ИВАН КОРОБКО, сертифицированный специалист MCP, автор более 50 статей и двух книг. Занимается созданием различных приложений для Active Directory

Автоматическое управление
распределением прав доступа на файловую систему

Управление правами файловой системы требует большого внимания от исполнителя. Автоматизация этого процесса позволит избежать ошибок и сэкономит время

Создание папки – тривиальная задача. Назначить на созданную папку определенные права – чуть более сложная задача, однако и она не относится к сколько-нибудь сложным. Сформировать сложную файловую структуру, позволяющую решить какие-либо задачи, достаточно непросто. Защита ее от случайного или намеренного изменения – тоже задача не из простых. Проектированию такой системы уделим внимание.

Разрабатывая файловую структуру, помимо стандартного функционала, системному администратору желательно реализовать два решения:

- Обеспечить защиту от случайного перемещения папки или ее переименования, удаления.
- Скрыть от глаз пользователя данные, к которым он не имеет прав доступа. Реализуется настройкой характеристик сетевых папок на файловом сервере с помощью технологии ABE.

Рассмотрим проектирование системы распределения прав на файловую структуру на конкретном примере.

Задача: создать сетевой диск, содержащий дистрибутивы программного обеспечения.

Способы решения задачи

Решить поставленную задачу можно двумя способами: вручную и программно. Оба из них имеют право на жизнь. Именно эту задачу проще всего выполнить вручную, однако цель статьи показать, как реализовать управление правами в автоматическом режиме.

Уделим внимание обоим способам, т.к. перед тем, как приступить к программированию, следует четко понимать, что необходимо сделать.

Управление правами можно реализовать как с помощью COM-объекта, так и библиотек .NET Framework. Поскольку COM-объекты работают медленно и морально устарели, то они рассматриваться не будут.

В качестве языка программирования используем VB.NET. В случае необходимости все примеры легко адаптировать под C#.

Рисунок 1. Защита папки от переименования

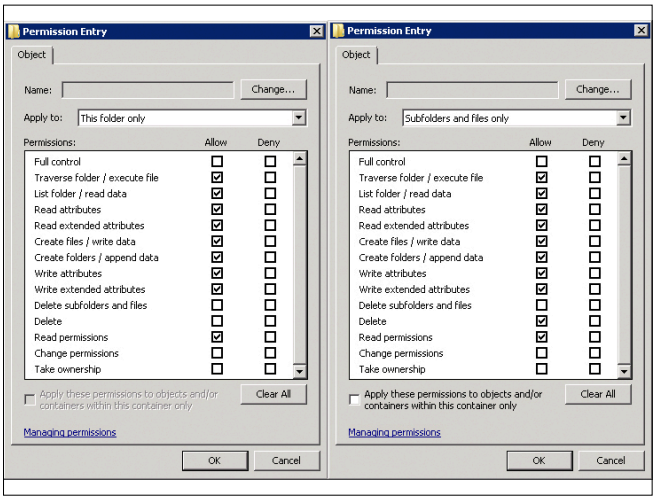
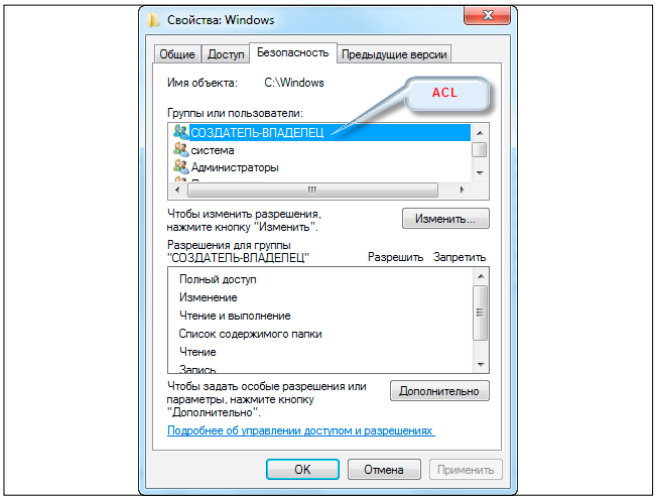


Рисунок 2. Графическое отображение ACL-списка



Рассмотрим проектирование архитектуры файловой структуры со всех сторон, а о технологии ABE расскажем в общих словах, поскольку подробно о программном управлении отображением сетевых папок пользователей рассказано в статье [1].

