Points to the dialog box procedure. For more information about the dialog box procedure, see the DialogProc callback function.

dwInitParam
Specifies the value to pass to the dialog box in the *lParam* parameter of the WM_INITDIALOG message.

Return Values

If the function succeeds, the return value is the value of the *nResult* parameter specified in the call to the **EndDialog** function used to terminate the dialog box.

If the function fails, the return value is –1.

Remarks

The **DialogBoxParam** function uses the **CreateWindowEx** function to create the dialog box. **DialogBoxParam** then

چیزی ک در اینجا برای ما اهمیت ویژه ایی دارد آدرس فراخوانی DLGPROCاست یعنی آدرس callback در برنامه ی ما 0040102B،که تمام پیام های ما را هندل میکند،به پنجره ی disassembly در دیباگر خود نگاه کنید :

```
<ModuleEn  $  6A 00          PUSH 0                                  pModule = NULL
00401002   .  E8 C5040000    CALL <JMP.&KERNEL32.GetModuleHandleA>   GetModuleHandleA
00401007   .  A3 28304000    MOV DWORD PTR [403028],EAX
0040100C   .  6A 00          PUSH 0                                  lParam = NULL
0040100E   .  68 2B104000    PUSH crackme1.0040102B                  DlgProc = crackme1.0040102B
00401013   .  6A 00          PUSH 0                                  hOwner = NULL
00401015   .  68 20030000    PUSH 320                                pTemplate = 320
0040101A   .  FF35 28304000  PUSH DWORD PTR [403028]                 hInst = NULL
00401020   .  E8 8F040000    CALL <JMP.&USER32.DialogBoxParamA>      DialogBoxParamA
00401025   .  50             PUSH EAX                                ExitCode
00401026   .  E8 9B040000    CALL <JMP.&KERNEL32.ExitProcess>        ExitProcess
0040102B   .  55             PUSH EBP
0040102C   .  8BEC           MOV EBP,ESP
0040102E   .  817D 0C 10010  CMP DWORD PTR [EBP+C],110
```

Main Dialog Callback Message Handler

بسیار خوب به آدرس 0040102B (شروع پردازش پیام ها DlgProc) بروید:

```
00401025   .  50             PUSH EAX                                ExitCode = 761F3388
00401026   .  E8 9B040000    CALL <JMP.&KERNEL32.ExitProcess>        ExitProcess
0040102B   .  55             PUSH EBP
0040102C   .  8BEC           MOV EBP,ESP
0040102E   .  817D 0C 10010000 CMP [ARG.2],110
00401035   .v 75 37          JNZ SHORT crackme1.0040106E
00401037   .  C705 48304000 00( MOV DWORD PTR DS:[403048],0
00401041   .  C705 38304000 ADD MOV DWORD PTR DS:[403038],0DEAD
0040104B   .  C705 3C304000 ADD MOV DWORD PTR DS:[40303C],0DEAD
00401055   .  C705 40304000 42( MOV DWORD PTR DS:[403040],42424242
0040105F   .  C705 4C304000 00( MOV DWORD PTR DS:[40304C],crackme1.0040  ASCII "An error occured"
00401069   .v E9 DE010000    JMP crackme1.0040124C
0040106E   >  837D 0C 10     CMP [ARG.2],10
00401072   .v 75 0D          JNZ SHORT crackme1.00401081
00401074   .  FF75 08        PUSH [ARG.1]                            hWnd = 7EFDE000
00401077   .  E8 32040000    CALL <JMP.&USER32.DestroyWindow>        DestroyWindow
0040107C   .v E9 CB010000    JMP crackme1.0040124C
00401081   >  817D 0C 11010000 CMP [ARG.2],111
00401088   .v 0F85 B5010000  JNZ crackme1.00401243
0040108E   .  8B45 10        MOV EAX,[ARG.3]
00401091   .  8B55 10        MOV EDX,[ARG.3]
00401094   .  C1EA 10        SHR EDX,10
00401097   .  66:0BD2        OR DX,DX
0040109A   .v 0F85 AC010000  JNZ crackme1.0040124C
004010A0   .  66:83F8 65     CMP AX,65
004010A4   .v 75 0C          JNZ SHORT crackme1.004010B2
004010A6   .  6A 01          PUSH 1
004010A8   .  E8 F8010000    CALL crackme1.004012A5
004010AD   .v E9 4001000     JMP crackme1.004011F2
004010B2   >  66:83F8 66     CMP AX,66
004010B6   .v 75 0C          JNZ SHORT crackme1.004010C4
004010B8   .  6A 02          PUSH 2
004010BA   .  E8 E6010000    CALL crackme1.004012A5
004010BF   .v E9 2E010000    JMP crackme1.004011F2
004010C4   >  66:83F8 67     CMP AX,67
004010C8   .v 75 0C          JNZ SHORT crackme1.004010D6
004010CA   .  6A 03          PUSH 3
004010CC   .  E8 D4010000    CALL crackme1.004012A5
004010D1   .v E9 1C010000    JMP crackme1.004011F2
004010D6   >  66:83F8 68     CMP AX,68
004010DA   .v 75 0C          JNZ SHORT crackme1.004010E8
004010DC   .  6A 04          PUSH 4
004010DE   .  E8 C2010000    CALL crackme1.004012A5
004010E3   .v E9 0A010000    JMP crackme1.004011F2
004010E8   >  66:83F8 69     CMP AX,69
004010EC   .v 75 0C          JNZ SHORT crackme1.004010FA
004010EE   .  6A 05          PUSH 5
004010F0   .  E8 B0010000    CALL crackme1.004012A5
004010F5   .v E9 F8000000    JMP crackme1.004011F2
004010FA   >  66:83F8 6A     CMP AX,6A
004010FE   .v 75 0C          JNZ SHORT crackme1.0040110C
00401100   .  6A 06          PUSH 6
00401102   .  E8 9E010000    CALL crackme1.004012A5
00401107   .v E9 E6000000    JMP crackme1.004011F2
0040110C   >  66:83F8 6B     CMP AX,6B
00401110   .v 75 0C          JNZ SHORT crackme1.0040111E
00401112   .  6A 07          PUSH 7
00401114   .  E8 8C010000    CALL crackme1.004012A5
00401119   .v E9 D4000000    JMP crackme1.004011F2
0040111E   >  66:83F8 6C     CMP AX,6C
00401122   .v 75 0C          JNZ SHORT crackme1.00401130
00401124   .  6A 08          PUSH 8
00401126   .  E8 7A010000    CALL crackme1.004012A5
0040112B   .v E9 C2000000    JMP crackme1.004011F2
00401130   >  66:83F8 6D     CMP AX,6D
00401134   .v 75 0C          JNZ SHORT crackme1.00401142
00401136   .  6A 09          PUSH 9
00401138   .  E8 68010000    CALL crackme1.004012A5
0040113D   .v E9 B0000000    JMP crackme1.004011F2
00401142   >  66:83F8 6E     CMP AX,6E
00401146   .v 75 0C          JNZ SHORT crackme1.00401154
00401148   .  6A 0A          PUSH 0A
0040114A   .  E8 56010000    CALL crackme1.004012A5
0040114F   .v E9 9E000000    JMP crackme1.004011F2
00401154   >  66:83F8 6F     CMP AX,6F
00401158   .v 75 0C          JNZ SHORT crackme1.00401166
0040115A   .  6A 0B          PUSH 0B
0040115C   .  E8 44010000    CALL crackme1.004012A5
00401161   .v E9 8C000000    JMP crackme1.004011F2
00401166   >  66:83F8 70     CMP AX,70
```

