50 1190 0101

Утвержден

РУСБ.10015-01-УД

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл.	

ОПЕРАЦИОННАЯ СИСТЕМА СПЕЦИАЛЬНОГО НАЗНАЧЕНИЯ «ASTRA LINUX SPECIAL EDITION»

Руководство по КСЗ. Часть 1

РУСБ.10015-01 97 01-1

Листов 138

2015

АННОТАЦИЯ

Настоящий документ является первой частью руководства по комплексу средств защиты (КСЗ) операционной системы специального назначения «Astra Linux Special Edition» РУСБ.10015-01 (далее по тексту — ОС).

Руководство по КСЗ состоит из двух частей:

– РУСБ.10015-01 97 01-1 «Операционная система специального назначения «Astra Linux Special Edition». Руководство по КСЗ. Часть 1»;

– РУСБ.10015-01 97 01-2 «Операционная система специального назначения «Astra Linux Special Edition». Руководство по КСЗ. Часть 2».

В первой части руководства приведены общие сведения о КСЗ, рассмотрены идентификация и аутентификация, дискреционное и мандатное управление доступом, очистка памяти, изоляция модулей, маркировка документов, защита ввода-вывода информации на отчуждаемый физический носитель, сопоставление пользователя с устройством, регистрация событий, надежное восстановление, средства ограничения прав доступа к страницам памяти, контроль целостности и генерация КСЗ, режим киоска, запуск ОС.

Во второй части руководства приведено описание тестов КСЗ.

СОДЕРЖАНИЕ

1. ОБЩИЕ СВЕДЕНИЯ

ОС предназначена для построения автоматизированных систем в защищенном исполнении, обрабатывающих информацию, содержащую сведения, составляющие государственную тайну с грифом не выше «совершенно секретно».

КСЗ (подсистема безопасности PARSEC) предназначен для реализации функций ОС по защите информации от НСД и предоставления администратору безопасности информации средств управления функционированием КСЗ.

ВНИМАНИЕ! После установки ОС интерактивный вход в систему суперпользователя `root` по умолчанию заблокирован. Создаваемый при установке операционной системы пользователь включается в группу `astra-admin`. Пользователям, входящим в названную группу, через механизм `sudo` предоставляются права для выполнения действий по настройке ОС, требующих привилегий суперпользователя `root`. Далее по тексту такой пользователь именуется администратором.

1.1. Состав КСЗ

В состав КСЗ входят следующие основные подсистемы:

- модули подсистемы безопасности PARSEC, входящие в состав ядра ОС;
- библиотеки;
- утилиты безопасности;
- подсистема протоколирования (регистрации);
- модули аутентификации;
- графическая подсистема;
- консольный вход в систему;
- средства контроля целостности;
- средства восстановления;
- средства разграничения доступа к подключаемым устройствам.

1.2. Контролируемые функции

КСЗ обеспечивает реализацию следующих функций ОС по защите информации от НСД:

- идентификацию и аутентификацию;
- дискреционное разграничение доступа;
- мандатное разграничение доступа;
- очистку памяти;
- изоляцию модулей;
- маркировку документов;

– защиту ввода-вывода информации на отчуждаемый физический носитель;

– сопоставление пользователя с устройством;

– регистрацию событий;

– надежное восстановление;

– контроль целостности КСЗ.

Реализация перечисленных выше функций основана на следующих основных положениях:

1) с каждым пользователем системы связан уникальный численный идентификатор — идентификатор пользователя (UID), который является ключом к соответствующей записи в БД пользователей, содержащей информацию о пользователях, включая их реальные и системные имена. БД пользователей поддерживается и управляется системным администратором. UID есть ярлык субъекта (номинальный субъект), которым система пользуется для определения прав доступа. БД пользователей в ОС может быть как локальной для системы, так и являться частью ЕПП, функционирующего на основе протокола LDAP;

2) каждый пользователь входит в одну или более групп. Группа — это список пользователей системы, имеющий собственный идентификатор (GID). Поскольку группа объединяет несколько пользователей системы, в терминах политики безопасности она соответствует понятию «множественный субъект». GID есть ярлык множественного субъекта, которых у номинального субъекта может быть более одного. Таким образом, одному UID соответствует список GID;

3) роль действительного (работающего с объектами) субъекта играет процесс. Каждый процесс снабжен единственным UID: это идентификатор запустившего процесс номинального субъекта, т. е. пользователя. Процесс, порожденный некоторым процессом пользователя, наследует его UID. Таким образом, все процессы, запускаемые по желанию пользователя, будут иметь его идентификатор. Все процессы, принадлежащие пользователю, образуют сеанс пользователя. Первый процесс сеанса пользователя порождается после прохождения процедур идентификации и аутентификации. При обращении процесса к объекту доступ предоставляется по результатам процедуры авторизации, т. е. обработки запроса на основе мандатных и дискреционных ПРД;

4) механизм ПРД реализован в ядре ОС, что обеспечивает его правильное функционирование при использовании любых компонент, предоставляемых ОС. Реализация мандатного разграничения доступа затрагивает все подсистемы ядра, в которых реализовано дискреционное разграничение доступа. При этом, оба вида разграничения доступа функционируют параллельно, не влияя на принятие решений

друг друга (непротиворечивость). Доступ разрешается в том случае, если он возможен относительно дискреционных и мандатных ПРД. Запрещается в случае, если доступ запрещен относительно любого из механизмов.

1.3. Средства организации ЕПП

Организация ЕПП обеспечивает:

– сквозную аутентификацию в сети;

– централизацию хранения информации об окружении пользователей;

– централизацию хранения настроек системы защиты информации на сервере.

Сетевая аутентификация и централизация хранения информации об окружении пользователя подразумевает использование двух основных механизмов: поддержки кроссплатформенных серверных приложений для обеспечения безопасности (NSS) и подгружаемых аутентификационных модулей (PAM). Сквозная аутентификация в сети реализуется на основе протокола Kerberos с использованием службы каталогов LDAP в качестве источника данных для базовых системных сервисов на базе механизмов NSS и PAM. Подобный подход обеспечивает централизацию хранения информации об окружении пользователей (в том числе предназначенную для обеспечения мандатного разграничения доступа):

– существующие в системе мандатные уровни и категории;

– минимальные и максимальные мандатные уровни, доступные пользователям при входе в систему;

– минимальные и максимальные наборы мандатных категорий, доступные пользователям при входе в систему;

– члены привилегированных групп, которые могут получать из БД службы каталогов LDAP определенную информацию о пользователях.

Кроме того, с использованием СЗФС CIFS обеспечено централизованное хранение домашних каталогов пользователей.

Для снижения нагрузки на сеть и повышения производительности в ЕПП может применяться кэширование редко изменяемой информации в локальном кэше.

ВНИМАНИЕ! Измененная на сервере информация может попасть в локальный кэш с задержкой. Период обновления локального кэша задается параметром `CACHE_REFRESH_PERIOD` в конфигурационном файле `/etc/ald/ald.conf`.

Более подробное описание ЕПП приведено в РУСБ.10015-01 95 01-1 «Операционная система специального назначения «Astra Linux Special Edition». Руководство администратора. Часть 1».

2. ИДЕНТИФИКАЦИЯ И АУТЕНТИФИКАЦИЯ

Функция идентификации и аутентификации пользователей в ОС основывается на использовании механизма PAM.

PAM представляют собой набор разделяемых библиотек (т. н. «модулей»), с помощью которых системный администратор может организовать процедуру аутентификации (подтверждение подлинности) пользователей прикладными программами. Каждый модуль реализует собственный механизм аутентификации. Изменяя набор и порядок следования модулей, можно построить сценарий аутентификации.

Подобный подход позволяет изменять процедуру аутентификации без изменения исходного кода и повторного компилирования PAM.

Сценарии аутентификации (т. е. работа этих функций) описываются в конфигурационном файле `/etc/pam.conf` и в ряде конфигурационных файлов, расположенных в каталоге `/etc/pam.d/`. Сама аутентификация выполняется с помощью PAM. Модули располагаются в каталоге `/lib/security` в виде динамически загружаемых объектных файлов.

Если ЕПП не используется, аутентификация осуществляется с помощью локальной БД пользователей `/etc/passwd`. Информация, которая хранится в `/etc/shadow` и используется для пользователей в локальной БД. В ОС реализована возможность хранения аутентификационной информации пользователей, полученной с использованием хеш-функций по ГОСТ Р 34.11-94 и по ГОСТ Р 34.11-2012.

При использовании ЕПП аутентификация пользователей осуществляется централизованно по протоколу Kerberos. Для защиты аутентификационной информации по умолчанию используются отечественные алгоритмы по ГОСТ 28147-89 и ГОСТ Р 34.11-2012.

В ЕПП в качестве источника данных для идентификации и аутентификации пользователей применяются службы каталогов LDAP. В результате вся служебная информация пользователей сети может располагаться на выделенном сервере в распределенной гетерогенной сетевой среде. Добавление новых сетевых пользователей в этом случае производится централизованно на сервере службы каталогов. Сетевые сервисы, поддерживающие возможность аутентификации пользователей (web, FTP, почта), могут вместо локальных учетных записей использовать тот же каталог LDAP проверки аутентификационной информации. Администратор сети может централизованно управлять конфигурацией сети, в т. ч. разграничивать доступ к сетевым сервисам.

Благодаря предоставлению информации LDAP в иерархической древовидной форме разграничение доступа в рамках службы каталогов LDAP может быть основано на введении доменов. В качестве домена в данном случае будет выступать поддерево службы

каталогов LDAP. Сервисы LDAP позволяют разграничивать доступ пользователей к разным поддеревьям каталога, хотя по умолчанию в ОС реализуется схема одного домена.

Для управления пользователями, группами и настройками их атрибутов используется графическая утилита `fly-admin-smc`. Описание графической утилиты см. в электронной справке.

Для управления БД ALD в режиме командной строки используется утилита `ald-admin`, подробное описание которой приведено в `man ald-admin`.

Примечание. При создании локальных пользователей или пользователей ЕПП необходимо обязательно устанавливать для них диапазоны допустимых мандатных уровней и категорий: минимальный и максимальный мандатные уровни, минимальный и максимальный наборы мандатных категорий (раздел 4). Отсутствие установленных допустимых диапазонов мандатных атрибутов приводит к запрещению доступа при обращении к сетевым сервисам защищенных комплексов программ гипертекстовой обработки данных, электронной почты, СУБД и печати.

По умолчанию в сценарии `/etc/pam.d/common-auth`, содержащий настройки, общие для всех служб, предоставляющих сервис для входа в систему, используется PAM-модуль `pam_tally.so`. Данный PAM-модуль при начале процедуры аутентификации пользователя увеличивает счетчик неуспешных попыток аутентификации пользователя на единицу. Число неуспешных попыток аутентификации для пользователя может быть просмотрено следующей командной:

```
$faillog -u user_name
```

После успешного завершения попытки аутентификации пользователя счетчик неуспешных попыток аутентификации пользователя сбрасывается в ноль. Максимальное число неуспешных попыток аутентификации пользователя для пользователя определяется утилитой `faillog` и значениями параметров `deny` и `per_user`:

```
auth [success=ignore default=die] pam_tally.so per_user deny=10
```

Использование параметра `per_user` означает, что при установке утилитой `faillog` максимального значения неуспешных попыток аутентификации для пользователя не равного `0` применяется указанное значение. Иначе применяется значение, определяемое параметром `deny`. При отсутствии установленного параметра `per_user` используется значение параметра `deny`.

Для сброса счетчика неуспешных попыток аутентификации необходимо выполнить следующую команду:

```
faillog -r -u user_name
```

Более подробное описание см. в руководстве `man` на `faillog` и `pam_tally`.

3. ДИСКРЕЦИОННОЕ РАЗГРАНИЧЕНИЕ ДОСТУПА

3.1. Общие сведения

В ОС реализован механизм дискреционных ПРД именованных субъектов (пользователей) к именованным объектам. Реализация механизма дискреционных ПРД обеспечивает наличие для каждой пары (субъект-объект) явное и недвусмысленное перечисление допустимых типов доступа.

Дискреционный контроль доступа применяется к каждому объекту и каждому субъекту и заключается в том, что на защищаемые именованные объекты устанавливаются (автоматически при их создании) базовые ПРД в виде идентификаторов номинальных субъектов (UID и GID), которые вправе распоряжаться доступом к данному объекту и прав доступа к объекту. Определяются три вида доступа: чтение (read, r), запись (write, w) и исполнение (execution, x). Права доступа включают список (битовую маску) из девяти пунктов: по три вида доступа для трех классов - пользователя-владельца, группы-владельца и всех остальных. Каждый пункт в этом списке может быть либо разрешен, либо запрещен (равен 1 или 0).

При обращении процесса к объекту (с запросом доступа определенного вида, т.е. на чтение, запись или исполнение) система проверяет совпадение идентификаторов владельцев процесса и владельцев файла в определенном порядке, и в зависимости от результата, применяет ту или иную группу прав.

Права доступа файлового объекта могут быть изменены, если это разрешено (санкционировано) текущими правилами разграничения доступа.

Существуют также специальные биты, такие как:

– `SUID` — Set User ID - бит смены идентификатора пользователя;

– `SGID` — Set Group ID - бит смены идентификатора группы;

– `Sticky` — определяет владельца объектов в каталоге.

Когда пользователь или процесс запускает исполняемый файл с одним из установленных битов `SUID` или `SGID`, то файлу временно назначаются права его (файла) владельца или группы (в зависимости от того, какой бит задан). Таким образом, пользователь может даже запускать файлы от имени суперпользователя.

Каталог с установленным `Sticky`-битом означает, что удалить файл из этого каталога может только владелец файла или суперпользователь. Другие пользователи лишаются права удалять файлы. Установить `Sticky`-бит в каталоге можно только от имени суперпользователя с использованием механизма `SUDO`. `Sticky`-бит каталога остается до тех пор, пока владелец каталога или суперпользователь не удалит каталог явно или не изменит права доступа. Владелец может удалить `Sticky`-бит, но не может его установить.

