Алексей Федоров

Обеспечение совместимости приложений в Microsoft® Windows Vista® и Windows® 7

Руководство для IT-специалистов

Алексей Федоров

Обеспечение совместимости приложений в Microsoft Windows Vista и Windows 7

Руководство для IT-специалистов

Алексей Федоров — руководитель партнерской группы
Департамента стратегических технологий
российского представительства компании Microsoft.
(alexeif@microsoft.com)

© Федоров А. Г., 2009

Содержание

Введение ... 4
Инфраструктура обеспечения совместимости приложений ... 5
Подсистема User Account Control ... 8
Использование манифеста приложения ... 13
Проблемы при установке приложений ... 15
Расширенная поддержка подсистемы User Account Control ... 16
Файлы, папки и реестр ... 19
Изоляция процессов интерфейса (UIPI) ... 30
Защита ресурсов Windows ... 32
Версия операционной системы ... 35
Устаревшие компоненты ... 40
Внешний вид системы ... 41
Дополнительные заплатки ... 42
Использование системных заплаток. Практика ... 47
1. Использование Standard User Analyzer ... 47
2. Создание заплаток для улучшения работы приложения ... 51
3. Run Level Specification ... 54
4. Исправление различных ошибок, связанных с UAC ... 56
5. Виртуальный реестр ... 57
6. Исправление работы с папками ... 58
7. User Interface Process Isolation ... 58
8. Windows Resource Protection ... 59
9. Version Lies ... 60
10. FakeLunaTheme ... 61
Развертывание заплаток ... 63
Заключение ... 64

Введение

Совместимость существующих приложений с операционной системой Microsoft Windows Vista (и выходящей в этом году операционной системой Windows 7, построенной на ядре Windows Vista) является одной из основных проблем, с которой могут столкнуться пользователи, переходящие на новую версию операционной системы. Несмотря на усилия, прилагаемые компанией Microsoft, некоторые производители программного обеспечения продолжают использовать устаревшие функции операционной системы, некорректно выполняют операции по проверке версий ОС, не следуют рекомендациям по работе с файловой системой и зачастую не руководствуются советами по обеспечению корректной работы приложений в новых версиях системы. Все это приводит к тому, что в операционной системе Microsoft Windows Vista есть более 5600 системных заплаток (shims) для обеспечения корректной работы приложений различных производителей — от утилит китайских производителей до крупных продуктов известных фирм. В Windows 7 число системных заплаток увеличилось — в бета-версии новой операционной системы их насчитывается более 5700!

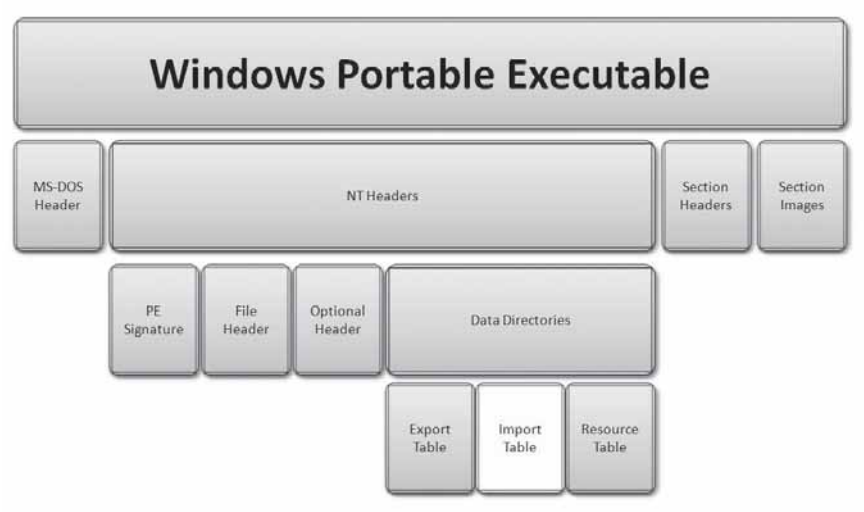
Можно выделить три основных подхода к обеспечению совместимости приложений: использование упомянутых выше системных заплаток, запуск приложения в виртуальной среде (терминальные сервисы или использование Microsoft Application Virtualization) и изменение кода приложения таким образом, чтобы оно соответствовало требованиям по корректной работе в операционной системе, — для этого служат руководства по сертификации приложений для получения логотипов Works With Windows Vista и Certified for Windows Vista, а также соответствующие тесткейсы, которые можно найти на сайте **http://www.innovateon.com** в разделе, посвященном Windows Vista.

В данном обзоре мы рассмотрим использование системных заплаток, а также приведем рекомендации по изменению кода приложений на основании анализа примененных к ним системных заплаток.

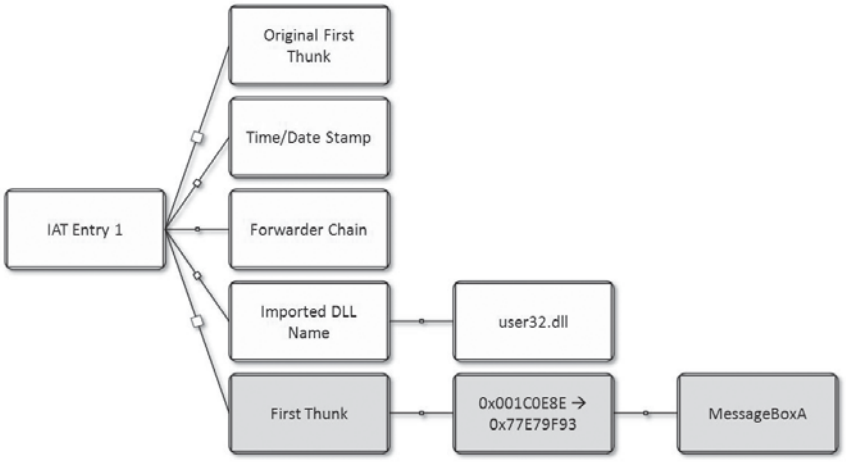
Инфраструктура обеспечения совместимости приложений

Подсистема системных заплаток (в Windows 98 и Windows 2000 они назывались «слоями») реализована в виде т. н. **Application Compatibility Infrastructure** — набора динамически загружаемых библиотек (основная функциональность реализована в библиотеке **shimeng.dll** — Shim Engine и **apphelp.dll** — Application Compatibility Interface, Application Compatibility Client Library), а также на уровне загрузчика **ntdll.dll**, ряда ключей реестра и системной базы данных. Когда на то или иное приложение накладываются соответствующие системные заплатки, это отображается в соответствующих записях реестра и служит индикатором для загрузчика на то, что при запуске данного приложения также необходимо запустить **Shim Engine**, выбрать указанные для приложения системные заплатки из системной базы данных **sysmain.sdb** и применить их к приложению. Системные заплатки выполняют роль перехватчиков вызовов системных функций, реализованы на основе механизма замещения элементов **Import Address Table (IAT)** для загруженного в память исполняемого файла и позволяют обеспечить приложению максимально комфортную среду для выполнения.

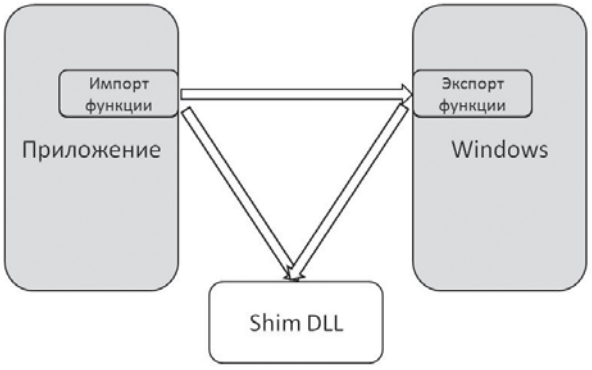
На следующих рисунках показана структура исполняемого файла Windows (Windows Portable Executable), таблицы импорта и процесса применения системной заплатки как перехватчика вызова системных функций с использованием таблиц импорта и экспорта функций в структуре исполняемого файла Windows.



Структура исполняемого файла Windows



Структура таблицы импорта



Работа перехватчика вызова системной функции

Системная база данных (в которой располагаются данные о всех системных заплатках, доступных для конкретной версии операционной системы, а также о дополнительных заплатках, созданных для конкретных приложений) располагается в каталоге **%systemroot%\AppPath**, здесь же находятся ключевые загружаемые библиотеки (dll-файлы с префиксом «Ac»), которые содержат код функций, поведение которых должно отличаться от стандартного. Системные заплатки находятся в библиотеках **acgenral.dll** и **aclayers.dll**, заплатки для конкретных приложений — в библиотеке **acspecfic.dll**. Другие группы системных заплаток — это флаги для LUA и

механизмов установки приложений, а также системные заплатки, которые накладываются на образ исполняемого файла в памяти и используются в тех случаях, когда невозможно решить проблемы совместимости простой заменой вызовов системных функций.

Необходимость применения системной заплатки определяется при создании каждого нового процесса. Подсистема Win32 отвечает за создание и удаление процессов и потоков. Одна из системных динамически загружаемых библиотек — **kernel32.dll** — содержит функцию **CreateProcess()**, которая используется для создания новых процессов и к которой добавляется код для обеспечения совместимости приложений (внутри ядра Windows эта функциональность реализована на уровне внутренней системной функции **Kernel32!CreateProcessInternalW()**).

Системный процесс **csrss.exe** (сокращение от «client/server run-time subsystem»; отвечает за консольные приложения, создание/удаление потоков и за 16-битную виртуальную среду MS-DOS) выполняет поиск наличия заплаток для загружаемого приложения. Загрузчик образов исполняемых файлов (ntdll.dll) выполняет загрузку EXE-файлов в память и именно в этот момент вступают в действие заплатки для конкретного приложения.

Процесс загрузки системных заплаток показан на следующем рисунке.

Процесс загрузки системных заплаток

После того как мы рассмотрели общие вопросы реализации подсистемы заплаток в Windows Vista, давайте обратимся к группам заплаток, которые обеспечивают решение ключевых проблем совместимости приложений. Начнем с проблем, связанных с появлением в Windows Vista подсистемы User Account Control (UAC).

Подсистема User Account Control

Основной задачей подсистемы User Account Control является обеспечение работоспособности пользователей под стандартной учетной записью. Это позволяет решить проблемы безопасности, снизить уязвимость системы для вредоносного кода и повысить надежность системы. Подсистема User Account Control работает следующим образом. Сначала компонент системы winlogon создает для пользователя локальный уникальный идентификатор (LUID) с полным токеном (его еще называют «маркером») безопасности. Затем создается LUID с защищенным токеном, который содержит только те права, которые доступны для пользователей, ассоциированных с учетной записью «стандартный пользователь». Идентификатор с защищенным токеном используется при вызове функции **CreateProcess()** для запуска среды Windows — explorer.exe. Работа подсистемы User Account Control показана на следующем рисунке.

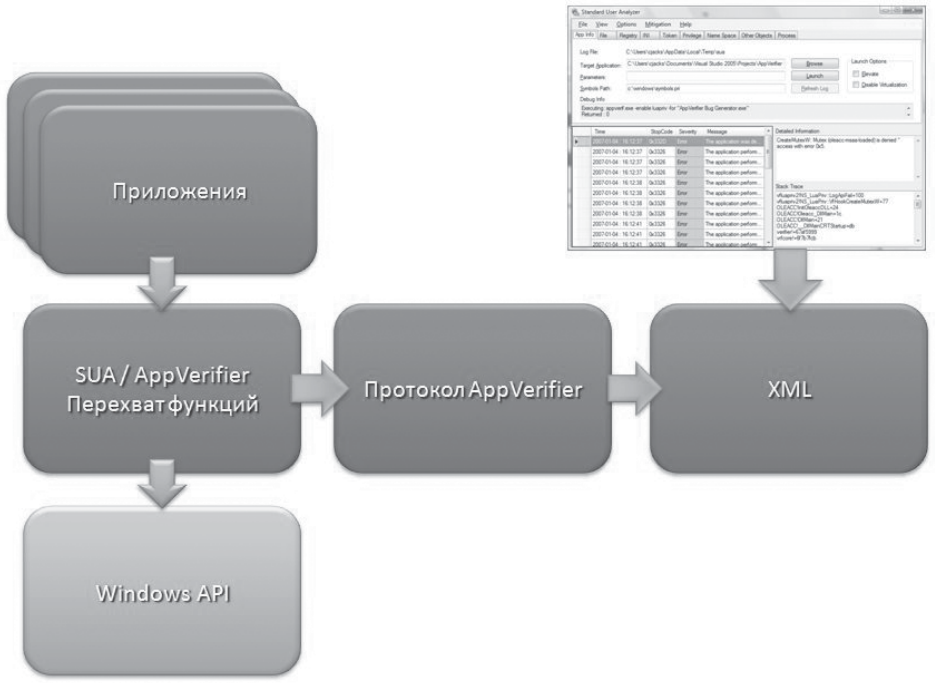
Работа User Account Control

Для нахождения возможных несовместимостей приложений с подсистемой User Account Control можно использовать утилиту **Standard User Analyzer**, которая поможет предсказать поведение приложений при работе под стандартной учетной записью. Действие утилиты основано на перехвате вызовов ряда системных функций и распространяется на следующие функциональные группы:

- доступ к файловой системе (например, к папке **Program Files**);
- доступ к реестру (например, к ветви **HKEY/Local Machine**);
- запись в INI-файлы (использование функции **WriteProfile()** для записи в системные папки);
- проверка токенов;

- использование функций, требующих привилегий (например, **SeDebugPrivilege()**);
- использование пространств имен (**Global**);
- использование защищенных объектов (например, событий, мьютексов, таймеров, семафоров и т. п.);
- создание процессов (привилегированная операция — запуск процесса, который требует повышенных привилегий).

Архитектура утилиты **Standard User Analyzer** показана на следующем рисунке.



Архитектура Standard User Analyzer

В процессе работы **Standard User Analyzer** может порекомендовать использование следующих системных заплаток, которые позволят решить ряд проблем, возникающих при использовании подсистемы User Account Control:

- **ElevateCreateProcess**
- **ForceAdminAccess**
- **LocalMappedObject**
- **VirtualizeDeleteFile**
- **VirtualizeHKCRLite**

Назначение этих заплаток показано в следующей таблице.

