Window subclassing

Vložil/a RubberDuck [1], 11 Říjen, 2012 - 13:20

- Programming [2]

Snad téměř každý si při práci v nějakém programu s GUI posteskl, že mu v menu chybí nějaká volba. Nebo v programu, kde by měl uživatel do textového políčka zadávat pouze čísla může zadávat libovolné znaky.

Co teď s tím? V zásadě máme tři hlavní možnosti:
i. nechat to tak a spoléhat se, že při zadání neplatného vstupu aplikace nespadne
ii. upravit fyzicky kód a dopsat do něj požadovaný kód
iii. využít možností Window subclassingu

Ad i. nechat to tak a spoléhat se, že při zadání neplatného vstupu aplikace nespadne
Nejpohodlnější řešení, které dokáže realizovat každý člověk ;) Pro nás nevhodné.

Ad ii. upravit fyzicky kód a dopsat do něj požadovaný kód
Poměrně náročný úkol s nejistým výsledkem. Pro nás použitelné jen tehdy, pokud jsme sadomasochisti nebo zručtí reverzní inženýři a za předpokladu, že aplikaci nebudeme chtít více updatovat z oficiálních zdrojů.

Ad iii. využít možností Window subclassingu
Jednoduché, přímočaré, (téměř vždy) snadno realizovatelné řešení. Nejlepší cesta.

Co je to Window subclassing
Window subclassing je metoda stará jako okno samotné. Tato technika měla umožnit pohodlněji upravovat aplikace bez nutnosti zasahovat do jejich kódu. Ale jak už to bývá, ajťák je tvor hloubavý a tak netrvalo dlouho a zjistil, že window subclassing je možné využít i k jiným účelům než ke kterým byl vytvořen. Pojďme se podívat na celý problém blíže. V operačním systému Windows má každá okenní aplikace tzv. 'třídu okna (Window Class)'. Tato 'třída okna' popisuje, jak bude okno po spuštění programu vypadat. Pro tyto účely slouží struktura WNDCLASS, respektive WNDCLASSEX.

```
typedef struct tagWNDCLASSEX {
  UINT      cbSize;
  UINT      style;
  WNDPROC   lpfnWndProc;
  int       cbClsExtra;
  int       cbWndExtra;
  HINSTANCE hInstance;
  HICON     hIcon;
  HCURSOR   hCursor;
  HBRUSH    hbrBackground;
  LPCTSTR   lpszMenuName;
  LPCTSTR   lpszClassName;
  HICON     hIconSm;
} WNDCLASSEX, *PWNDCLASSEX;
```

Důležitý je prvek lpfnWndProc, který odkazuje na proceduru okna. Tato procedura je zodpovědná za reakce na události nad okny aplikace a také za zpracování zpráv určených pro danou aplikaci a její okna. Procedura okna je volána pomocí funkce CallWindowProc.

```
LRESULT WINAPI CallWindowProc(
  __in  WNDPROC lpPrevWndFunc,
  __in  HWND hWnd,
  __in  UINT Msg,
  __in  WPARAM wParam,
  __in  LPARAM lParam
);
```

Funkce CallWindowProc bere jako první parametr pointer na předchozí proceduru okna. A zde se dostáváme k podstatě Window Subclassingu. Pokud jsme schopni zjistit adresu originální procedury okna a zajistit spuštění našeho kódu v rámci daného procesu okenní aplikace, můžeme nahradit originální volání okna procedury naším vlastním kódem. Tento náš kód spustí požadovaný kód a zavolá originální proceduru okna, čímž se zachová původní fukčnost okenní aplikace.

Pro demonstraci zvolíme aplikaci Notepad.exe, která je součástí každé verze Windows. Cílem bude přidat do menu tlačítko, které zobrazí MessageBox se zprávou. První věc, kterou musíme udělat, je získání handle okna. Možností je vícero. Například s využitím fukce GetForegroundWindow

```
HWND WINAPI GetForegroundWindow(void);
```

nebo s využitím funkce FindWindow:

```
HWND WINAPI FindWindow(
  __in_opt  LPCTSTR lpClassName,
  __in_opt  LPCTSTR lpWindowName
);
```