همانطور که میبینید تعداد سویچ ها (شرط ها)زیاد است ، چیزی شبیه به آموزش قبلی ،با این تفاوت که در اینجا چیزی به نام Case Lable مثله (WM_TIMER Case 113) نداریم،این زیربرنامه ایی که میبینید صرفا فقط برای پاسخ با توجه به نوع پیام است ،این زیربرنامه با توجه به نوع message ID که ویندوز برای برنامه میفرستد مقایسه هایی انجام میدهد اگر پیام ارسالی مچ باشد کد مخصوص آن قسمت اجرا خواهد شد و اگر کد (پیام)در لیست هندل ما نباشد این کار را ویندوز انجام خواهد داد،حال برنامه را تحت دیباگر اجرا کنید:

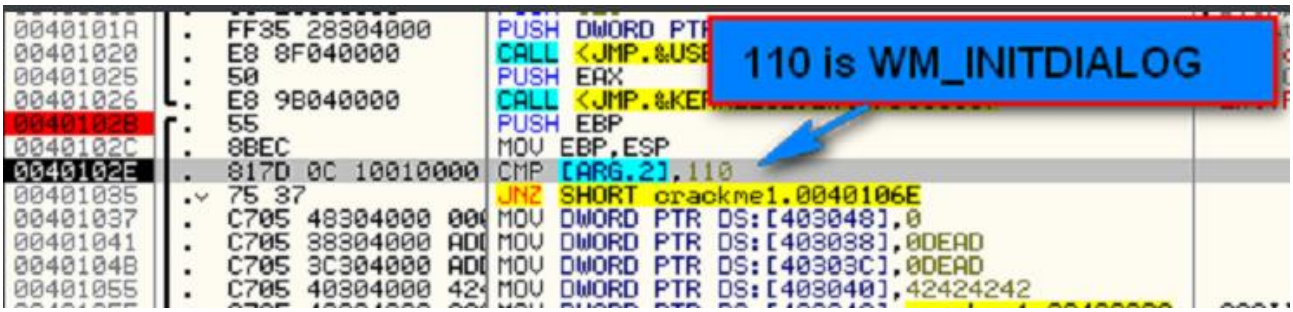
کرک می عجیبی است ،با دکمه ها کار کنید و روی آن کلیک کنید ،به زودی خواهید دید که هیچ اتفاق خاصی ظاهرا نمی افتد.

بروی آدرس 40102B (DlgProc) یک BP بگذارید و برنامه را ری- استارت کنید، یک بار F9 را بفشارید تا دیباگر در آدرس فوق متوقف شود:

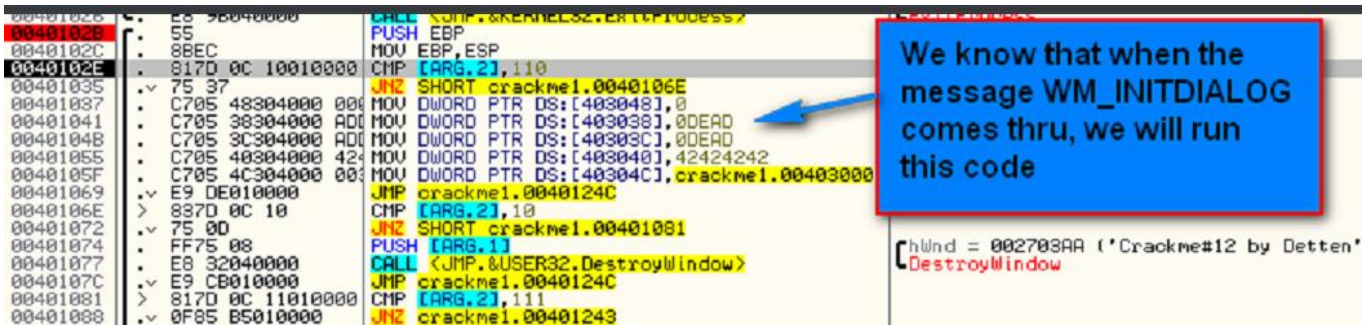
به اولین مقایسه نگاه کنید :

