ЛЕКЦИЯ №4

«Управление доступом»

по дисциплине

«Безопасность операционных систем»

Текст лекции рассмотрен и одобрен на заседании кафедры протокол № _____ от " " 201__ г.

(Слайд 1. Титульный слайд)

Уважаемые студенты! Сегодня вы продолжаете изучение дисциплины «Безопасность операционных систем». Лекция №4 «Управление доступом». Продолжительность лекции - 4 академических часа.

Слайд 2 (план проведения занятия)

Наш курс состоит из лекций и практических занятий по отдельным подсистемам и механизмам безопасности, реализованным в ОС.

В данной лекции мы начнем изучать "Управление доступом". В частности: Понятие управления доступом. Модели управления доступом (Take Grant, Бела-Лападулы). Реализации мандатного управления доступом в современных ОС (например, SELinux, AppArmor). Реализации управления доступом в защищённых операционных системах МСВС, Заря, Astra-Linux. Аппаратные средства контроля доступа.

Понятие управления доступом.

После выполнения идентификации и аутентификации необходимо установить полномочия (совокупность прав) субъекта для последующего контроля санкционированного использования вычислительных ресурсов.

Такой процесс называется разграничением (логическим управлением) доступа.

В руководящих документах выделяется два вида (принципа) разграничения доступа:

- дискреционное (избирательное) управление доступом;
- мандатное (принудительное) управление доступом.

Дискретное управление доступом - разграничение доступа между поименованными субъектами и поименованными объектами.

Субъект с определенным правом доступа может передать это право любому другому субъекту.

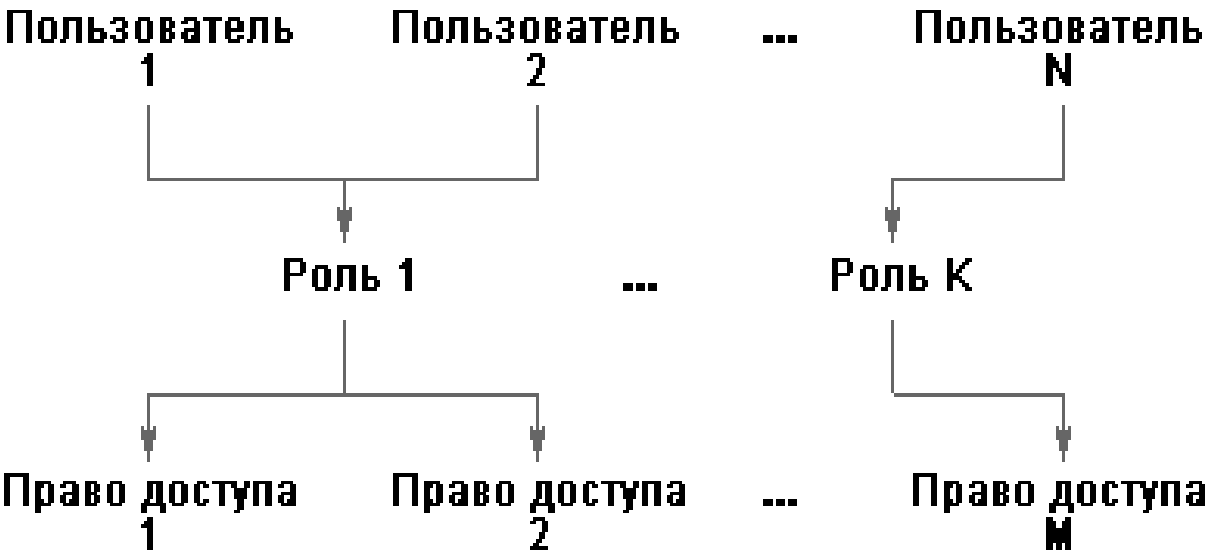
Данный вид организуется на базе методов разграничения по спискам или с помощью матрицы.

Возможны несколько подходов к построению дискреционного управления доступом:

- Каждый объект системы имеет привязанного к нему субъекта, называемого **владельцем**. Именно владелец устанавливает права доступа к объекту.
- Система имеет одного выделенного субъекта — суперпользователя, который имеет право устанавливать права владения для всех остальных субъектов системы.
- Субъект с определенным правом доступа может передать это право любому другому субъекту.

Например, **-rw-r--r-- 1 root root 1810 авг. 4 22:53 /etc/passwd**

Управление доступом на основе ролей (Role Based Access Control, RBAC) — развитие политики избирательного управления доступом, при этом права доступа субъектов системы на объекты группируются с учетом специфики их применения, образуя роли.



Мандатное управление доступом регламентирует разграничение доступа субъектов к объектам, основанное на характеризуемой меткой конфиденциальности информации, содержащейся в объектах, и официальном разрешении (допуске) субъектов обращаться к информации такого уровня конфиденциальности.

Проект АНБ SELinux реализация мандатного контроля доступа в ядре Linux.

Мандатная система разграничения доступа реализована в ОС FreeBSD Unix.

AppArmor - архитектура мандатного контроля доступа для SUSE Linux и Ubuntu.

В Oracle Database есть подсистема Oracle Label Security.

Существуют различные формальные модели для анализа систем управления доступом:

- мандатная (модель Белла— Лападулы);
- дискретная (модель Take-Grant).

Модели управления доступом.

Дискреционное управление доступом - это метод ограничения доступа к объектам, который основан на том, что некоторый субъект (обычно владелец объекта) может по своему усмотрению давать другим субъектам или отбирать у них права доступа к объекту.

Текущее состояние прав доступа в случае дискреционного управления доступом описывается матрицей, в строках которой перечислены субъекты, а в столбцах - объекты (таблица 1). В клетках, расположенных на пересечении строк и столбцов, записываются способы доступа для субъекта по отношению к объекту, например: чтение, запись, выполнение, возможность передачи прав другим субъектам и т.д.

Прямолинейное представление подобной матрицы невозможно (поскольку она очень большая), да и не нужно (поскольку она разрежена, то есть большинство клеток в ней пусто). В операционных системах более компактное представление матрицы доступа основывается либо на структуризации совокупности субъектов (владелец/группа/другие у ОС UNIX), либо на механизме списков управления доступом (ACL, Access Control List), то есть на представлении матрицы по столбцам, когда для каждого объекта перечисляются субъекты вместе с их правами доступа. За счет использования метасимволов можно компактно описывать группы субъектов, удерживая тем самым размеры ACL в разумных пределах.

Каждая колонка в матрице может быть реализована как список доступа для одного объекта.

Очевидно, что пустые клетки могут не учитываться. В результате для каждого объекта имеем список упорядоченных пар <субъект, набор полномочий>, который определяет все субъекты с непустыми наборами прав для данного объекта.

Большинство операционных систем (Windows, Linux) и систем управления базами данных реализует именно дискреционное управление доступом и, как правило, на механизме ACL.

Главное его преимущество - гибкость.

Главные недостатки - рассредоточенность управления и сложность централизованного контроля, а также оторванность прав доступа от данных, что позволяет копировать секретную информацию в общедоступные файлы.

Многоуровневые модели предполагают формализацию процедуры назначения прав доступа посредством использования так называемых меток конфиденциальности или мандатов, назначаемых субъектам и объектам доступа.

Так, для субъекта доступа метки, например, могут определяться в соответствии с уровнем допуска лица к информации, а для объекта доступа (собственно данные) – признаками конфиденциальности информации. Признаки конфиденциальности фиксируются в метке объекта.

Права доступа каждого субъекта и характеристики конфиденциальности каждого объекта отображаются в виде совокупности уровня конфиденциальности и набора категорий конфиденциальности. Уровень конфиденциальности может принимать одно из строго упорядоченного ряда фиксированных значений, например: конфиденциально, секретно, для служебного пользования, не секретно и т.п.

Основу реализации управления доступом составляют:

1. Формальное сравнение метки субъекта, запросившего доступ, и метки объекта, к которому запрошен доступ.

2. Принятие решений о предоставлении доступа на основе некоторых правил, основу которых составляет противодействие снижению уровня конфиденциальности защищаемой информации.

С помощью многоуровневых моделей возможно существенное упрощение задачи администрирования. Причем это касается как исходной настройки разграничительной политики доступа (не требуется столь высокого уровня детализации задания отношения субъект-объект), так и последующего включения в схему администрирования новых объектов и субъектов доступа.

Самое важное достоинство заключается в том, что пользователь не может полностью управлять доступом к ресурсам, которые он создаёт.

Такая система запрещает пользователю или процессу, обладающему определённым уровнем доверия, получать доступ к информации, процессам или устройствам более защищённого уровня.

Механизм управления доступом реализует на практике некоторую абстрактную (или формальную) модель, определяющую правила задания разграничительной политики доступа к защищаемым ресурсам и правила обработки запросов доступа к защищаемым ресурсам.

Модель Гогена-Мезигера

Модель Гогена-Мезигера (Goguen-Meseguer), представленная ими в 1982 году, основана на теории автоматов. Согласно этой модели система может при каждом действии переходить из одного разрешенного состояния только в несколько других. Субъекты и объекты в данной модели защиты разбиваются на группы — домены.

Переход системы из одного состояния в другое выполняется только в соответствии с так называемой таблицей разрешений, в которой указано, какие операции может выполнять субъект, например, из домена C над объектом из домена D. В данной модели при переходе системы из одного разрешенного состояния в другое используются транзакции, что обеспечивает общую целостность системы.

