Использование Vista UAC в Delphi

Это заключительная часть серии из трех частей о UAC и Delphi.
Перейти к части 1
Перейти к части 2

Здесь я затрону наиболее интересные подходы к обеспечению подлинного UAC соответствия – в том виде, в каком вероятно, следует реализовывать большинство Vista-совместимых приложений.

Повышение прав по запросу с использованием COM Elevation Moniker

Первая вещь, которую в которой вы должны быть уверены, это то, что исполняемый файл приложения содержит UAC манифест. Манифест, практически идентичный тому, который вы могли видеть в части 2, с одним, но очень важным отличием: атрибут requestedExecutionLevel должен иметь значение "**asInvoker**":

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
  <assemblyIdentity
    version="1.0.0.0"
    processorArchitecture="*"
    name="UACAwareApplication"
    type="win32"/>
  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
    <security>
      <requestedPrivileges>
        <requestedExecutionLevel level="asInvoker"/>
      </requestedPrivileges>
    </security>
  </trustInfo>
</assembly>
```

Этот манифест заставляет Vista назначить процессу привилегии по умолчанию при запуске приложения. Если в этот момент уровень доступа более низким, приложение будет вынуждено запрашивать повышение привилегий, каждый раз, когда пользователь выполняет действия, требующие административных привилегий. Это именно то, чего мы пытались добиться ранее.

Прежде чем продолжить, я объясню, как повышение привилегий работает изнутри…

Когда процесс стартует, ему присваивается уровень доступа Windows, который контролирует то, что этот процесс может или не может сделать, с некоторыми объектами, такими как файлы и каталоги, имеющими списки управления доступом (Access Control Lists или ACL's, для краткости) к себе. Что здесь важно, так это то, что уровень доступа, который присваивается фактически единожды, в момент запуска процесса, не может быть изменен на более поздних этапах жизни процесса. Следовательно, повышение привилегий, возможно одним единственным способом - порождением нового процесса с более высоким уровнем привилегий. Зная об этом, менее требовательные приложения, которые не нуждаются в использовании сложных состояний пользовательского интерфейса, могут просто перезапускать себя в админском режиме и работать.

Для большинства приложений, однако, это не приемлемо, потму, что перезапуск обычно означает длительную паузу и сложности, связанные с сохранением и восстановлением состояний пользовательского интерфейса.

Вместо этого, есть другое решение для таких случаев. Код приложения может быть разбит на две части – не привилегированный код размещается в основном исполняемом файле, в то время, как код, требующий привилегий выносится в отдельную dll, оформленную как один или несколько COM объектов. COM объекты могут вызываться COM посредствам Elevation Moniker в контексте выделенного COM-сервера, который и повышает привилегии.

Довольно теории. Выполним некоторые действия.

Здесь мы должны сделать следующее:

написать dll, которая будет загружаться выделенным COM – сервером;

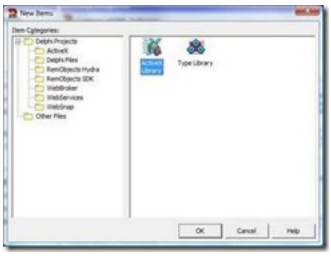
добавить COM-объект, реализующий функциональность, требующую привилегий;

добавить COM class factory, которая будет использоваться для создания экземпляра COM объекта и регистрации библиотеки, и

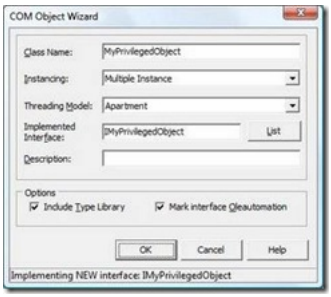
написать какой-то код для вызова COM объекта из главного исполняемого файла, который изначально не обладает привилегиями.

До настоящего момента, ничего не говорилось о создании dll. Эта обычная COM библиотека может быть создана в Delphi путем следующих действий:

"New" -> "Other..." -> "Delphi Projects"\"ActiveX" -> "ActiveX Library", как это показано на скриншоте.



Как только мы создали и сохранили новый COM library проект, нам необходимо добавить новый COM объект, в котором мы реализуем функционал повышения привилегий.



В демонстрационных целях мы определим единственный метод COM объекта, и опишем его с помощью интерфейса объекта по умолчанию:

```
1. IMyPrivilegedObject = interface(IUnknown)
2.   ['{04004D01-2115-40D5-991F-D258C8CEF07E}']
3.   function CreateFile(const aFileName: WideString): HResult; stdcall;
4. end;
```

Этот метод позволит нам проверить, действительно ли COM объект привязывается к процессу повышения привилегий.

Теперь нам нужно создать пользовательскую фабрику классов (Class Factory) для нашего COM объекта. Это нужно по той причине, что Class Factory отвечает за регистрацию и удаление (unregistering) COM объекта в системном реестре. Мы не можем просто по умолчанию использовать TTypedComObjectFactory, потому существуют некоторые специальные значения, которые должны быть прописаны в реестре, для того, чтобы сделать возможным размещение вне процесса и независимое повышение привилегий.

Приложения не могут просто запрашивать у Windows повышение привилегий подобно тому, как это делает любой OLE код. Если бы оно могло делать это, это была бы серьезная дыра в системе безопасности. Любой код, который должен быть выполнен с повышенными привилегиями, должен быть помещен в COM объект, расположенный в dll, который должен быть подготовлен к повышению привилегий.

Здесь основная часть реализации фабрики классов.

```
 1. type
 2.   TPrivilegedClassFactory = class(TTypedComObjectFactory)
 3.   private
 4.     fResourceId: AnsiString;
 5.   public
 6.     constructor Create(
 7.       const aResourceId: AnsiString;
 8.       aComServer: TComServerObject;
 9.       aTypedComClass: TTypedComClass; const aClassID: TGUID;
10.       aInstancing: TClassInstancing;
11.       aThreadingModel: TThreadingModel = tmSingle
12.     );
13.     procedure UpdateRegistry(aRegister: Boolean); override;
14.   end;
15.
16. implementation
17.
18. constructor TPrivilegedClassFactory.Create(const aResourceId: AnsiString;
```

```
    aComServer: TComServerObject; aTypedComClass: TTypedComClass;
    const aClassID: TGUID; aInstancing: TClassInstancing;
    aThreadingModel: TThreadingModel);
begin
  inherited Create(
    aComServer, aTypedComClass, aClassID, aInstancing, aThreadingModel
  );
  { Save the id of the string resource, that holds the application name. }
  fResourceId := aResourceId;
end;

procedure TPrivilegedClassFactory.UpdateRegistry(aRegister: Boolean);
var
  ID, ClassKey, FullFileName, FileName: AnsiString;
begin
  ID := GUIDToString(Self.ClassID);
  ClassKey := 'CLSID\' + ID;
  FullFileName := ComServer.ServerFileName;
  FileName := ExtractFileName(FullFileName);
  try
    if aRegister then begin
      inherited UpdateRegistry(aRegister);
      { DLL out-of-process hosting requirements. }
      CreateRegKey('AppID\' + ID, '', Description);
      CreateRegKey('AppID\' + ID, 'DllSurrogate', '');
      CreateRegKey('AppID\' + FileName, 'AppID', ID);
      { Over-The-Shoulder activation requirements. }
      SetAccessPermissionsForLUAServer('AppID\' + ID, 'AccessPermission');
      { COM object elevation requirements. }
      CreateRegKey(ClassKey, 'AppID', ID);
      CreateRegKey(ClassKey, 'LocalizedString', '@' + FullFileName + ',-' + fResourceId);
      CreateRegKeyEx(ClassKey + '\Elevation', 'Enabled', '1', nil, 0, REG_DWORD);
    end else begin
      DeleteRegKey(ClassKey + '\Elevation');
      DeleteRegKey('AppID\' + ID);
      DeleteRegKey('AppID\' + FileName);
      inherited UpdateRegistry(aRegister);
    end;

  except
    on E: EOleRegistrationError do raise;
    on E: Exception do raise EOleRegistrationError.Create(E.Message);
  end;
end;
```

Как отмечено в комментариях, существуют три требования, которые должны быть удовлетворены: регистрация разм

Я только хочу отметить, что такие фабрики классов являются типичными, и могут быть использованы в любых Del

Теперь мы должны использовать новую фабрику классов для создания COM объекта, и встроить строковый ресурс в

```
initialization
  TPrivilegedClassFactory.Create(
    '101', // resource string id
    ComServer, TMyPrivilegedObject, Class_MyPrivilegedObject,
    ciMultiInstance, tmApartment
  );
```

Это файл ресурсов со строкой, которая будет отображаться в запросе на превышение привилегий.