40102E CMP [ARG.2], 110

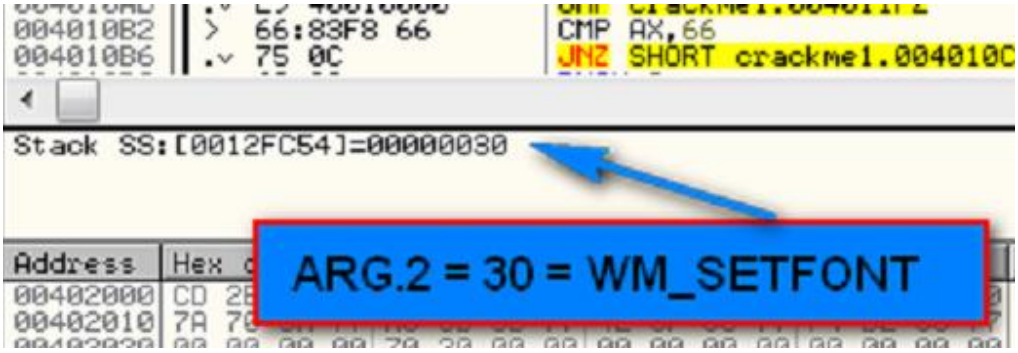
اگر به لیست پیامهای ضمیمه شده نگاه کنید میبینید شناسه ی110مربوط به پیام WM_InitDialog است:



این پیام اجازه ی آماده سازی یک سری کارها را به برنامه ی ما میدهد،و نتیجه مقایسه:



اما به پنجره ی میانی (Info Area) دیباگر را نگاه کنید مقدار ARG.2 برابر با 30 است نه 110:



در تصویر بالا مقدار 30 درواقع شناسه ی WM_SetFont است.و همچنین این اولین پیام فرستاده شده است ، مقایسه ی بعدی با شناسه ی 10 صورت میگیرد :

```
00401028  .  55                 PUSH EBP
0040102C  .  8BEC               MOV EBP,ESP
0040102E  .  817D 0C 10010000   CMP [ARG.2],110
00401035  .v 75 37              JNZ SHORT crackme1.0040106E
00401037  .  C705 48304000 000  MOV DWORD PTR DS:[403048],0
00401041  .  C705 38304000 ADE  MOV DWORD PTR DS:[403038],0DEAD
0040104B  .  C705 3C304000 ADE  MOV DWORD PTR DS:[40303C],0DEAD
00401055  .  C705 40304000 424  MOV DWORD PTR DS:[403040],42424242
0040105F  .  C705 4C304000 003  MOV DWORD PTR DS:[40304C],crackme1.00403000   ASCII "An error occured"
00401069  .v E9 DE010000        JMP crackme1.0040[illegible]C
0040106E  >  837D 0C 10         CMP [ARG.2],10
00401072  .v 75 0D              JNZ SHORT crackme1.00401081
00401074  .  FF75 08            PUSH [ARG.1]                                      hWnd = 002703AA ('Crackme#12 by
00401077  .  E8 32040000        CALL <JMP.&USER32.DestroyWindow>                  DestroyWindow
0040107C  .v E9 CB010000        JMP crackme1.0040124C
00401081  >  817D 0C 11010000   CMP [ARG.2],111
00401088  .v 0F85 B5010000      JNZ crackme1.00401243
```

Compare with 10, which is WM_CLOSE

با توجه به لیست پیام های ضمیمه شده شناسه ی 10 مربوط به پیام WM_CLOSE است ، یعنی وقتی بروی دکمه Close در برنامه کلیک شود این کد اجرا میشود

مقایسه بعدی با شناسه ی 111 صورت میگیرد،طبق فایل ضمیمه شده مربوط به پیام WM_COMMAND است:

```
00401074  .  FF75 08            PUSH [ARG.1]                                      hWnd = 00
00401077  .  E8 32040000        CALL <JMP.&USER32.DestroyWindow>                  DestroyWi
0040107C  .v E9 CB010000        JMP crackme1.0040124C
00401081  >  817D 0C 11010000   CMP [ARG.2],111
00401088  .v 0F85 B5010000      JNZ crackme1.00401[illegible]
0040108E  .  8B45 10            MOV EAX,[ARG.3]
00401091  .  8B55 10            MOV EDX,[ARG.3]
00401094  .  C1EA 10            SHR EDX,10
00401097  .  66:0BD2            OR DX,DX
0040109A  .v 0F85 AC010000      JNZ crackme1.00401[illegible]
004010A0  .  66:83F8 65         CMP AX,65
004010A4  .v 75 0C              JNZ SHORT crackme1.004010B2
004010A6  .  6A 01              PUSH 1
004010A8  .  E8 F8010000        CALL crackme1.004012A5
004010AD  .v E9 40010000        JMP crackme1.004011F2
004010B2  >  66:83F8 66         CMP AX,66
004010B6  .v 75 0C              JNZ SHORT crackme1.004010C4
004010B8  .  6A 02              PUSH 2
004010BA  .  E8 E6010000        CALL crackme1.004012A5
004010BF  .v E9 2E010000        JMP crackme1.004011F2
004010C4  >  66:83F8 67         CMP AX,67
004010C8  .v 75 0C              JNZ SHORT crackme1.004010D6
004010CA  .  6A 03              PUSH 3
004010CC  .  E8 D4010000        CALL crackme1.004012A5
004010D1  .v E9 1C010000        JMP crackme1.004011F2
004010D6  >  66:83F8 68         CMP AX,68
004010DA  .v 75 0C              JNZ SHORT crackme1.004010E8
004010DC  .  6A 04              PUSH 4
004010DE  .  E8 C2010000        CALL crackme1.004012A5
004010E3  .v E9 0A010000        JMP crackme1.004011F2
004010E8  >  66:83F8 69         CMP AX,69
004010EC  .v 75 0C              JNZ SHORT crackme1.004010FA
004010EE  .  6A 05              PUSH 5
004010F0  .  E8 B0010000        CALL crackme1.004012A5
004010F5  .v E9 F8000000        JMP crackme1.004011F2
004010FA  >  66:83F8 6A         CMP AX,6A
004010FE  .v 75 0C              JNZ SHORT crackme1.0040110C
```