Проектирование архитектуры файловой структуры

Поставленная задача состоит из двух подзадач. Первая – разработка архитектуры или структуры файловой системы. Вторая – права, назначаемые на каждую из подпапок.

Структура папок может изменяться в зависимости от рода деятельности компании. Однако в любом случае в ней существуют каталоги, на которые необходимо установить защиту от переименования.

Масштабируемость решения реализуется распределением прав доступа не на конкретные учетные записи пользователей, на группы безопасности, которым соответствуют используемые роли, например, члены группы soft_read могут только читать данные, а soft_modify – изменять.

Для обеспечения защиты от переименования папок, имеющих права больше, чем только чтение, необходимо отключить право «Удалить» (Delete) из стандартного «Изменить» (Modify).

В результате проделанных манипуляций необходимо создать два правила, как показано на рис. 1.

На внутреннюю структуру папок установленные права распространяются через механизм наследования папок.

Описатели безопасности

С точки зрения программирования, управляя файловой системой NTFS, выполняем манипуляции со списками ACL и состоящими из них записями ACE.

Набор пользователей и групп, отображаемый во вкладке Security («Безопасность») свойств любой папки, называется ACL-списком (см. рис. 2). С его элементами возможны следующие операции: добавление, удаление и копирование. Каждый из этих элементов называется Access Control Entry (ACE).

Каждая запись характеризуется набором параметров (см. рис. 3):

FileSystemRights. Определяет уровень доступа к ресурсу (FullControl, ReadOnly, Modify и др.). Каждому из них соответствует число (см. таблицу 1);

AccessControlType. Управляет типом доступа, разрешающим (Allow | 0) или запрещающим (Deny | 1) запись данных.

Таблица 1. Возможные значения свойства FileSystemRights

Псевдоним	Значение	Описание
ListDirectory	1	Право на чтение содержимого каталога
ReadData	1	Право открывать и копировать объекты файловой системы (файлы и папки). Не включает в себя право на чтение атрибутов
WriteData	2	Право открытия и записи файла или папки. Не включает в себя право на открытие и запись атрибутов файловой системы
CreateFiles	2	Право на создание файлов
CreateDirectories	4	Право на создание каталогов
AppendData	4	Право добавлять данные в конце файла
ReadExtendedAttributes	8	Право открывать и копировать расширенные атрибуты файловой системы для папок и файлов
WriteExtendedAttributes	16	Право открывать и записывать расширенные атрибуты файловой системы для папок и файлов
Traverse	32	Право получить список содержимого папки и право на запуск приложений, находящихся в этой папке
ExecuteFile	32	Право на запуск файла приложения
DeleteSubdirectoriesAndFiles	64	Право на удаление папки и всего содержимого
ReadAttributes	128	Право открывать и копировать атрибуты файловой системы для папки или файла
WriteAttributes	256	Право открывать и записывать атрибуты файловой системы для папки или файла
Write	278	Право создавать папки и файлы, добавлять или удалять данные в файле
Delete	65536	Право на удаление файлов и папок
ReadPermissions	131072	Право открывать и копировать правила безопасности и аудита для файлов и папок
Read	131209	Право открывать и копировать файлы и папки с разрешениями «только для чтения». Включает в себя ReadData, ReadPermissions, ReadAttributes, ReadExtendedAttributes
ReadAndExecute	131241	Право открывать и копировать файлы и папки, запуска приложения. Включает в себя Read и ExecuteFile
Modify	197055	Право на чтение, запись, получение содержимого папки, удаление файлов и папок, запуск приложений. Включает в себя Read, ExecuteFile, Write и Delete
ChangePermissions	262144	Право на изменение правил безопасности и аудита, для файлов и папок
TakeOwnership	524288	Право изменить владельца файлов и папок. Для изменения владельца необходимо иметь полный доступ к ресурсу
Synchronize	1048576	Определяет возможность синхронизации приложения и дескриптора файла после операции ввода-вывода
FullControl	2032127	Право на полный доступ к файлам и папкам, изменение правил управления доступом и аудита

IdentityReference. Определяет объект, к которому применяется ACE. Все объекты условно делятся на группы и пользователей, на локальные и сетевые. В ACL-списке отображается имя объекта. Имена системных объектов (система, СОЗДАТЕЛЬ-ВЛАДЕЛЕЦ) меняются в зависимости от языковых настроек операционной системы. Чтобы обеспечить масштабируемость решения, используют соответствующие им уникальные идентификаторы безопасности – SID (см. таблицу 2).

