*Математические структуры и моделирование*
*2017. № 2(42). С. 131–140*

УДК 004.42

# РАЗРАБОТКА СИСТЕМЫ МАНДАТНОГО УПРАВЛЕНИЯ ДОСТУПОМ ДЛЯ ОПЕРАЦИОННЫХ СИСТЕМ СЕМЕЙСТВА WINDOWS

**Д.М. Бречка**
к.т.н., доцент кафедры компьютерных технологий и сетей,
e-mail: dbrechkawork@yandex.ru
**А.А. Литвиненко**
студент, e-mail: a.litvinenko@outlook.com

Омский государственный университет им. Ф.М. Достоевского

**Аннотация.** В статье описывается разработка системы мандатного управления доступом для операционных систем семейства Windows. Предлагается подход, основанный на разработке собственного драйвера мини-фильтра файловой системы, для перехвата обращений пользователей к программам. Описывается интерфейс полученной системы, приводятся результаты тестирования.

**Ключевые слова:** мандатное управление доступом, операционная система, Windows, драйвер, управление доступом, субъекты, объекты.

## Введение

Операционные системы (ОС) семейства Windows NT используют дискреционные политики для управления доступом к файлам, но при этом не предоставляют возможности управления доступом на основе мандатов [1–3]. В то же время мандатное управление доступом (помимо прочего) необходимо для соответствия 4 классу защищённости согласно руководящему документу ФСТЭК «Средства вычислительной техники. Защита от несанкционированного доступа к информации. Показатели защищённости от несанкционированного доступа к информации» [4].

Для соответствия этому требованию ОС должна содержать следующие компоненты:

1. Диспетчер доступа — средство, перехватывающее все обращения субъектов (пользователей) к объектам (файлам) и принимающее решение о разрешении (или запрете) доступа, исходя из заданных мандатных правил разграничения доступа (ПРД). При этом пользователь должен иметь право читать только те файлы, уровень безопасности которых не превышает его собственный уровень безопасности (это обеспечивает защиту информации более высокоуровневых пользователей от доступа со стороны низкоуровневых пользователей), и записывать информацию только в те документы, уровень безопасности которых не ниже его собственного уровня

безопасности (это правило предотвращает нарушение режима доступа со стороны высокоуровневых участников процесса обработки информации к низкоуровневым пользователям).

2. Средство администрирования мандатных ПРД — пользовательский интерфейс, обеспечивающий сопровождение средства защиты, а именно, предоставляющий возможность изменения классификационных уровней субъектов и объектов специально выделенными субъектами.

Целью данной работы является разработка и реализация системы мандатного управления доступом в операционных системах семейства Windows NT.

Разработка и тестирование системы проводились в виртуальной машине с установленной ОС Windows 10 Professional (версия 1511, сборка 10586). В качестве ПО для виртуализации был выбран Oracle VM VirtualBox (версия 5.1.6). В установленной ОС не производилось никаких дополнительных модификаций, помимо описанных в работе.

## 1. Формат и способ обмена данными

В первую очередь, опишем формат и способ хранения и передачи конфигурационных данных об уровнях доступа объектов и субъектов между средством администрирования и диспетчером доступа. Для простоты реализации было принято решение хранить эти данные в отдельных файлах в папке Windows текущей ОС. Список необходимых конфигурационных файлов приведён в таблице 1. Формат конфигурационных файлов приведён в таблице 2.

Таблица 1. Список конфигурационных файлов и их назначение

| **Расположение** | **Имя файла** | **Назначение** |
|---|---|---|
| %WINDIR% | macLevels.dat | Содержит существующие в системе уровни доступа |
| %WINDIR% | macObjects.dat | Содержит сопоставление уровней доступа объектам |
| %WINDIR% | macSubjects.dat | Содержит сопоставление уровней доступа субъектам |

## 2. Безопасность настроек системы защиты

Хранение конфигурации в файлах на жёстком диске влечёт соответствующие проблемы, связанные с безопасностью. Необходимо обеспечить защиту файлов конфигурации от изменения (и удаления) неавторизованными пользователями, эти возможности должны быть предоставлены только специально