Pokud chceme získat hlavní okno procesu, můžeme zavolat funkci GetCurrentProcessId a uložit si ID aktuálního procesu:

```
DWORD WINAPI GetCurrentProcessId(void);
```

následně pomocí funkce EnumWindows procházet všechna okna v rámci daného procesu:

```
BOOL WINAPI EnumWindows(
  __in  WNDENUMPROC lpEnumFunc,
  __in  LPARAM lParam
);
```

a na každé zavolat funkci GetWindowThreadProcessId a porovnat výsledek s uloženým ID aktuálního procesu:

```
DWORD WINAPI GetWindowThreadProcessId(
  __in       HWND hWnd,
  __out_opt  LPDWORD lpdwProcessId
);
```

Pokud budou obě hodnoty stejné, pravděpodobně jsme nalezli hlavní okno procesu.

Nyní potřebujeme získat handle menu aktuálního okna. Nejjednodušší cestou je použít funkci GetMenu:

```
HMENU WINAPI GetMenu(
  __in  HWND hWnd
);
```

Teď již máme vše, co potřebujeme pro přidání nové volby v menu hlavního okna aplikace. Využijeme funkci AppendMenu a přidáme požadovanou volbu:

```
BOOL WINAPI AppendMenu(
  __in      HMENU hMenu,
  __in      UINT uFlags,
  __in      UINT_PTR uIDNewItem,
  __in_opt  LPCTSTR lpNewItem
);
```

Nakonec zaměníme původní proceduru okna za naši novou pomocí funkce SetWindowLong, čímž rozšíříme původní proceduru o nové možnosti - v našem případě nová položka menu a reakce po kliknutí na ni.:

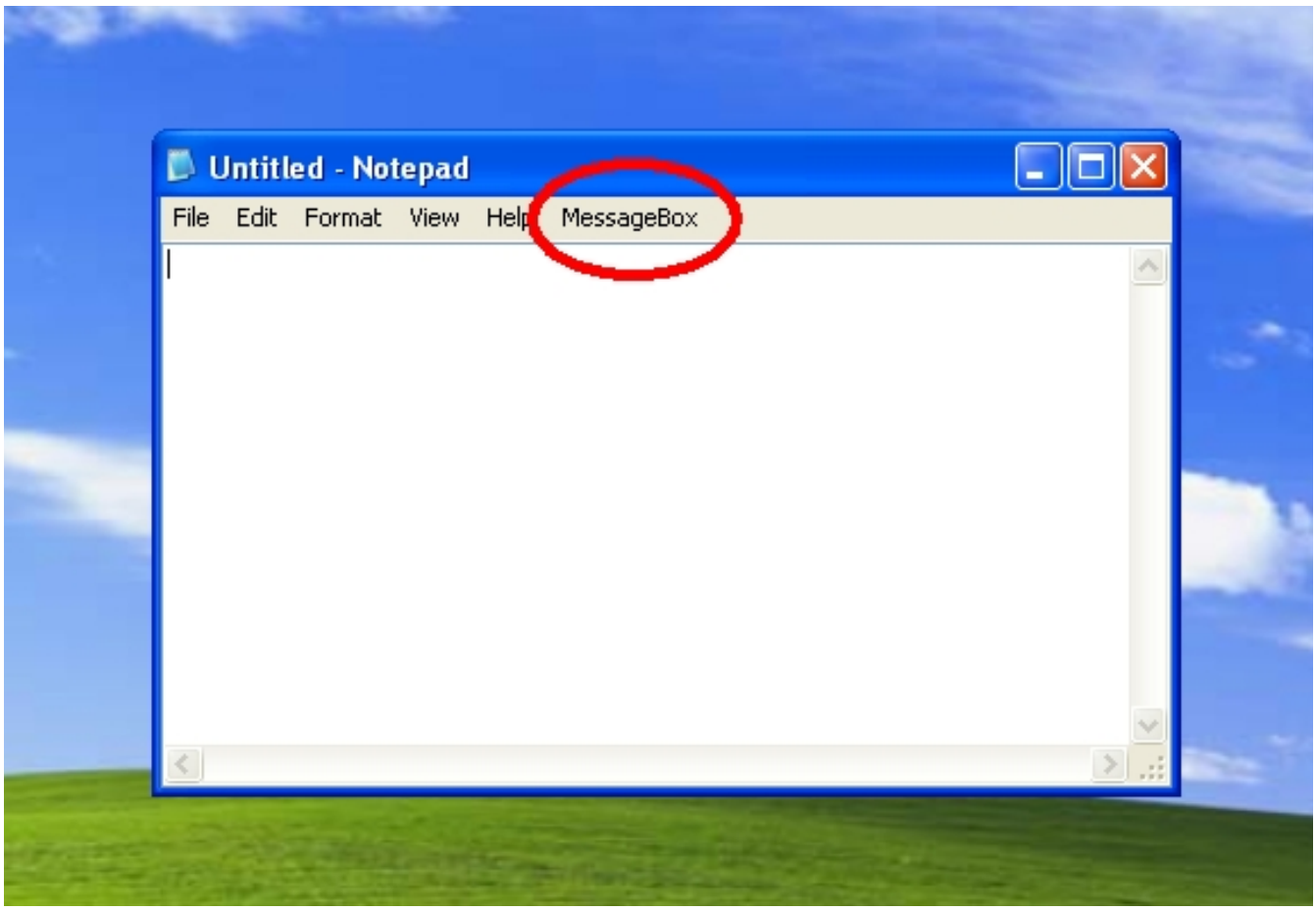
```
LONG WINAPI SetWindowLong(
  __in  HWND hWnd,
  __in  int nIndex,
  __in  LONG dwNewLong
);
```