IsInherited. Управление наследованием данных (True/False).

InheritanceFlags и PropagationFlags. В совокупности два параметра позволяют управлять областью применения прав (этот раздел и его подразделы, только подразделы и т.д.). Значения, присваиваемые параметру InheritanceFlags, см. в таблице 3, а PropagationFlags – в таблице 4.

Программное управление безопасностью

Во время формирования файловой структуры необходимо обеспечить выполнение ряда действий для назначения желаемого набора прав на папку:

- Отключить наследование прав родительской папки. В противном случае некоторые ACE-элементы удалить будет невозможно.
- Удаление всех элементов из списка за исключением группы Domain Admins, наличие которой является обязательным условием, т.к. с ее помощью осуществляется изменение прав на сервере.
- Создание необходимых ACE-элементов для файловой структуры.

Отключение наследования

Отключение наследования осуществляется с помощью метода SetAccessRuleProtection()[2], который имеет два параметра (см. рис. 4). Первый из них отвечает за управление наследованием (FALSE – разрешает, TRUE – запрещает), второй – за его сохранение (TRUE – сохраняет, FALSE – удаляет). Пример использования метода приведен в листинге 1.

Листинг 1. Отключение наследования прав на папку

```
Function RemoveInherit(ByVal Path As String)
    Dim dSecurity As DirectorySecurity = ↵
        System.IO.Directory.GetAccessControl(path)
    dSecurity.SetAccessRuleProtection(True, True)
    System.IO.Directory.SetAccessControl(path,dSecurity)
End Function
```

В приведенном примере в функцию RemoveInherit передается UNC-путь. Затем создается объект dSecurity, содержащий описатель безопасности указанной папки.Далее осуществляются отключение наследования с копированием прав с родительского каталога и запись сделанных изменений в файловую структуру.

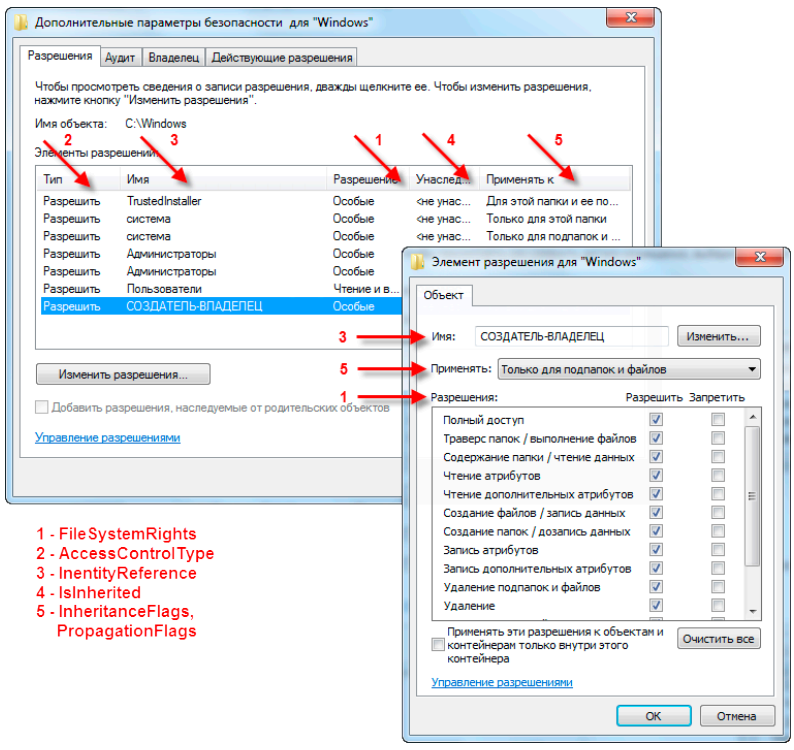
Удаление ACE-элементов

Удаление ACE-элементов выполняется с помощью метода RemoveAccessRule(), в качестве его параметра указывается правило (FileSystemAccessRule). В листинге 2 приведен пример удаления правила [3].