Модель Take-Grant

Модель Take-Grant - это формальная модель, используемая в области компьютерной безопасности, для анализа систем дискреционного разграничения доступа; подтверждает либо опровергает степени защищенности данной автоматизированной системы, которая должна удовлетворять регламентированным требованиям. Модель представляет всю систему как направленный граф, где узлы - либо объекты, либо субъекты.

Дуги между ними маркированы, и их значения указывают права, которые имеет объект или субъект (узел). В модели доминируют два правила: "давать" и "брать". Они играют в ней особую роль, переписывая правила, описывающие допустимые пути изменения графа.

В общей сложности существует 4 правила преобразования:

- правило "брать";
- правило "давать";
- правило "создать";
- правило "удалить";

Используя эти правила, можно воспроизвести состояния, в которых будет находиться система в зависимости от распределения и изменения прав доступа. Следовательно, можно проанализировать возможные угрозы для данной системы.

O - множество объектов (файлы, сегменты памяти и т.д.)

S - множество субъектов (пользователи, процессы системы)

R = {r1, r2, r3, r4, ..., rn } U {t,g} - множество прав доступа

t [take] - право брать "права доступа"

g [grant] - право давать "права доступа"

G = (S, O, E) - конечный, помеченный, ориентированный граф без петель.

× - объекты, элементы множества O

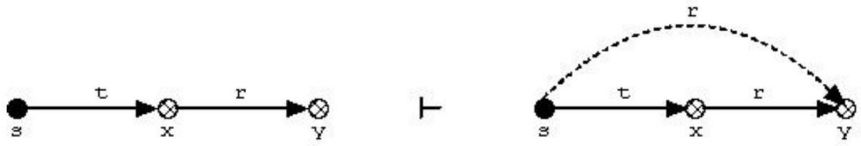
• - субъекты, элементы множества S

E ∈ O x O x R - дуги графа. Состояние системы описывается ее графом.

Правило "Брать"

Брать = take(r, x, y, s), r ∈ R,
Пусть s ∈ S, x,y ∈ O - вершины графа **G**

Тогда граф G:

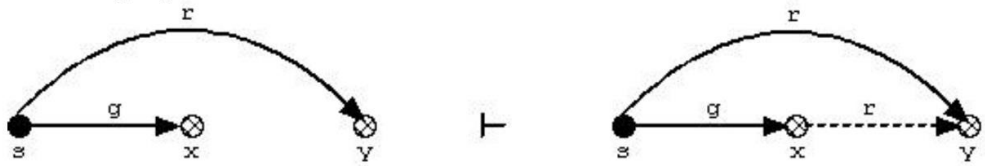


Т.е. субъект *S* берет у объекта *X* права r на объект Y.

Правило "Давать"

Давать = grant(r, x, y, s), r ∈ R,
Пусть s ∈ S, x,y ∈ O - вершины графа **G**

Тогда граф G:



Т.е. субъект *S* дает объекту *X* права r на объект Y.

Правило "Создать"

Создать = create(r, x, s), r ∈ R,

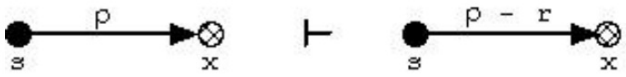
Пусть s ∈ S, x,y ∈ O - вершины графа **G**
Тогда граф G:

Т.е. субъект *S* берет r-доступный объект Y.

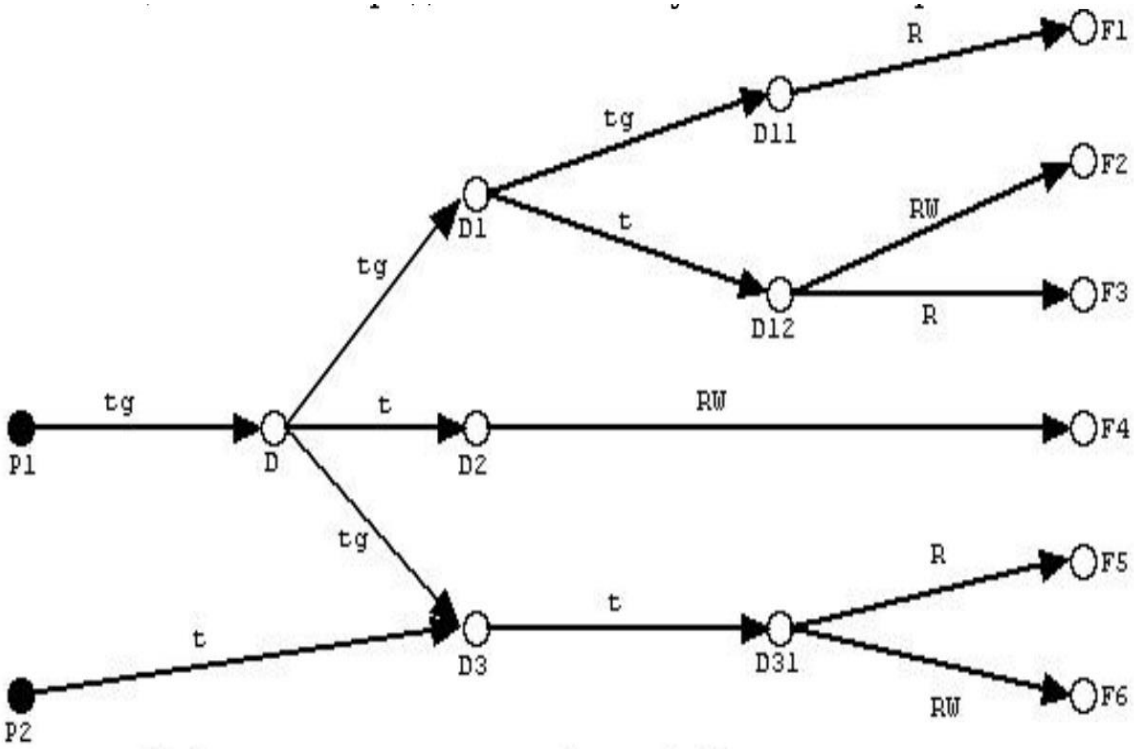
Правило "Удалить"

Удалить = remove(r, x, s), r ∈ R,
Пусть s ∈ S, x,y ∈ O - вершины графа **G**

Тогда граф G:

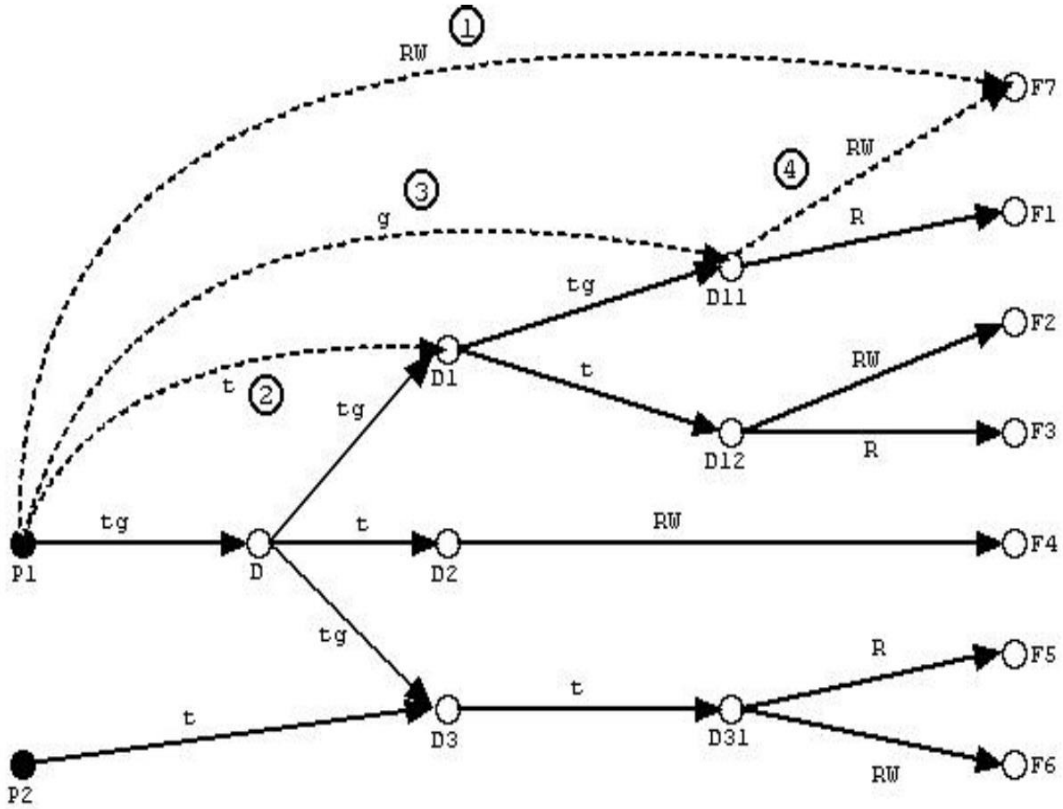


На рисунке ниже, показано графическое представление структуры каталогов. В этом графе P1 и P2 составляют субъекты (возможные пользователи), а также Ds и Fs представляют объекты, каталоги и файлы, соответственно. Право "чтение" было изменено на правило "брать" для всех уровней, за исключением фактически файловых уровней в каталогах. Право "записи" также изменилось правилом "давать". Становится ясно из этого графа: если субъект имеет право чтения(брать) объекта, то он может иметь право чтения любых других объектов, на которые этот первый объект имеет какие-то права. Аналогично, если субъект имеет право записи (давать) объекта, он может предоставить любую из своих прав на этот объект.



Take-grant representation of directory structure

На рисунке ниже показано, что посредством комбинации из указанных выше четырех правил, новый файл может быть добавлен в каталог структуры Примера 1. И права на чтение/запись, будут назначены согласно правилам использующимся каталогом, в которой этот файл записывается.