Samotná procedura okna má pevně danou strukturu:

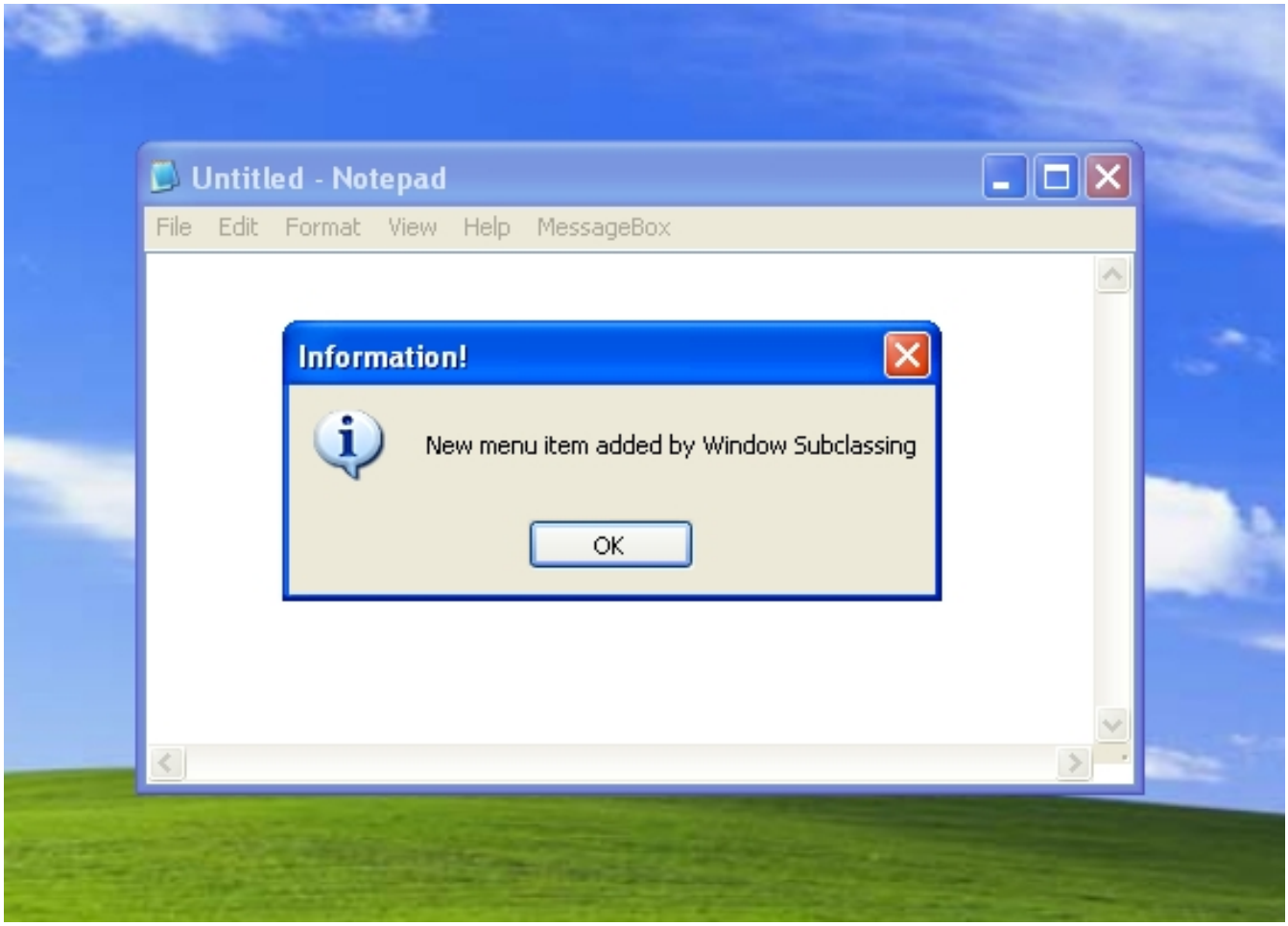
```
LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam){
        switch(iMsg){
                default:
                break;
        }

        return CallWindowProc(hwnd, iMsg, wParam, lParam);
}
```