Таблица 2. Формат конфигурационных файлов

| **Имя файла** | **Формат** |
|---|---|
| macLevels.dat | Название уровня:{GUID-уровня} |
| macObjects.dat | Полный путь к файлу:{GUID-назначенного-ему-уровня} |
| macSubjects.dat | SID-пользователя:{GUID-назначенного-ему-уровня} |

выделенным субъектам, которые отвечают за администрирование системы защиты (СЗ). В данном случае полагаем, что все администраторы СЗ состоят в локальной группе администраторов, а всем прочим пользователям такой доступ не предоставлен. Тогда необходимо реализовать выполнение двух требований.

1. Запуск интерфейса администрирования должен выполняться только от имени пользователя с правами администратора.
2. Доступ к файлам конфигурации должен контролироваться мандатными или дискреционными политиками, разрешающими их модификацию только пользователям, состоящим в группе локальных администраторов.

Первое требование легко выполняется созданием манифеста для приложения, в котором прописан необходимый уровень запуска [5]:

*<requestedExecutionLevel level="requireAdministrator"uiAccess="false"/>*

Второе требование выполнимо путём изменения списков контроля доступа в конфигурационных файлах соответствующим образом. Стандартная библиотека .NET Framework предоставляет такую возможность. Управление ACL осуществляется путём последовательного вызова следующих методов:

1. *FileSecurity File.GetAccessControl(String filePath)*
   Получает список контроля доступа для файла, путь к которому передан в качестве аргумента filePath в экземпляр класса FileSecurity.
2. *FileSystemAccessRule(IdentityReference identity, FileSystemRights fileSystemRights, AccessControlType type)*
   Конструктор, формирующий экземпляр класса *FileSystemAccessRule* и принимающий в качестве своих аргументов идентификатор безопасности (*identity*) пользователя или группы, к которым применяется правило, тип операции, к которой применяется правило (*fileSystemRights*), и тип этого правила — разрешение или запрет (*type*).
3. *FileSecurity.AddAccessRule(FileSystemAccessRule rule)*
   Добавляет к экземпляру *FileSecurity* новое правило доступа, переданное в качестве аргумента *rule*.
4. *File.SetAccessControl(String filePath, FileSecurity fileSecurity)*
   Применяет список контроля доступа из экземпляра *FileSecurity*, переданного в качестве аргумента *fileSecurity*, к файлу, путь к которому передан в качестве аргумента *filePath*.

Чтобы избежать проблем с открытием или записью конфигурационных файлов в ситуациях, когда доступ к этим файлам, по тем или иным причинам, оказался закрыт даже для группы локальных администраторов, процес-