Листинг 2. Удаление указанного правила

```
Dim rule As New System.Security.AccessControl. ↵
```

Риунок 3. Графическое отображение ACE-списка



```
        FileSystemAccessRule(Domain + "\Domain Admins", ↵
        2032127, 3, 0, 0)
seq.RemoveAccessRule(rule)
System.IO.Directory.SetAccessControl(Path, seq)
```

Обычно неизвестно, какие правила назначены на указанного пользователя, и задача звучит несколько иначе: удалить все правила для учетной записи. В этом случае необходимо перебрать все учетные записи пользователей и удалить соответствующие заданному критерию (см. листинг 3).

Листинг 3. Удаление всех правил для указанного пользователя

```
For Each ace As FileSystemAccessRule ↵
        In seq.GetAccessRules(True, True, ↵
        GetType(System.Security.Principal.NTAccount))
    If StrComp(UCase(domain) + "\Domain Users"), ↵
        UCase(ace.IdentityReference.Value.ToString)) = 0 ↵
                Then
        seq.RemoveAccessRule(ace)
        System.IO.Directory.SetAccessControl(Path, seq)
    End If
Next
```

Вспомним, что изначально стоит задача удалить все учетные записи, кроме группы Domain Admins, членам которой должен быть обеспечен доступ к файловой структуре в полном объеме. Поскольку неизвестно, находится ли эта группа в ACL-списке и какие она имеет права, рекомендуется перед удалением всех элементов из списка, кроме указанного, в любом случае создать новое правило (см. листинг 4).

Листинг 4. Удаление всех правил, кроме группы Domain Admins

```
Sub Clear(ByVal Path As String)
Dim seq As System.Security.AccessControl. ↵
        DirectorySecurity = System.IO.Directory. ↵
        GetAccessControl(Path)
Dim rule As New System.Security.AccessControl. ↵
        FileSystemAccessRule(domain + "\Domain Admins", ↵
        2032127, 3, 0, 0)
seq.AddAccessRule(rule)
System.IO.Directory.SetAccessControl(Path, seq)

For Each ace As FileSystemAccessRule ↵
        In seq.GetAccessRules(True, True, ↵
        GetType(System.Security.Principal.NTAccount))
    If StrComp(UCase(domain) + "\Domain Admins"), ↵
        UCase(ace.IdentityReference.Value.ToString)) <> 0 ↵
                Then
        seq.RemoveAccessRule(ace)
        System.IO.Directory.SetAccessControl(Path, seq)
End If
Next
End Sub
```

Создание ACE-элементов

Для создания ACE-элемента используется метод AddAccess Rule [4] (см. листинг 5). Приведенный сценарий работает по следующему алгоритму: сначала получают описатель безопасности папки, права которой необходимо изменить (переменная seq).

Далее создается правило безопасности (rule), указывая значения пяти ранее описанных параметров, и добавляем его в полученный ACL-список.

Затем выполняем запись сделанных изменений с помощью метода SetAccessControl.

Листинг 5. Создание ACE-правила в ACL-листе

```
Sub SetProperty(FileSystemRights, AccessControlType, ↵
        IdentityReference, IsInherited, InheritanceFlags, ↵
        PropagationFlags)
    Dim seq As System.Security.AccessControl. ↵
        DirectorySecurity = System.IO.Directory. ↵
        GetAccessControl(path)
    Dim rule As New System.Security.AccessControl. ↵
        FileSystemAccessRule(FileSystemRights, ↵
        AccessControlType, IdentityReference, IsInherited, ↵
        InheritanceFlags, PropagationFlags)
    seq.AddAccessRule(rule)
```

Таблица 2. Стандартные учетные записи

Описание в ACL-списке	Значение IdentityReference	SID	Описание
СОЗДАТЕЛЬ-ВЛАДЕЛЕЦ (Creator Owner)	СОЗДАТЕЛЬ-ВЛАДЕЛЕЦ	S-1-3-0	Стандартная учетная запись Creator Owner
Система	NT AUTHORITY/Система	S-1-5-18	Встроенная локальная учетная запись System
1619pc/Администраторы (1619pc/Administrators)	BUILTIN/Администраторы	S-1-5-32-544	Встроенная локальная группа Administrators
1619pc/Пользователи (1619pc/Users)	BUILTIN/Пользователи	S-1-5-32-545	Встроенная локальная группа Users
ОГРАНИЧЕННЫЕ (Restricted Code)	NT AUTHORITY/ОГРАНИЧЕННЫЕ	S-1-5-12	Запрещает доступ. Зарезервирован для использования в будущем
Все (Everyone)	NT AUTHORITY/Все	S-1-1-0	Встроенная локальная группа Everyone