Дополнительно в ОС механизмом дискреционных ПРД поддерживаются списки контроля доступа ACL (Access Control List), реализованные на основе расширенных атрибутов файловых систем. С использованием ACL можно дополнительно для каждого объекта задавать права на доступ субъектов к нему.

ACL состоит из набора записей. Права доступа к объекту для пользователя-владельца, группы-владельца и всех остальных имеют соответствующее представление в ACL в виде отдельных записей. ACL соответствующий базовым ПРД называется минимальным ACL. Таким образом, к каждому объекту доступа всегда сопоставляется минимальный ACL, включающий три записи: для пользователя-владельца, группы-владельца и всех остальных. Права доступа для дополнительных субъектов определяются в дополнительных записях ACL.

ACL включающий более трех записей называется расширенным ACL. Он дополнительно содержит запись для маски доступа и набор записей для именованных пользователей и именованных групп.

В общем случае ACL включает записи следующих типов:

– пользователь-владелец (текстовое представление: `user::rwx`);

– именованный пользователь (текстовое представление: `user:user_name:rwx`);

– группа-владельца (текстовое представление: `group::rwx`);

– именованная группа (текстовое представление: `user:group_name:rwx`);

– маска доступа (текстовое представление: `mask::rwx`);

– все остальные (текстовое представление: `other::rwx`).

Запись маски доступа используется для ограничения распространения прав доступа именованных пользователей и групп.

В механизме дискреционных ПРД реализовано отображение прав доступа к объекту, указанных в битовой маске для трех классов (пользователя-владельца, группы-владельца и всех остальных) в соответствующие записи ACL.

При использовании минимального ACL:

– права доступа из битовой маски для класса пользователь-владелец (например, `rwx`) отображаются в идентичные права доступа записи ACL типа пользователь-владелец (например, `user::rwx`);

– права доступа из битовой маски для класса группа-владельца (например, `rw-`) отображаются в идентичные права доступа записи ACL типа группа-владельца (например, `group::rw-`);

– права доступа из битовой маски для класса все остальные (например, `r--`) отображаются в идентичные права доступа записи ACL типа все остальные (например, `group::r--`).

При использовании расширенного ACL:

– права доступа из битовой маски для класса пользователь-владелец (например, `rwx`) отображаются в идентичные права доступа записи ACL типа пользователь-владелец (например, `user::rwx`);

– права доступа из битовой маски для класса группа-владельца (например, `rw-`) отображаются в идентичные права доступа записи ACL типа маска доступа (например, `mask::rw-`);

– права доступа из битовой маски для класса все остальные (например, `r--`) отображаются в идентичные права доступа записи ACL типа все остальные (например, `group::r--`).

Реализованная в механизме дискреционных ПРД проверка прав доступа субъекта к объекту, выполняется в два этапа. На первом этапе выбирается запись ACL, соответствующая субъекту доступа. Записи ACL для объекта доступа просматриваются в следующем порядке:

1) запись для пользователя-владельца;
2) записи именованных пользователей;
3) запись группы-владельца;
4) записи именованных групп;
5) запись для всех остальных.

Решение о доступе принимается только на основе одной выбранной записи ACL. На втором этапе проверяется, что запись ACL содержит необходимые права доступа.

Алгоритм проверки прав доступа имеет следующий вид:

1) Проверяется является ли субъект доступа пользователем-владельцем объекта. Если субъект доступа не является пользователем-владельцем объекта, то проверка продолжается. Если субъект доступа является пользователем-владельцем объекта, то проверяется разрешен ли в соответствии с выбранной записью ACL запрошенный субъектом вид доступа. По результатам проверки доступ либо разрешается либо запрещается.

2) Проверяется является ли субъект доступа одним из именованных пользователей, указанных в записях ACL. Если субъект доступа не является именованным пользователем, указанным в записях ACL, то проверка продолжается. Если субъект доступа является именованным пользователем, то проверяется разрешен ли в соответствии с выбранной записью ACL и записью маски доступа ACL запрошенный субъектом вид доступа. По результатам проверки доступ либо разрешается либо запрещается.

3) Проверяется, является ли одна из групп субъекта доступа группой-владельца

объекта. Если ни одна из групп субъекта доступа не является группой-владельца объекта, то проверка продолжается. Если одна из групп субъект доступа является группой-владельца объекта, то проверяется разрешен ли в соответствии с выбранной записью ACL запрошенный субъектом вид доступа. По результатам проверки доступ либо разрешается либо запрещается.

4) Проверяется является ли одна из групп субъекта доступа одной из именованных групп, указанных в записях ACL. Если ни одна из групп субъекта доступа не является именованной группой, указанной в записях ACL, то проверка продолжается. Если одна из групп субъект доступа является именованной группой, то проверяется разрешен ли в соответствии с выбранной записью ACL и записью маски доступа ACL запрошенный субъектом вид доступа. По результатам проверки доступ либо разрешается либо запрещается.

5) Проверяется разрешен ли в соответствии с записью ACL для всех остальных запрошенный субъектом вид доступа. По результатам проверки доступ либо разрешается либо запрещается.

6) Если доступ не был разрешен при проведении предыдущих проверок, то доступ запрещается.

Реализованный в ОС механизм дискреционных ПРД предусматривает наличие у объектов-контейнеров ACL, используемого по умолчанию. Названный ACL наследуется объектами, создаваемыми в объекте-контейнере.

Объектами доступа являются:

– файлы;
– соединения (сокеты);
– сетевые пакеты;
– механизмы IPC (разделяемая память, очереди сообщений и др.).

Механизм, реализующий дискреционное разграничение доступа, обеспечивает возможность санкционированного изменения списка пользователей и списка защищаемых файловых объектов.

Право изменения ПРД предоставлено выделенному субъекту - суперпользователю root (пользователю с UID, имеющим значение 0). Администратор может изменять права с использованием механизма `sudo`. Кроме того, права доступа к объекту, как указанные в битовой маске для трех классов (пользователя-владельца, группы-владельца и всех остальных), так указанные в записях ACL могут быть изменены субъектом являющимся пользователем-владельцем объекта.

Реализация в ОС механизма дискреционных ПРД обеспечивает непротиворечивость правил изменения дискреционных ПРД.

При использовании для объекта минимального ACL прямое и обратное отображение ПРД обеспечивается следующим образом:

1) При изменении прав доступа в битовой маске для пользователя-владельца идентичным образом изменяются права доступа для пользователя-владельца в записи ACL.

2) При изменении прав доступа для пользователя-владельца в записи ACL идентичным образом изменяются права доступа в битовой маске для пользователя-владельца.

3) При изменении прав доступа в битовой маске для группы-владельца идентичным образом изменяются права доступа для группы-владельца в записи ACL.

4) При изменении прав доступа для группы-владельца в записи ACL идентичным образом изменяются права доступа в битовой маске для группы-владельца.

5) При изменении прав доступа в битовой маске для всех остальных идентичным образом изменяются права доступа для всех остальных в записи ACL.

6) При изменении прав доступа для всех остальных в записи ACL идентичным образом изменяются права доступа для всех остальных в битовой маске.

При использовании для объекта расширенного ACL прямое и обратное отображение ПРД обеспечивается следующим образом:

1) При изменении прав доступа в битовой маске для пользователя-владельца идентичным образом изменяются права доступа для пользователя-владельца в записи ACL.

2) При изменении прав доступа для пользователя-владельца в записи ACL идентичным образом изменяются права доступа в битовой маске для пользователя-владельца.

3) При изменении прав доступа в битовой маске для группы-владельца идентичным образом изменяется маска доступа в записи ACL.

4) При изменении маски доступа в записи ACL идентичным образом изменяются права доступа в битовой маске для группы-владельца.

5) При изменении прав доступа в битовой маске для всех остальных идентичным образом изменяются права доступа для всех остальных в записи ACL.

6) При изменении прав доступа для всех остальных в записи ACL идентичным образом изменяются права доступа для всех остальных в битовой маске.

Таким образом, реализованный в ОС механизм, регулирующий дискреционный принцип контроля доступа, предусматривает санкционированное изменение дискреционных ПРД, включая санкционированное изменение списка субъектов и списка защищаемых объектов.

3.2. Linux-привилегии

Linux-привилегии предназначены для передачи отдельным пользователям прав выполнения определенных административных действий и являющихся стандартными для системы Linux: CAP_CHOWN, CAP_DAC_OVERRIDE, CAP_DAC_READ_SEARCH, CAP_FOWNER, CAP_FSETID, CAP_KILL, CAP_SETGID, CAP_SETUID, CAP_SETPCAP, CAP_LINUX_IMMUTABLE, CAP_NET_BIND_SERVICE, CAP_NET_BROADCAST, CAP_NET_ADMIN, CAP_NET_RAW, CAP_IPC_LOCK, CAP_IPC_OWNER, CAP_SYS_MODULE, CAP_SYS_RAWIO, CAP_SYS_CHROOT, CAP_SYS_PTRACE, CAP_SYS_PACCT, CAP_SYS_ADMIN, CAP_SYS_BOOT, CAP_SYS_NICE, CAP_SYS_RESOURCE, CAP_SYS_TIME, CAP_SYS_TTY_CONFIG, CAP_MKNOD, CAP_LEASE.

Linux-привилегии наследуются процессами от своих «родителей». Процессы, запущенные от имени суперпользователя, независимо от наличия у них привилегий имеют возможность осуществлять все перечисленные привилегированные действия.

Система привилегий ОС расширена привилегиями, относящимися к системе PARSEC и обеспечивающими работу с механизмом мандатного разграничения доступа (4.2).

Все привилегии пользователя наследуются запущенными от имени его учетной записи процессами. При запуске процесса с установленными привилегиями загрузчик динамических библиотек осуществляет сброс переменных среды окружения, позволяющих осуществлять загрузку динамических библиотек из нестандартных каталогов LD_LIBRARY_PATH и LD_PRELOAD. Таким образом, установка Linux-привилегий для пользователя может привести к невозможности запуска приложений, использующих динамическую загрузку библиотек из нестандартных каталогов (например, Firefox, Thunderbird, LibreOffice, fly-scan).

Командный интерфейс КСЗ может использовать как Linux-, так и PARSEC-привилегии для настройки ОС (4.2).

3.3. Средства управления дискреционными ПРД

Для управления дискреционными ПРД используется графическая утилита fly-fm («Менеджер файлов»). Более подробное описание утилиты см. в электронной справке.

Для управления Linux-привилегиями пользователей системы используется графическая утилита fly-admin-smc («Управление политикой безопасности»). Более подробное описание утилиты см. в электронной справке.

Далее рассмотрены средства управления дискреционными ПРД в режиме командной строки.

3.3.1. chown

Синтаксис:

```
chown [OPTION]... OWNER[:[GROUP]] FILE...
chown [OPTION]... :GROUP FILE...
chown [OPTION]... --reference=RFILE FILE...
```

Команда `chown` изменяет владельца и/или группу, владеющую каждым из указанных файлов, согласно заданным аргументам, которые интерпретируются в последовательном порядке. Если задано только имя пользователя (или его числовой идентификатор), то данный пользователь становится владельцем каждого из указанных файлов, а группа этих файлов не изменяется. Если за именем пользователя через двоеточие следует имя группы (или числовой идентификатор группы) без пробелов между ними, то изменяется также и группа файлов. Если двоеточие или точка следует за именем пользователя, но группа не задана, то данный пользователь становится владельцем указанных файлов, а группа указанных файлов изменяется на основную группу пользователя. Если опущено имя пользователя, а двоеточие или точка вместе с группой заданы, то будет изменена только группа указанных файлов; в этом случае `chown` выполняет ту же функцию, что и `chgrp` (3.3.2). Команда `chown` изменяет владельца и/или группу каждого `FILE` на `OWNER` и/или `GROUP`.

Опции приведены в таблице 1.

Таблица 1

Опция	Описание
`-c`, `--changes`	То же, что и `--verbose`. Подробно описывать только файлы, чей владелец действительно изменяется
`--dereference`	Изменить владельца файла, на который указывает символьная ссылка, вместо самой символьной ссылки
`-h`, `--no-dereference`	Работать с самими символьными ссылками, а не с файлами, на которые они указывают. Данная опция доступна, только если имеется системный вызов `lchown`
`--from=CURRENT_OWNER:CURRENT_GROUP`	Изменить владельца и/или группу каждого файла, только если текущий владелец и/или группа совпадает с `CURRENT_OWNER:CURRENT_GROUP`. Как группа, так и владелец могут быть опущены, в этом случае совпадение для данного атрибута не обязательно
`-f`, `--silent`, `--quiet`	Не выводить сообщения об ошибках на файлы, чей владелец не может быть изменен
`--reference=RFILE`	Вместо заданных значений `OWNER:GROUP` использовать владельца и группу файла, которые имеют `RFILE`
`-R`, `--recursive`	Рекурсивно изменять владельца каталогов и всего их содержимого
`--help`	Вывести справку и выйти
`--version`	Вывести информацию о версии и выйти

Владелец не изменяется, если он не существует. Группа также не изменяется, если отсутствует, но изменяется на группу по умолчанию, если не задан пользователь.

3.3.2. chgrp

Синтаксис:

```
chgrp [OPTION]... GROUP FILE...
chgrp [OPTION]... --reference=RFILE FILE...
```

Команда `chgrp` изменяет группу, владеющую каждым из указанных файлов `FILE`, на группу `GROUP`, которая может быть задана именем группы или числовым идентификатором группы.

Опции приведены в таблице 2.