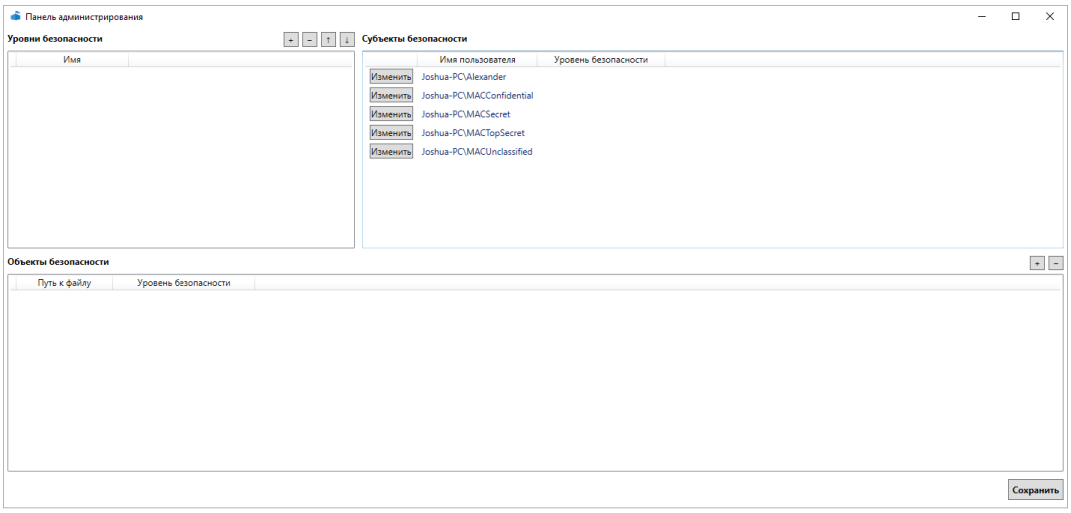


Рис. 1. Главное окно панели администрирования

су панели администрирования необходимо получить от системы привилегию на доступ на запись во все файлы вне зависимости от установленных для них ACL («SeRestorePrivilege»). Для реализации этого необходимо вызывать ряд методов WinAPI, в частности, метод *AdjustTokenPrivileges* из баблиотеки ADVAPI32.DLL, которые необходимо импортировать в программу на C#, подключив пространство имён *System.Runtime.InteropServices*, например, так:

```
[DllImport(ADVAPI32, CharSet = CharSet.Unicode, SetLastError = true)]
internal static extern bool AdjustTokenPrivileges([In] IntPtr TokenHandle,
[In] bool DisableAllPrivileges, [In] ref TOKEN_PRIVILEGE NewState,
[In] uint BufferLength,[In, Out] ref TOKEN_PRIVILEGE PreviousState,
[In, Out] ref uint ReturnLength).
```

## 3. Интерфейс пользователя

Роль панели администрирования выполняет Win32-приложение, написанное на языке C#. Графический интерфейс был реализован на Windows Presentation Foundation с использованием паттерна MVVM. Главное окно панели администрирования (рис. 1) позволяет быстро увидеть все созданные в системе уровни безопасности и их приоритет, список всех пользователей в системе и назначенные им уровни безопасности, а также список всех файлов, которым назначены метки доступа.

Создание новых и редактирование уже существующих уровней и объектов безопасности, а также назначение уровней безопасности субъектам, выполняется с помощью диалоговых окон, вызываемых кнопками «изменить» и «+». Сохранение параметров (т. е. запись конфигурационных файлов на жёсткий диск) происходит при нажатии на кнопку «Сохранить».

## 4. Диспетчера доступа

Диспетчер доступа должен гарантированно перехватывать все запросы к файлам от всех пользователей и принимать решение об их разрешении или запрете. Реализация подобной функциональности неизбежно влечёт за собой необходимость перехвата системных вызовов. Существует несколько коммерческих продуктов, позволяющих перехватывать системные вызовы, одни из них позволяют перехватывать только доступ к объектам файловой системы, другие предоставляют более широкий спектр возможностей. В то же время эти продукты различаются по уровню реализации — одни представляют собой SDK, которые можно использовать при разработке своего продукта, другие же работают на более высоком уровне и скорее являются конечными продуктами. Список официальных инструментов и SDK от Microsoft, позволяющих добиться необходимой функциональности, если отсортировать их по сложности разработки от более сложных к более простым, сводится к [5–7]:

- разработке собственного драйвера файловой системы;
- разработке собственного драйвера фильтра файловой системы;
- разработке собственного драйвера мини-фильтра файловой системы.

Последний подход представляет собой самый простой способ перехвата обращений к файловой системе, особенно по сравнению с первыми двумя вариантами. Microsoft также предоставляет набор примеров мини-фильтров, исходный код которых открыт и выложен в свободный доступ в репозитории на Github [8].

Драйвера мини-фильтров ФС позволяют перехватывать обращение к объектам файловой системы как непосредственно перед самим вызовом, так и после его завершения. Очевидно, что в данном случае необходим перехват обращения до его совершения.

Любой из этих трёх подходов требует Windows Driver Kit, установленного на машине разработчика. Последняя версия WDK, 10.0, позволяет разрабатывать драйвера мини-фильтров файловых систем для всех операционных систем, начиная с Windows 7, а при выборе Windows 10 в качестве целевой платформы ещё и позволяет разрабатывать мини-фильтры одновременно для трёх архитектур процессоров (x86, x64, ARM) и двух платформ (Desktop и Mobile). Но для большей совместимости с выпущенными ранее ОС в качестве целевой ОС была выбрана Windows 7 (что, впрочем, означает, что мини-фильтр должен успешно функционировать и в Windows 8.x и в десктопной Windows 10, при условии соответствия архитектур процессора).