Таблица 3. Значения, принимаемые InheritanceFlags

Псевдоним	Значение	Описание
None	0	Флаги наследования не установлены
ContainerInherit	1	Наследуются контейнеры
ObjectInherit	2	Наследуются объекты

Таблица 4. Значения, принимаемые PropagationFlags

Псевдоним	Значение	Описание
None	0	Флаги наследования не установлены
NoPropagateInherit	1	Не распространяет наследование
InheritOnly	2	Наследование распространяется только на дочерние объекты

```
    System.IO.Directory.SetAccessControl(path, seq)
End Sub
```

Многократное выполнение данного листинга с разным набором «путь-права» позволит создать сколь угодно сложную структуру папок с необходимыми правами доступа.

Устанавливаемые настройки необходимо считывать из какого-либо внешнего источника данных. Из разнообразных способов рекомендуется остановить выбор на XML-файле (см. листинг 6), который имеет древовидную структуру. Создав один раз правильно такой файл, его можно использовать многократно.

Листинг 6. Пример статического XML-файла

```
<?xml version="1.0" encoding="utf-8" ?>
<Permissions>
  <Folder>
<Path>\\ESMIRALDA\SOFTWARE$\Операционные системы</Path>
<Sequrity>
<Object>\\ISLAND\DOMAIN USERS</Object>
<Mask>1179817</Mask>
<Type>0</Type>ы
<IFlag>1</IFlag>
<PFlag>0</PFlag>
</Sequrity>
  </Folder>
</Permissions>
```

Хранилище настроек безопасности

Для описания иерархической структуры оптимально использовать XML-файл, позволяющий формировать не стандартную систему безопасности.

Например, после приема нового сотрудника на работу системному администратору требуется создать на сервере персонализированный набор папок. Согласитесь, что использовать для каждого нового пользователя персональный XML-файл, в котором будет изменяться только его сетевое имя, – нерациональное решение. Наращивать существующий XML-файл – верный путь запутаться.

Логично использовать псевдонимы (см. листинг 7), которые при выполнении сценария на лету будут преобразовываться в соответствующие значения. В приведенном скрипте необходимо заменить $DOMAIN на заранее определенное значение (см. таблицу 5).

Листинг 7. Пример динамического XML-файла

```
<?xml version="1.0" encoding="utf-8" ?>
<Permissions>
  <Folder>
<Path>\\ESMIRALDA\SOFTWARE$\Операционные системы</Path>
<Sequrity>
<Object>\\$DOMAIN\DOMAIN USERS</Object>
<Mask>1179817</Mask>
<Type>0</Type>
<IFlag>1</IFlag>
<PFlag>0</PFlag>
</Sequrity>
  </Folder>
</Permissions>
```

К написанию псевдонимов логично предъявить два требования:

Он должен содержать идентификатор. В данном случае это символ доллара, иначе псевдоним не будет найден соответствующим обработчиком.

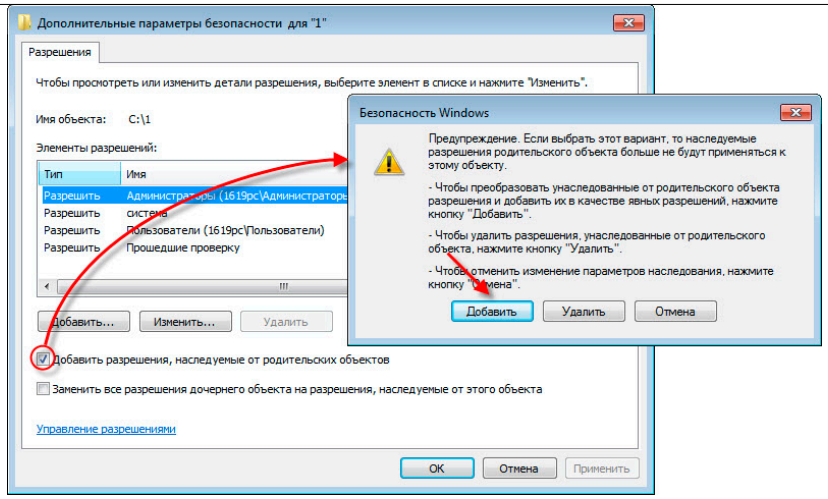
В файле он указывается прописными буквами. Данное требование не является обязательным, однако его выполнение поможет человеку быстро найти псевдонимы в тексте XML-файла в случае необходимости.