Таблица 2

Опция	Описание
`-c`, `--changes`	То же, что и `--verbose`, но выводить сообщение только тогда, когда действительно была изменена группа файла
`--dereference`	Изменить владельца файла, на который указывает символьная ссылка, вместо самой символьной ссылки
`-h`, `--no-dereference`	Изменить владельца символьной ссылки, а не владельца файла, на который указывает эта ссылка (доступна только на системах, имеющих системный вызов `lchown`)
`-f`, `--silent`, `--quiet`	Не выводить сообщения об ошибках
`--reference=RFILE`	Изменить группу файла `FILE` на ту, что владеет файлом `RFILE`
`-R`, `--recursive`	Рекурсивно изменять владельца каталогов и их содержимого
`-v`, `--verbose`	Выводить диагностическое сообщение об изменении владельца для каждого файла
`--help`	Вывести справку и выйти
`--version`	Вывести информацию о версии и выйти

3.3.3. chmod

Синтаксис:

```
chmod [OPTION]... MODE[,MODE]... FILE...
chmod [OPTION]... OCTAL-MODE... FILE...
chmod [OPTION]... --reference=RFILE FILE...
```

Команда `chmod` изменяет права доступа указанного файла `FILE` в соответствии с правами доступа, указанными в параметре `MODE`, который может быть представлен как в символьном виде, так и в виде восьмеричного числа, представляющего битовую маску новых прав доступа.

Формат символьного режима:

```
[ugoa...][[+-=][rwxXstugo...]...][,...]
```

Каждый аргумент — это список символьных команд изменения прав доступа, разделенных запятыми. Каждая такая команда начинается с нуля или более букв `ugoa`, комбинация которых указывает, чьи права доступа к файлу будут изменены: пользователя, владеющего файлом (`u`); пользователей в данной группе (`g`); остальных пользователей, не входящих в данную группу (`o`), или же всех пользователей (`a`). Буква `a` эквивалентна `ugo`. Если не задана ни одна буква, то автоматически будет использоваться буква `a`, но биты, установленные в `umask`, не будут затронуты.

Оператор «+» добавляет выбранные права доступа к уже имеющимся у каждого файла; «-» удаляет эти права; а «=» присваивает только эти права каждому указанному файлу.

Буквы `rwxXstugo` выбирают новые права доступа для пользователя, заданного одной из букв `ugoa`: чтение (`r`); запись (`w`); исполнение (или доступ к каталогу) (`x`); выполнение, если файл является каталогом или уже имеет право на выполнение для какого-нибудь пользователя (`X`); setuid- или setgid-биты (`s`); sticky-бит (`t`); установка для остальных таких же прав доступа, которые имеет пользователь, владеющий этим файлом (`u`); установка для остальных таких же прав доступа, которые имеет группа файла (`g`); установка для остальных таких же прав доступа, которые имеют остальные пользователи (не входящие в группу файла) (`o`). (Так, `chmod g-s file` снимает бит `set-group-ID` (`sgid`), `chmod ug+s file` устанавливает биты `suid` и `sgid`, в то время как `chmod o+s file` ничего не делает.)

Числовой режим состоит из не более четырех восьмеричных цифр (от нуля до семи), которые складываются из битовых масок 4, 2 и 1. Любые пропущенные разряды дополняются лидирующими нулями. Первая цифра выбирает установку идентификатора пользователя (`setuid`) (4) или идентификатора группы (`setgid`) (2) или sticky-бита (1). Вторая цифра выбирает права доступа для пользователя, владеющего данным файлом: чтение (4), запись (2) и исполнение (1); третья цифра выбирает права доступа для пользователей, входящих в данную группу, с тем же смыслом, что и у второй цифры; и четвертый разряд выбирает права доступа для остальных пользователей (не входящих в данную группу), опять с тем же смыслом.

`chmod` никогда не изменяет права на символьные ссылки, т. к. этого не делает системный вызов `chmod`. Это не является проблемой: права символьных ссылок никогда не используются. Однако для каждой символьной ссылки, заданной в командной строке, `chmod` игнорирует символьные ссылки, встречающиеся во время рекурсивной обработки каталогов.

Команда `chmod` изменяет права доступа каждого файла `FILE` на `MODE`.

Опции приведены в таблице 3.

Таблица 3

Опция	Описание
`-c, --changes`	То же, что и `--verbose`, но выводить сообщение только тогда, когда были произведены изменения
`-f, --silent, --quiet`	Не выдавать сообщения об ошибках на те файлы, чьи права не могут быть изменены
`-v, --verbose`	Подробно описывать измененные права доступа
`--reference=RFILE`	Изменить права доступа к файлу на те права, что имеет `RFILE`
`-R, --recursive`	Рекурсивное изменение прав доступа для каталогов и их содержимого
`--help`	Вывести справку и выйти
`--version`	Вывести информацию о версии и выйти

Каждый `MODE` представляет собой комбинацию из одного или более символов `ugoa` в начале и один из символов «+», «-», «=», затем одна или несколько букв из `rwxXstugo`.

Символьная форма приведена в таблице 4.

Таблица 4

Опция	Описание
`u`	Пользователь (владелец файла) — от user (пользователь)
`g`	Группа — от group (группа)
`o`	Остальные пользователи — от other (остальные)
`a`	Все пользователи — от all (все)
`+`	Добавить разрешения к текущим правам доступа
`-`	Удалить разрешения из текущих прав доступа
`=`	Установить разрешения вне зависимости от текущих прав доступа
`r`	Разрешение на чтение — от read (читать)
`w`	Разрешение на изменение — от write (писать)
`x`	Разрешение на исполнение — от execute (выполнять)
`l`	Блокировка файла для других пользователей при доступе

3.3.4. umask

Синтаксис:

```
umask [-p] [-S] [маска]
```

Пользовательская маска создания файла устанавливается равной аргументу `маска`. Если маска начинается с цифры, она интерпретируется как восьмеричное число, иначе — как маска в символьном формате, аналогичном используемому в команде `chmod` (см. 3.3.3). Если маска не указана или задана опция `-S`, выдается текущее значение маски.

Опция `-S` вызывает выдачу маски в символьном формате; по умолчанию выдается восьмеричное число. Если указана опция `-p`, а маска не задана, результат выдается в виде, который можно использовать во входной команде. Статус выхода — 0, если маска была успешно изменена или не указана, и 1 — в противном случае.

Команда `umask` распознается и выполняется оболочкой `shell`.

Команду `umask` целесообразно включить в пользовательский pro-файл. Тогда она будет автоматически вызываться при входе в систему и установит нужный режим доступа к создаваемым файлам и каталогам.

3.3.5. getfacl

Синтаксис:

```
getfacl [-dRLP] ƒайл ...
```

Для каждого файла `getfacl` выводит имя файла, владельца, группу-владельца и ACL. Если каталог имеет ACL-по умолчанию, то `getfacl` выводит также ACL-по умолчанию. Файлы не могут иметь ACL-по умолчанию.

Формат вывода:

```
1:  # file: somedir/
2:  # owner: lisa
3:  # group: staff
4:  user::rwx
5:  user:joe:rwx               #effective:r-x
6:  group::rwx                 #effective:r-x
7:  group:cool:r-x
8:  mask:r-x
9:  other:r-x
10:  default:user::rwx
11:  default:user:joe:rwx       #effective:r-x
12:  default:group::r-x
13:  default:mask:r-x
14:  default:other:---
```

Строки 4, 6 и 9 относятся к традиционным битам прав доступа к файлу, соответственно, для владельца, группы-владельца и всех остальных. Эти три элемента являются базовыми. Строки 5 и 7 являются элементами для отдельных пользователя и группы. Строка 8 — маска эффективных прав. Этот элемент ограничивает эффективные права, предоставляемые всем группам и отдельным пользователям. Маска не влияет на права для владельца файла и всех других. Строки 10–14 показывают ACL-по умолчанию, ассоциированный с данным каталогом.

Команда `getfacl` выводит ACL файлов и каталогов по умолчанию.

Для большого количества файлов `getfacl` выводит ACL, разделенные пустыми строками. Результаты команды `getfacl` могут использоваться как входные данные для команды `setfacl` (3.3.6).

Опции приведены в таблице 5.

Таблица 5

Опция	Описание
`--access`	Вывести только ACL файла
`-d, --default`	Вывести только ACL-по умолчанию
`--omit-header`	Не показывать заголовок (имя файла)
`--all-effective`	Показать все эффективные права
`--no-effective`	Не показывать эффективные права
`--skip-base`	Пропускать файлы, имеющие только основные записи
`-R, --recursive`	Для подкаталогов рекурсивно
`-L, --logical`	Следовать по символическим ссылкам, по умолчанию символические ссылки, не указанные в командной строке, игнорируются
`-P, --physical`	Не следовать по символическим ссылкам, даже если они указаны в командной строке
`--tabular`	Использовать табулированный формат вывода
`--numeric`	Показывать числовые значения пользователя/группы
`--absolute-names`	Не удалять ведущие «/» из пути файла
`--help`	Вывести справку и выйти
`--version`	Вывести информацию о версии и выйти

3.3.6. setfacl

Синтаксис:

```
setfacl [-bkndRLP] { -m|-M|-x|-X ... } файл ...
```

Эта команда изменяет ACL к файлам или каталогам. В командной строке за последовательностью команд идет последовательность файлов (за которой, в свою очередь, также может идти последовательность команд и т. д.).

Опции приведены в таблице 6.

Таблица 6

Опция	Описание
`-m, --modify=acl`	Изменить текущий ACL для файла
`-M, --modify-file=file`	Прочитать записи ACL для модификации из файла
`-x, --remove=acl`	Удалить записи из ACL файла
`-X, --remove-file=file`	Прочитать записи ACL для удаления из файла

Окончание таблицы 6

Опция	Описание
`-b`, `--remove-all`	Удалить все расширенные записи ACL
`-k`, `--remove-default`	Удалить ACL-по умолчанию
`--set=acl`	Установить ACL для файла, заменив текущий ACL
`--set-file=datei`	Прочитать записи ACL для установления из файла
`--mask`	Пересчитать маску эффективных прав
`-n`, `--no-mask`	Не пересчитывать маску эффективных прав, обычно `setfacl` пересчитывает маску (кроме случая явного задания маски) для того, чтобы включить ее в максимальный набор прав доступа элементов, на которые воздействует маска (для всех групп и отдельных пользователей)
`-d`, `--default`	Применить ACL-по умолчанию
`-R`, `--recursive`	Для подкаталогов рекурсивно
`-L`, `--logical`	Следовать по символическим ссылкам, по умолчанию ссылки, не указанные в командной строке, игнорируются
`-P`, `--physical`	Не следовать по символическим ссылкам, даже если они указаны в командной строке
`--restore=file`	Восстановить резервную копию прав доступа, созданную командой `getfacl -R` или ей подобной. Все права доступа дерева каталогов восстанавливаются, используя этот механизм. Если вводимые данные содержат элементы для владельца или группы-владельца и команда `setfacl` выполняется пользователем с именем `root`, то владелец и группа-владелец всех файлов также восстанавливаются. Эта опция не может использоваться совместно с другими опциями, за исключением опции `--test`
`--test`	Режим тестирования (ACL не изменяются)
`--help`	Вывести справку и выйти
`--version`	Вывести информацию о версии и выйти

При использовании опций `--set`, `-m` и `-x` должны быть перечислены записи ACL в командной строке. Элементы ACL разделяются одинарными кавычками.

При чтении ACL из файла при помощи опций `--set-file`, `-M` и `-X` команда `setfacl` принимает множество элементов в формате вывода `getfacl`. В строке обычно содержится не больше одного элемента ACL.

3.3.6.1. Элементы ACL

Команда `setfacl` использует следующие форматы элементов ACL:

1) `[d[efault]:] [u[ser]:]uid [:[+|^]perms]`

Права доступа отдельного пользователя. Если не задан `uid`, то права доступа владельца файла.

2) `[d[efault]:] g[roup]:gid [:[+|^]perms]`

Права доступа отдельной группы. Если не задан `gid`, то права доступа группы-владельца.

3) `[d[efault]:] m[ask]:[+|^] perms`

Маска эффективных прав.

4) `[d[efault]:] o[ther]:[+|^] perms`

Права доступа всех остальных.

Элемент ACL является абсолютным, если он содержит поле `perms` и является относительным, если он включает один из модификаторов: «+» или «^». Абсолютные элементы могут использоваться в операциях установки или модификации ACL. Относительные элементы могут использоваться только в операции модификации ACL. Права доступа для отдельных пользователей, группы, не содержащие никаких полей после значений `uid`, `gid` (поле `perms` при этом отсутствует), используются только для удаления элементов.

Значения `uid` и `gid` задаются именем или числом. Поле `perms` может быть представлено комбинацией символов `r`, `w`, `x`, `-` или цифр (0–7).

3.3.6.2. Автоматически созданные права доступа

Изначально файлы и каталоги содержат только три базовых элемента ACL: для владельца, группы-владельца и всех остальных пользователей. Существует ряд правил, которые следует выполнять:

1) не могут быть удалены сразу три базовых элемента. Должен присутствовать хотя бы один;

2) если ACL содержит права доступа для отдельного пользователя или группы, то ACL также должен содержать маску эффективных прав;

3) если ACL содержит какие-либо элементы ACL-по умолчанию, то в последнем должны также присутствовать три базовых элемента (т. е. права доступа по умолчанию для владельца, группы-владельца и всех остальных);

4) если ACL-по умолчанию содержит права доступа для отдельных пользователей или групп, то в ACL также должна присутствовать маска эффективных прав.

Для того чтобы помочь пользователю выполнять эти правила, `setfacl` создает права доступа, используя уже существующие, согласно следующим условиям:

1) если права доступа для отдельного пользователя или группы добавлены в ACL, а маски прав не существует, то создается маска с правами доступа группы-владельца;

2) если создан элемент ACL-по умолчанию, а трех базовых элементов не было, тогда делается их копия и они добавляются в ACL-по умолчанию;

3) если ACL-по умолчанию содержит какие-либо права доступа для конкретных пользователя или группы и не содержит маску прав доступа по умолчанию, то при

создании эта маска будет иметь те же права, что и группа по умолчанию.

3.4. Дискреционное разграничение доступа в СУБД PostgreSQL

В качестве защищенной СУБД в составе ОС используется PostgreSQL, доработанная в соответствии с требованием интеграции с ОС в части мандатного разграничения доступа к информации.