111 = WM_COMMAND

WM_COMMAND چند کاره است اما معمولا به منابع متصل میشود برای مثال :کلیک بروی دکمه ها،انتخاب منوها،یا کلیک کردن روی آیکن های تولبار،توجه داشته باشید که ARG ها در اینجا یک مقدار صحیح هستند،اگر بروی یک دکمه کلیک کنید WM_COMMAND مقدار ARG.3 را با مقدار شناسه ی دکمه ی کلیک شده ارزش گذاری میکند.

و اگر موس را حرکت دهیدWM_COMMAND مقدار ARG.3 رابا مقدار X,Y ارزش گذاری میکند که نمایانگر محل فعلی موس است

```
00401081  >  817D 0C 11010000  CMP [ARG.2],111
00401088  .v 0F85 B5010000     JNZ crackme1.00401243
0040108E  .  8B45 10           MOV EAX,[ARG.3]
00401091  .  8B55 10           MOV EDX,[ARG.3]
00401094  .  C1EA 10           SHR EDX,10
00401097  .  66:0BD2           OR DX,DX
0040109A  .v 0F85 AC010000     JNZ crackme1.0040124C
004010A0  .  66:83F8 65        CMP AX,65
004010A4  .v 75 0C             JNZ SHORT crackme1.004010B2
004010A6  .  6A 01             PUSH 1
004010A8  .  E8 F8010000       CALL crackme1.004012A5
004010AD  .v E9 40010000       JMP crackme1.004011F2
004010B2  >  66:83F8 66        CMP AX,66
004010B6  .v 75 0C             JNZ SHORT crackme1.004010C4
004010B8  .  6A 02             PUSH 2
004010BA  .  E8 E6010000       CALL crackme1.004012A5
004010BF  .v E9 2E010000       JMP crackme1.004011F2
004010C4  >  66:83F8 67        CMP AX,67
004010C8  .v 75 0C             JNZ SHORT crackme1.004010D6
004010CA  .  6A 03             PUSH 3
004010CC  .  E8 D4010000       CALL crackme1.004012A5
004010D1  .v E9 1C010000       JMP crackme1.004011F2
004010D6  >  66:83F8 68        CMP AX,68
004010DA  .v 75 0C             JNZ SHORT crackme1.004010E8
004010DC  .  6A 04             PUSH 4
004010DE  .  E8 C2010000       CALL crackme1.004012A5
004010E3  .v E9 0A010000       JMP crackme1.004011F2
004010E8  >  66:83F8 69        CMP AX,69
004010EC  .v 75 0C             JNZ SHORT crackme1.004010FA
004010EE  .  6A 05             PUSH 5
004010F0  .  E8 B0010000       CALL crackme1.004012A5
004010F5  .v E9 F8000000       JMP crackme1.004011F2
004010FA  >  66:83F8 6A        CMP AX,6A
004010FE  .v 75 0C             JNZ SHORT crackme1.0040110C
00401100  .  6A 06             PUSH 6
00401102  .  E8 9E010000       CALL crackme1.004012A5
00401107  .v E9 E6000000       JMP crackme1.004011F2
0040110C  >  66:83F8 6B        CMP AX,6B
```

با دقت نگاه کنید ،ما WM_COMMAND را با تعدادی پیام در زیر میبینیم ،اما اگر یکبار F8 را بفشارید ما به انتهای زیر برنامه هدایت میشویم این به این معنی است که این پیام (SetFont) را ما هندل نکرده ایم بلکه این کار را ویندوز برای ما هندل میکند:

```
00401228  >  68 11304000        PUSH crackme1.004030 11
0040122D  .  6A 03              PUSH 3
0040122F  .  FF75 08            PUSH [ARG.1]
00401232  .  E8 89020000        CALL <JMP.&USER32.SetDlgItemTextA>
00401237  .  C705 44304000 000  MOV DWORD PTR DS:[403044],0
00401241  >v EB 09              JMP SHORT crackme1.0040124C
00401243  >  B8 00000000        MOV EAX,0
00401248  .  C9                 LEAVE
00401249  .  C2 1000            RETN 10
0040124C  >  B8 01000000        MOV EAX,1
```