Заплатка	Симптом	Исправление
ElevateCreateProcess	Хост-процесс вызвал функцию **CreateProcess()** и получил ошибку с кодом **ERROR_ELEVATION_ REQUIRED** (740L)	Обрабатывает код ошибки и пытается запустить процесс с повышенными привилегиями. Если у приложения есть манифест, заплатка игнорируется. Данная заплатка перехватывает вызовы следующих системных функций: • **CreateProcessA()** • **CreateProcessW()** • **WinExec()** • **RegisterWaitFor-InputIdle ()** **Примечание:** перехват функции **RegisterWaitForInputIdle()** требуется для обработки последующих вызовов функции **WinExec()**
ForceAdminAccess	Приложение выполняется с ошибкой при отсутствии прав администратора	Возвращает **true** при выполнении проверки наличия прав администратора. Данная заплатка перехватывает вызовы следующих системных функций: • **AccessCheck()** • **CheckTokenMember-ship()** • **RegOpenKeyExW()** • **GetTokenInformation()** • **IsUserAnAdmin()** • **NetUserGetInfo()** • **SetActivePwrScheme()**
LocalMappedObject	Приложение пытается создать объект в пространстве имен **Global**	Перехватывает вызов и создает объект в пространстве имен **Local**. Данная заплатка перехватывает вызовы следующих системных функций: • **CreateFileMappingA()** • **CreateFileMappingW()** • **OpenFileMappingA()** • **OpenFileMappingW()**

Заплатка	Симптом	Исправление
VirtualizeDeleteFile	Приложение не может удалить файлы	Перехватывает вызов функции **DeleteFile()** и отключает виртуализацию для удаляемого файла
VirtualizeHKCRLite	В процессе работы приложение не может зарегистрировать COM-объекты	Перенаправляет записи в реестр — из ветви **HKLM** в ветвь **HKCU** для регистрации COM-объектов в режиме per-user. Заплатка распространяется на составную ветвь, которая является объединением **HKCU\Software\Classes** и **HKLM\Software\Classes**. Эту «заплатку» следует использовать совместно с заплаткой **VirtualizeRegisterTypeLib**

Приведем некоторые рекомендации по исправлению кода приложения, к которому применены одна или более из перечисленных выше системных заплаток. Если к приложению применена «системная заплатка **ElevateCreateProcess**, то следует заменить перехватываемые ей функции на вызов функции **ShellExecute()**, т. к. эта функция обладает большими возможностями, обрабатывает указанную ошибку, а также потенциальные новые типы ошибок, которые могут появиться в будущих версиях Windows. Несмотря на то что можно выполнить следующее сравнение:

```
GetLastError() == ERROR_ELEVATION_REQUIRED(740L)
```

и, соответственно, вызывать функцию **ShellExecute()** с глаголом «**runas**», это все равно приведет к вызову **ShellExecute()** и поэтому проще заменить начальный вызов.

При использовании системной заплатки **ForceAdminAccess** следует обратить внимание на тот факт, что приложению, возможно, вообще не требуется работа под учетной записью администратора. Если это не так, то следует выделить в отдельный блок функции, требующие повышенных привилегий для выполнения и задать требуемый уровень привилегий с помощью манифеста.

Использование системной заплатки **LocalMappedObject** указывает на то, что необходимо пересмотреть код приложения на предмет использования объектов в глобальном пространстве имен и, при возможности, создавать такие объекты в локальном пространстве имен.

Системные заплатки, связанные с включением или отключением механизмов виртуализации на уровне файловой системы (**VirtualizeDeleteFile**) и

реестра (**VirtualizeHKCRLite**) указывают на то, что необходимо пересмотреть подходы к хранению файлов — использовать рекомендованные каталоги и пытаться регистрировать COM-объекты в режиме per-user, а не per-machine.

В частности, заплатка **VirtualizeDeleteFile** помогает в тех случаях, когда приложение не может удалить какой-либо файл из-за отсутствия соответствующих привилегий. Заплатка **VirtualizeDeleteFile** возвращает код успешного завершения операции удаления.

Заплатка **VirtualizeHKCRLite** используется в случае возникновения проблем с регистрацией COM-компонентов в режиме выполнения приложений. Данная заплатка перенаправляет запись из ветви реестра HKCR (HKLM) в ветвь HKCU для регистрации COM-компонентов в режиме per-user. Данная заплатка более эффективно справляется с задачей, по сравнению с использованием заплатки **VirtualRegistry** с параметром **VIRTUALIZEHKCR**. Обычно системная заплатка **VirtualizeHKCRLite** используется совместно с заплаткой **VirtualizeRegisterTypeLib**.