Примечание. В текущей версии ОС представлены две версии защищенной СУБД (на базе версий СУБД PostgreSQL 9.2 и 9.4), отличающиеся в части реализации мандатного разграничения доступа к информации. Различия в части реализации дискреционного разграничения доступа отсутствуют. Дальнейшее описание дискреционного разграничения доступа в СУБД справедливо для обеих версий.

СУБД PostgreSQL является объектно-реляционной. На низком уровне данные хранятся в отношениях (таблицах, видах), и доступ к данным разграничивается в понятиях реляционной СУБД.

Данные в реляционной БД хранятся в отношениях (таблицах), состоящих из строк и столбцов. При этом единицей хранения и доступа к данным является строка, состоящая из полей, идентифицируемых именами столбцов. Кроме таблиц, существуют другие объекты БД (виды, процедуры и т. п.), которые предоставляют доступ к данным, хранящимся в таблицах.

С каждым типом объектов БД ассоциируется определенный набор типов доступа (возможных операций). Для каждого объекта явно задается список разрешенных для каждого из поименованных субъектов БД (пользователей, групп или ролей) типов доступа (т. е. ACL). И в дальнейшем при разборе запроса к БД осуществляется проверка возможности предоставления доступа субъекта к объекту типа, соответствующего запросу.

В общем случае отдельная строка таблицы не является однозначно идентифицируемым объектом (каждая строка идентифицируется только набором содержимого своих полей, но без специальных действий, например создания первичного ключа или физического уникального идентификатора строки в БД, такая идентификация не является уникальной), и дискреционные правила разграничения доступа к ней применены быть не могут. В PostgreSQL объектами дискреционного разграничения доступа могут являться и столбцы объектов, поскольку могут быть однозначно идентифицированы по составному имени объекта и столбца, т. к. имя столбца внутри объекта является уникальным.

В рамках дискреционных ПРД определены следующие операции над таблицами и хранящимися в них данными:

– `SELECT` — чтение данных из таблицы;

– `INSERT` — вставка новых данных в таблицу;

– DELETE — удаление некоторых/всех данных в таблице;

– UPDATE — изменение данных в таблице;

– REFERENCES — использование данных таблицы для внешних ключей;

– TRIGGER — создание и назначение для таблицы триггеров;

– TRUNCATE — очистка таблицы (удаление всех данных).

Для более гибкой работы с данными в СУБД введены следующие объекты, к каждому из которых так же существует набор операций:

1) вид — способ организации предварительно подготовленных запросов. Набор операций совпадает с набором операций для таблиц, за исключением создания триггеров и внешних ключей:

– SELECT — чтение данных из вида;

– INSERT — вставка новых данных в вид;

– DELETE — удаление некоторых/всех данных в виде;

– UPDATE — изменение данных в виде;

2) последовательность — способ получения уникальных значений (счетчик). Определены следующие операции:

– SELECT — чтение значения счетчика;

– UPDATE — установка значения счетчика;

– USAGE — выполнение функций манипулирования счетчиком;

3) БД — способ организации области данных, содержащих все остальные объекты СУБД. Определены следующие операции:

– CREATE — создание БД;

– CONNECT — установка соединения с БД;

– TEMPORARY/TEMP — создание временных таблиц в БД;

4) функция — программный код манипулирования данными на сервере. Определена операция EXECUTE — выполнение функции;

5) язык — язык написания функций на сервере. Определена операция USAGE — использование языка для написания функций;

6) схема — способ организации объектов в пределах отдельной БД. Определены следующие операции:

– CREATE — создание объектов в указанной схеме;

– USAGE — использование объектов указанной схемы;

7) табличное пространство — способ организации БД в ФС ОС. Определена операция CREATE — создание объектов в указанном табличном пространстве.

8) бинарный объект — способ хранения больших двоичных объектов (файлов, документов, фотографий, и т.п.) в БД. Определены следующие операции:

– SELECT — чтение бинарного объекта;

– UPDATE — изменение бинарного объекта;

9) В БД могут присутствовать дополнительные объекты, для использования которых определена операция USAGE.

Для контроля выполнения всех перечисленных операций дискреционных ПРД существуют соответствующие права доступа. Право на предоставление прав доступа к объектам не может быть предоставлено другим пользователям и доступно только администратору БД (при соответствующих настройках сервера может быть предоставлено и владельцу объекта).

Кроме рассмотренных (делегируемых) прав доступа, существует ряд прав, которые всегда принадлежат владельцам объектов и администраторам СУБД. Эти права не могут быть делегированы или отменены средствами СУБД. К таким правам относятся: удаление и модификация объекта и назначение пользователям делегируемых прав доступа к объектам.

Сразу же после создания объекта только его владелец и администраторы СУБД могут использовать его каким-либо образом. Для того чтобы с этим объектом могли работать другие пользователи, владелец объекта или администратор СУБД должен явно предоставить им соответствующие дискреционные права доступа.

Модификация метаданных возникает каждый раз при изменении структуры БД, что включает в себя создание, модификацию и удаление объектов БД.

Разграничение доступа к перечисленным операциям на уровне СУБД так же реализуется применением дискреционных ПРД. Для этого используется право владения объектом, право на создание объектов. Право владения объектом предоставляет владельцу объекта возможность модифицировать и удалять объект. Как правило, владельцем является создатель объекта или суперпользователь (администратор БД). Право на создание (CREATE) существует к объектам БД, являющихся контейнерами для других объектов, а именно: непосредственно сама БД, схема, табличное пространство.

При выполнении любого запроса пользователя (субъекта БД) к защищаемому ресурсу (объекту БД) выполняется дискреционное разграничение доступа на основе установленных пользователю прав. Для каждой выполняемой операции производится проверка наличия права у пользователя на выполнение данной конкретной операции.

Дискреционные ПРД применяются после разбора запроса пользователя и построения плана его выполнения. Дискреционные ПРД к столбцам объекта применяются только при отсутствии явного разрешения на доступ к самой таблице. Таким образом, права доступа к объекту являются доминирующими. При этом при отсутствии явно заданных прав на объект нельзя сказать определенно о предоставлении доступа до тех пор, пока не будут

проверены права на столбцы объекта.

В СУБД PostgreSQL параметр конфигурации ac_enable_trusted_owner позволяет администратору запретить владельцам объектов передавать права на доступ к ним другим пользователям СУБД. В случае установки значения этой переменной конфигурации в FALSE распределение прав доступа к объектам БД разрешено только администраторам СУБД.

Параметр конфигурации ac_allow_grant_options позволяет администратору запретить передачу уже имеющихся прав доступа на объект другим ролям. Если ac_allow_grant_options установлен в FALSE, то запрещается использовать команду GRANT с привилегией WITH GRANT OPTION. Если у роли есть привилегия GRANT OPTIONS и ac_allow_grant_options = false, то передача прав доступа другим ролям также запрещается. Изъятие (REVOKE) привилегии GRANT OPTIONS разрешается всегда.

Параметр конфигурации ac_allow_admin_options позволяет администратору запретить передачу прав членства роли другим ролям. Если ac_allow_admin_options установлен в FALSE, то запрещается использовать GRANT с привилегией WITH ADMIN OPTION. Если у роли есть привилегия ADMIN OPTIONS и ac_allow_admin_options = false, то передача прав членства другим ролям также запрещается. Изъятие (REVOKE) привилегии ADMIN OPTIONS разрешается всегда.

Параметр конфигурации ac_enable_truncate позволяет администратору запретить владельцам объектов и любым пользователям, обладающим соответствующим правом TRUNCATE, выполнять удаление всех записей из таблиц. В случае установки значения этой переменной конфигурации в FALSE выполнение команды TRUNCATE запрещено всем пользователям.

3.4.1. Средства управления дискреционными ПРД к объектам БД СУБД PostgreSQL

Для управления дискреционными ПРД к объектам БД СУБД PostgreSQL используется графическая утилита pgadmin3 («Средство администрирования СУБД PostgreSQL»).

Для делегирования дискреционных прав доступа к объектам используется команда SQL GRANT, а для отмены — команда REVOKE. Например, если в системе существует пользователь ivanov, то ему может быть предоставлено право на изменение данных в таблице Счета с помощью следующей команды:

```
GRANT UPDATE ON "Счета" TO ivanov
```

Для предоставления прав доступа к объекту сразу всем пользователям системы существует специальное «имя пользователя» PUBLIC, а для предоставления всех прав — специальное «право» ALL. Например, чтобы дать всем пользователям полный доступ к таблице Счета, следует использовать следующую команду:

```
GRANT ALL ON "Счета" TO PUBLIC
```

При необходимости право доступа может быть предоставлено пользователю (но не группе) с возможностью делегирования данного права другим ролям. Для этого используется ключевая фраза `WITH GRANT OPTION`:

```
GRANT UPDATE ON "Счета" TO ivanov WITH GRANT OPTION
```

Владелец объекта может отменить собственные делегируемые права, например, переведя объект в режим «только для чтения» для себя, так же как и для всех остальных пользователей.

4. МАНДАТНОЕ РАЗГРАНИЧЕНИЕ ДОСТУПА

4.1. Общие сведения

Механизм контроля мандатного разграничения доступа реализован, как и механизм дискреционного разграничения доступа, в ядре ОС. При этом, принятие решения о запрете или разрешении доступа субъекта к объекту принимается на основе типа операции (чтение/запись/исполнение), мандатного контекста безопасности субъекта и мандатной метки объекта.

Правила принятия решения могут быть записаны следующим образом. Пусть контекст безопасности субъекта содержит уровень `L0`, уровень целостности `iL0` и категории `C0`, а мандатная метка объекта содержит уровень `L1`, уровень целостности `iL1` и категории `C1`. Определим операции сравнения для уровней и категорий:

1) уровень `L0` меньше уровня `L1` (`L0<L1`), если численное значение `L0` меньше численного значения `L1`;

2) уровень `L0` равен уровню `L1` (`L0==L1`), если численные значения `L0` и `L1` совпадают;

3) уровень целостности `iL0` меньше уровня `iL1` (`iL0<iL1`), если численное значение `iL0` меньше численного значения `iL1`;

4) уровень целостности `iL0` равен уровню целостности `iL1` (`iL0==iL1`), если численные значения `iL0` и `iL1` совпадают;

5) категории `C0` меньше категорий `C1` (`C0<C1`), если все биты набора `C0` являются подмножеством набора бит `C1`;

6) категории `C0` равны категориям `C1` (`C0==C1`), если значения `C0` и `C1` совпадают;

7) операция записи разрешена, если `L0==L1`, `iL0>=iL1` и `C0==C1`;

8) операция чтения разрешена, если `L0>=L1`, `C0>=C1`, $\forall$`iL0`, $\forall$`iL1`;

9) операция исполнения разрешена, если `L0>=L1` и `C0>=C1`, $\forall$`iL0`, $\forall$`iL1`.

В настоящий момент в системе могут использоваться два уровня целостности: высокий (`hi`) – значение 1; низкий (`low`) – значение 0.

В остальных случаях анализируются полномочия субъекта (см. 4.2) и тип мандатной метки объекта. В ОС предусмотрено существование объектов-контейнеров (например, каталогов), т.е. объектов, которые могут содержать другие объекты. Метка объекта-контейнера определяет максимальную метку вложенных объектов. Тип метки может использоваться для того, чтобы изменять ее эффективное действие:

– тип метки `ccnr` применяется к объектам-контейнерам и определяет, что объект контейнер может содержать объекты с различными мандатными метками, но не превышающими метку объекта контейнера;

– тип метки ccnri применяется к объектам-контейнерам и определяет, что объект контейнер может содержать объекты с различными уровнями целостности, но не превышающими уровень целостности объекта-контейнера;

– тип метки ehole применяется к объектам-контейнерам и простым объектам для игнорирования мандатных правил разграничения доступа к ним.

Ненулевой тип метки может быть установлен только привилегированным процессом. Перечисленные типы могут использоваться совместно. Таким образом, объект-контейнер может иметь тип: ccnr,ccnri,ehole.

Для создания в объекте-контейнере, имеющем тип ccnr, вложенного объекта с уровнем и категориями меньшими чем у объекта-контейнера необходимо обладать специальными привилегиями (см. 4.2). Для создания в объекте-контейнере, имеющем тип ccnri, вложенного объекта с уровнем целостности меньшим чем у объекта-контейнера необходимо обладать специальными привилегиями (см. 4.2). Названные действия могут быть выполнены администратором через механизм sudo.

Следует учесть, в контейнере с меткой типа ehole уровень, категории и уровень целостности вложенных объектов не могут превышать уровень, категории и уровень целостности объекта-контейнера.

ВНИМАНИЕ! Мандатные атрибуты на корне файловой системы определяет максимальный мандатный контекст безопасности объектов. Мандатные атрибуты, устанавливаемые по умолчанию на корень файловой системы и ряд вложенных объектов определены в сценарии pdp-init-fs, который расположен в каталоге /usr/sbin.

ВНИМАНИЕ! По умолчанию максимальный уровень целостности для объектов файловой системы равен 0 и вход в систему пользователей с высокой целостностью невозможен. Для использования мандатного контроля целостности необходимо модифицировать сценарий pdp-init-fs, установив в 1 значение переменной sysmaxilev и флаг ccnri для контейнеров с флагом ccnr, и выполнить сценарий или перезагрузку ОС.

Указанная модификация сценария позволяет установить для определенных объектов файловой системы значение уровня целостности в 1, обеспечив их защиту от изменения пользователем, работающим в сеансе с низким уровнем целостности и не имеющим привилегий игнорирования мандатного контроля доступа.

Механизм мандатного разграничения доступа затрагивает следующие подсистемы:

– механизмы IPC;

– стек TCP/IP (IPv4);

– слой ВФС;

– ФС Ext2/Ext3/Ext4;

– сетевые ФС CIFS;

– ФС proc, tmpfs.

С каждым субъектом и объектом связаны: мандатный контекст безопасности и мандатная метка, соответственно.