دوباره F9 را بفشارید تا دیباگر در آدرس 40102B (بروی پیام بعدی)متوقف شود:

```
00401026  L. E8 9B040000        CALL <JMP.&KERNEL32.ExitProcess>
0040102B  r. 55                 PUSH EBP
0040102C  . 8BEC                MOV EBP,ESP
0040102E  . 817D 0C 10010000    CMP [ARG.2],110
00401035  .v 75 37              JNZ SHORT crackme1.0040106E
00401037  . C705 48304000 000   MOV DWORD PTR DS:[403048],0
00401041  . C705 38304000 ADD   MOV DWORD PTR DS:[403038],0DEAD
0040104B  . C705 3C304000 ADD   MOV DWORD PTR DS:[40303C],0DEAD
00401055  . C705 40304000 424   MOV DWORD PTR DS:[403040],42424242
0040105F  . C705 4C304000 003   MOV DWORD PTR DS:[40304C],crackme1.00403000
00401069  .v E9 DE010000        JMP crackme1.0040124C
0040106E  > 837D 0C 10          CMP [ARG.2],10
00401072  .v 75 0D              JNZ SHORT crackme1.00401081
00401074  . FF75 08             PUSH [ARG.1]
00401077  . E8 32040000         CALL <JMP.&USER32.DestroyWindow>
0040107C  .v E9 CB010000        JMP crackme1.0040124C
00401081  > 817D 0C 11010000    CMP [ARG.2],111
00401088  .v 0F85 B5010000      JNZ crackme1.00401243
0040108E  . 8B45 10             MOV EAX,[ARG.3]
00401091  . 8B55 10             MOV EDX,[ARG.3]
00401094  . C1EA 10             SHR EDX,10
00401097  . 66:0BD2             OR DX,DX
0040109A  .v 0F85 AC010000      JNZ crackme1.0040124C
004010A0  . 66:83F8 65          CMP AX,65
004010A4  .v 75 0C              JNZ SHORT crackme1.004010B2
004010A6  . 6A 01               PUSH 1
004010A8  . E8 F8010000         CALL crackme1.004012A5
004010AD  .v E9 40010000        JMP crackme1.004011F2
004010B2  > 66:83F8 66          CMP AX,66
004010B6  .v 75 0C              JNZ SHORT crackme1.004010C4

Stack SS:[0012F644]=00000111
```

این بار شناسه ی مربوط به پیام WM_COMMAND را میبینید یعنی (111) با کلید F8 ادامه دهید تا به آدرس 401088 برسید،بیایید دقیق تر به مسئله نگاه کنیم :

```
00401074  . FF75 08             PUSH [ARG.1]
00401077  . E8 32040000         CALL <JMP.&USER32.DestroyWindow>
0040107C  .v E9 CB010000        JMP crackme1.0040124C
00401081  > 817D 0C 11010000    CMP [ARG.2],111
00401088  .v 0F85 B5010000      JNZ crackme1.00401243
0040108E  . 8B45 10             MOV EAX,[ARG.3]
00401091  . 8B55 10             MOV EDX,[ARG.3]
00401094  . C1EA 10             SHR EDX,10
00401097  . 66:0BD2             OR DX,DX
0040109A  .v 0F85 AC010000      JNZ crackme1.0040124C
004010A0  . 66:83F8 65          CMP AX,65
004010A4  .v 75 0C              JNZ SHORT crackme1.004010B2
004010A6  . 6A 01               PUSH 1
004010A8  . E8 F8010000         CALL crackme1.004012A5
004010AD  .v E9 40010000        JMP crackme1.004011F2
004010B2  > 66:83F8 66          CMP AX,66
004010B6  .v 75 0C              JNZ SHORT crackme1.004010C4
004010B8  . 6A 02               PUSH 2
004010BA  . E8 E6010000         CALL crackme1.004012A5
004010BF  .v E9 2E010000        JMP crackme1.004011F2
004010C4  > 66:83F8 67          CMP AX,67
004010C8  .v 75 0C              JNZ SHORT crackme1.004010D6
004010CA  . 6A 03               PUSH 3
004010CC  . E8 D4010000         CALL crackme1.004012A5
004010D1  .v E9 1C010000        JMP crackme1.004011F2
004010D6  > 66:83F8 68          CMP AX,68
004010DA  .v 75 0C              JNZ SHORT crackme1.004010E8
004010DC  . 6A 04               PUSH 4
004010DE  . E8 C2010000         CALL crackme1.004012A5
004010E3  .v E9 0A010000        JMP crackme1.004011F2
004010E8  > 66:83F8 69          CMP AX,69
```

در خط 40108E، ARG.3 درون ثبات EAXوEDX ریخته میشود، سپس بروی ثبات EDX ،به مقدار 10یا 16 در مد دسیمال شیفت به راست صورت میگیرد،و در خط بعدی این مقدار OR میشود و اگر نتیجه صفر نباشد ،پرش صورت میگیرد،در واقع 5بیت از این آرگمان چک میشود که آیا صفر است یا نه

```
0040107C  .v E9 CB010000     JMP crackme1.0040
00401081  > 817D 0C 11010000 CMP [ARG.2],111
00401088  .v 0F85 B5010000   JNZ crackme1.00401
0040108E  . 8B45 10          MOV EAX,[ARG.3]
00401091  . 8B55 10          MOV EDX,[ARG.3]
00401094  . C1EA 10          SHR EDX,10
00401097  . 66:0BD2          OR DX,DX
0040109A  .v 0F85 AC010000   JNZ crackme1.00401
004010A0  . 66:83F8 65       CMP AX,65
004010A4  .v 75 0C           JNZ SHORT crackme1.004010B2
004010A6  . 6A 01            PUSH 1
004010A8  . E8 F8010000      CALL crackme1.004012A5
004010AD  .v E9 40010000     JMP crackme1.004011F2
004010B2  > 66:83F8 66       CMP AX,66
004010B6  .v 75 0C           JNZ SHORT crackme1.004010C4
```