Для успешного построения проекта, первым делом, для всех конфигураций и платформ в решении необходимо в свойствах проекта драйвера мини-фильтра на странице «Driver Signing»→«General» создать новый тестовый сертификат. Также в параметрах проекта необходимо указать целевую ОС и платформу для сборки драйвера.

Пустой проект мини-фильтра уже содержит один файл с расширением «.c» для написания программного кода (этот файл содержит ряд пустых методов-заглушек) и один файл с расширением «.inf», содержащий всю нужную инфор-

мацию о драйвере и необходимый как для сборки проекта, так и для последующей установки драйвера в систему [9]. Для реализации монитора безопасности в файл исходного кода необходимо внести следующие изменения:

1. Раскомментировать содержимое массива Callbacks, тем самым включив перехват всех событий. Каждый элемент массива представляет собой структуру FLT_OPERATION_REGISTRATION с 4 полями — номер перехватываемой функции, дополнительные флаги, метод пре-обработчик, метод пост-обработчик.
2. Изменить логику работы метода пре-обработчика.
3. Реализовать чтение конфигурационных файлов.

**Чтение конфигурации**

Для представления конфигурации были реализованы классы *MACLevel*, *MACObject* и *MACSubject* для уровней, объектов и субъектов безопасности соответственно.

Для чтения конфигурационных файлов были реализованы функции:

- *vector<MACLevel> ReadLevelsFromFile(const char* macLevelPath)*;
- *vector<MACObject> ReadObjectsFromFile(const char* macObjectPath)*;
- *vector<MACSubject> ReadSubjectsFromFile(const char* macSubjectPath)*.

**Обработка событий доступа**

Для получения идентификатора безопасности текущего пользователя была реализована функция *bool GetAccountSid (PSID* psid)*. Для преобразования файловых путей реализована функция *bool GetWin32FileName (const char* nativeFileName, char* *win32FileName)*.

Для определения того, какая операция над файлом в данный момент перехвачена, выполняется проверка значений *Data→Iopb→IrpFlags* и *Data→Iopb→Parameters.Create.SecurityContext→DesiredAccess*. Если *IrpFlags* содержит флаг *IRP_READ_OPERATION*, то выполняется операция чтения, если же *IrpFlags* содержит флаг *IRP_WRITE_OPERATION* и *DesiredAccess* содержит флаг *FILE_APPEND_DATA*, то выполняется операция записи в конец файла.

**Принятие решений**

Для определения, доступен ли файл для чтения/записи для текущего пользователя, реализованы функции:

- *bool CanUserReadFile(PSID userSid, char* filePath)*;
- *bool CanUserAppendFile(PSID userSid, char* filePath)*.

В случае, если результатом вызова функции является значение FALSE, то:

- в поле *Data→IoStatus.Status* устанавливается значение *STATUS_ACCESS_DENIED*;
- в поле *Data→Iopb→TargetFileObject* устанавливается значение *NULL*;
- пре-обработчик возвращает значение *FLT_PREOP_COMPLETE*.

## 5. Использование системы

Для демонстрации работы системы выполним вход под учётной записью с правами администратора и создадим 4 файла, к которым в дальнейшем будут