Преобразование псевдонимов осуществляется с помощью функции Replace(), которой передается строка, содержащая его имя (см. листинг 8).

Листинг 8. Преобразование псевдонимов

```
Function Replace(ByVal Str As String) As String
    Try
        Str = Str.Replace(UCase("$DOMAIN"), domain)
        Str = Str.Replace(UCase("$DESCRIPTION"), ↵
                description)
        Str = Str.Replace(UCase("$DEPARTMENT"),department)
        Str = Str.Replace(UCase("$ROOT"), root)
        Return Str
    Catch ex As Exception
```

Риунок 4. Управление наследованием



```
        Trace.Warn("ERROR: " + ex.Message.ToString)
    End Try
End Function
```

Проанализировав листинг функции, убедимся, что подстановка значений осуществляется с помощью многократного вызова метода Replace(). Заменяемые характеристики должны быть определены перед первым вызовом функции. Можно создать соответствующую функцию или определить их значения в процедуре Page_Load().

Рекомендуется использовать второй способ (см. листинг 9), добавив условие IsPostBack, позволяющее загружать данные при первом запуске страницы.

Листинг 9. Определение значений для замены псевдонимов

```
Public domain As String
Public description As String
Public department As String
Public root As String = "…"
Public userName As String = "…" 'значение поля sAMAccountName

Protected Sub Page_Load(ByVal sender As Object, ↵
        ByVal e As System.EventArgs) Handles Me.Load
    If IsPostBack = False Then
        Dim objAD As New DirectoryEntry("LDAP://RootDSE")
        Dim LDAPDomain As String = "LDAP://" + ↵
            objAD.Properties("DefaultNamingContext").Value
        domain = LDAPDomain.Replace("DC=","").Split(",")(0)
        Dim search As New DirectorySearcher()
        search.SearchRoot = New DirectoryEntry(LDAPDomain)
        Dim query As String = "(&(objectclass=person) ↵
            (sAMAccountname=" + userName + "))"
        search.Filter = query
        Dim Result As SearchResult = search.FindOne()
        description = Result. GetDirectoryEntry. ↵
            Properties("description").Value.ToString()
        department = Result. GetDirectoryEntry. ↵
            Properties("department").Value.ToString()
    End If
End Sub
```

Чтение данных динамического XML-файла

Чтение XML-файла реализовано с помощью метода Load() встроенного .NET объекта XmlDocument (см. листинг 10). Затем осуществляется перебор всех тегов. При нахождении тега <Folder> выполняется чтение вложенных в него тегов с преобразованием встречающихся псевдонимов в путях и названиях групп безопасности. После этого данные записываются в многомерный динамический массив.

Листинг 10. Чтение данных из динамического XML-файла

```
Dim ACE_Array()()

Sub ReadXML(ByVal XMLFileName As String)
    Try
        Dim i As Integer = 0
        Dim xDoc As New XmlDocument
        xDOC.Load("file:///" + XMLFileName)

        Dim Path, Obj, Mask, Type, IFlag, PFlag As String
        For Each b As XmlNode In a.DocumentElement.ChildNodes
            Select Case UCase(b.Name)
                Case UCase("folder")
                    For Each c As XmlNode In b.ChildNodes
                        Dim temp As String = ↵
                            c.InnerText.ToString
                        Select UCase(c.Name)
                            Case UCase("Path")
                                Path = Replace(temp)
                            Case UCase("Object")
                                Obj = Replace(temp)
                        End Select
                    Next
                ReDim Preserve ACE_Array(i)
                ACE_Array(i) = New String() {Path, Obj, ↵
                        Mask, Type, IFlag, PFlag}
                i ++
            End Select
        Next
    Catch ex As Exception
        Trace.Warn("ERROR: " + ex.Message.ToString)
    End Try
End Sub
```

Назначение прав на основе данных из XML-файла

Формирование файловой структуры на основе считанных данных состоит из нескольких этапов (листинг 11):

- Создание папки, если она не создана. Отметим, что файловые операции выполняются небыстро, поэтому необходимо убедиться, что операция закончена. В противном случае часть манипуляций с назначением прав не будет завершена.
- Отключение наследования прав с родительской папки с помощью метода SetAccessRuleProtection.
- Удаление всех ACE-элементов, кроме группы Domain Admins.
- Назначение прав в соответствии с данными, полученными из XML-файла.