Adding a new file F7 to directory D11

На данный момент эта модель практически не используется в программном обеспечении. В основном, данная модель используется для анализа уязвимости различных систем.

Модель Белла — Лападулы

Модель Белла — Лападулы — модель контроля и управления доступом, основанная на мандатной модели управления доступом. В модели анализируются условия, при которых невозможно создание информационных потоков от субъектов с более высоким уровнем доступа к субъектам с более низким уровнем доступа.

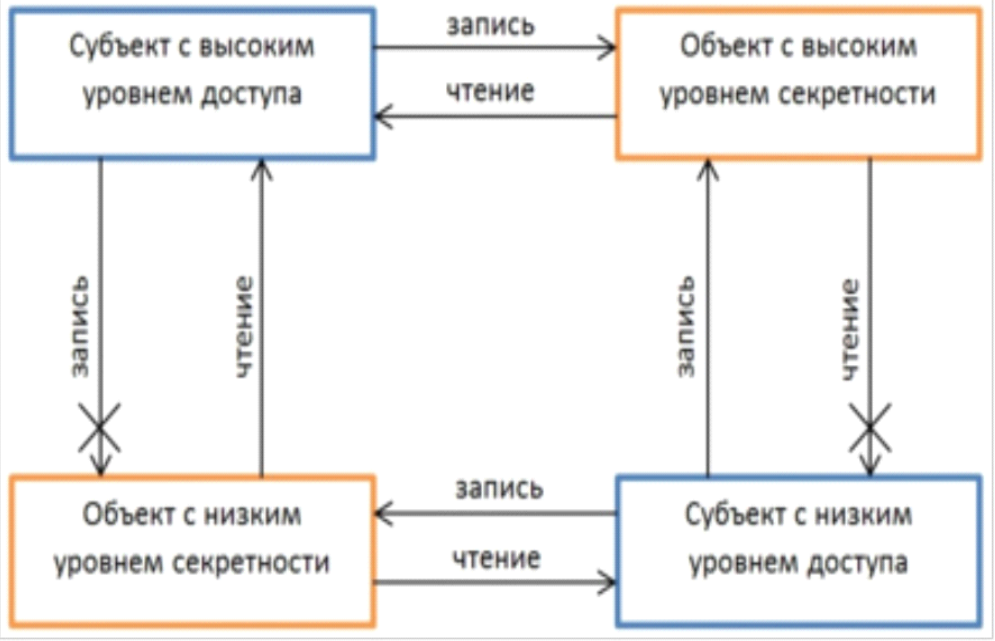


Диаграмма информационных потоков

Модель Белла — Лападулы является моделью разграничения доступа к защищаемой информации. Она описывается конечным автоматом с допустимым набором состояний, в которых может находиться информационная система. Все элементы, входящие в состав информационной системы, разделены на две категории — субъекты и объекты. Каждому субъекту присваивается свой уровень доступа, соответствующий степени конфиденциальности. Аналогично, объекту присваивается уровень секретности. Понятие защищённой системы определяется следующим образом: каждое состояние системы должно соответствовать политике безопасности, установленной для данной информационной системы. Переход между состояниями описывается функциями перехода. Система находится в безопасном состоянии в том случае, если у каждого субъекта имеется доступ только к тем объектам, к которым разрешен доступ на основе текущей политики безопасности. Для определения, имеет ли субъект права на получение определенного вида доступа к объекту, уровень секретности субъекта сравнивается с уровнем

секретности объекта, и на основе этого сравнения решается вопрос, предоставить или нет запрашиваемый доступ.

Простое свойство безопасности (The Simple Security)

Субъект с уровнем доступа может читать информацию из объекта с уровнем секретности тогда и только тогда, когда преобладает над . Это правило также известно под названием «нет чтения верхнего» (NRU).

Например, если субъект, имеющий доступ только к несекретным данным, попытается прочесть объект с уровнем секретности совершенно секретно, то ему будет отказано в этом.

Свойство * (The *-property)

Субъект с уровнем секретности может писать информацию в объект с уровнем безопасности только если преобладает над . Это правило также известно под названием «нет записи вниз» (NWD). Например, если субъект, имеющий уровень доступа совершенно секретно, попытается записать в объект с уровнем секретности секретно, то ему будет отказано в этом.

Дискреционное свойство безопасности (The Discretionary Security Property)

Заключается в том, что права дискреционного доступа субъекта к объекту определяются на основе матрицы доступа.

Система в модели Белла — Лападулы состоит из следующих элементов:

— начальное состояние системы;

— множество прав доступа;

— функция перехода, которая в ходе выполнения запросов переводит систему из одного состояния в другое.

Состояние системы называется **безопасным по чтению** (или simpleбезопасным), если для каждого субъекта, осуществляющего в этом состоянии доступ по чтению к объекту, уровень безопасности субъекта доминирует над уровнем безопасности объекта

Состояние системы называется **безопасным по записи** (или * — безопасным) в случае, если для каждого субъекта, осуществляющего в этом состоянии доступ по записи к объекту, уровень безопасности объекта доминирует над уровнем безопасности субъекта.

Система называется **безопасной**, если её начальное состояние v0 безопасно, и все состояния, достижимые из v0 путём применения конечной последовательности запросов из R, безопасны.

Понятие авторизации.

Authorization (авторизация, проверка полномочий, проверка уровня доступа) — сопоставление учётной записи в системе (и персоны, прошедшей аутентификацию) и определённых полномочий (или запрета на доступ). В общем случае авторизация может быть «негативной» (пользователю А запрещён доступ к серверам компании).

Управление доступом в Windows

После аутентификации пользователя процессом Winlogon, все процессы, запущенные от имени этого пользователя будут идентифицироваться специальным объектом, называемым ***маркером доступа (access token)***. Если процесс пользователя запускает дочерний процесс, то его маркер наследуются, поэтому маркер доступа олицетворяет пользователя для системы в каждом запущенном от его имени процессе. Основные элементы маркера представлены на рисунке.

SID пользо-вателя	**SID1 … SIDn** **Идентификаторы групп пользователя**	**DACL по умолчанию**	**Привилегии**	**Прочие па-раметры**

Маркер доступа содержит идентификатор доступа самого пользователя и всех групп, в которые он включен. В маркер включен также DACL по умолчанию - первоначальный список прав доступа, который присоединяется к создаваемым пользователем объектам. Еще одна важная для определения прав пользователя в системе часть маркера – список его привилегий.

Привилегии - это права доверенного объекта на совершение каких-либо действий по отношению ко всей системе.

Имя и идентификатор привилегии	Описание привилегии
Увеличение приоритета диспетчирования SeIncreaseBasePriorityPrivilege	Пользователь, обладающий данной привилегией может изменять приоритет диспетчирования процесса с помощью интерфейса Диспетчера задач
Закрепление страниц в памяти SeLockMemoryPrivilege	Процесс получает возможность хранить данные в физической памяти, не прибегая к кэшированию данных в виртуальной памяти на диске.
Управление аудитом и журналом безопасности SeAuditPrivilege	Пользователь получает возможность указывать параметры аудита доступа к объекту для отдельных ресурсов, таких как файлы, объекты Active Directory и разделы реестра.
Овладение файлами или иными объектами SeTakeOwnershipPrivilege	Пользователь получает возможность становиться владельцем любых объектов безопасности системы, включая объекты Active Directory, файлы и папки NTFS, принтеры, разделы реестра, службы, процессы и потоки
Завершение работы системы SeShutdownPrivilege	Пользователь получает возможность завершать работу операционной системы на локальном компьютере
Обход перекрестной проверки SeChangeNotifyPrivilege	Используется для обхода проверки разрешений для промежуточных каталогов при проходе многоуровневых каталогов

Остальные параметры маркера носят информационный характер и определяют, например, какая подсистема создала маркер, уникальный идентификатор маркера, время его действия. Необходимо также отметить возможность создания ***ограниченных маркеров (restricted token)***, которые отличаются от обычных тем, что из них удаляются некоторые привилегии и его SID-идетификаторы проверяются только на запрещающие правила. Создать ограниченный маркер можно программно, используя API-функцию CreateRestrictedToken, а можно запустить процесс с ограниченным маркером, используя пункт контекстного меню Windows ***“Запуск от имени…”*** и отметив пункт ***“Защитить компьютер от несанкционированных действий этой программы”***

Маркер доступа может быть создан не только при первоначальном входе пользователя в систему. Windows предоставляет возможность запуска процессов от имени других пользователей, создавая для этих процессов соответствующий маркер.

Для этих целей можно использовать:

- API-функции CreateProcessAsUser, CreateProcessWithLogon;
- оконный интерфейс, инициализирующийся при выборе пункта контекстного меню “Запуск от имени…”;
- консольную команду runas:

runas /user:имя_пользователя program,

где *имя_пользователя* - имя учетной записи пользователя, которая будет использована для запуска программы в формате пользователь@домен или домен\пользователь;

program – команда или программа, которая будет запущена с помощью учетной записи, указанной в параметре /user.