При создании субъектом любого из вышеприведенных объектов объект наследует метку на основе мандатного контекста безопасности процесса. При этом тип метки устанавливается в 0. Если ФС поддерживает только нулевые метки (например, VFAT), то на ней невозможно создание объектов с меткой, отличной от нулевой.

ВНИМАНИЕ! Не рекомендуется устанавливать ненулевые мандатные атрибуты (уровень и категории) для учетных записей администраторов (ОС, системы печати, СУБД и т. д.)

4.2. PARSEC-привилегии

PARSEC-привилегии приведены в таблице 7.

Таблица 7

Привилегия	Описание
PARSEC_CAP_SIG	Посылать сигналы процессам, игнорируя дискреционные и мандатные права
PARSEC_CAP_SETMAC	Изменить мандатную метку и установить другие привилегии
PARSEC_CAP_CHMAC	Менять мандатные метки файлов
PARSEC_CAP_AUDIT	Управление политикой аудита
PARSEC_CAP_READSEARCH	Игнорировать мандатную политику при чтении и поиске файлов (но не при записи)
PARSEC_CAP_PRIV_SOCK	Создавать привилегированный сокет и менять его мандатную метку. Привилегированный сокет позволяет осуществлять сетевое взаимодействие, игнорируя мандатную политику
PARSEC_CAP_UPDATE_ATIME	Изменять время доступа к файлу
PARSEC_CAP_IGNMACLVL	Игнорировать мандатную политику по уровням
PARSEC_CAP_IGNMACCAT	Игнорировать мандатную политику по категориям
PARSEC_CAP_FILE_CAP	Устанавливать привилегии на файлы
PARSEC_CAP_CAP	Устанавливать любой непротиворечивый набор привилегий для другого процесса
PARSEC_CAP_MAC_SOCK	Смена мандатной точки соединения
PARSEC_CAP_UNSAFE_SETXATTR	Устанавливать мандатные атрибуты объектов ФС без учета мандатных атрибутов родительского объекта-контейнера. Привилегия используется для восстановления объектов ФС из резервных копий (см. 11.2) и только после установки значения 1 для параметра /parsecfs/unsecure_setxattr
PARSEC_CAP_IGNMACINT	Игнорировать мандатную политику по уровням целостности

PARSEC-привилегии так же, как и Linux-привилегии наследуются процессами от своих «родителей». Процессы, запущенные от имени суперпользователя, независимо от наличия у них привилегий имеют возможность осуществлять все перечисленные привилегированные действия.

Командный интерфейс КСЗ может использовать как PARSEC-, так и Linux-привилегии для настройки ОС (см. 3.2).

4.3. Сетевое взаимодействие

В качестве основной сетевой ФС используется CIFS, которая является расширением SMB и поддерживает атрибуты ФС UNIX и имеет ограниченную поддержку расширенных атрибутов. Кроме того, эта ФС широко распространена и работает в гетерогенных сетях (поддерживается многими ОС). Поддерживает аутентификацию средствами PAM и Kerberos.

Сетевые соединения могут рассматриваться как IPC, поэтому должны подвергаться мандатному контролю доступа. Для этого в сетевые пакеты протокола IPv4 в соответствии со стандартом RFC1108 внедряются мандатные метки, соответствующие метке объекта — сетевое соединение (сокет). При этом метка сокета наследуется от субъекта (процесса). Прием сетевых пакетов подчиняется мандатным ПРД. Следует отметить, что метка сокета может иметь тип, позволяющий создавать сетевые сервисы, принимающие соединения с любыми уровнями секретности.

В рамках стандарта RFC1108 метка снабжается классом `0xAB` (Unclassified), при этом последующий битовый список (последовательность байт, в которых младший бит указывает на наличие следующего байта в потоке) опции представляет собой упакованную в соответствии со стандартом структуру мандатного контекста, где уровень занимает 8 бит, а категории — 64 бита (порядок байт — от младших к старшим). Последние (старшие) нулевые биты в соответствии со стандартом могут быть отброшены.

Пример кодирования метки для протокола IPv4:

```
IPOPT_SEC,5,0xAB,03,12 /* Битовый список: 00000011, 00001100, Контекст:
   уровень 1, категории 3) */
```

Отсутствие метки на объекте доступа является синонимом нулевой мандатной метки. Таким образом, ядро ОС, в которой все объекты и субъекты доступа имеют уровень секретности «несекретно», функционирует аналогично стандартному ядру ОС Linux.

Для ряда сетевых сервисов (сервера LDAP, DNS, Kerberos и т. д.) необходимо обеспечить возможность их работы с клиентами, имеющими разный мандатный контекст безопасности, без внесения изменений в исходные тексты сервиса. Для предоставления названной возможности в подсистеме безопасности PARSEC реализован механизм запуска сетевых сервисов с использованием привилегированного сокета для ожидания входящих

соединений (механизм privsock, 4.3.1).

4.3.1. Механизм privsock

Механизм privsock предназначен для обеспечения функционирования системных сетевых сервисов, не осуществляющих обработку информации с использованием мандатного контекста, но взаимодействующих с процессами, работающими в мандатном контексте субъекта доступа.

Для его использования при функционировании сетевого сервиса необходимо отредактировать файл /etc/parsec/privsock.conf, добавив в него строку, содержащую полный путь к исполняемому файлу сервиса. Далее приведен пример строки из файла /etc/parsec/privsock.conf для запуска DNS-сервера с использованием механизма privsock.

Пример

```
/usr/sbin/named
```

Для использования механизма privsock необходимо, чтобы переменная PATH, используемая при запуске сервиса, содержала следующий путь:

```
/usr/lib/parsec/bin
```

Далее приведен пример задания требуемого пути в переменной окружения для запуска DNS-сервера.

Пример

Установка значения переменной окружения PATH может быть выполнена добавлением в файл /etc/default/bind9 следующей строки:

```
PATH=/usr/lib/parsec/bin:$PATH
```

Подробное описание механизма privsock приведено в man privsock.

4.4. Средства управления мандатными ПРД

Для управления мандатными ПРД используются следующие графические утилиты:

– fly-fm («Менеджер файлов») — управление мандатными атрибутами файлов;

– fly-admin-smc («Управление политикой безопасности») — управление протоколированием, привилегиями и мандатными атрибутами пользователей, работа с пользователями и группами.

Более подробное описание утилит см. в электронной справке.

Для управления мандатными ПРД в режиме командной строки используются следующие утилиты:

– pdpl-file — управление мандатными атрибутами файлов (см.4.4.1), утилита доступна в том числе через символическую ссылку pdp-flbl;

– pdp-id — отображение мандатных атрибутов сессии пользователя ОС (см.4.4.2);

– pdp-init-fs — скрипт инициализации мандатных атрибутов ФС (см.4.4.3);

– pdp-ls — вывод аналогично стандартной команде ls информации о файлах с отображением мандатных атрибутов (см.4.4.4);

– pdpl-ps — управление мандатными атрибутами процессов (см.4.4.5), утилита доступна в том числе через символическую ссылку pdp-pslbl;

– pdpl-user — управление допустимыми мандатными уровнями и категориями пользователей ОС (см.4.4.6), утилита доступна в том числе через символическую ссылку pdp-ulbls;

– sumac — запуск процесса с заданными мандатными уровнем и категорией в отдельной графической сессии (см.4.4.7);

– userlev — изменение БД мандатных уровней (см.4.4.8);

– usercat — изменение БД мандатных категорий (см.4.4.9).

Для совместимости с предыдущими версиями ОС сохранены следующие утилиты командной строки для управления мандатными ПРД:

– chmac — управление мандатными атрибутами файлов (см.4.4.10.1);

– lsm — вывод аналогично стандартной команде ls информации о файлах с отображением мандатных атрибутов (см.4.4.10.3);

– macid — отображение мандатных атрибутов сессии пользователя ОС (см.4.4.10.2);

– psmac — управление мандатными атрибутами процессов (см.4.4.10.4);

– usermac — управление допустимыми мандатными уровнями и категориями пользователей ОС (см.4.4.10.5);

– getfmac — получение мандатных меток файловых объектов (см.4.4.10.6);

– setfmac — изменение мандатных меток файловых объектов (см.4.4.10.7).

Далее рассмотрены средства управления мандатными ПРД в режиме командной строки.

4.4.1. pdpl-file

Синтаксис:

```
pdpl-file [опции]...[уровень][:уровень целостности[:категория[:флаги]]] [файл]
```

Команда pdpl-file изменяет мандатные атрибуты файлов ОС, которые включают мандатную метку и специальные мандатные атрибуты файла.

Опции приведены в таблице 8.

Таблица 8

Опция	Описание
`-f`, `--silent`, `--quiet`	Не выводить сообщений об ошибках
`-v`, `--verbose`	Выводить диагностические сообщения для каждого файла
`-c`, `--changes`	То же, что и `--verbose`, но сообщать только об изменениях
`-R`, `--recursive`	Применить рекурсивно
`-h`, `--help`	Вывести справку и выйти
`--version`	Вывести информацию о версии и выйти

Уровень и категория могут быть заданы именем или шестнадцатеричным значением.

Пример

```
pdpl-file -Rv Секретно:0:Категория_A /tmp
```

Данная команда рекурсивно для всех файлов каталога `/tmp` изменит уровень на `Секретно` и категорию на `Категория_A` (уровень и категория должны быть определены в системе).

Флаги могут быть заданы значением или именами через запятую:

– `ccnr` — неприменение метки конфиденциальности контейнера (каталога) при просмотре его содержимого;

– `ccnri` — неприменение уровня целостности контейнера (каталога) при просмотре его содержимого;

– `ehole` — игнорирование мандатных уровней и категорий при выполнении операций записи.

Флаги могут быть заданы одним из псевдонимов:

– `CCNRA` — псевдоним для сочетания `ccnr` и `ccnri`;

– `ALL` — псевдоним для сочетания всех возможных флагов (в настоящее время `ccnr`, `ccnri` и `ehole`).

Пример

```
pdpl-file 2:0:0:ccnr /tmp
```

4.4.2. pdp-id

Синтаксис:

```
pdp-id [опции]
```

Команда `pdp-id` выводит мандатные атрибуты сессии пользователя ОС.

Опции приведены в таблице 9.

Таблица 9

Опция	Описание
-l,--level	Вывести только мандатный уровень конфиденциальности
-i,--ilevel	Вывести только мандатный уровень целостности
-c,--categories	Вывести только мандатную категории
-r,--categories	Вывести только роли
-n,--name	Для опций -lc выводить имена вместо числовых значений
-h,--help	Вывести справку и выйти
--version	Вывести информацию о версии и выйти

При отсутствии опций выводит строку текущих мандатных свойств.

Пример

```
pdp-id
Уровень конф: Секретно Уровень целостности: low Категории: Категория_A
```

В этом примере текущая сессия пользователя имеет уровень секретности Секретно, низкий уровень целостности и категорию Категория_A.

4.4.3. pdp-init-fs

Синтаксис:

```
pdp-init-fs
```

Скрипт инициализации мандатных атрибутов ФС pdp-init-fs вызывается при инициализации и перезапуске системы для установки корректных мандатных атрибутов на системные файловые объекты (файлы и каталоги), начина с корня ФС.

Скрипт располагается в каталоге /usr/sbin и доступен для правки только администратору.

ВНИМАНИЕ! Настоятельно не рекомендуется вносить изменения в указанный скрипт без необходимости. Изменения требуются только в случае изменения максимального уровня конфиденциальности или целостности в системе.

4.4.4. pdp-ls

Синтаксис:

```
pdp-ls [опции] [имя файла]
```

Команда pdp-ls выводит аналогично стандартной команде ls информацию о файлах (по умолчанию — о текущем каталоге).

Использование данной команды в целом не отличается от использования pdp-ls, за исключением следующих особенностей:

– если на файле установлены ACL, то к ним добавляется символ «+»;

– если на файле установлена ненулевая мандатная метка, то к ACL добавляется символ «m»;

– если на файле установлены списки протоколирования, то к ACL добавляется символ «a»;

– доступна опция `-M`, которая может быть использована для просмотра мандатных меток на файловых объектах.

4.4.5. pdpl-ps

Синтаксис:

```
pdpl-ps [-nzhv] <идентификатор процесса>
```

Команда `pdpl-ps` позволяет считать мандатный контекст безопасности с процесса, заданного параметром (идентификатор процесса).

Утилита `pdpl-ps` доступна в том числе через символическую ссылку `pdp-pslbl`.

Только администратор может устанавливать и считывать мандатный контекст безопасности произвольного процесса, обычный пользователь может только считывать контекст с собственного процесса, для этого параметр должен иметь нулевое значение.

Опции приведены в таблице 10.

Таблица 10

Опция	Описание
`-n, --numeric`	Вывести информацию о контексте в численном виде
`-z, --iszero`	Если метка безопасности нулевая, то завершиться с кодом `0`, иначе `1`, если не удалось получить метку безопасности процесса, то `74`
`-h, --help`	Вывести справку и выйти
`--version`	Вывести информацию о версии и выйти

4.4.6. pdpl-user

Синтаксис:

```
pdpl-user [-dzhv [-m минимальный:максимальный уровень конфиденциальности]
  [-i максимальный уровень целостности]
  [-c минимальная категория:максимальная категория]] <пользователь>
```

Команда `pdpl-user` отображает и устанавливает допустимые мандатные уровни и категории пользователей ОС.

Опции приведены в таблице 11.

Таблица 11

Опция	Описание
-d, --delete	Удалить строку пользователя из файла
-z, --zero	Обнулить значения уровней и категорий
-l, --levels	Установить допустимые уровни конфиденциальности
-i, --ilevel	Установить максимальный уровень целостности
-m, --maclabels	то же, что и -l (используется для совместимости)
-c, --category	Установить допустимые категории
-h, --help	Вывести справку и выйти
-v, --version	Вывести информацию о версии и выйти

Если в качестве параметра ключей -m или -c указано одно значение или одно значение с предшествующим двоеточием, это значение интерпретируется как максимальное значение, если одно значение с последующим двоеточием — как минимальное.