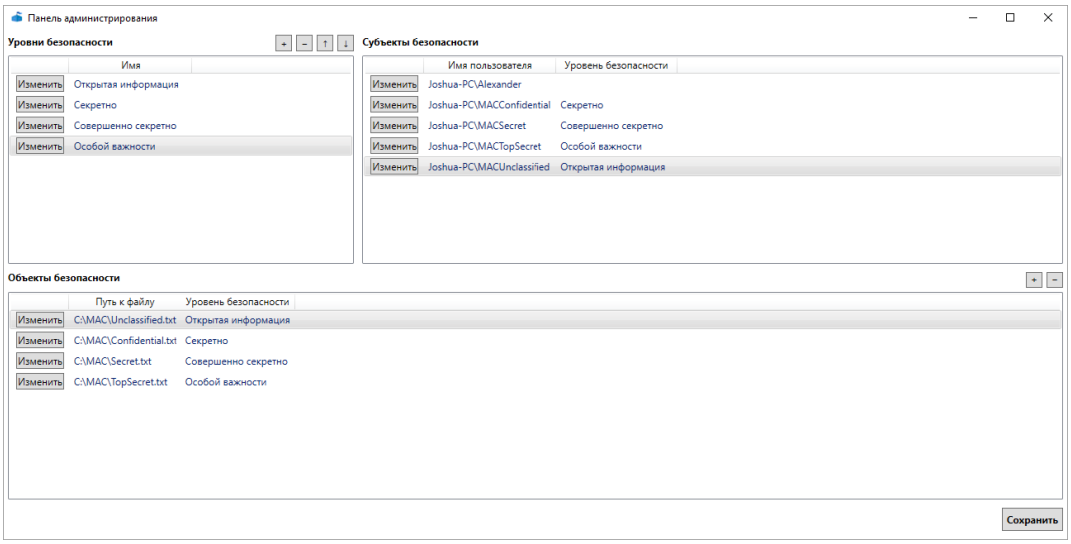


Рис. 2. Панель администрирования с настроенными политиками

применены мандатные политики контроля доступа. Затем запустим панель администрирования и в ней создадим 4 уровня безопасности («Открытая информация», «Секретно», «Совершенно секретно» и «Особой важности»), сопоставим эти уровни соответствующим пользователям и файлам (рис. 2) и сохраним конфигурацию нажатием кнопки «Сохранить».

Получившиеся после выполнения указанных настроек файлы конфигурации будут иметь следующее содержание:

**C:\Windows>type macLevels.dat**
Открытая информация: beeebadf-0793-488d-8c7b-3148b30e6ed4
Секретно: 48e05eaf-dl46-41e1-8765-291be39996a7
Совершенно секретно: 99ee53b8-f4e2-4556-8a49-74a7474c207b
Особой важности: 52f8963c-lal9-409f-9ba0-a5el614bcde0

**C:\Windows>type macSubjects.dat**
S-l-5-21-3757206296-3124401388-62139004-1002:{48e05eaf-dl46-41el-S765-291be39996a7}
S-l-5-21-3757206296-3124401388-62139004-1003:{99ee53b8-f4e2-4556-8a49-74a7474c207b}
S-l-5-21-3757206296-3124401388-62139004-1004:{52f8963c-lal9-409f-9ba0-a5el614bcde0}
S-l-5-21-3757206296-3124401388-62139004-1001:{beeebadf-0793-488d-8c7b-3148b30e6ed4}

**C:\Windows>type macObjects.dat**
C:\MAC\Unclassified.txt:{beeebadf-0793-488d-8c7b-3148b30e6ed4}
C:\MAC\Confidenfial.txt:{48e05eaf-dl46-41el-8765-291be39996a7}
C:\MAC\Secret.txt:{99ee53b8-f4e2-4556-8a49-74a7474c207b}
C:\MAC\TopSecret.txt:{52f8963c-lal9-409f-9baQ-a5el614bcdeQ}

Таблица 3. Матрица доступов

| Объект / Субъект | Unclassified.txt | Confidential.txt | Secret.txt | TopSecret.txt |
|---|---|---|---|---|
| MACUnclassified | rw | w | w | w |
| MACConfidential | r | rw | w | w |
| MACSecret | r | r | rw | w |
| MACTopSecret | r | r | r | rw |

Для проверки корректности работы фильтра теперь достаточно для каждого из четырёх пользователей проверить доступ на чтение и на запись к каждому из четырёх файлов. Исходя из использованной модели безопасности, диаграмма доступа должна выглядеть, как показано в таблице 3.