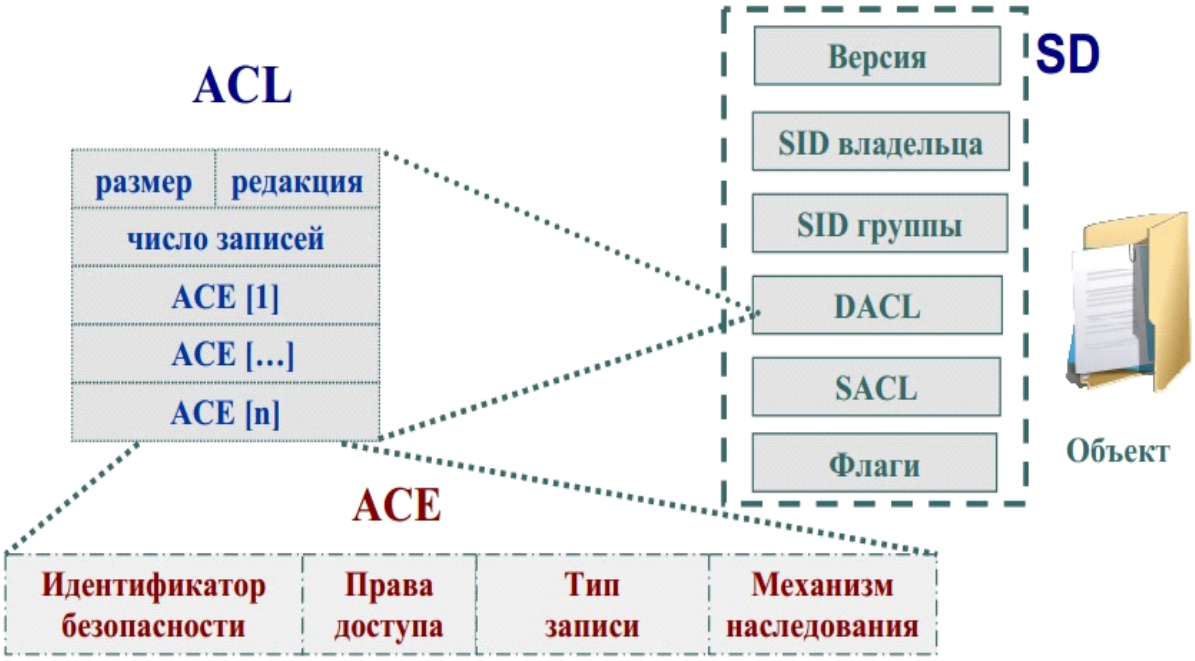
Защита объектов системы

Маркер доступа идентифицирует субъектов-пользователей системы. С другой стороны, каждый объект системы, требующий защиты, содержит описание прав доступа к нему пользователей. Для этих целей используется ***дескриптор безопасности (Security Descriptor, SD***). Каждому объекту системы, включая файлы, принтеры, сетевые службы, контейнеры Active Directory и другие, присваивается дескриптор безопасности

SD определяет права доступа к объекту и содержит следующие основные атрибуты:

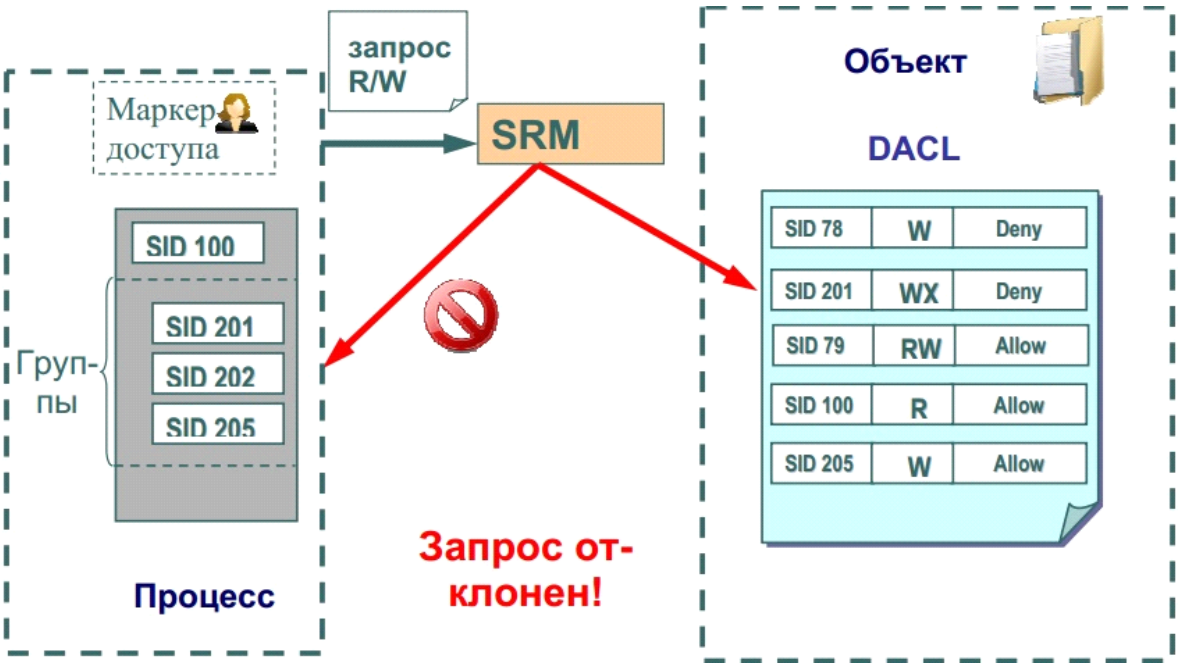
- SID владельца, идентифицирующий учетную запись пользователя-владельца объекта;
- пользовательский список управления доступом (Discretionary Access Control List, DACL), который позволяет отслеживать права и ограничения, установленные владельцем данного объекта. DACL может быть изменен пользователем, который указан как текущий владелец объекта.
- системный список управления доступом (System Access Control List, SACL), определяющий перечень действий над объектом, подлежащих аудиту;
- флаги, задающие атрибуты объекта.

Авторизация Windows основана на сопоставлении маркера доступа субъекта с дескриптором безопасности объекта. Управляя свойствами объекта, администраторы могут устанавливать разрешения, назначать право владения и отслеживать доступ пользователей.

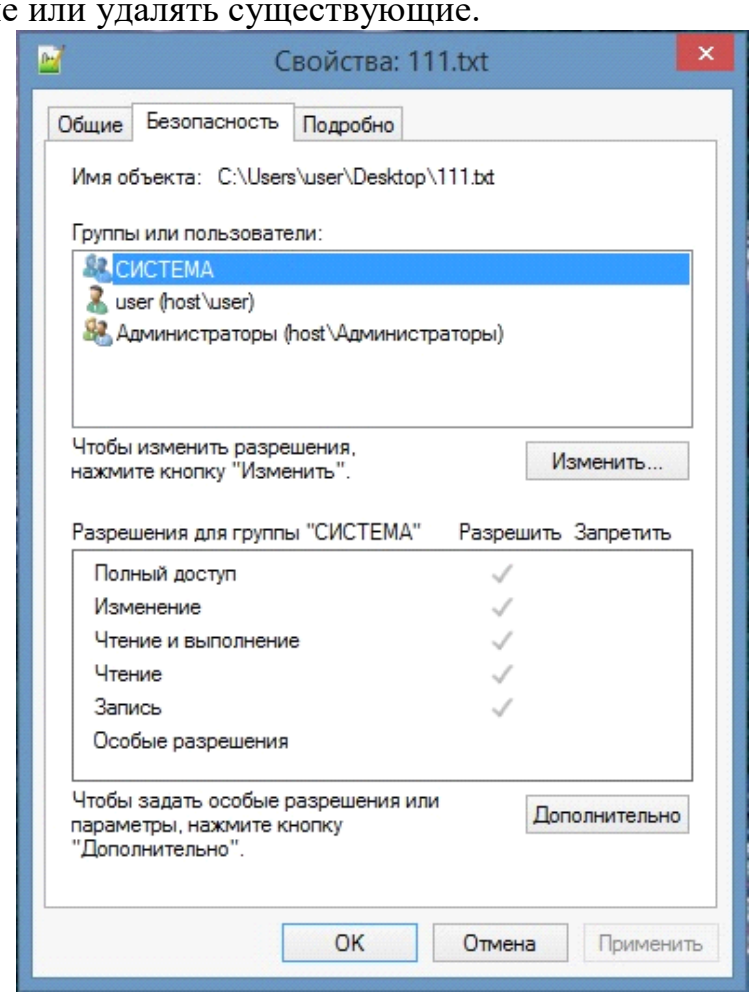


Список управления доступом содержит набор элементов (Access Control Entries, ACE). В DACL каждый ACE состоит из четырех частей: в первой указываются пользователи или группы, к которым относится данная запись, во второй – права доступа, а третья информирует о том, предоставляются эти права или отбираются.