We are not dealing with this message, so we jump to end

درواقع بیت های بالایی ثبات DX که ایدی (شناسه) منابع دراینجا تحت تاثیر است، که در اینجا مقدار آن صفر است بنابراین این پرش صورت میگیردو اگر به قسمت info area نگاه کنید خواهید دید:

```
004010DA  .v 75 0C           JNZ SHORT crackm
004010DC  . 6A 04            PUSH 4

Stack SS:[0012F644]=00000111

Address   Hex dump
00402000  CD 2B 12 76 F3 D8 12 76 E2 BB 13 76
```

بله پیام WM_COMMAND با شناسه ی 111 ،و در نهایت پرش:

```
00401081  > 817D 0C 11010000 CMP [ARG.2],111
00401088  .v 0F85 B5010000   JNZ crackme1.00401243
0040108E  . 8B45 10          MOV EAX,[ARG.3]
00401091  . 8B55 10          MOV EDX,[ARG.3]
00401094  . C1EA 10          SHR EDX,10
00401097  . 66:0BD2          OR DX,DX
0040109A  .v 0F85 AC010000   JNZ crackme1.0040124C
004010A0  . 66:83F8 65       CMP AX,65
004010A4  .v 75 0C           JNZ SHORT crackme1.004010B2
004010A6  . 6A 01            PUSH 1
004010A8  . E8 F8010000      CALL crackme1.004012A5
004010AD  .v E9 40010000     JMP crackme1.004011F2
004010B2  > 66:83F8 66       CMP AX,66
004010B6  .v 75 0C           JNZ SHORT crackme1.004010C4
004010B8  . 6A 02            PUSH 2
004010BA  . E8 E6010000      CALL crackme1.004012A5
004010BF  .v E9 2E010000     JMP crackme1.004011F2
004010C4  > 66:83F8 67       CMP AX,67
004010C8  .v 75 0C           JNZ SHORT crackme1.004010D6
004010CA  . 6A 03            PUSH 3
004010CC  . E8 D4010000      CALL crackme1.004012A5
004010D1  .v E9 1C010000     JMP crackme1.004011F2
004010D6  > 66:83F8 68       CMP AX,68
004010DA  .v 75 0C           JNZ SHORT crackme1.004010E8
004010DC  . 6A 04            PUSH 4
004010DE  . E8 C2010000      CALL crackme1.004012A5
004010E3  .v E9 0A010000     JMP crackme1.004011F2
004010E8  > 66:83F8 69       CMP AX,69
004010EC  .v 75 0C           JNZ SHORT crackme1.004010FA
004010EE  . 6A 05            PUSH 5
004010F0  . E8 B0010000      CALL crackme1.004012A5
004010F5  .v E9 F8000000     JMP crackme1.004011F2
004010FA  > 66:83F8 6A       CMP AX,6A
004010FE  .v 75 0C           JNZ SHORT crackme1.0040110C
00401100  . 6A 06            PUSH 6
```

برنامه با F9 یکبار اجرا کنید میبینید که دوباره بروی BP خودمان متوقف میشویم و متوجه پیام WM_INITDIALOG میشویم:

```
004010D6  >  66:83F8 68     CMP AX,68
004010DA  .v 75 0C          JNZ SHORT
004010DC  .  6A 04          PUSH 4

Stack SS:[0012FC54]=00000110

Address   Hex dump
00402000  CD 2B 12 76 F3 D8 12 76 E2 BB
```

دو خط پایین تر بخشی از مقدار دهی اولیه دیالوگ باکس را میبینیم:

```
0040102B  .  55              PUSH EBP
0040102C  .  8BEC            MOV EBP,ESP
0040102E  .  817D 0C 10010000 CMP [ARG.2],110
00401035  .v 75 37           JNZ SHORT crackme1.0040106E
00401037  .  C705 48304000 00 MOV DWORD PTR DS:[403048],0
00401041  .  C705 38304000 AD MOV DWORD PTR DS:[403038],0DEAD
0040104B  .  C705 3C304000 AD MOV DWORD PTR DS:[40303C],0DEAD
00401055  .  C705 40304000 42 MOV DWORD PTR DS:[403040],42424242
0040105F  .  C705 4C304000 00 MOV DWORD PTR DS:[40304C],crackme1.00403000
00401069  .v E9 DE010000     JMP crackme1.0040124C
0040106E  >  837D 0C 10      CMP [ARG.2],10

It's 110 (WM_INITDIALOG)
so we run this code

ASCII "An error occured"
```

در این قسمت اتفاقات مهمی رخ میدهد اگر به تصویر بالا نگاه کنید چند متغییر صحیح درون حافظه با آدرس 403038 میبینید به پنجره ی window dumpنگاه کنید چه اتفاقی رخ میدهد :

```
Address   Hex dump                                         ASCII
00403038  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00403048  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00403058  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00403068  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00403078  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00403088  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00403098  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
004030A8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
004030B8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
004030C8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
004030D8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
004030E8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
004030F8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00403108  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00403118  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00403128  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00403138  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

در ابتدا ما میبینیم که با مقدار صفر ، مقدار دهی اولیه میشود، حال دستور mov را (آدرس 401037) را اجرا کنید (F8):

```
00401037  .  C705 48304000 00[  MOV DWORD PTR DS:[403048],0
00401041  .  C705 38304000 AD[  MOV DWORD PTR DS:[403038],0DEAD
0040104B  .  C705 3C304000 AD[  MOV DWORD PTR DS:[40303C],0DEAD
00401055  .  C705 40304000 42[  MOV DWORD PTR DS:[403040],42424242
0040105F  .  C705 4C304000 00[  MOV DWORD PTR DS:[40304C],crackme1.00403000
```

،حالا به پنجره ی windo dumpنگاه کنید میبینید در آدرس 403038 مقدار صفر کپی میشود،حال یکبار دیگر F8 را فشارید تا خط بعد اجرا شود و دوباره به پنجره ی Window dump نگاهی بیندازید:

```
Address   Hex dump
00403038  AD DE 00 00 00 00 00 00 00 00 00
00403048  00 00 00 00 00 00 00 00 00 00 00
00403058  00 00 00 00 00 00 00 00 00 00 00
00403068  00 00 00 00 00 00 00 00 00 00 00
00403078  00 00 00 00 00 00 00 00 00 00 00
00403088  00 00 00 00 00 00 00 00 00 00 00
00403098  00 00 00 00 00 00 00 00 00 00 00
004030A8  00 00 00 00 00 00 00 00 00 00 00
```

خب همانطور که واضح است مقدار 0xDEAD در آدرس 403038 کپی شده (به صورت little endian)

دوباره F8 را فشارید تا خط بعد در آدرس 40104B اجرا شود:

```
Address   Hex dump
00403038  AD DE 00 00 AD DE 00 00 00 00 00 00
00403048  00 00 00 00 00 00 00 00 00 00 00 00
00403058  00 00 00 00 00 00 00 00 00 00 00 00
00403068  00 00 00 00 00 00 00 00 00 00 00 00
00403078  00 00 00 00 00 00 00 00 00 00 00 00
00403088  00 00 00 00 00 00 00 00 00 00 00 00
00403098  00 00 00 00 00 00 00 00 00 00 00 00
004030A8  00 00 00 00 00 00 00 00 00 00 00 00
004030B8  00 00 00 00 00 00 00 00 00 00 00 00
```

همانطور که میبینید در تصویر بالا مقدار 0xDEAD در آدرس 40303C کپی شده،در واقع اینها یک مقدار Word در مد هگزا هستند ،در خط بعدی مقدار 42 42 42 42 در آدرس 403040 کپی میشود،که اینها هم در مد هگزا هستند و اگر آنها را به کد اسکی (ASCII) تبدیل کنیم معادل کاراکتر B هستند ، به پنجره ی window dumpنگاه کنید:

```
Address   Hex dump                                          ASCII
00403038  AD DE 00 00 AD DE 00 00 42 42 42 42 00 00 00 00  ↓▌..↓▌..BBBB....
00403048  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00403058  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00403068  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00403078  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00403088  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00403098  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
004030A8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
004030B8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
004030C8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
004030D8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

و در آخر مقدار عدد صحیح 00403000 در آدرس 40304C کپی میشود :

```
Address   Hex dump                                          ASCII
00403038  AD DE 00 00 AD DE 00 00 42 42 42 42 00 00 00 00  ↓▌..↓▌..BBBB....
00403048  00 00 00 00 00 30 40 00 00 00 00 00 00 00 00 00  .....0@.........
00403058  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00403068  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00403078  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00403088  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00403098  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
004030A8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
004030B8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

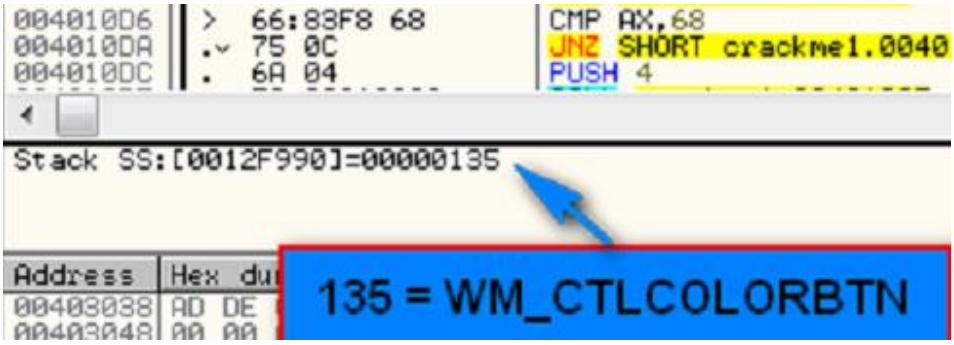
در واقع دیباگر اشاره میکند به آدرس شروع سکشن data ، یعنی همان 00403000 (به صورت little endian)

در پایان در آدرس 401069یک پرش انتهای کد داریم به این معنی که منتظر برای پیام ارسالی میماند :