Команда pdpl-user при успешном выполнении всегда выводит значения установленных допустимых мандатных меток.

Чтобы просмотреть текущие допустимые метки, выполнить команду без ключей:

```
pdpl-user пользователь
```

Пример

```
pdpl-user -m Уровень_0:Уровень_3 -i 0 -c 0:Категория_2 user1
```

Данная команда для пользователя user1 установит:

– минимальный уровень — Уровень_0(0);

– максимальный уровень — Уровень_3(3);

– максимальный уровень целостности — Низкий(0);

– минимальную категорию — 0x0(0) (без категорий);

– максимальную категорию — 0x2(2).

Уровни Уровень_0, Уровень_3, категория Категория_2 должны быть определены в системе. Значения уровней и категорий могут быть заданы в числовой форме.

4.4.7. sumac

Синтаксис:

```
sumac [-h, --help] [-v, --version] [-l, --level=] [-c, --category=]
  [-i, --stdin=] [-o, --stdout=] [-e, --stderr=] [-x, --xauth] [command]
```

Команда sumac используется для запуска процесса с заданными мандатными уровнем и категорией в отдельной графической сессии с использованием виртуального графического сервера Xephyr. Пользователь может запускать процесс только в пределах разре-

шенных ему уровней и категорий.

Если указанный мандатный уровень выше текущего, т. е. происходит увеличение уровня, то переменные окружения наследуются от текущего процесса. Если происходит уменьшение мандатного уровня, то текущие переменные окружения сбрасываются, чтобы избежать утечки информации. Аналогично при порождении нового процесса закрываются все файловые дескрипторы, мандатная метка которых не совпадает с указанной в командной строке. В том числе закрываются `stdin`, `stdout`, `stderr`. Перенаправить стандартный ввод и вывод для нового процесса можно с помощью опций `-i`, `-o`, `-e` для `stdin`, `stdout` и `stderr`, соответственно.

ВНИМАНИЕ! Запуск процесса с понижением мандатного уровня или с сокращением набора мандатных категорий запрещен для предотвращения утечки информации на более низкие уровни секретности.

Примеры:

1. Запуск графического приложения `xterm` с мандатным уровнем 2 и категорией `0xffff`

```
$ sumac -l 2 -c 0xffff xterm
```

Опции приведены в таблице 12.

Таблица 12

Опция	Описание
`-l , --level=`	Запустить процесс с указанным мандатным уровнем
`-c , --category=`	Запустить процесс с указанной мандатной категорией
`-i , --stdin=`	Перенаправить `stdin` запущенного процесса в указанный файл
`-o , --stdout=`	Перенаправить `stdout` запущенного процесса в указанный файл
`-e , --stderr=`	Перенаправить `stderr` запущенного процесса в указанный файл
`-x, --xauth`	Попытаться создать запись в `.Xauthority`. В случае неудачи прервать выполнение процесса
`-h, --help`	Вывести справку и выйти
`-v, --version`	Вывести информацию о версии и выйти

4.4.8. userlev

Синтаксис:

```
userlev [-d, --delete] [-a, --add<значение>] [-r, --rename<имя>]
  [-m, --modify<значение>] [-h, --help] [--version] <уровень>
```

Команда `userlev` служит для просмотра и изменения в БД мандатных уровней. Для просмотра всех уровней команду следует запускать без параметров. Вносить изменения в БД уровней может только администратор.

Опции приведены в таблице 13.

Таблица 13

Опция	Описание
-d, --delete	Удалить уровень из БД
-a, --add<значение>	Добавить новый уровень в БД
-r, --rename<новое имя>	Переименовать существующий уровень
-m, --modify<новое значение>	Изменить значение уровня
-h, --help	Вывести справку и выйти
--version	Вывести информацию о версии и выйти

4.4.9. usercat

Синтаксис:

```
usercat [-d, --delete] [-a, --add<значение>] [-r, --rename<имя>]
  [-m, --modify<значение>] [-h, --help] [--version] [категория]
```

Команда `usercat` служит для просмотра и изменения БД категорий. Для просмотра всех категорий команду следует запускать без параметров. Вносить изменения в БД категорий может только администратор.

Опции приведены в таблице 14.

Таблица 14

Опция	Описание
-d, --delete	Удалить категорию из БД
-a, --add<значение>	Добавить новую категорию в БД
-r, --rename<новое имя>	Переименовать существующую категорию
-m, --modify<новое значение>	Изменить значение категории
-h, --help	Вывести справку и выйти
--version	Вывести информацию о версии и выйти

4.4.10. Устаревшие утилиты управления мандатными ПРД

Для совместимости с предыдущими версиями ОС сохранен ряд утилит командной строки для управления мандатными ПРД.

ВНИМАНИЕ! Настоятельно не рекомендуется применять устаревшие утилиты. Данные утилиты не позволяют работать с уровнями целостности и отображают мандатные атрибуты в формате предыдущих версий ОС.

4.4.10.1. chmac

Синтаксис:

```
chmac [опции] [уровень][:категория[:специальные атрибуты]] [имя файла]
```

Команда chmac изменяет мандатные атрибуты файлов ОС, которые включают мандатную метку и специальные мандатные атрибуты файла.

Опции приведены в таблице 15.

Таблица 15

Опция	Описание
-f, --silent, --quiet	Не выводить сообщений об ошибках
-v, --verbose	Выводить диагностические сообщения для каждого файла
-c, --changes	То же, что и --verbose, но сообщать только об изменениях
-R, --recursive	Применить рекурсивно
-h, --help	Вывести справку и выйти
--version	Вывести информацию о версии и выйти

Уровень и категория могут быть заданы именем или шестнадцатеричным значением.

Пример

```
chmac -Rv Секретно:Категория_A /tmp
```

Данная команда рекурсивно для всех файлов каталога /tmp изменит уровень на Секретно и категорию на Категория_A (уровень и категория должны быть определены в системе).

Специальные атрибуты могут быть заданы значением или строкой символов rwxrwx, в которой любой из символов может быть заменен на «-» для снятия соответствующего атрибута.

Пример

```
chmac 0:0:rwxrwx /tmp
```

Данная команда для каталога /tmp установит игнорирование мандатных уровней и категорий при выполнении операций чтения, записи и исполнения.

При задании специальных атрибутов вместо rwx могут быть использованы следующие сокращения:

– equ — игнорирование мандатных уровней и категорий при выполнении операций чтения, записи и исполнения;

– equ_w — игнорирование мандатных уровней и категорий при выполнении операции записи;

– low — игнорирование мандатных уровней и категорий при выполнении операций чтения и исполнения.

ВНИМАНИЕ! В настоящее время используется другой набор специальных атрибутов (см.`pdp-flbl`). Данная утилиты не позволяет работать с ними. Существует соответствие между старым атрибутом `equ` (`rwxrwx`) и новым `ehole`. Таким образом, раздельное управление специальными атрибутами вида `rwxrwx` не предусмотрено.

Пример

```
chmac 0:0:equ /tmp
```

4.4.10.2. macid

Синтаксис:

```
macid [опции]
```

Команда `macid` выводит мандатные атрибуты сессии пользователя ОС.

Опции приведены в таблице 16.

Таблица 16

Опция	Описание
`-l,--level`	Вывести только мандатный уровень
`-c,--categories`	Вывести только мандатную категорию
`-n,--name`	Для опций `-lc` выводить имена вместо числовых значений
`-h,--help`	Вывести справку и выйти
`--version`	Вывести информацию о версии и выйти

При отсутствии опций выводит строку текущих мандатных свойств.

Пример

```
macid
Уровень=2(Секретно) Категория=1(Категория_А) Привилегии=0
```

В этом примере текущая сессия пользователя имеет уровень секретности `Секретно` и категорию `Категория_А`.

4.4.10.3. lsm

Синтаксис:

```
lsm [опции] [имя файла]
```

Команда `lsm` выводит аналогично стандартной команде `ls` информацию о файлах (по умолчанию — о текущем каталоге).

Использование данной команды в целом не отличается от использования `ls`, за исключением следующих особенностей:

– если на файле установлены ACL, то к ним добавляется символ «+»;

– если на файле установлена ненулевая мандатная метка, то к ACL добавляется символ «m»;

– если на файле установлены списки протоколирования, то к ACL добавляется символ «a»;

– доступна опция -M, которая может быть использована для просмотра мандатных меток на файловых объектах.

4.4.10.4. psmac

Синтаксис:

```
psmac [-d, --delete] [-n, --numeric] [-h, --help] [--version]
  <идентификатор процесса> [мандатная метка...]
```

Команда psmac позволяет изменять или считать мандатный контекст безопасности выбранного процесса. Если в качестве аргумента не указана метка, то команда считывает мандатный контекст с процесса, заданного параметром (идентификатор процесса).

Если аргумент-метка присутствует, то команда устанавливает заданную мандатную метку на процесс. Мандатная метка задается в виде:

<Уровень>[:<Категории>]

где <Уровень> и <Категории> могут быть заданы как в численном, так и в символьном виде. Сложные <Категории> могут быть заданы в виде списка своих составляющих.

Только администратор может устанавливать и считывать мандатный контекст безопасности произвольного процесса, обычный пользователь может только считывать контекст с собственного процесса, для этого параметр должен иметь нулевое значение.

Опции приведены в таблице 17.

Таблица 17

Опция	Описание
-d, --delete	Обнулить мандатный контекст безопасности процесса
-n, --numeric	Вывести информацию о контексте в численном виде
-h, --help	Вывести справку и выйти
--version	Вывести информацию о версии и выйти

4.4.10.5. usermac

Синтаксис:

```
usermac [-dzhv [-m минимальный уровень:максимальный уровень]
  [-c минимальная категория:максимальная категория]] пользователь
```

Команда usermac изменяет допустимые мандатные уровни и категории пользователей ОС.

Опции приведены в таблице 18.

Таблица 18

Опция	Описание
-d, --delete	Удалить строку пользователя из файла
-z, --zero	Обнулить значения уровней и категорий
-l, --levels	Установить допустимые мандатные уровни конфиденциальности
-i, --ilevel	Установить максимальный уровень целостности
-m, --maclabels	то же, что и -l (используется для совместимости)
-c, --category	Установить допустимые категории
-h, --help	Вывести справку и выйти
-v, --version	Вывести информацию о версии и выйти

Если в качестве параметра ключей -m или -c указано одно значение или одно значение с предшествующим двоеточием, это значение интерпретируется как максимальное значение, если одно значение с последующим двоеточием — как минимальное.

Команда usermac при успешном выполнении всегда выводит значения установленных допустимых мандатных меток.

Чтобы просмотреть текущие допустимые метки, выполнить команду без ключей:

```
usermac пользователь
```

Пример

```
usermac -m несекретно:секретно -c категория_A:категория_B user1
```

Данная команда для пользователя user1 установит:

– минимальный уровень — несекретно;

– максимальный уровень — секретно;

– минимальную категорию — категория_A;

– максимальную категорию — категория_B.

Уровни несекретно, секретно и категории категория_A, категория_B должны быть определены в системе. Значения уровней и категорий могут быть заданы в числовой форме.

4.4.10.6. getfmac

Синтаксис:

```
getfmac [-R, --recursive] [-L, --logical] [-P, --physycal] [-n, --numeric]
  [-p, --absolute-names] [-c, --omit-header] [-s, --skip-empty]
  [-h, --help] [-v, --version] файлы и/или каталоги
```

Команда getfmac служит для получения мандатных меток файловых объектов. Информация о метках посылается на стандартный вывод и может являться входными данными для команды setfmac (см. 4.4.10.7).

Опции приведены в таблице 19.

Таблица 19

Опция	Описание
-R, --recursive	Для подкаталогов рекурсивно
-L, --logical	Следовать по символическим ссылкам
-P, --phisical	Не следовать по символическим ссылкам
-n, --numeric	Выводить информацию о компонентах метки в цифровой форме
-p, --absolute-names	Абсолютные имена
-c, --omit-header	Не показывать заголовок (имя файла)
-s, --skip-empty	Пропускать файлы с пустыми атрибутами
-h, --help	Вывести справку и выйти
-v, --version	Вывести информацию о версии и выйти

4.4.10.7. setfmac

Синтаксис:

```
setfmac [-s, --set] [-m, --modify] [-S, --set-file] [-B, --restore]
  [-R, --recursive] [-L, --logical] [-P, --physical] [-h, --help]
  [-v, --version] [мандатная метка] Файлы и/или каталоги
```

Команда setfmac устанавливает мандатные метки на файлы. Метки задаются или в командной строке (параметры -s, -m), или в файле (параметры -S, -B). При этом, файлы могут быть сформированы с помощью перенаправления вывода команды getfmac (4.4.10.6).

Мандатная метка задается в виде:

```
<Уровень>[:<Категории>[:<Тип>]]
```

где <Уровень> и <Категории> могут быть заданы как в численном, так и в символьном виде. Сложные <Категории> могут быть заданы в виде списка своих составляющих, например: Танки, Самолеты.

Только администратор может устанавливать мандатные метки на файлы.

ВНИМАНИЕ! С помощью данной команды невозможно установить мандатную метку на файловый объект-сокет.

Опции приведены в таблице 20.

Таблица 20

Опция	Описание
-s, --set	Установить мандатную метку из командной строки
-m, --modify	Изменить мандатную метку из командной строки

Окончание таблицы 20

Опция	Описание
`-S, --set-file`	Установить метки из файла
`-B, --restore`	Восстановить метки из файла
`-R, --recursive`	Для подкаталогов рекурсивно
`-L, --logical`	Следовать по символическим ссылкам
`-P, --physical`	Не следовать по символическим ссылкам
`-h, --help`	Вывести справку и выйти
`-v, --version`	Вывести информацию о версии и выйти

4.5. Средства управления привилегиями пользователей и процессов

Для управления привилегиями файлов и процессов используется графическая утилита `fly-admin-smc` («Управление политикой безопасности»). Более подробное описание утилиты см. в электронной справке.