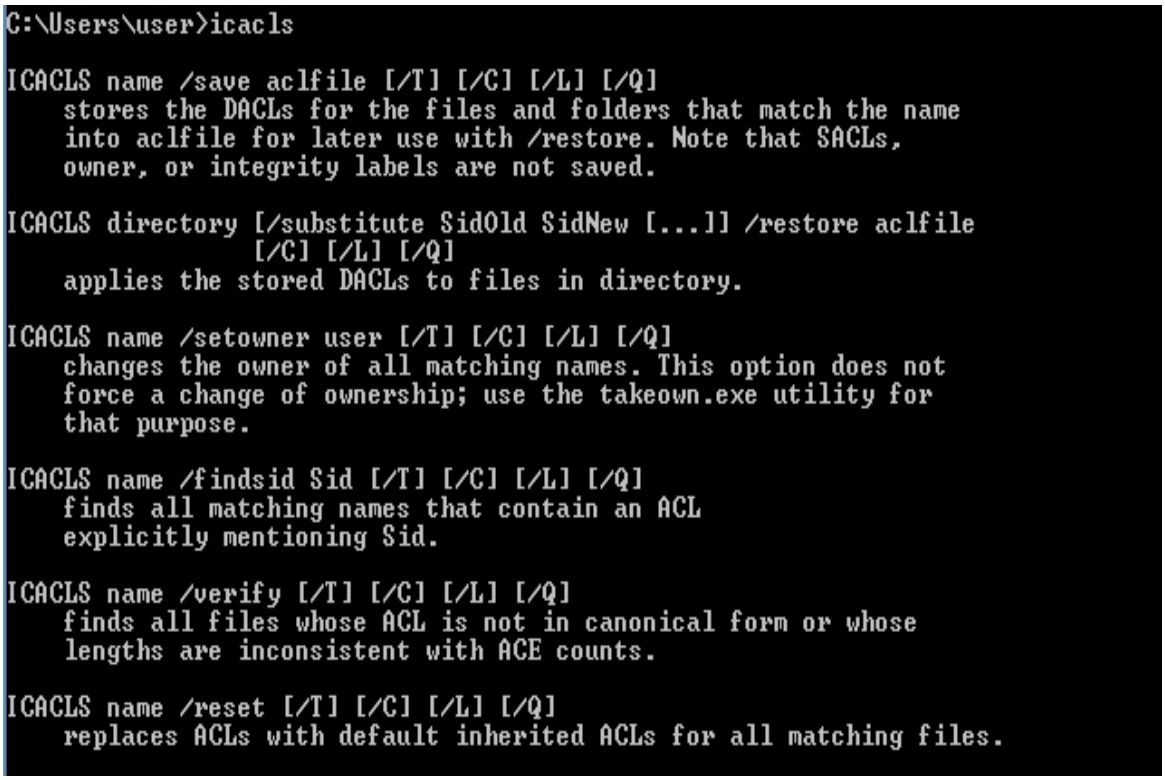
Четвертая часть представляет собой набор флагов, определяющих, как данная запись будет наследоваться вложенными объектами (актуально, например, для папок файловой системы, разделов реестра). Если список ACE в DACL пуст, к нему нет доступа ни у одного пользователя (только у владельца на изменение DACL). Если отсутствует сам DACL в SD объекта – полный доступ к нему имеют все пользователи. Если какой-либо поток запросил доступ к объекту, подсистема SRM осуществляет проверку прав пользователя, запустившего поток, на данный объект, просматривая его список DACL. Проверка осуществляется до появления разрешающих прав **на все** запрошенные операции. Если встретится запрещающее правило хотя бы **на одну** запрошенную операцию, доступ не будет предоставлен.



Для определения и просмотра прав доступа пользователей к ресурсам можно использовать как графические средства контроля, так и консольные команды. Стандартное окно свойств объекта файловой системы (диска, папки, файла) на вкладке ***Безопасность*** позволяет просмотреть текущие разрешения для пользователей и групп пользователей, редактировать их, создавать новые или удалять существующие.



Для просмотра и изменения прав доступа к объектам в режиме командной строки предназначена команда icacls (для Windows до Vista cacls)

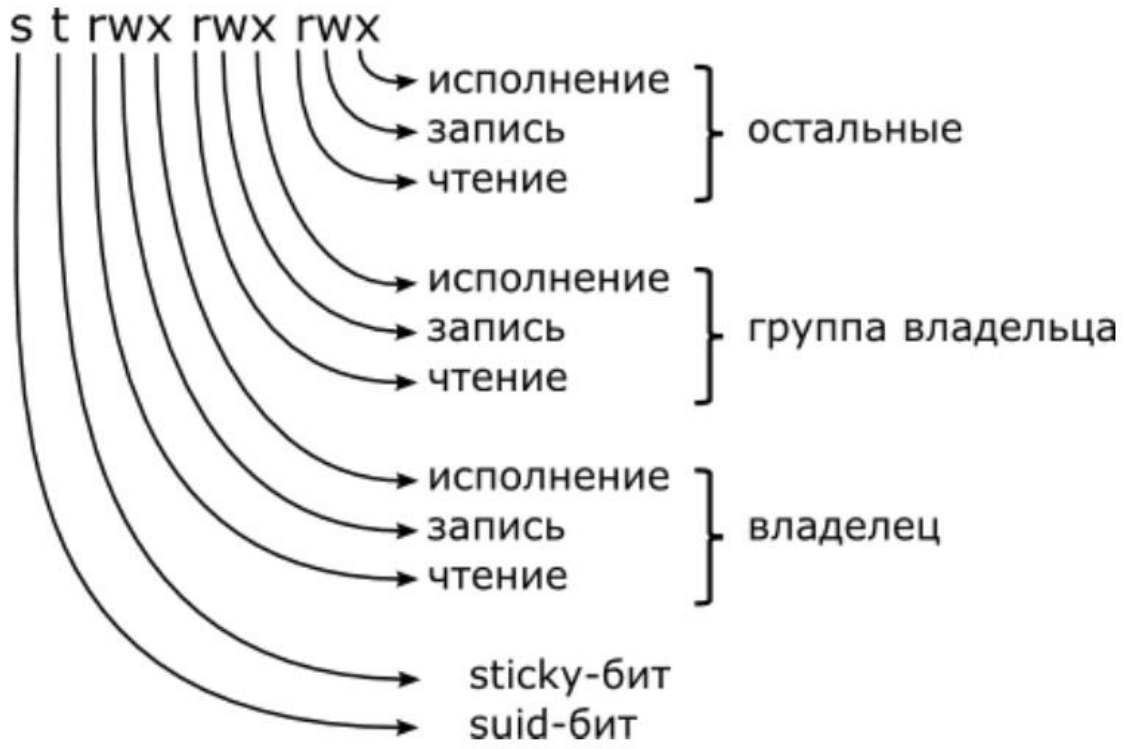


Управление доступом в Linux

Права доступа определяются по отношению к трём типам действий: чтение, запись и исполнение.

Эти права доступа могут быть предоставлены трём классам пользователей: владельцу файла (пользователю), группе, которой принадлежит файл, а также всем остальным пользователям, не входящим в эту группу. Право на чтение даёт пользователю возможность читать содержимое файла или, если такой доступ разрешён к каталогам, просматривать содержимое каталога (используя команду ls). Право на запись даёт пользователю возможность записывать или изменять файл, а право на запись для каталога — возможность создавать новые файлы или удалять файлы из этого каталога. Наконец, право на исполнение позволяет пользователю запускать файл как программу или сценарий командной оболочки (разумеется, это действие имеет смысл лишь в том случае, если файл является программой или сценарием). Владение правами на исполнение для каталога позволяет перейти (командой cd) в этот каталог.

Чтобы получить информацию о правах доступа, используйте команду ls с ключом -l. При этом будет выведена подробная информация о файлах и каталогах, в которой будут, среди прочего, отражены права доступа.



Помимо комбинации из этих девяти прав доступа, каждый файл может иметь дополнительные флаги доступа: sticky-бит, специфичный для директорий, и suid-бит, применяемый для исполняемых файлов. Если пометить директорию sticky-битом, удалять файл в ней смогут только владельцы, а не все те, кто имеют права записи на эту директорию - это необходимо в "общих директориях", где создавать и удалять файлы может любой. Suid-бит - большая головная боль системных администраторов, он нужен для повышения прав программы на время запуска.

Если t маленькое, значит x установлен. Если T большое, значит x не установлен. То же самое правило распространяется и на s.

Создание пользователя осуществляется командой **useradd** или **adduser**.

Удаление пользователя **userdel** или **deluser.**

Создание новых групп осуществляется командами **addgroup** и **groupadd.**

Для удаления группы можно удалить соответствующую строчку в файле /etc/group

Чтобы временно заблокировать пользователя в файле /etc/passwd в поле пароль необходимо поставить *.

Каждый пользователь в системе имеет свой уникальный идентификационный номер (user-ID, или UID). Также пользователи могут объединяться в группы, которые в свою очередь имеют group-ID, или GID.

Чтобы узнать свой UID и GID, т.е. уникальный номер пользователя и номер группы, к которой Вы принадлежите, необходимо ввести команду id:

[dmitry@localhost dmitry]$ id
uid=502 (dmitry) gid=503(users) groups=503(users)

В свою очередь файлы имеют двух владельцев: пользователя (user owner) и группу пользователей (group owner). Для каждого файла есть индивидуальные права доступа, которые разбиты на три группы:

1. Доступ для пользователя-владельца файла (owner).
2. Доступ для группы-владельца файла (group).
3. Доступ для остальных пользователей (others).

Права пользователя могут быть изменены только владельцем файла или пользователем с правами администратора системы. Для изменения прав используется команда

chmod [u | g | o | a] [+ | - | =] [r | w | x] name1 [name2 ...].