```
0040102C  .   8BEC              MOV EBP,ESP
0040102E  .   817D 0C 10010000  CMP [ARG.2],110
00401035  .v  75 37             JNZ SHORT crackme1.0040106E
00401037  .   C705 48304000 000 MOV DWORD PTR DS:[403048],0
00401041  .   C705 38304000 ADD MOV DWORD PTR DS:[403038],0DEAD
0040104B  .   C705 3C304000 ADD MOV DWORD PTR DS:[40303C],0DEAD
00401055  .   C705 40304000 424 MOV DWORD PTR DS:[403040],42424242
0040105F  .   C705 4C304000 003 MOV DWORD PTR DS:[40304C],crackme1.00403000
00401069  .v  E9 DE010000       JMP crackme1.0040124C
0040106E  >   837D 0C 10        CMP [ARG.2],10
00401072  .v  75 0D             JNZ SHORT crackme1.00401081
00401074  .   FF75 08           PUSH [ARG.1]
00401077  .   E8 32040000       CALL <JMP.&USER32.DestroyWindow>
0040107C  .v  E9 CB010000       JMP crackme1.0040124C
00401081  >   817D 0C 11010000  CMP [ARG.2],111
00401088  .v  0F85 B5010000     JNZ crackme1.00401243
0040108E  .   8B45 10           MOV EAX,[ARG.3]
00401091  .   8B55 10           MOV EDX,[ARG.3]
00401094  .   C1EA 10           SHR EDX,10
00401097  .   66:0BD2           OR DX,DX
0040109A  .v  0F85 AC010000     JNZ crackme1.0040124C
004010A0  .   66:83F8 65        CMP AX,65
004010A4  .v  75 0C             JNZ SHORT crackme1.004010B2
004010A6  .   6A 01             PUSH 1
004010A8  .   E8 F8010000       CALL crackme1.004012A5
004010AD  .v  E9 40010000       JMP crackme1.004011F2
004010B2  >   66:83F8 66        CMP AX,66
004010B6  .v  75 0C             JNZ SHORT crackme1.004010C4
004010B8  .   6A 02             PUSH 2
004010BA  .   E8 E6010000       CALL crackme1.004012A5
```

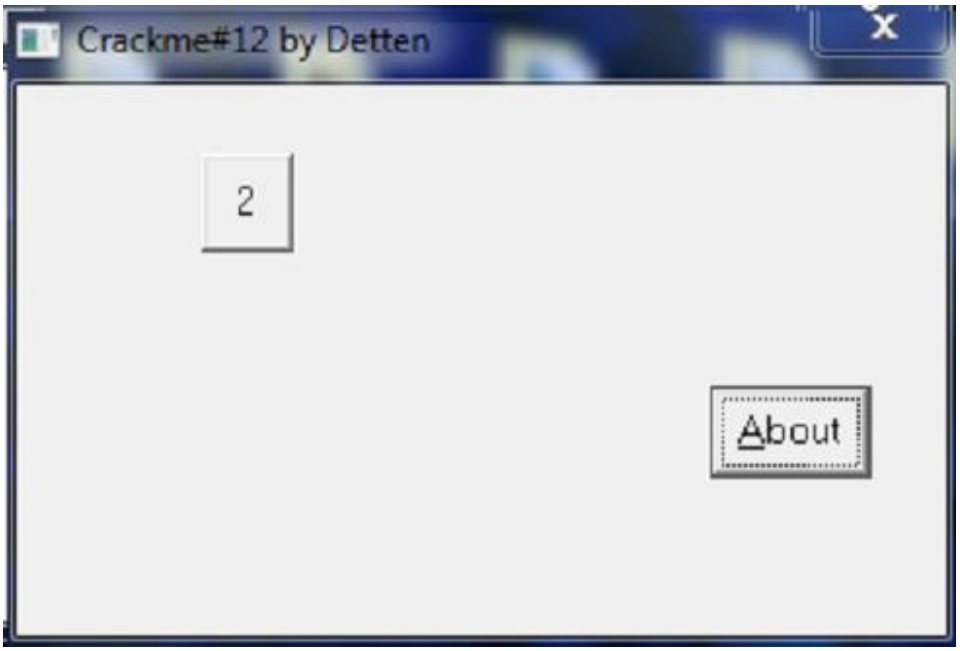
حال اگر کلید F9 را حدود 10 بار یا بیشتر بفشارید ،شما پنجره ی اصلی را میبینید که ساخته شده :

در این قسمت موضوع جالب میشود ،با فشردن هر بار کلید F9 اتفاقی می افتد و اگر 6 بار F9 را بفشارید ،پیام دریافت شده عملیات ترسیم را بروی صفحه اصلی انجام میدهد،این پیام 135 یا WM_CTLCOLORBUTTON است:



که اولین دکمه را در پنجره ی اصلی ترسیم میکند:

پس از فشردن یکبار کلید F9 دکمه بعدی 2 است:



درواقع پس از هربار کلیک F9 مشاهده میکنید که تمام متعلقات پنجره ساخته میشود و روی آن قرار میگیرد،هر دکمه در یک زمان،جالب است نه؟

میتوانید برای توضیحات پیام ها از گوگل استفاده کنید به هر حال در آخر Lable متن "No access" نوشته و در محل خود قرار میگیرد،برای تکمیل این پروسه (ساخت و تکمیل پنجره و متعلقات) حدود 35 بار جمعا باید کلید F9 فشرده شود :

حالا شما فرایند ساخت دیالوگ باکس را دیدید و با کلی ات آن آشنا شدید.همانطور که گفته شد callback functionتمام پیام های ارسالی از ویندوز رو مدیریت میکند و ویندوز یکی یکی آنها را به callback میدهد بسته به نوع پیام هایی که ما خود هندل کردیم عملیات مورد نظر صورت میگیرد و پس از ساخت کامل صفحه ما ، ویندوز درون یک حلقه قرار میگیرد و منتظر عملیات درخواستی ما میماند به محض درخواست ما پیام به callbackمافرستاده میشود (با یک شناسه) که مشخص کننده نوع آیتم مورد نظر است که عملیاتی رو آن صورت گرفته است ،البته یادتان باشد که وقتی برنامه ایی را درون olly دیباگ میکنید حتی حرکت دادن ماوس باعث توقف آن میشود که به معنی آغاز یک پیام است،در واقع شما هرکاری با پنجره و متعلقات آن انجام دهید تولید یک پیام میکند و آن را به callback ما ارسال میکند.

تمرین (مشق ☺)

1-امتحان کنید ببینید میتوانید کشف کنید که بعد از کلیک بروی یک دکمه چه اتفاقی می افتد مخصوصا در محتویات حافظه؟ از آدرس 403038 شروع کنید ،آیا دکمه های مختلف کارهای یکسانی انجام میدهند؟ایا مقادیر آدرس حافظه 403038 ویرایش میشوند؟

2-ببینید میتوانید بفهمید طول پسورد چند رقم است؟

تا آموزش بعدی....