Использование манифеста приложения

Использование манифеста позволяет указать требуемый уровень привилегий для выполнения приложений. Манифест — это XML-файл, который может быть прикреплен к исполняемому файлу в виде специального ресурса, либо помещен в виде отдельного файла в папку, в которой находится исполняемый файл. В общем случае манифест выглядит следующим образом:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
  <assemblyIdentity type="win32"
     processorArchitecture="*" version="1.0.0.0" name="MyApplication.exe"/>
  <description>My totally sweet Vista application</description>
  <ms_asmv2:trustInfo xmlns:ms_asmv2="urn:schemas-microsoft-com:asm.v2">
    <ms_asmv2:security>
      <ms_asmv2:requestedPrivileges>
        <ms_asmv2:requestedExecutionLevel level="asInvoker ||
         highestAvailable || requireAdministrator"/>
      </ms_asmv2:requestedPrivileges>
    </ms_asmv2:security>
  </ms_asmv2:trustInfo>
</assembly>
```

В зависимости от требуемого уровня привилегий можно указать три различных значения атрибута **requestedExecutionLevel**:

- asInvoker
- highestAvailable
- requireAdministrator

Использование данного атрибута эквивалентно применению следующих заплаток:

Заплатка	Симптомы	Исправление
RunAsAdmin	Приложение не запускается под стандартной учетной записью и не исправляется применением соответствующих заплаток	Помечает приложение как требующее повышенных привилегий. Эквивалент указанию в манифесте уровня **requireAdministrator**

Заплатка	Симптомы	Исправление
RunAsHighest	Приложению требуются повышенные привилегии, но необязательно на уровне администратора	Помечает приложение как требующее максимальных привилегий, доступных для данной учетной записи. Эквивалент указанию в манифесте уровня **highestAvailable**
RunAsInvoker	Приложение требует привилегий, которые на самом деле ему не нужны	Помечает приложение как не требующее привилегий выше, чем у создавшего его процесса. Эквивалент указанию в манифесте уровня **asInvoker**

В большинстве случаев использование манифеста и указание соответствующих значений атрибута **requestedExecutionLevel** позволяет отказаться от необходимости в применении перечисленных выше заплаток.

Проблемы при установке приложений

Две следующие заплатки призваны решить ряд проблем, которые могут возникнуть при установке приложений.

Заплатка	Симптомы	Исправление
SpecificInstaller	Приложение — это файл **setup.exe**, но компонент **GenericInstaller** не распознает его как инсталлятор	Помечает приложение как инсталлятор. При запуске отображается стандартный запрос на повышение привилегий
SpecificNonInstaller	Приложение не является инсталлятором и для его работы не требуется повышения привилегий, тем не менее, компонент **GenericInstaller** распознает приложение как инсталлятор	Помечает приложение таким образом, что оно перестает распознаваться как инсталлятор

Расширенная поддержка подсистемы User Account Control

Следующая группа заплаток также связана с возможными проблемами совместимости приложений с User Account Control и может помочь в тех ситуациях, когда:

- запрос на повышение привилегий не отображается;
- ссылка на деинсталлятор указывает на DLL;
- инсталлятор зависит от значений в ветви реестра HKCU;
- не удается запустить функцию **ElevateCreateProcess()**;
- требуется регистрация библиотеки типов (Type library);
- инсталляция и деинсталляция с помощью InstallShield завершается с ошибкой;
- требуются специальные привилегии для запуска сервисов.

Следующие заплатки могут использоваться для полного или частичного разрешения перечисленных проблем.

Заплатка	Симптомы	Исправление
CorrectShellExecuteHWND	Запуск приложения приводит к мерцанию панели задач, а не к появлению запроса на подтверждение повышения привилегий. При вызове функции **ShellExecute(Ex)** приложение указывает неверную ссылку на окно (HWND)	Заплатка перехватывает вызов функции **ShellExecute(Ex)**, исследует переданную ей ссылку на окно и при необходимости заменяет ее ссылкой на активное в данный момент окно (если оно принадлежит текущему процессу). Заплатка использует системную функцию **IsWindow()**
MakeShortcutRunas	При вызове ссылки на деинсталлятор не появляется запрос на подтверждение повышения привилегий. При вызове ссылки на деинсталлятор запускается **RunDll.exe**	Помечает ссылку на деинсталлятор как использующую команду «**runas**» что приводит к появлению запроса на подтверждение повышения привилегий

Заплатка	Симптомы	Исправление
CopyHKCUSettings_ FromOtherUsers	Инсталлятор выполняется с повышенными привилегиями, но зависит от значений в ветви реестра **HKCU**, относящейся к другому пользователю	Сканирует пользовательские профили и пытается скопировать ключи реестра в **HKCU**
RelaunchElevated	Не удается запустить дочерний процесс. Хост-приложение не может использовать функции **RunAsAdmin()** или **ElevateCreateProcess()** (пример — загружаемый из Интернета модуль запускает инсталлятор)	Помечает дочерний процесс соответствующими атрибутами. Приложение перезапускает само себя (если оно уже не выполняется с повышенными привилегиями), ожидает завершения перезапущенного процесса и возвращает код успешного завершения
VirtualizeRegister_TypeLib	В процессе выполнения приложение регистрирует библиотеку типов и получает ошибку Access Denied	Совместно с заплаткой **VirtualizeHKCRLite** управляет одновременной регистрацией библиотеки типов и COM-классов. Заменяет вызовы функции **RegisterTypeLib** на вызовы функции **RegisterTypeLibForUser**
BlockRunAsInteractiveUser	Запуск инсталлятора или деинсталлятора на базе InstallShield приводит к ошибкам, сбоям или зависанию приложения	Заменяет настройки реестра с **RunAs** на **InteractiveUser**
RetryOpenServiceWith_ ReadAccess	Приложение пытается обратиться к сервису и получает ошибку **ERROR_ACCESS_ DENIED**	Заменяет вызов функции **OpenService()** таким образом, чтобы вызов осуществлялся в режиме «только чтение»

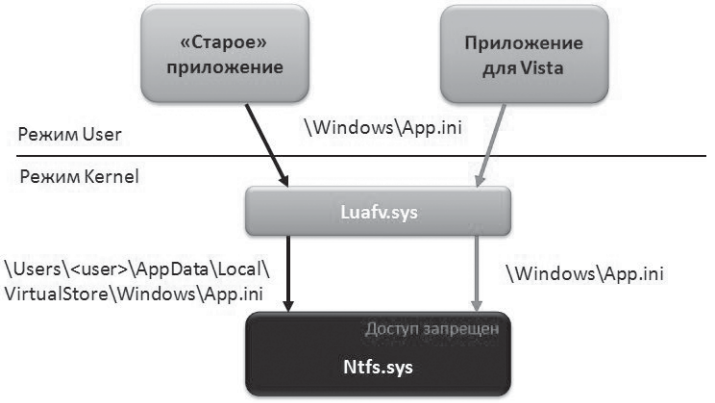
Заплатка	Симптомы	Исправление
RetryOpenSCManagerWith_ ReadAccess	Приложение пытается открыть менеджер управления сервисами (service control manager) и получает ошибку **ERROR_ ACCESS_DENIED**	Заменяет вызов функции **OpenSCMManager()** таким образом, чтобы вызов осуществлялся в режиме «только чтение»

Выше мы рассмотрели около 20 системных заплаток, которые можно использовать для решения различных проблем, связанных с совместимостью приложений и компонента операционной системы User Account Control.

В следующем параграфе мы рассмотрим заплатки, используемые для обеспечения совместимости приложений с файловой системой.

Файлы, папки и реестр

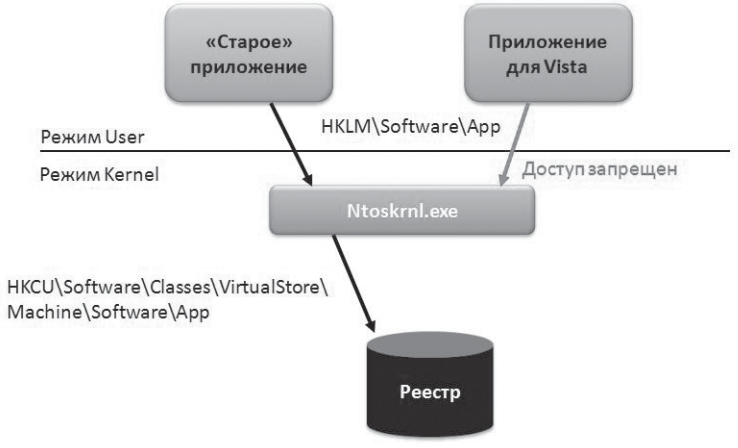
Как мы знаем, для обеспечения совместимости приложений с рядом изменений в структуре файловой системы (местоположение для хранения настроек, документов, изменения в наименовании стандартных папок и т. п.), а также для поддержки доступа к ряду областей реестра, в Microsoft Windows Vista реализованы два механизма виртуализации — для операций с файловой системой и для операций с реестром. Виртуализация работает только для клиентских приложений, поддерживается только для приложений, у которых нет манифеста, и реализована только в 32-битных версиях операционных систем семейства Microsoft Windows Vista. Принцип работы виртуализации файловой системы показан на следующем рисунке.



Принцип работы виртуализации файловой системы

Рассмотрим следующий сценарий. Приложение, созданное для предыдущих версий операционной системы (не имеющее встроенный манифест), пытается записать данные в конфигурационный файл **\Windows\App.ini**. В этом случае срабатывает драйвер **Luafv.sys**, который реализует виртуализацию на уровне файловой системы, и соответствующие программные интерфейсы системного модуля **Ntfs.sys** получат запрос на запись в файл, который физически будет располагаться в **\Users\<user>\AppData\Local\VirtualStore\Windows\App.ini** — т. е. в безопасном для манипуляций из пользовательских приложений месте. Операция завершится успешно. Если же приложение, написанное для Windows Vista (имеющее встроенный манифест), попытается выполнить аналогичную операцию, оно получит сообщение об ошибке — в этом случае виртуализация файловых операций выполняться не будет.

Точно по таким же принципам работает и виртуализация реестра — для приложений, не имеющих встроенного манифеста действия над некоторыми областями реестра (например, **HKLM\Software\App**), перенаправляются в «безопасные» области (например, **HKCU\Software\Classes\VirtualStore\Machine\Software\App**), а для приложений, имеющих встроенный манифест, возвращается сообщение об ошибке. Принцип работы виртуализации реестра показан на следующем рисунке.



Принцип работы виртуализации реестра

Отметим ряд ограничений описанных выше механизмов виртуализации. Виртуализация поддерживается только для файловых областей **Program Files**, **Program Data** и Windows, а также для ветви реестра **HKLM\Software**.

В ряде случаев описанных выше механизмов виртуализации может оказаться недостаточно для обеспечения совместимости существующих приложений с Microsoft Windows Vista. Если мы сталкиваемся с проблемами, использования приложений, подобно описанным ниже, возможно, следует использовать системные заплатки.

Системная заплатка **VirtualRegistry** может помочь в тех случаях, когда приложениям не удается удалить определенные ключи реестра — она позволяет активизировать функции реестра для перенаправления виртуальных ключей. Используя данную заплатку вы сможете:

- перенаправлять ключи реестра;
- скрывать существование ключа и/или его значения;
- превращать ключи типа **REG_EXPAND_SZ** в ключи типа **REG_SZ**;
- удалять начальный символ "\" из имени ключа;

- добавлять атрибуты безопасности **MAXIMUM_ALLOWED** ко всем ключам;
- управлять параметрами функции **QueryInfoKey()** для обеспечения совместимости с Windows 95;
- разрешать удаление ключей, которые имеют подключи для того, чтобы обеспечить совместимость с Windows 95;
- предохранять значения ключей и сами ключи от удаления.

Ниже показаны системные функции, которые перехватываются данной системной заплаткой.

Функция	Действия системной заплатки
RegOpenUserClassesRoot()	Виртуализирует вызовы функции **RegOpenUserClassesRoot()** — виртуальные ключи являются дочерними ключами **HKEY_CLASSES_ROOT** для данного виртуального ключа
RegConnectRegistryA(), RegConnectRegistryW()	Виртуализирует вызовы для подключения к реестру на указанном компьютере
RegOpenKeyExA(), RegOpenKeyExW()	Виртуализирует вызовы для открытия ключа реестра
RegQueryValueExA(), RegQueryValueExW()	Виртуализирует вызовы для получения значения ключа реестра
RegCloseKey()	Виртуализирует вызовы для закрытия ключа реестра
RegOpenKeyA(), RegOpenKeyW()	Виртуализирует вызовы для открытия ключа реестра
RegQueryValueA(), RegQueryValueW()	В данной версии не реализована, возвращает оригинальное значение
RegCreateKeyA(), RegCreateKeyW()	Виртуализирует вызовы для создания ключа реестра
RegCreateKeyExA(), RegCreateKeyExW()	Виртуализирует вызовы для создания ключа реестра
RegEnumValueA(), RegEnumValueW()	Виртуализирует вызовы для получения всех значений ключа реестра
RegEnumKeyA(), RegEnumKeyW()	Виртуализирует вызовы для получения для всех вложенных ключей данного ключа
RegEnumKeyExA(), RegEnumKeyExW()	Виртуализирует вызовы для получения для всех вложенных ключей данного ключа
RegQueryInfoKeyA(), RegQueryInfoKeyW()	Виртуализирует вызовы для получения информации о ключе реестра
RegSetValueExA(), RegSetValueExW()	Виртуализирует вызовы для задания значения ключа реестра

Функция	Действия системной заплатки
RegDeleteKeyA(), RegDeleteKeyW()	Виртуализирует вызовы для удаления ключа реестра
RegSetValueA(), RegSetValueW()	Виртуализирует вызовы для задания значения ключа реестра

Параметры, которые могут использоваться совместно с этой системной заплаткой, позволяют эмулировать работу определенных версий операционной системы или учитывают специфику поведения отдельных приложений.

Ключевыми рекомендациями по исправлению кода при возникновении проблем совместимости с реестром могут быть отказ от получения значений из недокументированных ключей реестра, а также более корректный подход к обработке ошибок, связанных с отсутствием тех или иных ключей реестра, а также ошибок, связанных с запретом доступа к определенным ключам или ветвям реестра.

Помимо несовместимостей с реестром, приложения могут столкнуться с изменениями, коснувшимися новых названий и местоположений ряда системных папок. Ключевые изменения показаны ниже.

- Папка **Documents and Settings** теперь называется **Users**.
- Папка **My Documents** теперь называется **Documents**.
- Отдельные папки для графических изображений (**Pictures**), звуковых файлов (**Music**) и видеофайлов (**Video**).
- Папка **All Users** теперь называется **Public\ProgramData**.
- Папка **C:\Programms** теперь называется **C:\Program Files**.
- Функция **ShGetFolderPath()** возвращает новые значения для системных папок.
- Механизм **Directory junction**s используется для обеспечения совместимости.
- Атрибут **DENY_READ** установлен для **Everyone**.

Для устранения несовместимостей можно использовать системную заплатку **CorrectFilePaths**, которая применяется в тех случаях, когда у приложений возникают проблемы с доступом к определенным областям файловой системы. К таким проблемам можно отнести ошибки класса «**access denied**» при попытке записи в файлы или «**file not found**» или «**path not found**» также при записи в файлы. Данная «системная заплатка модифицирует имена папок таким образом, чтобы они указывали на новые имена, используемые в Windows Vista. Ниже мы более подробно рассмотрим, как работает данная системная заплатка, но сначала поясним ее основные принципы на рисунке, который приведен ниже.

Последовательность коррекций заплатки CorrectFilePaths

Системная заплатка **CorrectFilePaths** является одной из самых мощных среди системных заплаток и довольно часто используется для решения проблем, связанных с обеспечением совместимости приложений. Используя данную системную заплатку можно изменить любой файловый путь на любой другой. Эта системная заплатка включает ряд коррекций, выполняемых по умолчанию и позволяет добавить дополнительные исправления, указываемые опциями командной строки, которые мы рассмотрим ниже.

Сначала выполняется набор коррекций по умолчанию для стандартных папок в стиле Windows 9x — в том случае, если данная возможность была сконфигурирована соответствующими опциями командной строки, которые описаны ниже. Выполняются следующие коррекции:

- преобразование всех символов "/" в символы "\";
- удаление пробелов из имен каталогов — "**abc\ \def**" становится "**abc\def**";
- удаление "**.**" из имен каталогов.

Затем происходит корректировка имен, которые были указаны в командной строке.

Существует несколько способов конфигурации коррекции, выполняемой по умолчанию. Доступны три следующие опции конфигурации — Base, User и All Users. Каждая из них описана ниже.

Опция **Base** указывает на то, что коррекции по умолчанию не должны применяться. Имена папок автоматически не изменяются.

Опция **User** указывает на то, что должны применяться коррекции по умолчанию, но только для папок в области данного пользователя. Коррекции выполняются в следующей последовательности:

- c:\windows становится %windir%;
- %windir%\rundll32.exe становится %systemdir%\rundll32.exe;
- %windir%\rundll.exe становится %systemdir%\rundll.exe;
- %windir%\write.exe становится %systemdir%\write.exe;
- %windir%\dxdiag.exe становится %systemdir%\dxdiag.exe;
- %windir%\wordpad.exe становится значением, полученным из HKEY_CLASSES_ROOT\Applications\wordpad.exe\shell\open\command;
- %programfiles%\accessories\wordpad.exe становится значением, полученным из HKEY_CLASSES_ROOT\Applications\wordpad.exe\shell\open\command;
- %windir%\Start Menu становится %userstartmenu%;
- %windir%\Desktop становится %userdesktop%;
- %windir%\Favorites становится %userfavorites%;