Несколько примеров:

$ **chmod go=w prog.pl** установить право на запись для всех пользователей кроме владельца

$ **chmod a+x prog.pl** предоставить право на запись для всех пользователей

$ **chmod g+x-w prog.pl** Добавить для группы право на выполнения файла, но снять право на запись Права доступа можно представить в виде битовой строки, в которой каждые 3 бита определяют права доступа для соответствующей категории пользователей, как представлено в таблице:

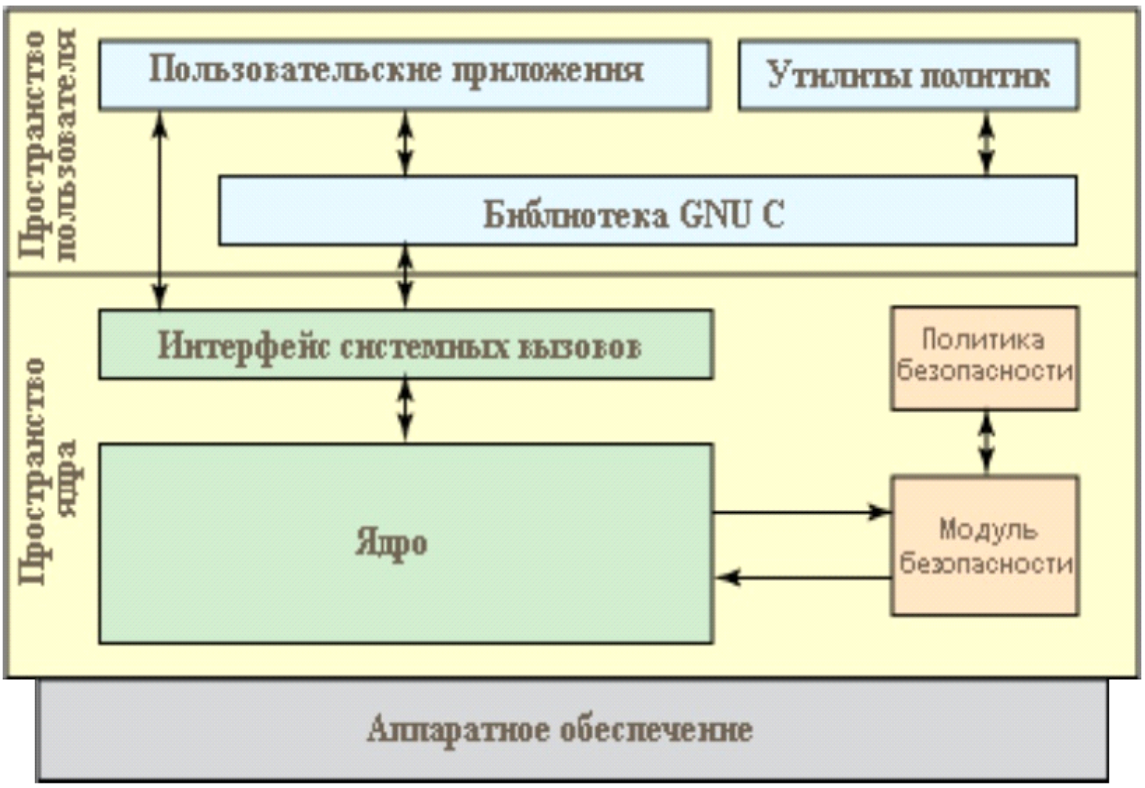
rwx rwx rwx
421 421 421
user group others
владелец группа остальные

chown – изменение владельца файла.
chmod – изменение прав доступа к файлу.
chgroup – изменение группы пользователя.
umask – определение прав пользователя по-умолчанию.

Проблема: любая программа, запущенная от имени пользователя, обладает всеми его правами - может читать конфигурационные файлы, устанавливать сетевые соединения и т.д. Другой большой проблемой безопасности таких систем является наличие прав суперпользователя (или администратора). Как правило, под этим названием понимается уровень доступа системы, с которым работают все системные службы.

Важным этапом развития ядра Linux стало внедрение интерфейса модулей безопасности (LSM, Linux Security Modules). В рамках этого проекта многие внутренние структуры ядра были расширены специальными полями, связанными с безопасностью. В код многих системных процедур были вставлены вызовы функций управления доступом (так называемые "hooks"), вынесенные во внешний модуль безопасности. Таким образом, каждый может написать собственный модуль, реализующий какую-то специфичную политику безопасности.

Общесистемная инфраструктура безопасности, предоставляемая модулем LSM, такова, что позволяет реализовывать модели безопасности в виде загружаемых модулей ядра.

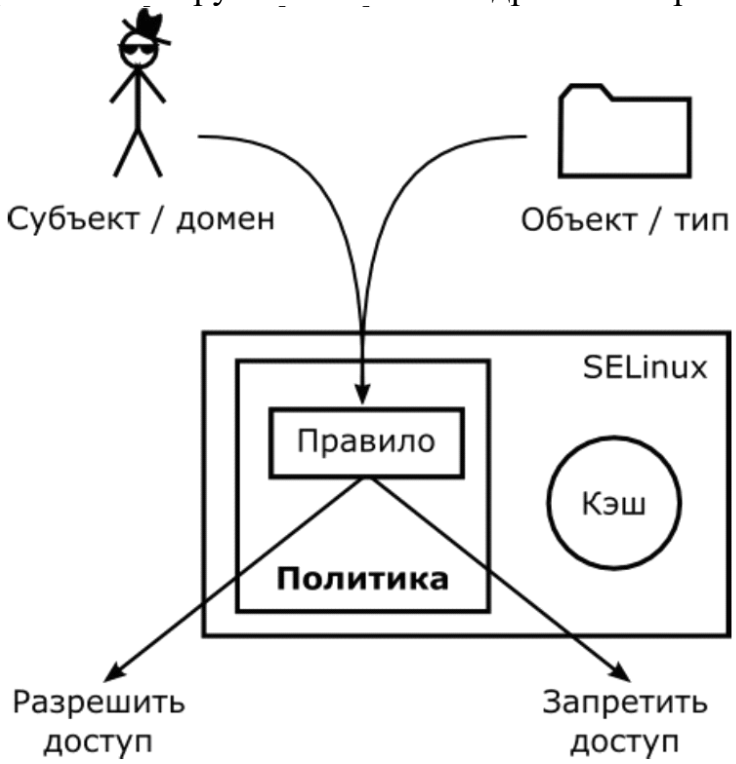


Существует несколько серьёзных проектов по расширению стандартной модели безопасности в Linux. Среди них можно выделить SELinux (Security-Enhanced Linux), RSBAC (Rule Set Base Access Control) и grsecurity.

SELinux - это расширение базовой модели безопасности операционной системы Linux, добавляющее механизм мандатного доступа. С помощью SELinux можно задать явные правила того, как субъекты (пользователи и программы) могут обращаться к объектам системы (файлы и устройства). Таким образом, можно ограничить программы, прописав возможности их поведения в виде политики, а операционная система обеспечит её соблюдение.

SELinux входит в официальное ядро Linux начиная с версии 2.6. Система разрабатывается Национальным агентством по безопасности США (NSA, National Security Agency) при сотрудничестве с другими исследовательскими лабораториями и коммерческими дистрибутивами Linux. Исходные тексты проекта доступны под лицензией GPL.

Мандатный доступ в SELinux реализован в рамках модели домен-тип. В этой модели каждый процесс запускается в определённом домене безопасности (уровень доступа), а всем ресурсам (файлы, директории, сокеты и т.п.) ставится в соответствие определённый тип (уровень секретности). Список правил, ограничивающих возможности доступа доменов к типам, называется политикой и задаётся один раз в момент установки системы. Описание политики в SELinux это набор текстовых файлов, которые могут быть скомпилированы и загружены в память ядра Linux при старте системы.

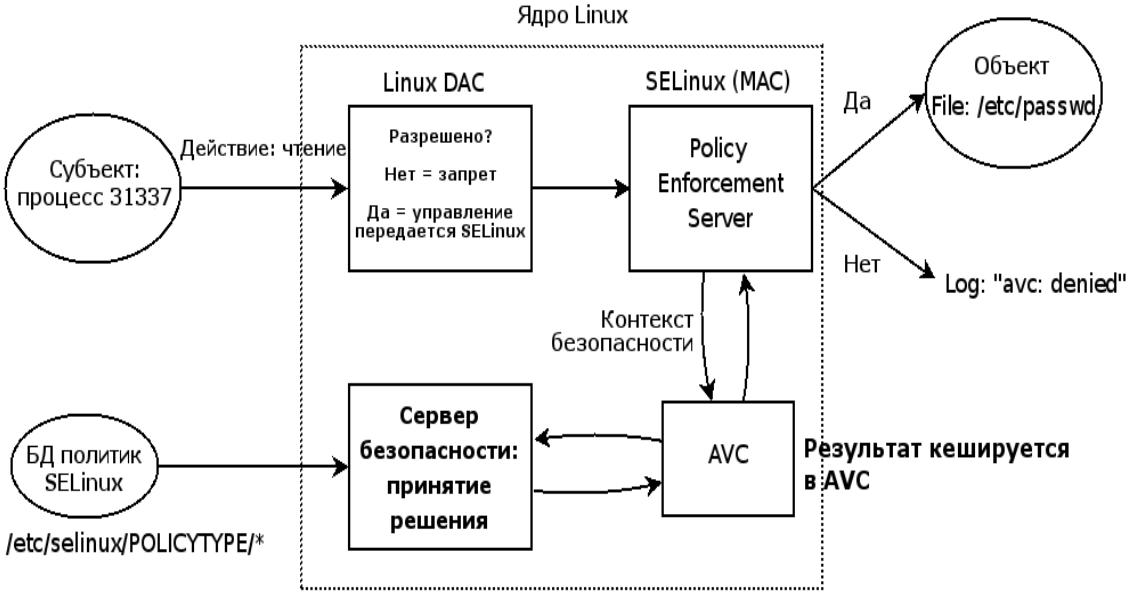


Тремя важнейшими концепциями структуры безопасности SELinux являются ***субъекты*** (subject), ***объекты*** (object) и ***действия*** (action).