```
STRINGTABLE

{

101, "Delphi Elevation Demo"

}
```

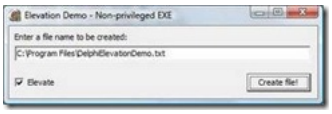
Последняя вещь, которую осталось сделать, это написать процедуру, которая будет вызываться из не привилегир

```
procedure CoCreateInstanceAsAdmin(
  aHWnd: HWND;           // parent for elevation prompt window
  const aClassID: TGUID; // COM class guid
  const aIID: TGUID;     // interface id implemented by class
  out aObj               // interface pointer
);

implementation

procedure CoCreateInstanceAsAdmin(aHWnd: HWND; const aClassID, aIID: TGUID;
 out aObj);
var
  BO: BIND_OPTS3;
  MonikerName: WideString;
begin
  if (not IsElevated) then begin
    { Request elevated out-of-process instance. }
    MonikerName := 'Elevation:Administrator!new:' + GUIDToString(aClassID);
    FillChar(BO, SizeOf(BIND_OPTS3), 0);
    BO.cbStruct := SizeOf(BIND_OPTS3);
    BO.dwClassContext := CLSCTX_LOCAL_SERVER;
    BO.hwnd := aHWnd;
    OleCheck(CoGetObject(PWideChar(MonikerName), @BO, aIID, aObj));
  end else
    { Request normal in-process instance. }
    OleCheck(CoCreateInstance(aClassID, nil, CLSCTX_ALL, aIID, aObj));
end;
```

Здесь находится полный исходный код ElevationDemo в формате Delphi 2006 for Win32 проекта вместе с скомпили



Чекбокс определяет будет ли вызываться COM объект, запрашивающий повышение привилегий или локальный COM объект, в случае вызова CreateFile(). Вы можете поэкспериментировать, вводя разные имена файлов в поле ввода.

Исходный код создан с использованием некоторых специфических Vista-структур, но для простоты я не включал код проверки ОС, пусть это будет домашним заданием для читателя.

Не забудьте сначала зарегистрировать DLL (для этого вам понадобятся привилегии Admin'а), в противном случае приложение не заработает. Обычно, это задача инсталлятора, но вы это можете сделать вручную:

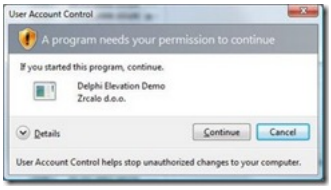
regsvr32.exe PrivilegedLib.dll

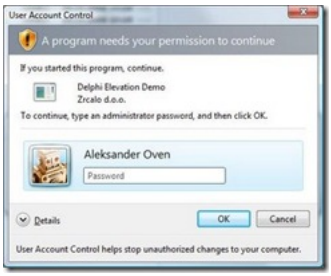
здесь показано, как будет выглядеть ваш реестр после полной инсталляции:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Classes
{
  AppID\PrivilegedLib.dll
  {
    (Default): REG_SZ = null
    AppID:     REG_SZ = '{6BCFB187-C1DD-4807-96AD-F91AB4AB08AC}'
  }
  AppID\{6BCFB187-C1DD-4807-96AD-F91AB4AB08AC}
  {
    (Default):        REG_SZ     = 'MyPrivilegedObject'
    AccessPermission: REG_BINARY = <BINARY VALUE>
    DllSurrogate:     REG_SZ     = ''

  }
```

Если все прошло нормально, вы должны будете увидеть один из следующих диалогов, после того, как вы выполните вызов запроса на превышение привилегий – вид диалога зависит от ваших настроек безопасности.





Ну вот и все. Я надеюсь, что вам понравилась эта серия. Я уверен, что показал все важнейшие шаги по созданию UAC-aware приложения в Delphi. Естественно, что я опустил массу деталей, чтобы не слишком вас утомлять – но это тема для отдельной публикации на будущее. Как всегда, присылайте свои комментарии и пожелания.