Для сокращения объёма излагаемого материала приведём результаты проверки работы только для пользователя MACUnclassified. Проверка возможности записи в файл проводится с помощью перенаправления вывода:
*[вывод другой команды]* $\gg$ *[имя файла]*.
Проверка возможности чтения производится с помощью команды *type*:
*type [имя файла]*.

Результаты тестирования:
$C:\backslash MAC > whoami$
$joshua - pc \backslash macunclassified$
*C:\ MAC > type Unclassified.txt*
*Открытая информация*
$C:\backslash MAC > whoami \gg Unclassified.txt$
$C:\backslash MAC$ *type Confidential.txt Access denied*
$C:\backslash MAC > whoami \gg Confidential.txt$
$C:\backslash MAC$ *type Secret.txt*
*Access denied*
$C:\backslash MAC > whoami \gg Secret.txt$
$C:\backslash MAC$ *type TopSecret.txt*
*Access denied*
$C:\backslash MAC > whoami \gg TopSecret.txt$

Следует заметить, что в результате успешной записи никакая информация на экран не выводится. В этом случае очевидно, что результаты тестирования соответствуют первой строке матрицы доступов (табл. 2).

Тестирование прочих пользователей также показало соответствие ожидаемым результатам. Таким образом, можно сделать вывод о корректности работы системы контроля доступа.

## Заключение

В результате выполнения работы был реализован драйвер мини-фильтра файловой системы, выполняющий роль диспетчера доступа, и панель администрирования, обеспечивающая сопровождение СЗ. Это позволило реализовать мандатное управление доступом в ОС Windows 10 (а также в ОС Windows 7 и более новых версиях ОС). Тестирование СЗ полностью совпадает с ожидаемым результатом, что позволяет сделать вывод о корректности реализации монитора безопасности.

## Литература

1. Белим С.В., Бречка Д.М. Исследование безопасности дискреционного разделения доступа в ОС Windows // Математические структуры и моделирование. 2011. Вып. 22. С. 121–130.
2. Бречка Д.М. Разработка программного продукта для анализа безопасности доступа к файлам в операционных системах Windows 7 // Математические структуры и моделирование. 2013. Т. 28, № 2. С. 88–102.
3. Бречка Д.М., Сыргий Е.В. Система составления матрицы доступов запущенных процессов в операционной системе Windows // Вопросы защиты информации. 2014. Вып. 3(106). С. 17–24.
4. Гостехкомиссия России. Руководящий документ «Средства вычислительной техники. Защита от несанкционированного доступа к информации. Показатели защищённости от несанкционированного доступа к информации». М., 1992.
5. Installable file systems driver design guide. URL:`https://msdn.microsoft.com/windows/hardware/drivers/ifs/index` (дата обращения: 27.09.2016).
6. File System Filter Drivers. URL:`https://msdn.microsoft.com/windows/hardware/drivers/ifs/file-system-filter-drivers` (дата обращения: 11.03.2017).
7. Filter Manager Concepts. URL: `https://msdn.microsoft.com/ru-ru/windows/hardware/drivers/ifs/filter-manager-concepts` (дата обращения: 11.03.2017).
8. Driver samples for Windows 10. URL: `https://github.com/Microsoft/Windows-driver-samples` (дата обращения: 11.03.2017).
9. File System Filter Driver Tutorial. URL: `http://www.codeproject.com/Articles/43586/File-System-Filter-Driver-Tutorial` (дата обращения: 11.03.2017).

## CREATING THE MANDATORY ACCESS CONTROL SYSTEM FOR WINDOWS

**D.M. Brechka**
Ph.D. (Eng.), Associate Professor, e-mail: dbrechkawork@yandex.ru
**A.A. Litvinenko**
Student, e-mail: a.litvinenko@outlook.com

Dostoevsky Omsk State University

**Abstract.** The article describes a development of Mandatory Access Control System for Windows operating system. It is proposed the approach of creating the minifilter driver for the file system to intercept application calls. The interface of the Mandatory Access Control system, and the results of testing are described.

**Keywords:** mandatory access control, operating system, Windows, driver, access control, subjects, objects..

*Дата поступления в редакцию: 10.04.2017*