Листинг 11. Создание файловой структуры и назначение на нее прав

```
Sub SetFSOPermissions()
Try
For i As Integer = 0 To ACE_Array.GetUpperBound(0)
    ' Создание папки
    If FileIO.FileSystem.DirectoryExists( ↵
        ACE_Array(i)(0))=False Then
        Do Until FileIO.FileSystem.DirectoryExists( ↵
                ACE_Array(i)(0))
            FileIO.FileSystem.CreateDirectory(ACE_Array(i)(0))
        Loop
    End If

    ' Проверка изменения пути папки
    If i > 0 And StrComp(ACE_Array(i)(0), ↵
        ACE_Array(i - 1)(0) > 0) Then
        RemoveInherit(ACE_Array(i)(0))
        Clear(ACE_Array(i)(0))
    End If
```

Таблица 5. Список псевдонимов для замены данных

Переменная	Описание
$domain	NetBIOS имя домена, в который совершил вход пользователь
$department	Подразделение, к которому относится пользователь. Определяется на основе поля department свойств учетной записи в каталоге Active Directory
$FIO	Полное ФИО сотрудника. Считывается из свойств учебной записи. Поскольку зарезервированного поля нет, рекомендуется использовать для хранения данных поле description
$ROOT	Путь к корневой папке создаваемой структуры

```
    ' Назначение прав на папку
    SetProperty(ACE_Array(i)(0).ToString, p, ↵
        ACE_Array(i)(2), ACE_Array(i)(3), ↵
        ACE_Array(i)(4), ACE_Array(i)(5))

Catch ex As Exception
    Trace.Warn("ERROR: " + ex.Message.ToString)
End Try
End Sub
```

В результате выполнения данного сценария осуществляется создание иерархической структуры папок и назначение в ней прав в соответствии с заданными в XML-файле правами. В приведенном листинге использован вызов процедур из 1, 4 и 5.

Управление отображением папок и файлов для пользователя

Начиная с Windows Server 2008 Microsoft встроил в операционную систему приложение, позволившее системным администраторам скрывать от глаз пользователя данные, на которые он не имеет прав, с помощью отличной технологии ABE (Access Based Enumeration). До этого его можно было установить на Windows Server 2003 SP1 в качестве надстройки.

Access Based Enumeration является серверным компонентом и может быть активирован только на папки, предоставленные в общее использование.

Активация ABE выполняется с помощью консоли Server Manager в разделе Share and Store Management (см. рис. 3), где необходимо вызвать свойства нужной сетевой папки и в отобразившемся окне нажать на кнопку Advanced, затем активировать опцию Enable Access Based Enumeration.

***

Автоматическое управление правами позволит администратору не только автоматически сформировать сложную файловую структуру с распределенной системой назначения прав, но и в случае сбоя легко восстановить, трансформировать имеющуюся структуру файлов и папок. Реализация такого функционала возможна с помощью универсального XML-файла, в котором описаны правила безопасности файловой системы в общем виде.

Применение ABE даст возможность скрыть от глаз пользователя лишнюю информацию, к которой у него нет прав доступа. EOF

1. Access Based Enumeration. Установка и управление. //«Системный администратор», №11, 2010 г. – С. 38-44.
2. Метод ObjectSecurity.SetAccessRuleProtection – http://msdn.microsoft.com/en-us/library/system.security.accesscontrol.objectsecurity.setaccessruleprotection.aspx.
3. Метод ObjectSecurity.RemoveAccessRule – http://msdn.microsoft.com/en-us/library/system.security.accesscontrol.filesystemsecurity.removeaccessrule.aspx.
4. Метод ObjectSecurity.AddAccessRule – http://msdn.microsoft.com/en-us/library/system.security.accesscontrol.filesystemsecurity.addaccessrule.aspx.

Рисунок 5. Управление ABE в Windows Server 2008