- %windir%\%CSIDL_STARTMENU_NAME% становится %userstartmenu%;
- %windir%\%CSIDL_DESKTOPDIRECTORY_NAME% становится %userdesktop%;
- %windir%\%CSIDL_FAVORITES_NAME% становится %userfavorites%;
- %windir%\Profiles\%username%\Start Menu становится %userstartmenu%;
- %windir%\Profiles\%username%\Desktop становится %userdesktop%;
- %windir%\Profiles\%username%\Favorites становится %userfavorites%;
- %windir%\Profiles\%username%\%CSIDL_STARTMENU_NAME% становится %userstartmenu%;
- %windir%\Profiles\%username%\%CSIDL_DESKTOPDIRECTORY_NAME% становится %userdesktop%;
- %windir%\Profiles\%username%\%CSIDL_FAVORITES_NAME% становится %userfavorites%;
- %windir%\Profiles\All Users\Start Menu становится %allstartmenu%;
- %windir%\Profiles\All Users\Desktop становится %alldesktop%;
- %windir%\Profiles\All Users\Favorites становится %userfavorites%;
- %windir%\Profiles\%ALLUSERSPROFILE_NAME%\%CSIDL_STARTMENU_NAME% становится %allstartmenu%;
- %windir%\Profiles\%ALLUSERSPROFILE_NAME%\%CSIDL_DESKTOP-DIRECTORY_NAME% становится %alldesktop%;
- %windir%\Profiles\%ALLUSERSPROFILE_NAME%\%CSIDL_FAVORITES_NAME% становится %userfavorites%;
- ddhelp.exe становится ddraw.dll;
- ddraw16.dll становится ddraw.dll;
- dsound.vxd становится ddraw.dll.

Опция **All Users** указывает на то, что коррекции должны применяться ко всем папкам. Коррекции выполняются в следующей последовательности — сначала все коррекции, что и для опции **User**, затем — **%userstartmenu%** становится **%allstartmenu%**.

И уже после выполнения коррекции стандартных папок выполняется коррекция папок, указанных в командной строке. Для этого можно использовать следующие переменные:

- %windir% = значение из GetWindowsDirectoryW;
- %systemroot% = значение из GetWindowsDirectoryW;
- %systemdrive% = первые да символа из GetWindowsDirectoryW;
- %systemdir% = значение из GetSystemDirectoryW;
- %username% = значение из GetUserNameW;
- %programfiles% = значение из ShGetFolderPathW(..., CSIDL_PROGRAM_FILES, ...);
- %userstartmenu% = значение из ShGetFolderPathW(..., CSIDL_STARTMENU, ...);
- %allstartmenu% = значение из ShGetFolderPathW(..., CSIDL_COMMON_STARTMENU, ...);
- %userdesktop% = значение из ShGetFolderPathW(..., CSIDL_DESKTOP-DIRECTORY, ...);
- %alldesktop% = значение из ShGetFolderPathW(..., CSIDL_COMMON_DESKTOPDIRECTORY, ...);
- %userfavorites% = значение из ShGetFolderPathW(..., CSIDL_FAVORITES, ...);
- %allfavorites% = значение из ShGetFolderPathW(..., CSIDL_COMMON_FAVORITES, ...);
- %userappdata% = значение из ShGetFolderPathW(..., CSIDL_APPDATA, ...);
- %allappdata% = значение из ShGetFolderPathW(..., CSIDL_COMMON_APPDATA, ...);
- %allusersprofile% = значение из GetAllUsersProfileDirectoryW;
- %userprofile% = значение из GetUserProfileDirectoryW;
- %appexedir% = значение из GetModuleFileNameW.

В следующей таблице показаны примеры применяемых данной системной заплаткой коррекций.

Старое значение	Новое значение
C:\Windows	%windir%
%windir%\rundll32.exe	%systemdir%\rundll32.exe
%windir%\rundll.exe	%systemdir%\rundll32.exe
%windir%\write.exe	%systemdir%\write.exe

Старое значение	Новое значение
%windir%\wordpad.exe	%programfiles%\windows nt\accessories\wordpad.exe
%programfiles%\accessories\wordpad.exe	%programfiles%\windows nt\accessories\wordpad.exe
%windir%\dxdiag.exe	%systemdir%\dxdiag.exe
%windir%\start menu	%userstartmenu%
%windir%\desktop	%userdesktop%
%windir%\favorites	%userfavorites%
%windir%\profiles\%username%\start menu	%userstartmenu%
%windir%\profiles\%username%\desktop	%userdesktop%
%windir%\profiles\%username%\favorites	%userfavorites%
%windir%\profiles\all users\start menu	%allstartmenu%
%windir%\profiles\all users\desktop	%alldesktop
%windir%\profiles\all users\favorites	%userfavorites%
ddhelp.exe	ddraw.dll
ddraw16.dll	ddraw.dll
dsound.vxd	ddraw.dll
%userstartmenu%	%allstartmenu%

Программные интерфейсы, которые перехватываются системной заплаткой CorrectFilePaths, могут быть сконфигурированы с помощью опций командной строки, описанных ниже. Ниже показаны перехватываемые программные интерфейсы, сгруппированные по функциональности.

Группа функций	Функции
Функции для создания процесса	• **CreateProcessA() / CreateProcessW()** • **WinExec ()** • **ShellExecuteA() / ShellExecuteW()** • **ShellExecuteExA() / ShellExecuteExW()**
Функции обработки командной строки	• **GetCommmandLineA / GetCommandLineW**

Группа функций	Функции
Получение профиля	• **GetPrivateProfileIntA / GetPrivateProfileIntW** • **GetPrivateProfileSectionA / GetPrivateProfileSectionW** • **GetPrivateProfileSectionNamesA / GetPrivateProfileSectionNamesW** • **GetPrivateProfileStringA / GetPrivateProfileStringW** • **GetPrivateProfileStructA / GetPrivateProfileStructW** • **WritePrivateProfileSectionA / WritePrivateProfileSectionW** • **WritePrivateProfileStringA / WritePrivateProfileStringW** • **WritePrivateProfileStructA / WritePrivateProfileStructW**
Файловые функции	• **CopyFileA / CopyFileW** • **CopyFileExA / CopyFileExW** • **CreateDirectoryA / CreateDirectoryW** • **CreateDirectoryExA / CreateDirectoryExW** • **CreateFileA / CreateFileW** • **DeleteFileA / DeleteFileW** • **FindFirstFileA / FindFirstFileW** • **FindFirstFileExA / FindFirstFileExW** • **GetBinaryTypeA / GetBinaryTypeW** • **GetFileAttributesA / GetFileAttributesW** • **GetFileAttributesExA / GetFileAttributesExW** • **SetFileAttributesA / SetFileAttributesW** • **GetTempFileNameA / GetTempFileNameW** • **GetLongPathNameA / GetLongPathNameW** • **MoveFileA / MoveFileW** • **MoveFileExA / MoveFileExW** • **MoveFileWithProgressA / MoveFileWithProgressW** • **RemoveDirectoryA / RemoveDirectoryW** • **SetCurrentDirectoryA / SetCurrentDirectoryW** • **OpenFile** • **_lopen** • **_lcreat**

Группа функций	Функции
Установка значений реестра	• **RegSetValueA / RegSetValueW** • **RegSetValueExA / RegSetValueExW**
Функции для Shell Link	• **IShellLinkA::SetPath / IShellLinkW::SetPath** • **IShellLinkA::SetArguments / IShellLinkW::SetArguments** • **IShellLinkA::SetIconLocation / IShellLinkW::SetIconLocation** • **IPersistFile::Save()**
Load Image	• **LoadImageA()**

Доступны следующие опции командной строки:

-a — выполнить все коррекции по умолчанию, учитывая все пользовательские настройки;

-c — не перехватывать функции группы **Create Process**;

-f — не перехватывать функции группы **File Routines**;

-p — не перехватывать функции группы **Get Private Profile**;

-s — не перехватывать функции группы **Shell Link**;

-b — не применять коррекции по умолчанию (опция **Base**);

-u — выполнить все коррекции по умолчанию, учитывая все пользовательские настройки;

+GetCommandLine — перехватывать функции группы **Get Command Line**;

+RegSetValue — перехватывать функции группы **Registry Set Value**;

+Win9xPath — применить коррекции, специфичные для Win9x;

-Profiles — выполнить все коррекции по умолчанию, учитывая все пользовательские настройки;

+LoadBitmap — перехватывать функции группы **Load Image**;

«**old path;new path**» — заменить части «**old path**» на «**new path**». Если в названии содержатся пробелы, поместить все название в кавычки.

Приведем несколько примеров использования параметров системной заплатки **CorrectFilePaths**:

- -b setup.exe;setupapp.exe;
- -f –p –s -b setup.exe;setupapp.exe;
- setup.exe;setupapp.exe «Bad Place;Correct Location»;
- wordpad.exe;%SystemDir%\wordpad.exe .\%SystemDir%;%SystemDir%.

В целом имена должны исправляться с помощью системной заплатки **CorrectFilePaths** в двух случаях: когда приложение пытается записывать данные в защищенные области файловой системы или когда в приложении используются «зашитые» имена папок, которые были изменены в новой версии операционной системы.

Необходимо внести корректировки в приложения таким образом, чтобы они сохраняли данные в папки, определенные на уровне пользовательского профиля или использовать ACL для задания установок на уровне всего компьютера (для всех пользователей).

Приложения не должны использовать «зашитые» имена папок — для определения имен системных папок следует использовать соответствующие программные интерфейсы, перечисленные выше.

Для корректной работы приложений под управлением Windows Vista и Windows 7 рекомендуется использовать новый набор функций для доступа к папкам, относящихся к профилю пользователя. Эти функции входят в группу Win32 Known Folders и заменяют собой функции из предыдущих версий Windows согласно следующей таблице.

Новая функция	Устаревшая функция	COM-эквивалент функции
SHGetKnownFolderPath()	**SHGetFolderPath()**	**GetPath()**
SHGetKnownFolderIDList()	**SHGetFolderLocation()**	**GetIDList()**
SHSetKnownFolderPath()	**SHSetFolderPath()**	**SetPath()**

Изоляция процессов интерфейса (UIPI)

В качестве дополнительного уровня защиты от вредоносного программного обеспечения в Windows Vista и Windows Server 2008 различные приложения могут работать под одним из трех уровней привилегий (High, Medium и Low). Приложения могут взаимодействовать с другими приложениями, выполняющимися с тем же или более низким уровнем привилегий, но не могут модифицировать или взаимодействовать с приложениями, выполняющимися с более высоким уровнем привилегий. В большинстве случаев приложения выполняются с уровнем привилегий Medium, а приложения, требующие наличия прав администратора, — с более высокими привилегиями. Процессы с ограниченными возможностями, например код внутри Internet Explorer, выполняются с минимальными привилегиями.

Таким образом, приложения, выполняющиеся с низким уровнем привилегий, не могут посылать сообщения приложениям, выполняющимся с более высоким уровнем привилегий за исключением тех случаев, когда последние непосредственно не разрешили получение сообщений используя функцию **ChangeWindowMessageFilter()**. Точно также, приложения с низкими привилегиями не могут модифицировать ссылки типа **HWND**, владельцами которых являются приложения с более высокими привилегиями. Для обеспечения совместимости функция **SendMessage()** и ряд других функций для отсылки сообщений возвращают код успешного завершения, даже если выполнение таких функций заблокировано из-за описанных выше проблем с уровнями привилегий.

Уровень доверия определяется с помощью функции **AccessCheck()**, а сам механизм Mandatory Integrity Control реализован на уровне идентификаторов SID со следующими значениями:

- Low S-1-16-4096
- Medium S-1-16-8192
- High S-1-16-12288
- System S-1-16-16384

Две системные заплатки позволяют решить возможные проблемы наличия дополнительных проверок уровней «доверия» при взаимодействии процессов и объектов — это заплатки **UIPIEnableStandardMessages** и **UIPIEnableCustomMessages**.

Системную заплатку **UIPIEnableStandardMessages** следует применять в тех случаях, когда приложение использует стандартные сообщения для общения с другими процессами и эти сообщения не доставляются до адресата. Данная заплатка позволяет доставить указанные сообщения от процесса с низким уровнем «доверия» текущему процессу, что эквивалентно использованию системной функции **ChangeWindowMessageFilter()**.

Параметры данной системной заплатки: MessageID1 MessageID2 MessageID3.

Пример использования: 1055 1056 1057 1058 1069.

Системную заплатку **UIPIEnableCustomMessages** следует использовать в тех случаях, когда возникают проблемы с передачей нестандартных сообщений. Функционирование данной заплатки эквивалентно вызову системной функции **RegisterWindowMessage()** с последующим вызовом системной функции **ChangeWindowMessageFilter()**.

Параметры данной системной заплатки: MessageString1 MessageString2.

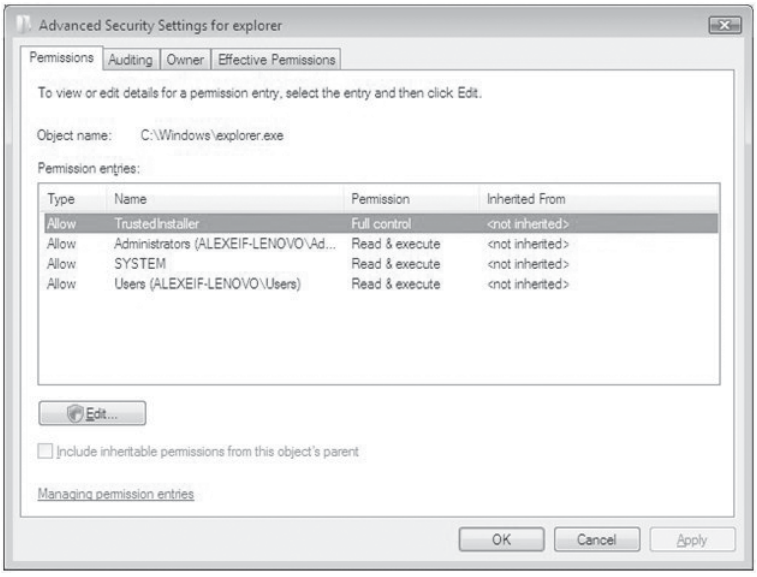
Защита ресурсов Windows

Для увеличения стабильности системы, ее надежности и предсказуемости поведения механизм Windows Resource Protection (WRP) защищает систему Windows, поддерживая ее в режиме «только чтение». Эта функциональность распространяется на файлы ядра операционной системы, папки и ключи реестра. Обновления системных ресурсов возможны только приложениями, которые известны операционной системе (**OS trusted installers**), такими как, например, Windows Servicing. Такая защита позволяет максимально защитить компоненты и приложения, входящие в состав операционной системы, от возможных воздействий со стороны других приложений и администраторов.

Данный механизм защищает следующие компоненты операционной системы:

- основные компоненты операционной системы (dll- и exe-файлы);
- основные ключи реестра, связанные с функционированием операционной системы;
- папки, используемые исключительно процессами операционной системы (например, части **inetpub**).

Для определения того, защищен ли компонент механизмом Windows Resource Protection, можно использовать либо соответствующие программные интерфейсы — функцию **SfcIsFileProtected**(), либо панель Advanced Security Settings, которую можно вызвать нажатием правой кнопки мыши на соответствующем файле и выбором команды Properties. Затем следует выбрать вкладку Security и нажать кнопку Advanced. Если полный доступ к файлу (Full control) имеет учетная запись **TrustedInstaller**, это означает, что компонент защищен механизмом Windows Resource Protection. Ниже показан пример панели **Advanced Security Settings** для файла **Explorer.exe**.



Windows Resource Protection

Для того чтобы определить, защищен ли элемент реестра механизмом Windows Resource Protection, можно выполнить следующие действия.

1. Из кода, используя функцию **SfcIsKeyProtected**().
2. В Registry Editor проверить свойства файла:
   - нажать правую кнопку мыши на интересующем нас элементе реестра и выбрать **Properties**;
   - элемент, который защищен механизмом WRP, покажет, что режим Full Control доступен только для учетной записи Trusted Installer. Для учетных записей **SYSTEM**, Administrators и Users доступен только режим Read.

Три системные заплатки — **WRPMitigation**, **WRPDllRegister** и **WRPRegDeleteKey** позволяют решить вопросы совместимости приложений с механизмом защиты компонентов операционной системы Windows Resource Protection.

Системную заплатку **WRPMitigation** следует использовать в тех случаях, когда приложение получает ошибки «Access Denied» при попытке манипуляции с защищенными системными ресурсами (помимо операций чтения). Заплатка не позволяет выполнять операции над системными файлами, но возвращает код успешного завершения при проверке возможности доступа к таким файлам (функция **AccessCheck()**).

Системную заплатку **WRPDllRegister** следует применять при возникновении ошибок при попытке регистрации COM-компонентов, поставля-

емых в составе Windows. Данная «системная заплатка позволяет пропустить регистрацию DLL, входящих в состав системы, или DLL, регистрируемых как параметры (используются функции **DllRegisterServer()** и **DllUnregisterServer()**).

Параметры заплатки: componet1.dll;component2.dll.

Пример использования: hhctrl.ocx;itircl.dll;itss.dll.

Системную заплатку **WRPRegDeleteKey** следует использовать при возникновении ошибок, возникающих при удалении ряда ключей реестра — в этом случае заплатка возвращает код успешного завершения операции, не удаляя указанный ключ реестра.

Версия операционной системы

Проверка версии операционной системы и некорректная интерпретация результатов является одной из наиболее частых причин невозможности запуска приложения под новой версией операционной системы. В следующей таблице показаны возможные версии операционных систем семейства Microsoft Windows.

Операционная система	Версия	Дата выпуска
Windows 1.0	1.04	1985
Windows 2.0	2.11	1987
Windows 3.0	3.00	1990
Windows NT 3.1	3.10.528	1993
Windows for Workgroups 3.11	3.11	1993
Windows NT Workstation 3.5	3.5.807	1994
Windows NT Workstation 3.51	3.51.1057	1995
Windows 95	4.0.950	1995
Windows NT Workstation 4.0	4.0.1381	1996
Windows 98	4.1.1998	1998
Windows 98 Second Edition	4.1.2222	1999
Windows Me	4.90.3000	2000
Windows 2000 Professional	5.0.2195	2000
Windows XP	5.1.2600	2001
Windows Vista	6.0.6000	2006
Windows 7 Beta 1	6.1.7000	2008