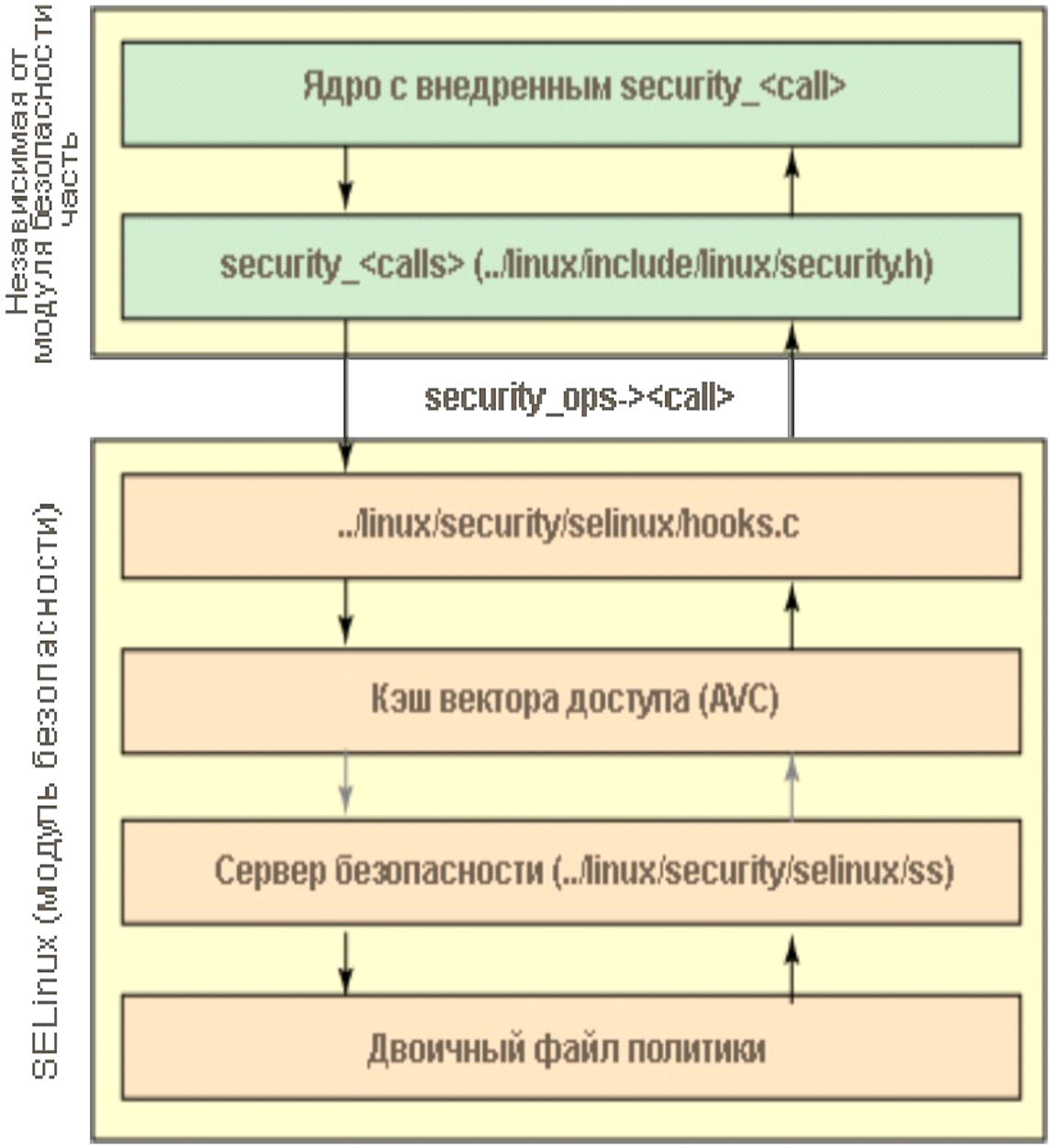
С точки зрения SELinux всю работу системы можно описать как выполнение субъектами действий над объектами. Субъектами считаются процессы, действующие как от имени определенных пользователей, так и самостоятельно (серверные процессы). Объектами являются, прежде всего, объекты файловой системы (файлы, каталоги, ссылки), процессы (когда один процесс-субъект выполняет операции с другим процессом, второй процесс выступает в роли объекта), а также дескрипторы файлов (в том числе сокеты, что повышает безопасность работы с сетью) и объекты межпроцессного взаимодействия, не связанные с дескрипторами файлов.

Действия в SELinux - это любые операции, которые субъект может выполнить над объектом. Собственно говоря, основная часть работы системы

безопасности как раз и заключается в принятии решения о том, имеет ли право данный субъект выполнить данное действие над данным объектом.



В кэше векторов доступа (Access Vector Cache, AVC) хранятся предыдущие решения SELinux (это сделано для улучшения производительности). Перед вызовом запрашиваемой функции вызывается определенный вызов, который содержит в себе идентификатор безопасности источника (source security identifier, sid), класс безопасности (составленный из подробной информации о запрашиваемых операциях), вызов особого сокета и дополнительные вспомогательные данные аудита. Если необходимое решение в кэше не находится, то для его получения производится вызов серверу безопасности.



Решение о допустимости или не допустимости действия принимается системой на основе политик, представляющих собой способ описания поведения системы безопасности, абстрагирующийся от таких низкоуровневых понятий как векторы доступа (аналоги масок доступа в обычной Linux-системе). Формирование политик безопасности в SELinux напоминает программирование: политика описывается на специальном языке, затем файл политики компилируется в двоичный модуль (обычно при этом используется хорошо знакомая программистам утилита make), а скомпилированный файл динамически загружается в ядро операционной системы.

Права субъектов и объектов определяются в SELinux контекстами безопасности, состоящими из идентификатора, роли и типа объекта.

Идентификатор субъекта - это идентификатор пользователя SELinux, создавшего процесс-субъект. ***Идентификатором объекта*** является идентификатор пользователя-владельца объекта (обычно это пользователь, создавший объект). Роли представляют собой наборы привилегий. В любой момент времени каждый пользователь может выступать в одной из доступных ему ролей. Роли позволяют предоставлять пользователю дополнительные привилегии, не утрачивая его идентичности, как это происходит при применении команды su в традиционных системах Unix/Linux. Политика безопасности SELinux может налагать ограничения не только на количество ролей, доступных процессу, но и определять правила перехода из одной роли в другую. Не всякий переход из одной роли в другую допустим.

Очевидно, что роли гораздо чаще используются субъектами, чем объектами, а некоторые объекты (скажем, дисковые файлы), вообще не нуждаются в ролях. Таким объектам присваиваются "пустые роли", не влияющие на безопасность.

Типы объединяют группы субъектов и объектов, предоставляя им определенные права. Важной функцией типов является ограничение возможных действий субъекта над объектом. По этой причине типы иногда называют "песочницами SELinux" (SELinux sandbox). В документации по SELinux можно встретить также термин "домен". В классических работах по безопасности систем (в частности, в модели Flask) понятия "тип" и "домен" имеют разные значения, однако в SELinux эти понятия почти синонимы. Мы говорим о типах, когда речь идет об объектах и о доменах, когда речь идет о субъектах. Домены можно описать как множества процессов (субъектов), обладающих одинаковыми правами. Например, Web-сервер Apache принадлежит домену (типу) httpd_t и обладает всеми правами, связанными с этим доменом. К этому же типу относятся файлы, к которым демон httpd должен иметь полный доступ. В SELinux действует механизм принудительного присвоения типов (type enforcement). В соответствии с этим механизмом каждый процесс оказывается принадлежащим к определенному типу (домену), определяющему права этого процесса. Очевидно, что без принудительного присвоения типов система обязательного контроля доступа не могла бы работать.

Каждый субъект и объект идентифицируется собственным контекстом безопасности, которому соответствует идентификатор безопасности SID, причем система контекстов сильно отличается от традиционной системы учетных записей в ОС Linux поэтому возникает задача их взаимодействия.

В соответствии с принципом независимости SELinux поддерживает таблицу контекстов безопасности, независимую от таблицы учетных записей Linux.

При этом возможно отображение нескольких учетных записей Linux на одну учетную запись SELinux. Таким образом, изменения в учетных записях Linux не влияют на параметры безопасности SELinux.

Модель безопасности SELinux требует, чтобы каждый файл в системе был связан с определенным контекстом безопасности, поэтому в ходе установки всегда выполняется маркировка (labeling) объектов файловой системы.

В соответствии с принципом независимости маркировка файлов не влияет на маски доступа к файлам, а в соответствии с принципом приоритета традиционной системы безопасности запреты, наложенные маской доступа, отменяют разрешения SELinux.

Операции SELinux делятся на операции доступа (access) и операции преобразования (transition). Первые система выполняет чаще всего, например, открытие и чтение данных из файла, вторые - операции, связанные с изменением контекста безопасности объектов. В результате операции преобразования объект получает контекст безопасности, отличный от того, который он получил бы по умолчанию. Например, созданный файл по умолчанию получает тот же контекст безопасности, что и каталог, в котором этот файл создан (в таких случаях говорят: "файл наследует контекст безопасности"). С помощью операции преобразования файлу можно присвоить новый контекст безопасности, отличающийся от контекста безопасности по умолчанию.

Из RHEL User and Admin Guide

SELinux - это модуль безопасности, встроенный в ядро Linux. Правила политики безопасности загружаются в ядро в процессе выполнения. Разрешения и запреты на доступ кэшируются в Access Vector Cache (AVC).

SELinux может работать в 3 режимах:

Enforcing - SELinux включен и выполняет правила политики.

Permissive - SELinux, но все нарушения не блокируются, а журналируются.

Disabled - SELinux выключена. Работают только DAC.

Управление режимом работы осуществляется утилитой ***setenforce***.