Далее рассмотрены средства управления привилегиями пользователей и процессов в режиме командной строки.

4.5.1. usercaps

Синтаксис:

```
usercaps [-d, --delete] [-z, --zero] [-f, --full] [-l, --linux] [-m, --parsec]
  [-l, --Linux] [-M, --PARSEC] [-n, --numeric] [-h, --help] [-v, --version]
  пользователь
```

Команда `usercaps` позволяет просматривать и устанавливать привилегии пользователей, должна использоваться только администратором через механизм `sudo`.

```
usercaps [-[l строка модифицирования linux-привилегий]
  [-[m строка модифицирования PARSEC-привилегий][dzLMnfhv]] пользователь
```

Строка модифицирования привилегий состоит из слов, разделенных «,» или «:», каждое слово может быть строкой в верхнем или нижнем регистре или числом, перед которым стоит знак «+» или «-». Список возможных привилегий с сокращенными именами и числами можно увидеть, введя для Linux-привилегий:

```
usercaps -L
```

для PARSEC-привилегий:

```
usercaps -M
```

Верхний регистр для слов и знак «+» для чисел устанавливает флаг для соответствующей привилегии, нижний регистр и знак «-» снимает его.

`usercaps <пользователь>` выводит все привилегии пользователя; `usercaps -L <пользователь>` или `usercaps -M <пользователь>` выводят только Linux- или только PARSEC-привилегии.

Опции приведены в таблице 21.

Таблица 21

Опция	Описание
-d, --delete	Удалить строку пользователя из файла привилегий
-z, --zero	Сбросить все привилегии пользователя
-f, --full	Присвоить все возможные привилегии пользователю
-l, --linux	Изменить Linux-привилегии пользователя
-m, --parsec	Изменить PARSEC-привилегии пользователя
-L, --Linux	Вывести список возможный Linux-привилегий
-M, --PARSEC	Вывести список возможных PARSEC-привилегий
-n, --numeric	Вывести привилегии в шестнадцатеричном формате
-h, --help	Вывести справку и выйти
-v, --version	Вывести информацию о версии и выйти

4.5.2. execaps

Синтаксис:

```
execaps [-v, --version] [-h, --help] [-c,--capability (привилегии)] [--]
  команда
```

Команда execaps может быть использована администратором для запуска процесса с одновременной установкой выбранных PARSEC-привилегий.

```
execaps -c <вектор привилегий> -- <программа и ее аргументы>
```

Привилегии задаются в виде числа — битовой маски (как правило, в шестнадцатеричном виде). Соответствие отдельных бит полномочиям приведено в man parsec_capset(2).

Пример

```
execaps -c 0x100 -- /etc/init.d/dbus restart
```

Будет выполнен перезапуск сервиса dbus с установленной привилегией PARSEC_CAP_PRIV_SOCK.

Опции приведены в таблице 22.

Таблица 22

Опция	Описание
-c, --capability	Установить привилегии
-f, --force	Вызвать программу, даже если не удалось установить привилегии
-h, --help	Вывести справку и выйти
-v, --version	Вывести информацию о версии и выйти

4.5.3. pscaps

Синтаксис:

```
pscaps [--version] [-h, --help] [действующие полномочия
  [разрешенные полномочия [наследуемые полномочия]]]
```

Команда `pscaps` может быть использована для просмотра и изменения (в численном виде) PARSEC-полномочий процесса.

```
pscaps [effective_caps [permitted_caps [inheritable_caps]]]
```

Если в качестве аргумента указан только идентификатор процесса, то команда показывает набор полномочий заданного процесса, в противном случае пытается установить заданные в командной строке в виде шестнадцатеричных чисел полномочия.

Опции приведены в таблице 23.

Таблица 23

Опция	Описание
`-h, --help`	Вывести справку и выйти
`-v, --version`	Вывести информацию о версии и выйти

4.6. Мандатное разграничение доступа в СУБД PostgreSQL

В качестве защищенной СУБД в составе ОС используется СУБД PostgreSQL, доработанная в соответствии с требованием интеграции с ОС в части мандатного разграничения доступа к информации.

Примечание. В текущей версии ОС представлены две версии защищенной СУБД (на базе версий СУБД PostgreSQL 9.2 и 9.4), отличающиеся в части реализации мандатного разграничения доступа к информации. В версии СУБД PostgreSQL 9.2 используется реализация мандатного разграничения доступа, применяемая в предыдущих версиях ОС, тогда как версия СУБД PostgreSQL 9.4 содержит реализацию ДП-модели управления доступом и информационными потоками, соответствующую текущей версии ОС.

В основе мандатного механизма разграничения доступа лежит управление доступом к защищаемым ресурсам БД на основе иерархических и неиерархических меток доступа. Это позволяет реализовать многоуровневую защиту с обеспечением разграничения доступа пользователей к защищаемым ресурсам БД и управление потоками информации. В качестве иерархических и неиерархических меток доступа при использовании СУБД в ОС используются метки конфиденциальности или метки безопасности ОС.

СУБД PostgreSQL не имеет собственного механизма назначения, хранения и модификации меток пользователей и использует для этого механизмы ОС.

4.6.1. Версия СУБД PostgreSQL 9.2

В реляционной модели в качестве структуры, обладающей меткой, естественно выбрать кортеж, поскольку именно на этом уровне детализации осуществляются операции чтения/записи информации в СУБД. При этом, местом хранения метки может быть выбран только сам кортеж, только так метка будет неразрывно связана с данными, содержащимися в кортеже. Кроме того, метка может быть определена для таких объектов БД, к которым применимы виды доступа на чтение/запись данных, а именно таблицы и виды. В этом случае метки объектов располагаются в записях системной таблицы, непосредственно описывающих защищаемый объект.

Так как мандатный контроль доступа может быть определен только для видов доступа на чтение и на запись информации, все множество операций с данными в защищаемых объектах приводится к ним следующим образом:

– `INSERT` — доступ на запись;

– `UPDATE`, `DELETE` — последовательное выполнение доступа на чтение и запись информации;

– `SELECT` — доступ на чтение.

При обращении пользователя к БД определяются его допустимый диапазон меток и набор специальных мандатных атрибутов. Если пользователю не присвоена метка, то он получает по умолчанию нулевую метку, соответствующую минимальному уровню доступа. Максимальная метка определяется по заданной при регистрации пользователя в ОС. Поскольку сервер БД может иметь метку, при превышении метки пользователя метки сервера ему будут разрешены только операции чтения. Текущая метка пользователя определяется по установленному соединению и может быть установлена в пределах его допустимого диапазона мандатных атрибутов при наличии соответствующей привилегии.

Применение мандатных ПРД осуществляется на уровне доступа к объектам БД и на уровне доступа непосредственно к данным (на уровне записей).

Проверка мандатных прав доступа к таблицам и видам осуществляется одновременно с проверкой дискреционных прав доступа к ним, после разбора и построения плана запроса, непосредственно перед его выполнением, когда определены все необходимые для проверки данные и проверяемые объекты. Таким образом, доступ предоставляется только при одновременном санкционировании дискреционными ПРД.

Проверка мандатных прав доступа к записям таблиц осуществляется в процессе выполнения запроса при последовательном или индексном сканировании данных.

Все записи, помещаемые в таблицы, для которых установлена защита на уровне записей, наследуют текущую метку пользователя. Обновляемые записи сохраняют свою метку при изменении. Доступ к существующим записям и возможность их обновления и

удаления определяются установленными мандатными правилами.

Для администратора БД предусмотрены системные привилегии игнорирования мандатного разграничения доступа, только таким образом можно производить регламентные работы с БД (например, восстановление резервной копии), т. к. это требует установки меток данных, сохраненных ранее.

В PostgreSQL объектами защиты являются столбцы, и для них реализованы мандатные ПРД. Метки столбцов объектов так же располагаются в записи соответствующей системной таблицы, непосредственно описывающей защищаемый столбец. Мандатные ПРД столбцов и самого объекта не могут быть применены одновременно. Режим применения мандатных ПРД только к самому объекту или только к его столбцам может быть задан для каждого объекта в отдельности. Защита на уровне записей может использоваться в любом случае.

Для настройки работы сервера с мандатным разграничением доступа существует ряд конфигурационных параметров, указываемых в конфигурационном файле `postgresql.conf` конкретного кластера данных (таблица 24).

Таблица 24

Параметр	Описание
`ac_ignore_socket_maclabel`	Определяет, будет ли сервер СУБД использовать метку входящего соединения. Если этот параметр установлен в `FALSE`, то метка входящего соединения будет учитываться при определении максимальной доступной метки сессии и после подключения будет доступна только информация с меткой не выше метки входящего соединения. При установке этого параметра в `TRUE` метки сеанса будут определяться максимальной меткой пользователя, полученной из ОС.
`ac_ignore_server_maclabel`	Определяет, будет ли сервер СУБД дополнительно использовать свою метку (метку пользователя `postgres`) при определении прав пользователя на занесение, удаление и модификацию данных или нет. Если этот параметр установлен в `FALSE`, то метка сервера используется для блокирования занесения в БД информации с меткой, превышающей метку сервера. Если этот параметр установлен в `TRUE`, то метка сервера не учитывается.
`ac_enable_trusted_owner`	Определяет, могут ли владельцы объектов назначать права на доступ к ним другим пользователям. Если этот параметр установлен в значение `FALSE`, то право назначать права на доступ к любым объектам БД имеют только суперпользователи. Это предотвращает неконтролируемое распространение прав на доступ к информации. Если этот параметр установлен в `TRUE`, то, кроме суперпользователей, каждый владелец объекта может назначать права на доступ пользователей к «своему» объекту.

Окончание таблицы 24

Параметр	Описание
ac_enable_truncate	Блокирует (FALSE) или разблокирует (TRUE) возможность выполнения команды TRUNCATE.
ac_enable_sequence_mac	Если параметр конфигурации установлен в FALSE, то мандатный принцип контроля доступа на последовательности не применяется.
ac_enable_dblink_mac	Если параметр конфигурации установлен в TRUE, разрешается использование dblink в условиях мандатного разграничения доступа.
ac_enable_copy_to_file	Блокирует (FALSE) или разблокирует (TRUE) возможность выполнения команды COPY с выводом результатов в файл, доступный серверу СУБД.
ac_caps_ttl	Время жизни информации в секундах о привилегиях пользователя подсистемы безопасности PARSEC (определяет время жизни кэшированной информации о PARSEC-привилегиях пользователя; уменьшение значения приводит к увеличению числа обращений сервера СУБД к подсистеме безопасности PARSEC и как следствие — к снижению производительности сервера СУБД).
ac_debug_print	Если установлен в TRUE, добавляет в журнал сервера отладочную информацию о работе механизмов защиты.

Для более гибкой настройки сервера СУБД расширен синтаксис конфигурационного файла pg_hba.conf опцией ignore_socket_maclabel. Данный параметр может быть указан после метода аутентификации.

Параметр ignore_socket_maclabel=1 позволяет пользователям подключаться к базам данных с игнорированием метки сокета соединения. Если этот параметр не указывается в конфигурационном файле pg_hba.conf или установлен в 0, то игнорирование метки сокета для этого подключения не происходит.

В ОС каждый пользователь может иметь множество меток, которое задается минимальной и максимальной метками диапазона. Чтобы поддержать эту модель в СУБД PostgreSQL каждой сессии пользователя назначаются три метки: максимальная, минимальная и текущая. Их начальная инициализация осуществляется по следующему алгоритму:

1) после прохождения пользователем стандартной процедуры аутентификации сервер считывает из ОС значения максимальной и минимальной меток пользователя и принимает их как максимальную и минимальную метки сессии. При этом, если запись о метках для пользователя не найдена, то максимальная и минимальная метки принимаются равными нулю. Следовательно, пользователи, зарегистрированные только в сервере СУБД PostgreSQL и не имеющие учетной записи в ОС сервера, всегда имеют минимальный уровень доступа к информации;

2) если параметр конфигурации ac_ignore_socket_maclabel установлен в FALSE, считывается метка входящего соединения, и, если она попадает в диапазон меток, считанных из ОС, то максимальная метка сессии устанавливается равной метке входящего соединения. При этом, если минимальная метка такого пользователя в ОС сервера выше нулевой, то он вообще не будет допущен к работе с БД;

3) если параметр конфигурации ac_ignore_server_maclabel установлен в FALSE, то считывается метка серверного процесса и, если она не совместима с максимальной меткой сессии, то процесс аутентификации прерывается;

4) текущей меткой сессии становится максимальная метка сформированного таким образом диапазона.

Если на любом из этих этапов возникает ситуация с несовместимостью меток или выходом за пределы диапазона, то процесс аутентификации клиента прерывается и доступ к БД блокируется.

Если пользователь имеет мандатный атрибут ac_capable_setmac, то он может изменять свою текущую мандатную метку в диапазоне от минимальной до максимальной.

СУБД PostgreSQL предоставляет пользователям возможность создавать функции (и, следовательно, триггеры), указывая при этом, будут ли они выполняться с уровнем доступа пользователя, прямо или косвенно вызвавшего функцию (SECURITY INVOKER), или с уровнем доступа пользователя, создавшего эту функцию (SECURITY DEFINER). При этом в понятие «уровня доступа» входят как дискреционный уровень доступа, так и мандатный, который в данном случае определяется текущими мандатными атрибутами пользователя СУБД, вызвавшего или создавшего функцию, соответственно. При этом метки текущей сессии пользователя, вызвавшего функцию, не изменяются.