Nyní nám tedy zbývá podle předchozích kroků sestavit celý kód. Pravděpodobně nejschůdnější cestou pro spuštění tohoto kódu v cizím procesu bude DLL injection.


Untitled - Notepad
File Edit Format View Help MessageBox

[3]


Untitled - Notepad
File Edit Format View Help MessageBox
Information!
New menu item added by Window Subclassing
OK

[4]

Z výše popsaného je jasně patrné, jak jednoduše je možné změnit vzhled nebo chování aplikace. Přidáním nové položkdy do menu vše pouze začíná. Důvtipný čtenář z rukávu vysype hromadu dalších využití.

```
#include <windows.h>

#define MYBUTTON 1337

LONG OriginalProcedure = 0;

LRESULT __stdcall NewProc(HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam);
DWORD __stdcall AddNewMenuItem();

BOOL APIENTRY DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved){
  switch(fdwReason){
    case DLL_PROCESS_ATTACH:
          CreateThread(0, NULL,
                   (LPTHREAD_START_ROUTINE)&AddNewMenuItem,
                   NULL, NULL, NULL);
      break;
    case DLL_THREAD_ATTACH:
      break;
    case DLL_THREAD_DETACH:
      break;
    case DLL_PROCESS_DETACH:
      break;
  }

  return TRUE;
}

LRESULT __stdcall NewProc(HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam){
  switch(iMsg){
    case WM_COMMAND:
      switch(wParam){
        case MYBUTTON:
          MessageBoxA(NULL,
                      "New menu item added with Window Subclassing",
                      "Information!",
                      MB_OK|MB_ICONINFORMATION);
          break;
      }
      break;
    }

  return CallWindowProc((WNDPROC)OriginalProcedure, hwnd, iMsg, wParam, lParam);
}

DWORD __stdcall AddNewMenuItem(){
  HWND hwnd = NULL;
  HMENU hMenu = NULL;

  hwnd = FindWindowA(NULL, "Untitled - Notepad");

  if(hwnd != INVALID_HANDLE_VALUE){
    hMenu = GetMenu(hwnd);
```

```
    if(hMenu != NULL){
      AppendMenuA(hMenu, MF_STRING, MYBUTTON, "MessageBox");
      DrawMenuBar(hwnd);
      OriginalProcedure = SetWindowLongA(hwnd, GWL_WNDPROC, (LONG)NewProc);

      return 1;
    }
  }

  return 0;
}
```

Originální článek: http://bflow.security-portal.cz/window-subclassing/ *[5]*

URL článku: https://www.security-portal.cz/clanky/window-subclassing

Odkazy:
[1] https://www.security-portal.cz/users/rubberduck
[2] https://www.security-portal.cz/category/tagy/programming
[3] http://bflow.security-portal.cz/images/WndSubclsng1.jpg
[4] http://bflow.security-portal.cz/images/WndSubclsng2.jpg
[5] http://bflow.security-portal.cz/window-subclassing/