 # setenforce 1 - включение режима enforce

 # setenforce 0 - включение режима permissive

Все процессы и файлы промаркированы контекстом SELinux, который содержит дополнительную информацию, такую как SELinux пользователь, роль, тип и опционально уровень. Вся эта информация используется SELinux для принятия решения о доступе.

SELinux обеспечивает комбинацию Role-Based Access Control (RBAC), Type Enforcement (TE), и, опционально, MultiLevel Security (MLS).

Просмотр контекста SELinux для файла:

$ ls -Z file1

-rwxrw-r-- user1 group1 unconfined_u:object_r:user_home_t:s0 file1

Контекст SELinux имеет синтаксис ***user:role:type:level***:

SELinux user

Идентификатор пользователя SELinux, который отождествлен с определенным набором ролей или для определенного диапазона MLS/MCS (MultiCategory Security). Каждый пользователь Linux отображается в SELinux пользователя через политику SELinux.

Просмотр соответствия между пользователями Linux и SELinux.

 # semanage login -l

Login Name	**SELinux User**	**MLS/MCS Range**	**Service**
__default__	**unconfined_u**	**s0-s0:c0.c1023**	*****
root	**unconfined_u**	**s0-s0:c0.c1023**	*****
system_u	**system_u**	**s0-s0:c0.c1023**	*****

Login Name - Linux users

Services - корректный SELinux контекст, в котором предполагается вход в систему пользователя Linux.

Role

Модель безопасности RBAC является частью SELinux. Role - это атрибут RBAC. Роли являются посредниками между пользователями SELinux и доменами.

Type

Тип является атрибутом TE (Type Enforcement). Type определяет домен для процесса и тип для файлов. Правила безопасности определяют правила для доступа домена к типу или домена к другому домену. Доступ возможен только при наличии соответствующего правила.

Level

Level - атрибут MLS и MCS. MLS уровень представляет собой два числа низший-высший, если уровни разные и одно, если одинаковые, например, s0.

Уровень может содержать категории чувствительности, обозначаются как c0,c1,… .

MLS реализует модель Бела-Лаппадула.

Transition domain

Приложение из одного домена может преобразоваться в момент запуска в другой домен если оно имеет entrypoint тип этого домена.

Например, утилита /usr/bin/passwd имеет тип **passwd_exec_t**

 # ls -Z /usr/bin/passwd
-rwsr-xr-x root root system_u:object_r:passwd_exec_t:s0 /usr/bin/passwd

Файл shadow имеет тип **shadow_t**
ls -Z /etc/shadow
-r--------. root root system_u:object_r:shadow_t:s0 /etc/shadow

Домен **passwd_t** имеет право чтения и записи файлов с типом **shadow_t**. Так как домен **passwd_t** имеет разрешения **entrypoint** для **passwd_exec_t,** то утилита passwd преобразуется в домен **passwd_t** и имеет право изменять пароли пользователей.

Домен процесса возможно посмотреть с помощью **ps -eZ**
 # ps -eZ | grep passwd
unconfined_u:unconfined_r:passwd_t:s0-s0:c0.c1023 13212 pts/1 00:00:00 passwd

Пример доменов системных сервисов
ps -eZ
system_u:system_r:dhcpc_t:s0 1869 ? 00:00:00 dhclient
system_u:system_r:sshd_t:s0-s0:c0.c1023 1882 ? 00:00:00 sshd
system_u:system_r:gpm_t:s0 1964 ? 00:00:00 gpm
system_u:system_r:crond_t:s0-s0:c0.c1023 1973 ? 00:00:00 crond
system_u:system_r:kerneloops_t:s0 1983 ? 00:00:05 kerneloops
system_u:system_r:crond_t:s0-s0:c0.c1023 1991 ? 00:00:00 atd

Контекст, ассоциированный с данным пользователем, определяется как
 # id -Z
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023

Файлы с которыми работает web-сервер Apache в домене httpd_t имеют тип **httpd_sys_content_t.**
Для перемаркировки файла используется команда **chcon.**
Например, **chcon -t samba_share_t /var/www/html/testfile**

Утилита **restorecon** - восстанавливает контекст файла по-умолчанию.

Управление пользователями SELinux осуществляется с помощью утилиты **semanage**.

seinfo - информация о SELinux.

Аппаратные средства управления доступом.

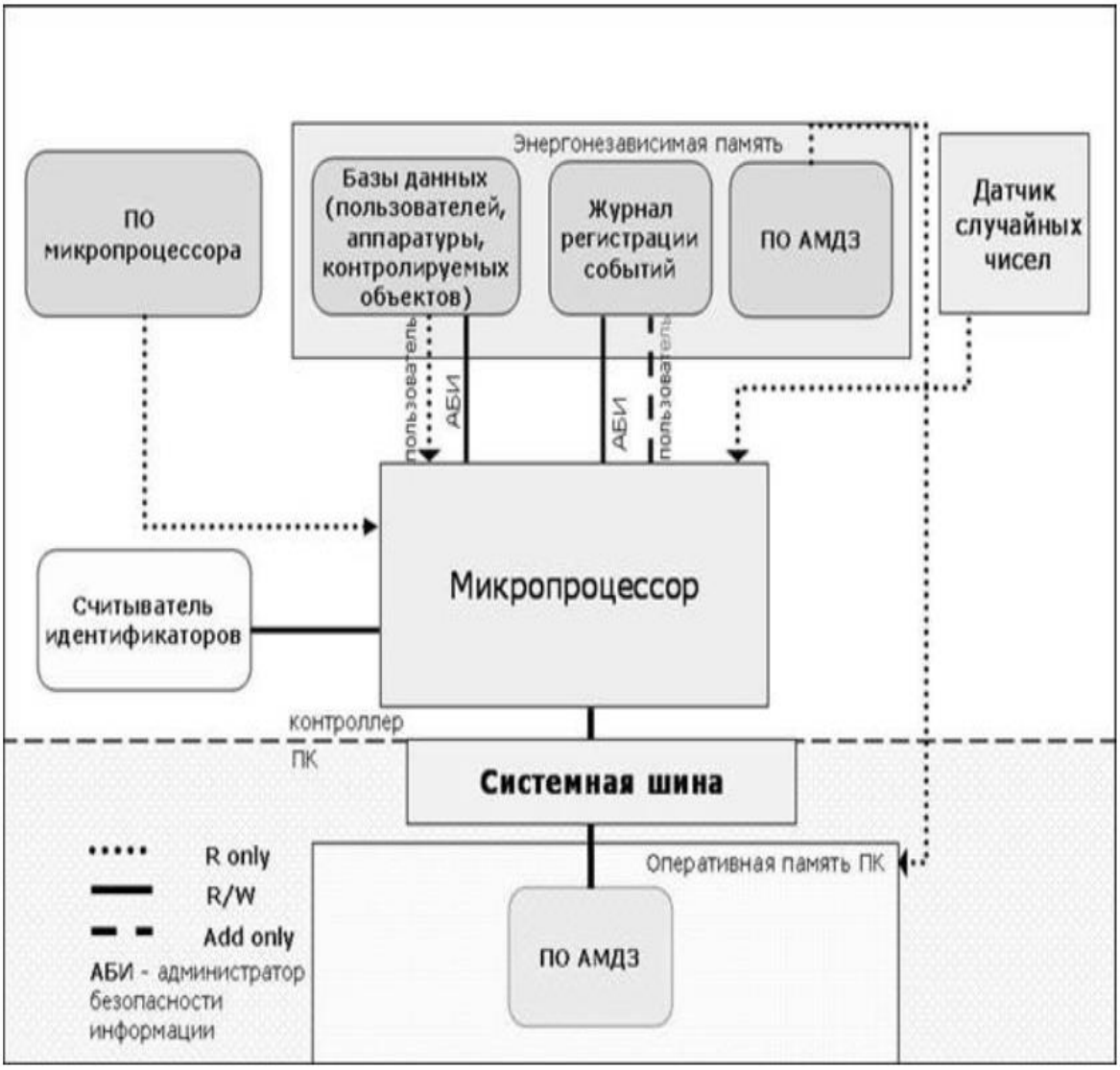
СЗИ НСД Аккорд-АМДЗ - это аппаратный модуль доверенной загрузки.

Доверенная загрузка - это загрузка различных операционных систем только с заранее определенных постоянных носителей (например, только с жесткого диска) после успешного завершения специальных процедур: проверки целостности технических и программных средств ПК (с использованием механизма пошагового контроля целостности) и аппаратной идентификации/аутентификации пользователя.

Иными словами, это загрузка строго определенной, не измененной ОС и только в том случае, если подтверждено, что в компьютере не произошло никаких несанкционированных изменений (на аппаратном уровне и в критичной части приложений) и что включает его именно тот пользователь, который имеет право на нем работать именно в это время.

Аккорд-АМДЗ физически представляет собой плату расширения, которая устанавливается в материнскую плату ПК и начинает работу сразу после выполнения штатного BIOS компьютера - до загрузки операционной системы.

Фактически же он представляет собой отдельный компьютер с собственным процессором, независимым от процессора ПК. На схемотехническом уровне процессор Аккорда защищен от несанкционированного перепрограммирования через ПК, поэтому его firmware не может быть искажено злоумышленником или недобросовестным пользователем. Санкционированное перепрограммирование контроллера возможно только в специальном защищенном режиме.



Другие модули доверенной загрузки

- Максим-М1
- Соболь
- Эшалон

Вывод

В данной лекции мы рассмотрели такой важный механизм безопасности как управление доступом. Изучили конкретные реализации механизмов авторизации в ОС Linux, Windows.