Набор заплаток в группе xxxVersionLie позволяет решить проблемы, связанные с определением версии операционной системы, — подменить значения, возвращаемые соответствующими системными функциями таким образом, чтобы проверка номера версии прошла успешно. Поддерживаются следующие заплатки:

- **Win95VersionLie**
- **WinNT4SP5VersionLie**
- **Win98VersionLie**
- **Win2000VersionLie**
- **Win2000SP1VersionLie**
- **Win2000SP2VersionLie**
- **Win2000SP3VersionLie**
- **WinXPVersionLie**
- **WinXPSP1VersionLie**

- **WinXPSP2VersionLie**
- **Win2K3RTMVersionLie**
- **Win2K3SP1VersionLie**

Эти заплатки входят в следующие уровни совместимости, которые, помимо собственно версии операционной системы, обеспечивают дополнительные функции совместимости: Win95, NT4SP5, Win98, Win2000, Win2000SP2, Win2000SP3, WinXP, WinXPSP1, WinXPSP2, WinXPSP2VersionLie, WinSrv03 и WinSrv03SP1. Отличие заплаток от уровней совместимости показано на следующих рисунках.

Применение системных заплаток

Применение уровня совместимости

Как видно из приведенных рисунков, действие системной заплатки распространяется только на текущее приложение, тогда как уровень совместимости может обеспечить корректную работу и дочерних приложений, используемых нашим приложением.

Помимо этих отличий, уровень совместимости включает использование целого ряда системных заплаток, а не только системной заплатки, корректирующей номер версии операционной системы. Например, уровень **XPSP2**, включает следующие системные заплатки:

- **DirectXVersionLie**
- **ElevateCreateProcess**
- **EmulateSorting**
- **EnableLegacyException HandlingInOLE**
- **FailObsoleteShellAPIs**
- **GlobalMemoryStatus2GB**
- **HandleBadPtr**
- **HardwareAudioMixer**
- **LoadLibraryCWD**
- **NoGhost**
- **RedirectMP3Codec**
- **VirtualRegistry**
- **WinXPSP2VersionLie**

Рассмотрим, как работает заплатка WinXPSP2VersionLie. Как следует из ее названия, она позволяет определить версию операционной системы как Windows XP Service Pack 2. Данная заплатка перехватывает следующие системные функции.

- **GetVersionExA** — возвращает структуру LPOSVERSIONINFOA со следующими значениями:
  - dwMajorVersion = 5
  - dwMinorVersion = 1
  - dwBuildNumber = 2600
  - dwPlatformId = VER_PLATFORM_WIN32_NT = 2
  - szCSDVersion = "Service Pack 2"
  - при передаче структуры LPOSVERSIONINFOEXA:
    - wServicePackMajor = 2
    - wServicePackMinor = 0

- **GetVersionExW** — возвращает структуру LPOSVERSIONINFOW со следующими значениями:
  - dwMajorVersion = 5
  - dwMinorVersion = 1
  - dwBuildNumber = 2600
  - dwPlatformId = VER_PLATFORM_WIN32_NT = 2
  - szCSDVersion = "Service Pack 2"
  - при передаче структуры LPOSVERSIONINFOEXW:
    - wServicePackMajor = 2
    - wServicePackMinor = 0
- **GetVersion** — возвращает DWORD со значением 0x0A280105
- **RtlGetVersion** — возвращает структуру PRTL_OSVERSIONINFOW со следующими значениями:
  - dwMajorVersion = 5
  - dwMinorVersion = 1
  - dwBuildNumber = 2600
  - dwPlatformId = VER_PLATFORM_WIN32_NT = 2
  - szCSDVersion = "Service Pack 2"
  - при передаче структуры RTL_OSVERSIONINFOEXW:
    - wServicePackMajor = 2
    - wServicePackMinor = 0

Приложения, которым для обеспечения работоспособности требуется одна из заплаток в группе xxxVersionLie, чаще всего некорректно выполняют проверку версии операционной системы, например, используют в коде операцию "=" вместо ">=". За исключением очень специфичных случаев следует выполнять проверку версии операционной системы с помощью операции ">=".

Для определения версии операционной системы рекомендуется использовать системную функцию **GetVersionEx()**, которая возвращает информацию об операционной системе, включая информацию об изданиях операционной системы. В приведенной ниже таблице показаны значения для ряда версий операционных систем, возвращаемые функцией **GetVersionEx()**.

ОС	Номер версии	dwMajor-Version	dwMinor-Version	Дополнительно
Windows 7	6.1	6	1	OSVERIONINFOEX.wProductType == VER_NT_WORKSTATION
Windows Server 2008 R2	6.1	6	1	OSVERIONINFOEX.wProductType != VER_NT_WORKSTATION
Windows Server 2008	6.0	6	0	OSVERIONINFOEX.wProductType != VER_NT_WORKSTATION
Windows Vista	6.0	6	0	OSVERIONINFOEX.wProductType == VER_NT_WORKSTATION
Windows Server 2003 R2	5.2	5	2	GetSystemMetrics(SM_SERVERR2) != 0
Windows Server 2003	5.2	5	2	GetSystemMetrics(SM_SERVERR2) == 0
Windows XP	5.1	5	1	-
Windows 2000	5.0	5	0	-

Устаревшие компоненты

Ряд компонентов, которые были доступны в предыдущей версии операционной системы, не поставляется в Windows Vista. К таким компонентам относятся:

- поддержка справочной системы на уровне WinHelp;
- библиотека GINA — Graphical Identification and Authentication — и соответствующие сервисы, позволявшие заменить данный графический интерфейс при запуске WinLogon;
- Logon Notifications — поддержка возможности загрузки динамических библиотек, которые получают сообщения от WinLogon;
- поддержка драйверов принтера в режиме Kernel Mode;
- набор компонентов FrontPage Server Extensions;
- набор компонентов Services for Macintosh;
- D3DRM — библиотека для Direct 3D, обеспечивающая работу в «retained mode»;
- Web Publishing Wizard;
- поддержка динамического обмена сообщениями на уровне NetDDE;
- компоненты Visual Basic 5.0 и Visual C 5.0.

Естественно, что заменить все перечисленные выше компоненты на уровне заплаток не представляется возможным — в каждом отдельном случае следует следовать рекомендациям, опубликованным в базе знаний Microsoft, но две рассматриваемые ниже заплатки могут помочь в тех случаях, когда существуют зависимости от несуществующих в системе сервисов или используется ряд устаревших функций Windows API.

Заплатка **DeprecatedServiceShim** может использоваться в тех случаях, когда возникает ошибка, связанная с тем, что приложение пытается установить сервис, который зависит от несуществующих в системе сервисов. Данная заплатка перехватывает вызов функции **CreateService()** и удаляет ссылку на устаревший сервис из параметра **lpDependencies**.

Заплатку F**ailObsoleteShellAPIs** следует использовать в тех случаях, когда в приложении возникает ошибка, связанная с использованием ряда устаревших функций **Shell API** (например, **FileMenu_***). В зависимости от конкретной функции данная заплатка либо полностью воспроизводит ее поведение, либо возвращает код ошибки. Отметим, что в Windows 7 из комплекта поставки исключен компонент Windows Mail.

Внешний вид системы

Отличия в подсистеме отрисовки в Windows Vista и Windows 7, заключающиеся в том, что приложения используют графический контекст, предоставляемый подсистемой Aero и DWM.EXE (Desktop Window Manager), а не GDI, как это было в предыдущих версиях Windows, могут привести к некорректному отображению ряда приложений — некоторые объекты могут не отрисовываться, тогда как другие могут оставлять после их удаления определенные артефакты.

Системная заплатка **DisableDWM** временно переключает тему на Windows Vista Basic, тем самым запрещая использование DWM и отрисовку на уровне GDI. В большинстве случаев приложения должны быть совместимы DWM — исключение составляют приложения, которые предполагают, что весь экран в их распоряжении и они даже могут осуществлять вывод за границами клиентских частей экрана. Приложениям, которым требуется полный контроль за экраном, например полноэкранным играм, следует использовать функцию **DWMIsCompositionEnabled()** для определения включенного режима композиции, и если этот режим включен, использовать функцию **DWMEnableComposition()** с аргументом **DWM_EC_DISABLECOMPOSITION** для отключения режима композиции.

Некоторые приложения, использующие визуальные стили (Visual Styles), используют основные и фоновые, заданные в виде цветов, а не цветовых пар, выбранных из соответствующего стиля. Под управлением Windows Vista/Windows 7 такие приложения могут выглядеть не так, как под управлением Windows XP, особенно при использования стиля Aero по умолчанию. Помимо этого, некоторые приложения могут поддерживать только темы Windows XP, в которых добавлено больше цветов, чем в визуальных стилях.

Системная заплатка, позволяющая решить перечисленные проблемы, называется **FakeLunaTheme**. Она перехватывает вызов функции **GetCurrentThemeName()** и возвращает тему Luna (**luna.msstyles**), которая является темой по умолчанию для Windows XP.

Упомянем еще две системные заплатки, связанные с отображением: **DisableFadeAnimations** — отключает анимации для приложений, которые могут некорректно их обрабатывать (некорректная отрисовка кнопок и других интерфейсных элементов), а системная заплатка **DisableThemeMenus** (и заплатка **DisableThemes**) позволяет временно отключить темы для меню и других элементов для тех приложений, которые не поддерживают темы.

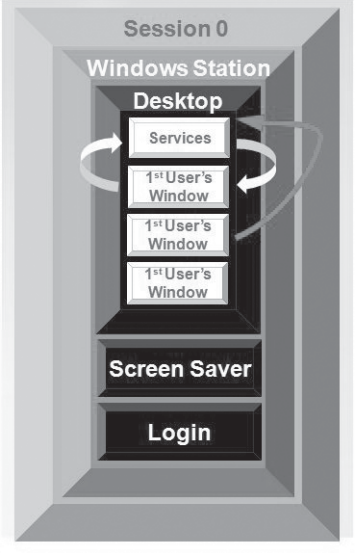
Дополнительные заплатки

В данном разделе мы рассмотрим ряд системных заплаток, которые трудно отнести к какой-то определенной категории.

Системная заплатка **EmulateSorting** может использоваться в тех случаях, когда приложения испытывают проблемы с выполнением функций поиска. Данная заплатка перехватывает следующие функции: **CompareStringW()**, **CompareStringA()**, **LCMapStringW()** и **LCMapStringA()** и заставляет приложения использовать таблицы сортировки от предыдущих версий Windows.

Системная заплатка **SessionShim** решает ряд проблем, связанных с изоляцией нулевой сессии (**Global Session**) и невозможностью приложений, запущенных в последующих сессиях, обмениваться сообщениями сервисами, запущенными в этой сессии.

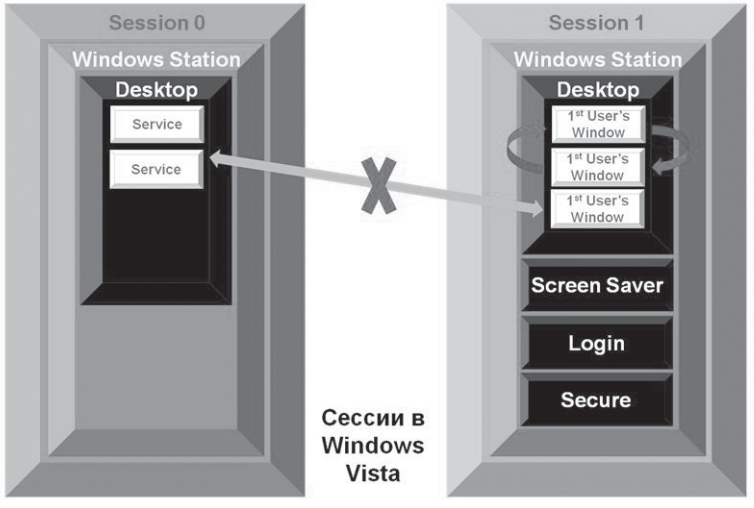
В Windows XP, Windows Server 2003 и более ранних версиях операционной системы Windows, все сервисы выполнялись в одной сессии — в той сессии, которая создавалась для первого пользователя, подключившегося к консоли. Эта сессия называется **Session 0**. Выполнение сервисов и пользовательских приложений в одной сессии может приводить к нарушению безопасности, так как сервисы выполняются с повышенными привилегиями и, таким образом, являются потенциальными жертвами для вредоносного кода.



Потенциальные возможности для атаки в Windows XP

В операционных системах Microsoft Windows Vista и Windows Server 2008 описанные выше возможности для атак со стороны вредоносного кода

сведены к минимуму за счет то, что сервисы изолированы в сессии 0 и, таким образом, сессия, в которой выполняются сервисы, не является интерактивной. В Windows Vista и Windows Server 2008 только системные процессы и сервисы выполняются в сессии 0. Первый пользователь, который подключается к консоли, загружается с сессию 1, следующие пользователи — в сессии с увеличивающимися номерами. Это означает, что сервисы никогда не выполняются в той же сессии, что и пользовательские приложения и, таким образом, защищены от атак со стороны прикладного кода.



Сессии в Windows Vista и Windows Server 2008

Можно привести несколько примеров возможных несовместимостей. Например, в ряде случаев могут возникнуть проблемы с драйверами принтеров, которые загружаются сервисами спулера (spooler service), а также с драйверами, созданными с помощью User Mode Driver Framework (UMDF), так как такие драйвера располагаются в сессии 0. Помимо этого, могут возникнуть проблемы с сервисами, которые создают интерфейсные элементы, сервисами, которые пытаются использовать функции **SendMessage()** и **PostMessage()** для взаимодействия с приложениями, а также с приложениями, создающими объекты с глобальными именами.

Системная заплатка **SessionShim** перехватывает следующие функции:

- **OpenEventA()/OpenEventW()**
- **OpenMutexA()/OpenMutexW()**
- **OpenSemaphoreA()/OpenSemaphoreW()**
- **OpenWaitableTimerA()/OpenWaitableTimerW()**

- **OpenJobObjectA()/OpenJobObjectW()**
- **OpenFileMappingA()/OpenFileMappingW()**

и пытается подменить названия глобальных объектов на локальные.

Следующая группа системных заплаток предназначена для решения проблем, которые могут возникать с приложениями, написанными для Windows 9.x

Заплатка **IgnoreAltTab** позволяет отключить комбинации клавиш **Alt-Tab**, **Ctrl-Esc**, **Alt-Esc** и клавишу **Windows** для приложений, которые могут некорректно обрабатывать эти клавиши.

Заплатка **EmulateGetDiskFreeSpace** служит для корректировки приложений, которые хранят размер дискового пространства в 32-битовой переменной, и при определении размера диска у них происходит ошибка, связанная с переполнением переменной. Данная заплатка возвращает реальное значение, когда размер свободного дискового пространства менее 2 Гбайт и 2 Гбайт в случаях, когда размер свободного дискового пространства превышает это значение.

В тех редких случаях, когда сразу же после запуска приложения оно зависает, можно использовать системную заплатку **IgnoreException**, которая позволяет приложению проигнорировать некоторые исключительные ситуации, которые указаны в параметрах этой системной заплатки. Данная заплатка использует функцию **AddVectoredExceptionHandler()**, которой передается адрес обработчика того или иного исключения, указанного в параметрах заплатки. Таким образом, заплатка **IgnoreException** помогает запустить приложения некорректно обрабатывающие системные исключения.

Параметры системной заплатки **IgnoreException**:

- EXCEPTION1_NAME:EXCEPTION1_LEVEL;EXCEPTION2_ NAME:EXCEPTION2_ LEVEL;...

Имя системного исключения может быть одним из следующих:

- ACCESS_VIOLATION_READ
- ACCESS_VIOLATION_WRITE
- ARRAY_BOUNDS_EXCEEDED
- BREAKPOINT
- DATATYPE_MISALIGNMENT
- FLT_DENORMAL_OPERAND
- FLT_DIVIDE_BY_ZERO
- FLT_INEXACT_RESULT
- FLT_INVALID_OPERATION
- FLT_OVERFLOW

- FLT_STACK_CHECK
- FLT_UNDERFLOW
- ILLEGAL_INSTRUCTION
- IN_PAGE_ERROR
- INT_DIVIDE_BY_ZERO
- INT_OVERFLOW
- INVALID_DISPOSITION
- NONCONTINUABLE_EXCEPTION
- PRIV_INSTRUCTION
- SINGLE_STEP
- STACK_OVERFLOW
- INVALID_HANDLE

Второй параметр, указывающий на уровень системного исключения, может принимать следующие значения:

- 0 — не игнорировать системное исключение;
- 1 — игнорировать первое исключение;
- 2 — игнорировать последующее исключение;
- 3 — завершить процесс после возникновения последующего исключения.

Системная заплатка для внутреннего использования **ShimViaEAT** позволяет попытаться решить проблемы совместимости приложений в тех случаях, когда не удалось использовать описанные выше системные заплатки. Данная заплатка позволяет непосредственно модифицировать таблицу экспорта функций в заголовке загруженного исполняемого файла.

Параметры системной заплатки **ShimViaEAT**:

- ApiToHook,OriginalDllName,OurShimName; ApiToHook2,OriginalDllName2, OurShimName2.

Примеры использования данной системной заплатки:

- GetOpenFileNameW,COMDLG32.dll,XPFileDialog
- RegCreateKeyExA,ADVAPI32.DLL,VirtualRegistry
- RegCreateKeyA,RegCreateKeyW,RegCreateKeyExA,RegCreateKeyExW, RegOpenKeyA,RegOpenKeyW,RegOpenKeyExA,RegOpenKeyExW,RegCloseKey, RegDeleteKeyA,RegDeleteKeyW,RegQueryInfoKeyA,RegQueryInfoKeyW, RegEnumKeyA,RegEnumKeyW,RegEnumKeyExA,RegEnumKeyExW,RegSetValueA, RegSetValueW,ADVAPI32.DLL,VirtualizeHKCRLite

И еще одна внутренняя системная заплатка — **PropagateProcessHistory**, с помощью которой можно «назначить» применение заплаток для дочернего процесса, порождаемого процессом, к которому уже применены

заплатки. Данная системная заплатка устанавливает переменную среды **__PROCESS_HISTORY**, что указывает загрузчику дочернего процесса на необходимость поиска информации о «закладках» в каталогах, связанных с родительским процессом.

Для отладки системных заплаток можно использовать следующий подход:

- установить следующий ключ реестра:
  - [HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags]"ShowDebugInfo"= dword:00000009
- воспользоваться утилитой **DebugView** из набора **Sysinternals** для просмотра отладочной информации;
- для протоколирования действия заплаток нужно установить следующие ключи реестра с помощью указанных команд:
  - reg add "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Environment" /v SHIM_DEBUG_LEVEL /t REG_SZ /d 9 /f
  - reg add "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Environment" /v SHIM_FILE_LOG /t REG_SZ /d logfile.txt /f
- протокол будет располагаться в файле **%appdata%\logfile.txt**

Использование системных заплаток. Практика

В этом разделе мы выполненим практическое упражнение, с помощью которого сможем познакомиться с использованием основных инструментов для обеспечения совместимости приложений без изменения их исходного кода. По завершении данного практического задания мы научимся:

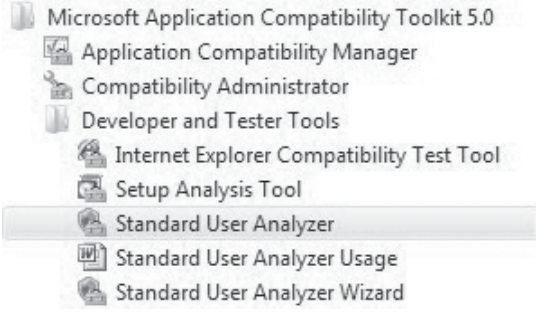
- анализировать приложения для нахождения проблем с совместимостью;
- создавать наборы системных заплаток для устранения обнаруженных проблем;
- развертывать наборы системных заплаток в компании.

Для выполнения данного практического занятия нам потребуется:

- набор утилит Application Compatibility Toolkit 5.0.2
  - можно бесплатно скачать с сайта Microsoft по адресу: **http://go.microsoft.com/fwlink/?LinkID=82101**
- утилита Microsoft Application Verifier 3.4
  - можно бесплатно скачать с сайта Microsoft по адресу: **http://msdl.microsoft.com/download/symbols/debuggers/Private/ApplicationVerifier.x86.msi**
- демонстрационное приложение StockViewer — его можно загрузить по адресу: **http://blogs.msdn.com/cjacks/archive/2008/01/03/stock-viewer-shim-demo-application.aspx**

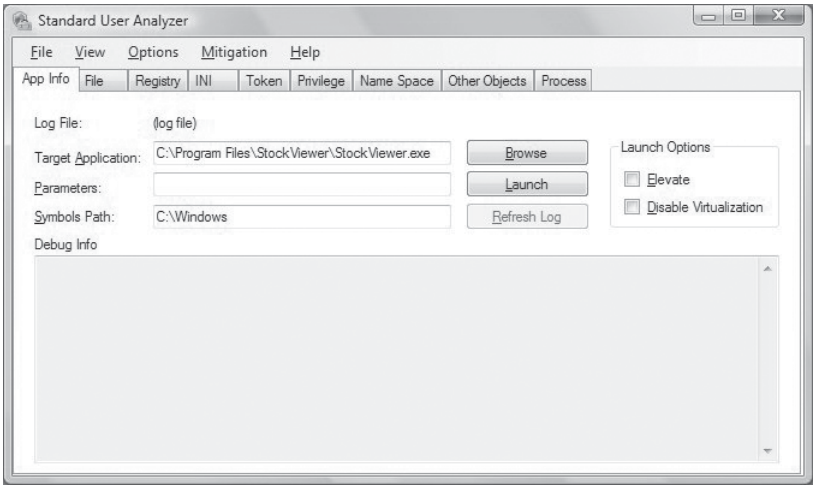
1. Использование Standard User Analyzer

Запустим утилиту Standard User Analyzer, которая входит в состав Application Compatibility Toolkit и расположена в меню Пуск / Все программы / Microsoft Application Compatibility Toolkit / Developer and Tester Tools.



Запуск Standard User Analyzer

На вкладке **AppInfo** выберем команду **Browse**, перейдем в папку **C:\Program Files\StockViewer** и выберем приложение **StockViewer.exe**. В группе **Launch Options** отключим опцию **Elevate** и нажмем кнопку **Launch**.



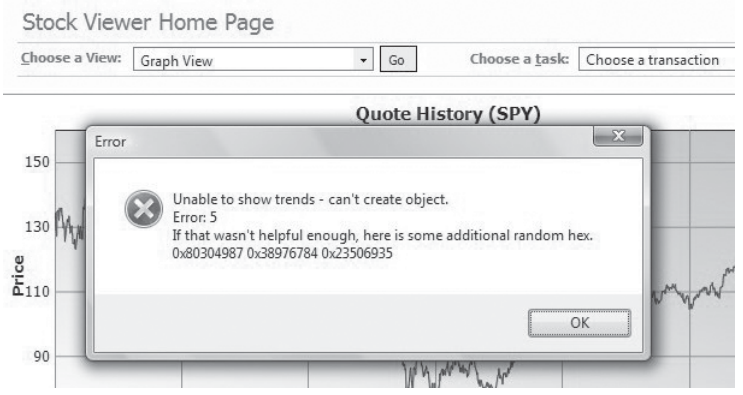
Программа Standard User Analyzer

В диалоговой панели **Warning**, которая предупреждает о том, что все предыдущие протоколы, созданные программой Application Verifier, будут удалены, нажмем кнопку **Ok**. Это приведет к запуску приложения — в диалоговой панели **Permission Denied** нажмем кнопку **Ok**.

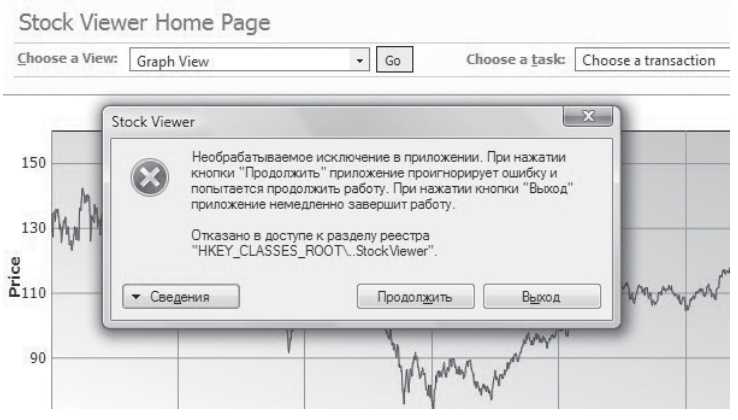
Приложение StockViewer

Затем в самом приложении выполним следующую последовательность действий.

- Нажмем иконку **Trends** и кнопку **Ok**, чтобы закрыть сообщение об ошибке.

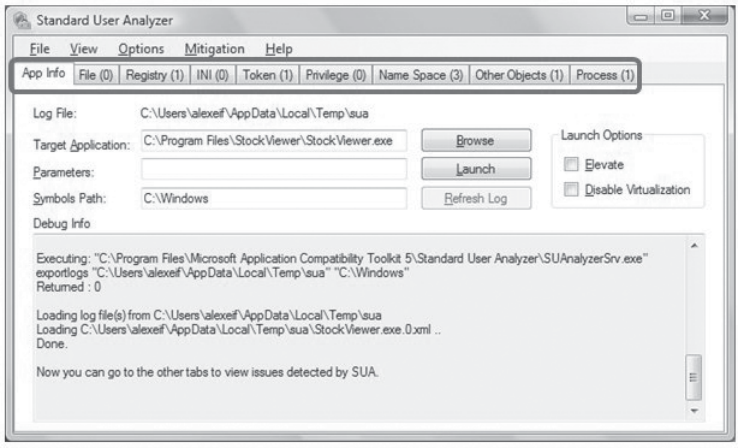


- В меню **Tools** выберем команду **Options** и нажмем кнопку **Продолжить**, игнорируя сообщение об ошибке.



- В меню **Help** выберем команду **Check for Updates** и в панели сообщения об ошибке выберем команду **Выход**, чтобы завершить работу приложения.

Вернувшись в программу Standard User Analyzer, обратим внимание на то, что вкладки **Registry**, **Token**, **Name Space**, **Other Objects** и **Process** содержат информацию.



Результат работы Standard User Analyzer

Вкладка **Registry** содержит следующую информацию:

- Использовалась функция **RegCreateKeyExW()** в режиме виртуализации и предпринималась попытка обратиться к ключам реестра, защищенным Windows Resource Protection — в попытке было отказано.

Вкладка **Token** содержит следующую информацию:

- Приложение использовало функцию **CheckTokenMembership()** для поиска принадлежности текущего пользователя (Security ID) к встроенной группе '**BUILTIN\Администраторы**'.

Вкладка **Name Space** содержит следующую информацию:

- Приложение использовало функции **CreateFileMappingW()** для создания объектов в области «**Global**», что в Windows Vista и Windows 7 запрещено для пользовательских процессов.

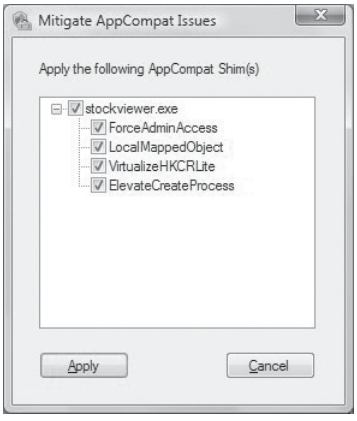
Вкладка **Other Objects** содержит следующую информацию:

- Приложение пыталось создать объект в области «**Global**», используя для этого фукнцию **CreateFileMappingA()**.

Вкладка **Process** содержит следующую информацию:

- Используя функцию **CreateProcessW()**, приложение пыталось запустить процесс "**StockViewerUpdater.exe**" с повышенными привилегиями.

Все эти ошибки были определены Standard User Analyzer в процессе анализа работы приложения. В меню **Mitigation** выберем команду **Apply Mitigations**, в появившейся диалоговой панели посмотрим на системные заплатки, которые рекомендуется применить к нашему приложению для устранения выявленных ошибок, и в диалоговой панели **Mitigate AppCompat Issues** нажмем кнопку **Apply**.



Рекомендованные заплатки для нашего приложения

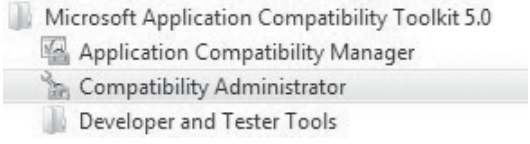
Запустим наше приложение, выполним описанную выше последовательность действий с меню и командами и убедимся в том, что примененные системные заплатки улучшили его работоспособность. У нас появилась новая проблема с командой **Check for Updates**, но ее мы исправим чуть позже.

2. Создание заплаток для улучшения работы приложения

Теперь рассмотрим, как мы сможем использовать утилиту **Compatibility Administrator**, входящую в состав **Application Compatibility Toolkit**, для создания набора заплаток и применения этого набора для устранения несовместимостей нашего приложения с операционной системой.

С помощью программы **Explorer** и на диске **c:** создадим папку **SDBShare**.

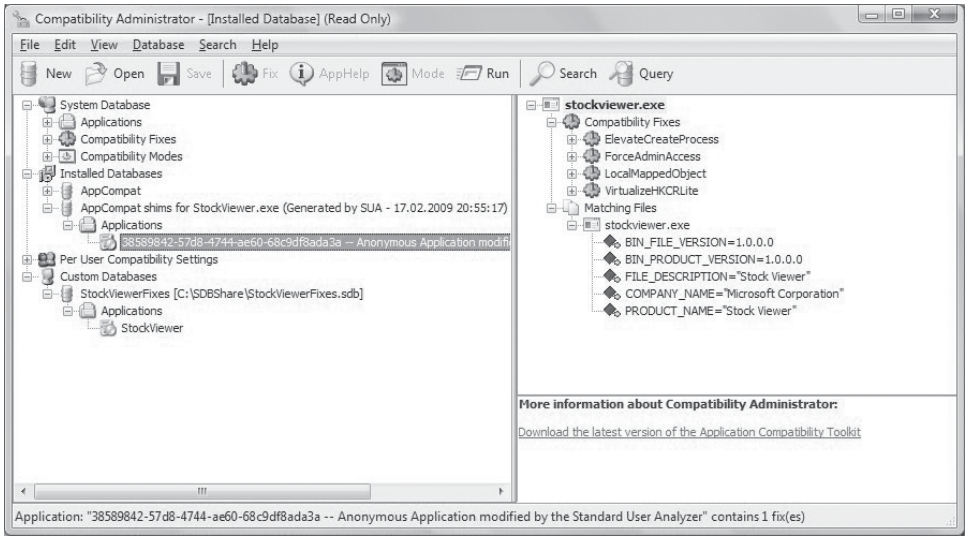
Запустим утилиту **Compatibility Administrator**, которая расположена в меню **Пуск / Все программы / Microsoft Application Compatibility Toolkit**. Утилиту необходимо запустить с правами администратора, поэтому нажмем на ссылку правой кнопкой мыши и выберем команду «**Запуск от имени администратора**».



Запуск Compatibility Administrator

После запуска утилиты **Compatibility Administrator** перейдем в левую панель и выберем ветвь **Installed Databases** и выберем **AppCompat shims for StockViewer.exe** — это набор системных заплаток, автомати-

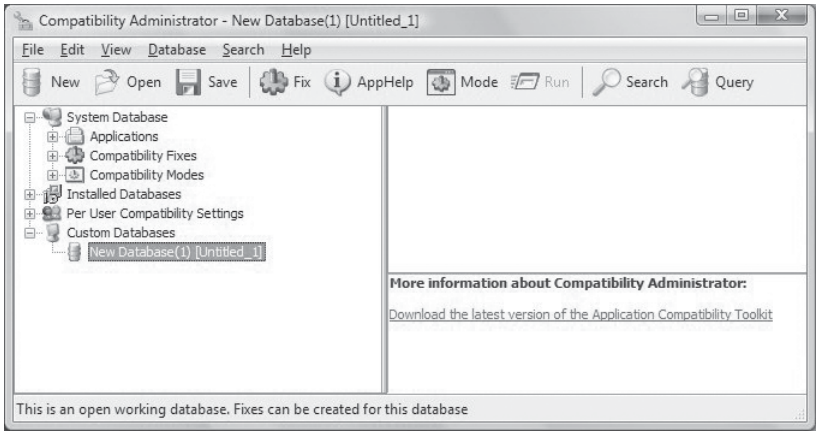
чески примененных к нашему приложению программой **Standard User Analyzer** в результате действий, которые мы предприняли в первой части нашего практического упражнения.



Заплатки, примененные к приложению

В меню File выберем команду Uninstall для того, чтобы удалить все автоматически созданные заплатки для нашего приложения.

Теперь зайдем в раздел **Custom Databases** и выберем элемент **New Databse(1) [Untitled_1]**.

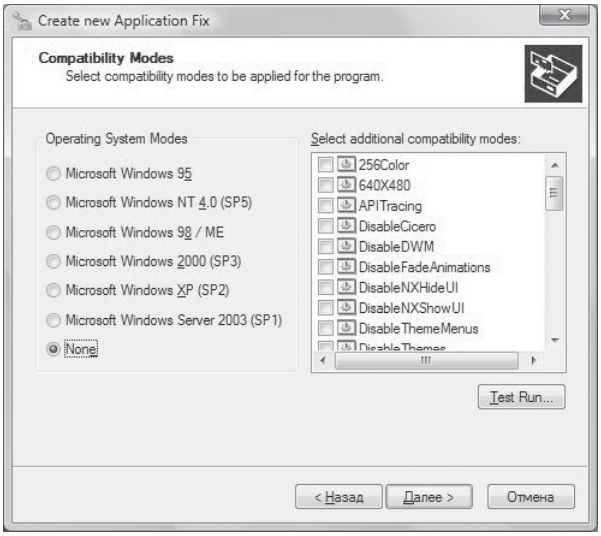


Утилита Compatibility Administrator

В панели команд выберем команду **Fix**, а в «мастере» **Create New Application Fix** зададим значения полей **Name** — **StockViewer** и **Location** —

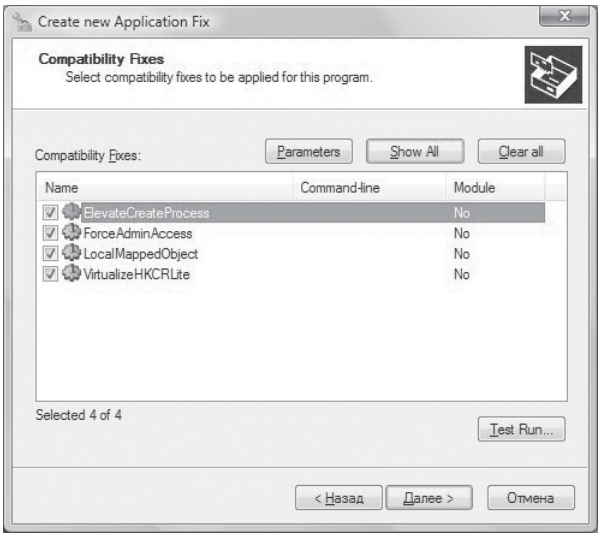
C:\Program Files\StockViewer\StockViewer.exe и после этого нажмем кнопку **Next**.

На экране **Compatibility Modes** в разделе **Operating System Modes** выберем опцию **None** и нажмем кнопку **Next**.



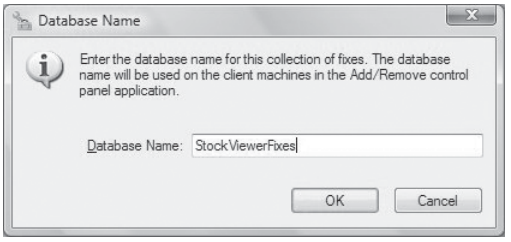
Экран Compatibility Modes

Далее, на странице **Compatibility Fixes** отметим системные заплатки **ElevateCreateProcess**, **ForceAdminAccess**, **LocalMappedObject** и **VirtualizeHKCRLite**, затем нажмем кнопку **Next**.



Экран Compatibility Fixes

На странице **Matching Information** нажмем кнопку **Finish**. После этого, в панели команд выберем «**Save**» и диалоговой панели **Database Name** укажем **StockViewerFixes** и нажмем кнопку **Ok**.



Панель Database Name

В панели **Save Database** укажем имя **c:\SDBShare\StockViewerFixes** и нажмем кнопку «**Сохранить**».

В левой панели в разделе **Custom Databases** выберем элемент **ShockViewerFixes**, затем в меню **File** выберем команду Install и нажмем кнопку **Ok** для закрытия диалоговой панели. Обратим внимание на то, что в разделе **Installed Databases** появился элемент **ShockViewerFixes**.

Выполним манипуляции с нашим приложением для того, чтобы убедиться в работоспособности примененных нами системных заплаток.

3. Run Level Specification

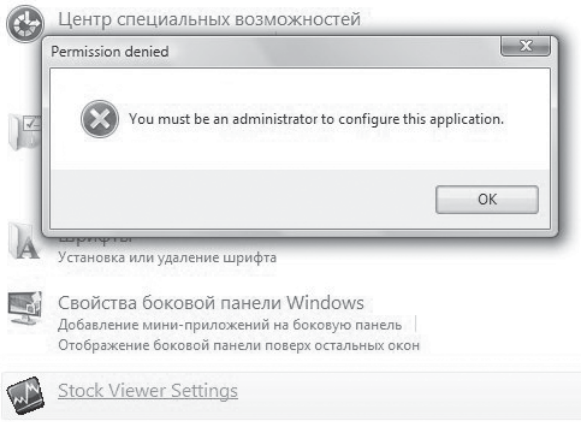
Ниже мы рассмотрим применение системных заплаток для исправления ошибок, связанных с Run Level Specification.

Для обеспечения максимального уровня совместимости с приложениями, написанными для предыдущих версий операционных системы, в Windows Vista и Windows 7 реализовано три уровня определения программ установки — так как большинство инсталляторов требует повышенных привилегий для нормальной работы: **SpecificInstaller**, **GenericInstaller** и **SpecificNonInstaller**. В ряде случаев эвристические механизмы, реализованные в Windows Vista и Windows 7, могут неверно определять программы установки.

Ниже мы рассмотрим, как используя системные заплатки исправить ошибки, возникающие при неверном определении программ установки на примерах программы **StockViewerUpdateOrders.exe** и компонента **StockViewerControlPanel.exe**.

Запустим приложение **StockViewer** и нажмем иконку **Orders**, что приведет к появлению системной панели User Account Control, где будет необходимо подтвердить действия программы **StockViewerUpdateOrders.exe** и по завершении ее работы закрыть основное приложение.

В меню «**Пуск**» выберем элемент **Панель управления**, а в нем — раздел **Оформление и персонализация**. Выберем элемент **Stock Viewer Settings**. Мы получим сообщение об ошибке, указывающее на отсутствие у нас административных прав.



Ошибка — требуются права администратора

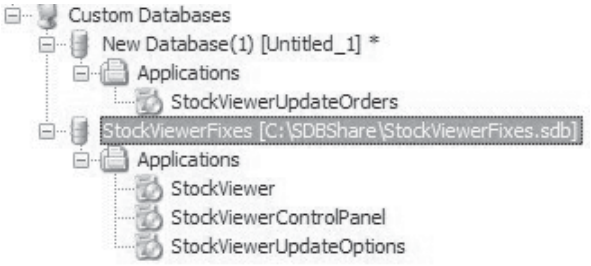
Нажмем кнопку **Ok**, закроем **Панель управления** и запустим программу **Compatibility Administrator**.

Выберем команду **Open** и откроем файл **c:\SDBShare\StockViewerFixes.sdb**. Выберем элемент **StockViewerFixes** и выполним команду Fix. В диалоговой панели укажем **Program Information — StockViewerUpdateOrders**, **Location — c:\Program Files\StockViewer\ StockViewerUpdateOrders.exe** и нажмем кнопку **Next**. В диалоговой панели **Compatibility Modes** выберем опцию **None** и нажмем кнопку **Next**. В панели **Application Fixes** выберем заплатку **SpecificNonInstaller** и нажмем кнопку **Next**. В панели **Matching Information** нажмем кнопку **Finish**.

Мы только что применили системную «заплатку» **SpecificNonInstaller** для программы **StockViewerUpdateOrders**, которая позволит нам запустить ее без повышения привилегий.

Нам осталось добавить еще одну заплатку. Выберем элемент **StockViewerFixes** и выполним команду **Fix**. В диалоговой панели укажем **Program Information — StockViewerControlPanel**, **Location — c:\Program Files\StockViewer\StockViewerControlPanel.exe** и нажмем кнопку **Next**. В диалоговой панели **Compatibility Modes** выберем опцию **None** и нажмем кнопку **Next**. В панели **Application Fixes** выберем заплатку **RunAsAdmin** и нажмем кнопку **Next**. В панели **Matching Information** нажмем кнопку **Finish**.

Наш набор заплаток теперь должен выглядеть так, как показано на следующем рисунке.



Набор заплаток для приложения

Выберем элемент **StockViewerFixes** и в меню **File** выполним команду **Install**. Затем нажмем кнопку **Ok**.

Для проверки наших действий повторим манипуляции с приложением, описанные в начале данного раздела, и убедимся в том, что проблемы с обновлением приложения и его настройкой из Панели управления решены. Отметим, что системная заплатка SpecificNonInstaller привнесла новую ошибку — Access Denied, которая теперь появляется из-за того, что все элементы приложения и его компоненты работают под учетной записью Standard User. Ниже мы решим и эту проблему.

4. Исправление различных ошибок, связанных с UAC

Выше мы столкнулись с проблемой, когда диалоговая панель UAC отображалась в области Taskbar, а не в области основного окна — это связано с тем, что функции **ShellExecute()** передается неверная ссылка на окно (HWND).

В приложении **Compatibility Administrator** выберем **StockViewerUpdateOrders** и удалим этот элемент командой **Edit | Delete**. В правой панели выберем элемент **StockViewer.exe** и выполним команду **Edit | Modify**, затем выберем опцию **Edit Application Fix**. Пропустим две первые панели, нажимая кнопку **Next**. В панели **Compatibility Fixes** добавим заплатку **CorrectShellExecuteHWND**, нажмем кнопку **Next**, затем — кнопку **Finish**.

В панели команд выберем **Save**, затем в левой панели — **StockViewerFixes** и выполним команду **File | Install** и нажмем кнопку **Ok**.

Запустим приложение **StockViewer** и убедимся в том, что теперь панель UAC отображается корректно — благодаря примененной системной заплатке панели UAC передается «правильная» ссылка на окно и панель отображается в нужном месте.

Повторим набор шагов из предыдущего раздела, чтобы восстановить заплатки для программы **StockViewerUpdateOrders.exe**. После этого выполним команды **Save** и **Install**.

Для того чтобы при выполнении команды **Orders** не запрашивалось повышение привилегий, мы используем системную заплатку **RetryOpenSCManagerWithReadAccess**.

В программе **Compatibility Administrator** в левой панели в разделе **Applications** выберем **StockViewerUpdateOrders**. В правой панели выберем элемент **StockViewerUpdateOrders.exe** и выполним команду **Edit | Modify**, затем выберем опцию **Edit Application Fix**. Пропустим две первые панели и в панели **Compatibility Fixes** добавим «заплатку» **RetryOpenSCManagerWithReadAccess**. Нажмем кнопки **Next** и **Finish**. После этого выполним команды **Save** и **Install**.

Запустим приложение **StockViewer** и убедимся в том, что команда **Orders** теперь работает корректно — без ошибок и без запросов на повышение привилегий.

5. Виртуальный реестр

Системная заплатка **VirtualRegistry** используется для имитации доступа к различным элементам реестра. Она может использоваться для перенаправления ключей реестра в другие области (в большинстве случаев это требуется для предотвращения возникновения ошибок типа Access Denied), для возвращения значений из несуществующих ключей, а также в ряде других сценариев.

Запустим приложение **StockViewer** и выполним команду **Browse**. Это приведет к сообщению об ошибке — программе требуется Internet Explorer 6.0, тогда как в состав Windows Vista входит версия 7.0, а в Windows 7 — Internet Explorer 8.0.

В программе **Compatibility Administrator** выберем элемент **Applications | StockViewer**. В правой панели для элемента **StockViewer.exe** выполним команду **Edit | Modify** и выберем опцию **Edit Appication Fix**. Пропустим две первые панели и в панели **Compatibility Fixes** добавим «заплатку» **VirtualRegistry**. Нажмем кнопку **Parameters** и в панели **Options for VirtualRegistry** в строке **Command Line** укажем **IE60**, затем нажмем кнопку **Ok**. Нажмем кнопки **Next** и **Finish**. После этого выполним команды **Save** и **Install**.

Запустим приложение **StockViewer** и выполним команду **Browse**. Теперь, после исправления ошибки, это приведет к появлению Internet Explorer. Закроем браузер и приложение **StockViewer**.

6. Исправление работы с папками

Системная заплатка **CorrectFilePaths** позволяет внести исправления в работу приложений, написанных для предыдущих версий операционной системы Windows. Это относится к обращениям к системным папкам, областям файловой системы и т. п. Имеется возможность сконфигурировать «заплатку» таким образом, что она будет перенаправлять обращения к файлам из одной области в другую и, таким образом, поможет исправить ряд ошибок в приложении.

Запустим приложение **StockViewer** и выполним команду **Options | Save Preferences**. Это приведет к ошибке **Access Denied** — в соответствующей диалоговой панели нажмем кнопку **Quit**.

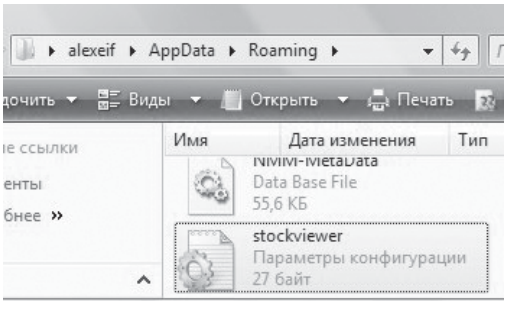
В программе **Compatibility Administrator** выберем элемент **Applications | StockViewer**. В правой панели для элемента **StockViewer.exe** выполним команду **Edit | Modify** и выберем опцию **Edit Appication Fix**. Пропустим две первые панели и в панели **Compatibility Fixes** добавим «заплатку» **CorrectFilePaths**. Нажмем кнопку **Parameters** и в строке **Command Line** укажем (кавычки необходимы!):

```
"c:\windows\downloaded program files\stockviewer.ini;
%userappdata%\stockviewer.ini"
```

затем нажмем кнопку **Ok**. Нажмем кнопки **Next** и **Finish**. После этого выполним команды **Save** и **Install**.

Запустим приложение **StockViewer** и выполним команду **Options | Save Preferences**. Убедимся, что приложение работает без ошибки.

В программе Explorer введем адрес **%appdata%** и убедимся в том, что файл настроек нашего приложения **StockViewer.ini** успешно сохранен.



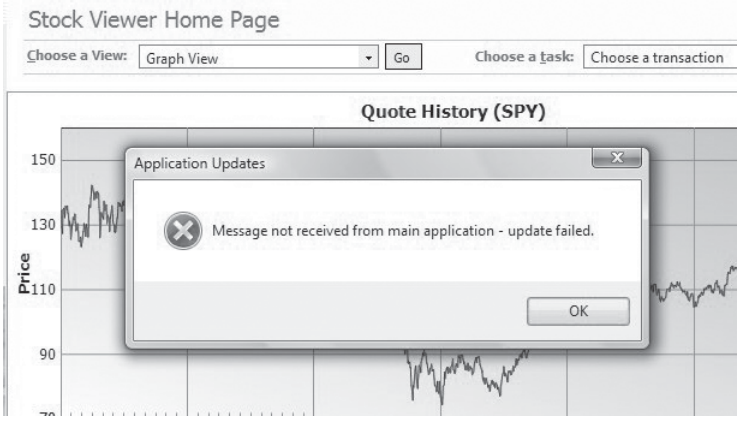
Файл настроек программы StockViewer

7. User Interface Process Isolation

User Interface Process Isolation (UIPI) — это технология, реализованная в Windows Vista и Windows 7, которая защищает и изолирует процессы, работающие на различных уровнях привилегий, от возможности посыл-

ки Windows-сообщений от процесса с низким приоритетом процессу с более высоким приоритетом.

Запустим приложение **StockViewer** и выполним команду **Help | Check for Updates**. Мы получим показанное ниже сообщение об ошибке.



Сообщение об ошибке UIPI

В программе **Compatibility Administrator** выполним команду **Fix**. В диалоговой панели укажем **Program Information — StockViewerUpdater**, **Location — c:\Program Files\StockViewer\StockViewerUpdater.exe** и нажмем кнопку **Next**. В диалоговой панели **Compatibility Modes** выберем опцию **None** и нажмем кнопку **Next**. В панели **Application Fixes** выберем «заплатку» **UIPIEnableStandardMsgs** и при нажатии кнопки **Parameters** укажем **739** в строке **Command Line**, затем нажмем кнопку **Ok**.

---

739 — это десятичное значение сообщения (его шестнадцатиричное значение — 0319). Идентификатор сообщения можно получить с помощью утилиты **Spy++**, входящей в состав Windows SDK.

Нажмем кнопку **Next**. В панели **Matching Information** нажмем кнопку **Finish**.

Запустим приложение **StockViewer** и выполним команду **Help | Check for Updates** и убедимся в том, что приложение работает без ошибок.

Как мы увидели выше, системная заплатка **UIPIEnableStandardMsgs** позволяет разрешить передачу определенных Windows-сообщений процессу, работающему с повышенными привилегиями.

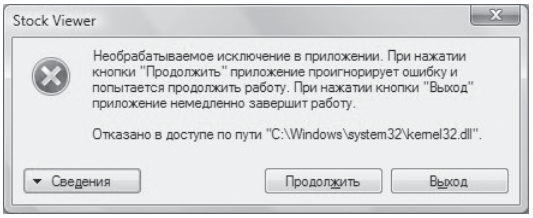
8. Windows Resource Protection

Windows Resource Protection — это механизм, защищающий критические компоненты операционной системы, файлы, ветви реестра от изменений. Только учетная запись **TrustedInstaller** может вносить изменения в эти

области и компоненты — под этой учетной записью, например, работает Windows Update.

Для обеспечения совместимости с приложениями и программами установки, созданными для предыдущих версий операционной системы, которые пытаются изменять системные файлы и не обрабатывают ошибку Access Denied, можно использовать системную заплатку **WRPMitigationSHim**. Данная заплатка просто возвращает код успешно завершенной операции записи, но на самом деле ничего не записывает на диск.

Запустим приложение **StockViewer**, выполним команду **Fille | Update Kernel** и убедимся в том, что эта операция приводит к ошибке Access Denied.



Ошибка Windows Resource Protection

Обычно это происходит в тех случаях, когда приложение пытается заместить какой-то компонент операционной системы более старым, от которого зависит данное приложение — пример показан в расшифровке сообщения об ошибке.

В программе **Compatibility Administrator** выберем элемент **Applications | StockViewer**. В правой панели для элемента **StockViewer.exe** выполним команду **Edit | Modify** и выберем опцию **Edit Appication Fix**. Пропустим две первые панели и в панели **Compatibility Fixes** добавим «заплатку» **WRPMitigation**. Нажмем кнопки **Next** и **Finish**. После этого выполним команды **Save** и **Install**.

Запустим приложение **StockViewer** и выполним команду **File | Update Kernel**. Убедимся, что приложение работает без ошибки.

9. Version Lies

Одной из наиболее частых проблем с запуском приложений, написанных для предыдущих версий операционной системы, является проверка версии ОС. Набор системных заплаток, входящих в группу Version Lies, перехватывает вызов системных функций **GetVersion()** и **GetVersionEx()** и подставляет номер версии, соответствующий ожиданиям приложения.

В следующем примере мы применим такую системную заплатку к приложению StockViewer так, чтобы ему «казалось», что оно работает под управлением Windows XP SP2.

Запустим приложение **StockViewer** и выполним команду **Tools | Show Me a Star**. В панели с сообщением о несоответствии версии нажмем кнопку **Ok**.

Ошибка, связанная с проверкой версии ОС

В программе **Compatibility Administrator** выполним команду **Fix**. В диалоговой панели укажем **Program Information — Star, Location — c:\Program Files\StockViewer\DMW Composing Rendering Demo.exe** и нажмем кнопку **Next**. В диалоговой панели **Compatibility Modes** выберем опцию **None** и нажмем кнопку **Next**. В панели **Application Fixes** выберем заплатку **WinXPSP2VersionLie**. Нажмем кнопку **Next**. В панели **Matching Information** нажмем кнопку **Finish**. После этого выполним команды **Save** и **Install**.

Запустим приложение **StockViewer** и выполним команду **Tools | Show Me a Star**. Убедимся, что приложение работает без ошибки.

10. FakeLunaTheme

Ряд приложений использует цветовые схемы, которые не соответствуют рекомендациям по созданию интерфейсов и основываются на определенных сочетаниях цветов, полученных из тем для рабочего стола в Windows XP. В ряде случаев можно использовать системную заплатку **FakeLunaTheme** для того, чтобы воспроизвестти внешний вид приложения, как если бы оно работало под управлением Windows XP с использованием темы Luna.

Запустим приложение **StockViewer** и нажмем инонку **Email** на панели команд. Закроем приложение **Outlook Look and Feel**. Завершим работу приложения **StockViewer**.

В программе **Compatibility Administrator** выполним команду **Fix**. В диалоговой панели укажем **Program Information — Email, Location — c:\Program Files\StockViewer\OLAF.exe** и нажмем кнопку **Next**. В диалоговой панели **Compatibility Modes** выберем опцию **None** и нажмем кнопку **Next**. В панели **Application Fixes** выберем заплатку **FakeLunaTheme**. Нажмем кнопку **Next**. В панели **Matching Information** нажмем кнопку **Finish**. После этого выполним команды **Save** и **Install**.

Запустим приложение **StockViewer**, нажмем инонку **Email** на панели команд и убедимся, что приложение, имитирующее Outlook, получило новое визуальное представление.

Развертывание заплаток

Сохраненный нами **.sdb**-файл при необходимости можно установить на других компьютерах — это может потребоваться в тех случаях, когда администратор или разработчик создает пакет системных заплаток для приложения, которым пользуются в компании. Для развертывания **.sdb**-файла на других компьютерах его необходимо перенести на соответствующий компьютер и под учетной записью администратора выполнить команду **SDBINST (%windir%\system32\sdbinst.exe)**:

```
C:\>SDBINST c:\SDBShare\ StockViewerFixes.sdb
```

При необходимости такая команда может быть включена в состав логон-скриптов, групповых политик, в состав SMS-пакета или выполнена как часть пакета установки приложения.

Для включения набора заплаток в пакет установки приложения можно использовать следующий скрипт, который затем разворачивается либо средствами SCCM, либо Active Directory Group Policies:

```
msidbCustomActionTypeVBScript + msidbCustomActionTypeInScript +
msidbCustomActionTypeNoImpersonate = 0x0C06
```

Для развертывания .SDB-файла средствами Visual Basic Script можно использовать следующий код:

```
Function Install
 Dim WshShell
 Set WshShell = CreateObject("WScript.Shell")
 WshShell.Run "sdbinst.exe -q " & CHR(34) &
 "c:\sdbshare\stockviewerfixes.sdb" & CHR(34), 0, true
 WshShell.Run "cmd.exe /c " & CHR(34) & "del " & CHR(34) &
 c:\sdbshare\stockviewerfixes.sdb " & CHR(34) & CHR(34), 0
 WshShell.Run "reg.exe delete
 HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\
 {SDB_GUID}.sdb /f", 0
End Function

Function UnInstall
 Dim WshShell
 Set WshShell = CreateObject("WScript.Shell")
 WshShell.Run "sdbinst.exe -q -u -g {SDB_GUID}", 0
End Function
```

Значение **SDB_GUID** можно получить, нажав правую кнопку мыши на соответствующем наборе заплаток в Compatibility Assistant и выбрав команду **Properties**.

Заключение

В данном руководстве мы рассмотрели вопросы, связанные с обеспечением совместимости существующих приложений с операционными системами Microsoft Windows Vista и Microsoft Windows 7. Мы обсудили использование основных системных заплаток, поддерживаемых на уровне Application Compatibility Infrastructure и привели примеры их использования и анализа кода приложения для внесения соответствующих исправлений. Правильным подходом при работе над обеспечением совместимости приложений без изменения исходного кода должна быть следующая последовательность действий:

1) любыми способами «заставить» приложение работать под учетной «standard user»,

2) добиться минимального числа сбоев в работе приложения и

3) оптимизировать работу приложения.

Несмотря на то что инфраструктура Application Compatibility Infrastructure обеспечивает мощный набор заплаток, способных решить множество проблем, связанных с совместимостью приложений, «латание» приложений является промежуточной мерой — компаниям-разработчикам следует стремиться к обеспечению совместимости на уровне кода, используя соответствующие рекомендации и тестовые сценарии.