При этом следует учитывать, что:

1) при определении функции как SECURITY DEFINER она будет всегда вызываться с переустановкой мандатных атрибутов на атрибуты создавшего ее пользователя;

2) при определении функции как SECURITY INVOKER она всегда будет выполняться без изменения текущего значения мандатных атрибутов;

3) при вызове функции в качестве триггера выполняются следующие правила в дополнение к указанным:

– перед вызовом в качестве триггера встроенной в СУБД функции к текущим мандатным атрибутам всегда добавляются флаги ac_capable_ignmaclvl и ac_capable_ignmaccat, чтобы обеспечить полноценную проверку ссылочной целостности БД;

– перед вызовом в качестве триггера не встроенной функции в качестве

текущих мандатных атрибутов всегда устанавливаются мандатные атрибуты пользователя, запустившего данную сессию (соединение) (т. е. пользователя с именем `SESSION_USER`). Это необходимо, чтобы предотвратить получение функцией-триггером пользователя с низким уровнем доступа высоких привилегий в случае каскадного вызова триггеров.

После возврата управления из функции значения текущих мандатных атрибутов всегда восстанавливаются в исходные (до вызова функции) значения.

Функции, написанные на языках низкого уровня, после их подключения имеют полный доступ ко всем внутренним структурам сервера СУБД PostgreSQL и могут произвольно их модифицировать. Кроме этого, поскольку они выполняются в рамках процесса сервера, они имеют соответствующие права доступа к объектам ОС в среде функционирования сервера. Именно поэтому права пользователя `postgres`, под которым запускается сервер, необходимо свести к необходимому минимуму, минуя какой-либо контроль с его стороны (включая текущие мандатные атрибуты).

При наличии мандатных меток на сам объект, его столбец и непосредственно строку возможны следующие варианты использования мандатных ПРД (рассмотрим на примере таблиц):

1) метки отсутствуют — мандатные ПРД не применяются.

В этом случае метка объекта не установлена, метки столбцов не установлены, а сам объект создан без защиты строк. СУБД функционирует в штатном режиме защиты с использованием только дискреционных ПРД.

ВНИМАНИЕ! Данный режим не обеспечивает мандатного разграничения доступа и не рекомендуется к использованию! Наличие даже одного объекта, функционирующего в подобном режиме, нарушает защищенность системы в целом;

2) метками защищаются только записи.

Метка объекта не установлена, метки столбцов не установлены, а сам объект создан с защитой строк. Дискреционные ПРД применяются перед выполнением запроса.

Мандатные ПРД применяются только на уровне записей. Создание записей разрешено всем субъектам, при этом записи наследуют метку субъекта. Операции чтения и модификации осуществляются над множествами записей, доступных субъекту по мандатным ПРД. Проверка мандатных ПРД осуществляется после успешного применения дискреционных ПРД, нарушение безопасности не возникает;

3) метками защищается только объект.

Метка объекта установлена, метки столбцов не установлены, а сам объект создан без защиты строк.

Мандатные ПРД применяются только на уровне объекта, все данные, содержащиеся в объекте, рассматриваются, как имеющие метку объекта. Создание записей разрешено субъектам с метками, над которыми доминирует метка объекта, при этом записи наследуют метку субъекта. Операции чтения и модификации осуществляются по мандатным ПРД к объекту.

Мандатные ПРД применяются только в случае успешной проверки дискреционных ПРД, которые к столбцам объекта применяются только при отсутствии явного разрешения на доступ к самой таблице;

4) метками защищается объект и его записи.

Метка объекта установлена, метки столбцов не установлены, а сам объект создан с защитой строк.

Аналогично предыдущему варианту создание записей разрешено субъектам с метками, над которыми доминирует метка объекта, при этом записи наследуют метку субъекта. Мандатные ПРД применяются как на уровне объекта, так и на уровне записей.

Операции модификации возможны только над данными, имеющими метку, равную метке таблицы;

5) метками защищаются столбцы объекта.

Метка объекта не установлена, метки столбцов установлены, а сам объект создан без защиты строк.

При этом мандатные ПРД применяются на уровне столбцов. Субъект может читать из столбцов, над метками которых доминирует его метка, вставлять данные в столбцы, чьи метки доминируют над его, и модифицировать те, чьи метки равны его.

Операции удаления невозможны при наличии разных меток на столбцы, т. к. операция применяется ко всей строке. Это связано с тем, что операция удаления интерпретируется как последовательное предоставление доступа на чтение и на запись, что возможно только при равенстве меток субъекта и объекта. В случае, когда столбцы имеют разные метки, данное условие выполниться не может.

Операция удаления доступна только для администратора и пользователей, обладающих привилегиями игнорирования мандатного разграничения доступа;

6) метками защищаются столбцы и записи объекта.

В этом случае метка объекта не установлена, метки столбцов установлены, а сам объект создан с защитой строк.

При этом мандатные ПРД применяются как на уровне столбцов, так и на уровне записей. Субъект может вставлять данные в столбцы, чьи метки доминируют над

его, при этом записи наследуют метку субъекта. Операции чтения и модификации осуществляются над множествами записей, доступными субъекту по мандатным ПРД на записи, и только по столбцам, доступных по мандатным ПРД на столбцы.

Поскольку в процессе работы с данными в СУБД возможно изменение организации их хранения путем изменения схемы объектов БД (метаданных), к подобным операциям так же применяются ПРД.

Модификация метаданных возникает каждый раз при изменении структуры БД, что включает в себя создание, модификацию и удаление объектов БД.

Так как некоторые действия над объектами БД могут влиять на хранящихся в них данных (как правило, модификация или удаление объекта или его части), при использовании мандатного разграничения доступа к данным объекта необходимо разграничивать и доступ к изменению метаданных в части, относящейся к этому объекту.

Аналогично операциям с данными: действия с объектами БД должны быть приведены к видам доступа на чтение и на запись информации для возможности применения к ним мандатных ПРД. Все множество операций с метаданными может быть приведено следующим образом:

– `CREATE`, `ADD` — доступ на запись;

– `ALTER`, `DROP` — последовательное выполнение доступа на чтение и запись информации;

– использование или обращение к объекту в других SQL-командах — доступ на чтение.

Проверка мандатных прав доступа к метаданным осуществляется одновременно с проверкой дискреционных прав доступа к ним после разбора и построения плана запроса непосредственно перед его выполнением, когда определены все необходимые для проверки данные и проверяемые объекты. Таким образом, доступ предоставляется только при одновременном санкционировании дискреционными ПРД.

Некоторые операции над объектами, такие как `DROP` всего объекта или его столбца и `TRUNCATE` влекут за собой удаление данных. В случае защиты метками записей объекта существуют ограничения на выполнение этих операций.

Операции удаления невозможны при наличии разных меток на записях, т. к. операция применяется ко множеству строк. Это связано с тем, что операция удаления интерпретируется как последовательное предоставление доступа на чтение и на запись, что возможно только при равенстве меток субъекта и объекта. В случае, когда строки имеют разные метки, данное условие выполниться не может.

Операция удаления доступна только для администратора и пользователей, обладающих привилегиями игнорирования мандатного разграничения доступа.

4.6.1.1. Средства управления мандатными ПРД к объектам БД

Для управления мандатными ПРД к объектам БД СУБД PostgreSQL используется графическая утилита `pgadmin3` («Средство администрирования СУБД PostgreSQL»).

При создании таблицы, вида или последовательности их мандатная метка устанавливается равной текущей мандатной метке создавшего их пользователя. Если параметр конфигурации сервера `ac_enable_sequence_mac` установлен в `FALSE`, то мандатный принцип контроля доступа на последовательности не применяется. Если пользователь имеет мандатный атрибут `ac_capable_chmac`, то он может менять мандатную метку принадлежащих ему таблиц и видов в пределах своего диапазона мандатных меток с помощью следующих команд:

```
ALTER TABLE имя_отношения SET MAC TO новое_значение_мандатной_метки
ALTER TABLE имя_отношения SET MAC TO NULL
```

Установка мандатной метки таблицы или любого другого защищаемого объекта в значение `NULL` означает снятие мандатного контроля с доступа к этому объекту.

Значения мандатных меток таблиц, видов и последовательностей содержатся в поле `relmaclabel` системной таблицы `pg_catalog.pg_class`, откуда могут быть выбраны соответствующим запросом.

Установка мандатных меток столбцов выполняется с помощью следующих команд:

```
ALTER TABLE имя_отношения ALTER COLUMN имя_столбца
  SET MAC TO новое_значение_мандатной_метки
ALTER TABLE имя_отношения ALTER COLUMN имя_столбца SET MAC TO NULL
```

Значения мандатных меток столбцов, таблиц и видов содержатся в поле `attmaclabel` системной таблицы `pg_catalog.pg_attribute`.

Мандатные ПРД столбцов и самого объекта не могут быть применены одновременно. Режим применения мандатных ПРД только к самому объекту или только к его столбцам может быть задан для каждого объекта в отдельности. Защита на уровне записей может использоваться в любом случае.

В систему управления мандатными ПРД СУБД введено понятие «режим использования меток столбцов», который может быть указан для каждого объекта независимо. Если он установлен для таблицы или вида, то в проверках по мандатным ПРД участвуют только метки столбцов, а факт наличия/отсутствия метки объекта игнорируется. Если флаг не установлен, то игнорируется уже факт наличия меток у столбцов, в отличие от метки объекта. Флаг включается следующей командой:

```
ALTER TABLE имя_отношения ENABLE COLUMN MACS
```

Выключается флаг командой:

```
ALTER TABLE имя_отношения DISABLE COLUMN MACS
```

По умолчанию флаг выключен. Физически флаг хранится в системной таблице

pg_class в столбце relusecolmacs.

По умолчанию записи создаваемых таблиц не защищены мандатными метками. Для того чтобы создать таблицы с защищенными метками записями, следует использовать следующий вариант команды CREATE TABLE:

```
CREATE TABLE имя_таблицы (
  ... -- список_столбцов
) WITH ( MACS = true, ... );
```

При этом все вставляемые записи по умолчанию наследуют текущие мандатные метки создавших их пользователей. Пользователи, имеющие установленный мандатный атрибут ac_capable_chmac, могут явно задать значение мандатной метки вставляемой записи. Задаваемая метка должна быть в пределах диапазона меток пользователя, либо пользователь должен иметь атрибуты игнорирования мандатного контроля ac_capable_ignmaclvl и ac_capable_ignmaccat с помощью варианта команды INSERT:

```
INSERT INTO имя_отношения (maclabel, ... список_столбцов)
  VALUES (значение_мандатной_метки, ... значения_столбцов)
```

Для изменения мандатных меток существующих записей пользователи с атрибутом ac_capable_chmac могут использовать стандартную команду UPDATE:

```
UPDATE имя_отношения SET maclabel = новое_значение_мандатной_метки
```

Просмотреть значения мандатных меток доступных записей можно с помощью команды SELECT:

```
SELECT maclabel FROM имя_отношения
```

В командах INSERT и UPDATE значения мандатных меток не обязательно должны быть заданы в явном виде. Для задания метки допускается использование любого скалярного выражения, возвращающего результат, приводимый к типу мандатной метки.

Для того чтобы сохранить записи вместе с их метками в архиве и в дальнейшем загрузить их обратно, предусмотрен специальный флаг MACS команды COPY. Вывести доступные пользователю данные вместе с метками может любой пользователь, загрузить же обратно — только пользователь с установленным мандатным атрибутом ac_capabel_chmac. При этом метки загружаемых записей должны находиться в пределах диапазона меток пользователя, либо пользователь должен иметь атрибуты игнорирования мандатного контроля ac_capable_ignmaclvl и ac_capable_ignmaccat. Например, выгрузка и обратная загрузка данных из/в таблицы test может выглядеть так:

```
COPY имя_отношения TO stdout WITH MACS
COPY имя_отношения FROM stdin WITH MACS
```

Использовать команду COPY без указания меток может любой пользователь. Загруженные таким образом данные будут иметь метки, равные текущей метке сессии пользователя.

Параметр конфигурации сервера ac_enable_copy_to_file разрешает выполнять команду COPY с выводом результатов в файл, доступный серверу СУБД. Для этого он должен быть установлен в TRUE.

4.6.1.2. Система привилегий СУБД и управление ими

Система привилегий СУБД PostgreSQL предназначена для передачи отдельным пользователям прав выполнения определенных административных действий. Обычный пользователь системы не имеет дополнительных привилегий.

Привилегии являются подклассом атрибутов пользователя СУБД PostgreSQL.

Привилегии ОС, используемые в СУБД PostgreSQL, кроме атрибута ac_session_maclabel, не могут быть изменены с помощью средств СУБД ни пользователями, ни администраторами СУБД:

– ac_session_maclabel — текущая мандатная метка сессии пользователя СУБД. Эта метка определяет доступные пользователю объекты БД и является меткой по умолчанию для создаваемых пользователем объектов. При соединении пользователя с СУБД значение этого атрибута устанавливается равным метки соединения или ac_user_max_maclabel;

– ac_user_max_maclabel — максимально возможное значение для ac_session_maclabel;

– ac_user_min_maclabel — минимально возможное значение для ac_session_maclabel;

– ac_capable_ignmaclvl — позволяет пользователю игнорировать мандатный контроль по уровням;

– ac_capable_ignmaccat — позволяет пользователю игнорировать мандатный контроль по категориям;

– ac_capable_mac_readsearch — позволяет пользователю игнорировать мандатный контроль по уровням и категориям при чтении данных;

– ac_capable_setmac — позволяет пользователю изменять текущую метку своей сессии в пределах, заданных ее минимальным и максимальным значением;

– ac_capable_chmac — позволяет пользователю изменять метки объектов БД.

В случае, если пользователь СУБД не зарегистрирован в ОС на стороне сервера СУБД, все его мандатные атрибуты имеют нулевое значение. Администраторам СУБД дополнительно к их атрибутам из ОС всегда добавляются атрибуты ac_capable_ignmaclvl, ac_capable_ignmaccat и ac_capable_chmac.

Для управления привилегиями СУБД PostgreSQL может быть использована графическая утилита pgadmin3 («Администрирование СУБД PostgreSQL»). Более подробное описание утилиты см. в электронной справке.

Просмотреть текущие значения привилегий (атрибутов пользователя) можно с помощью команды:

```
SHOW имя_атрибута
```

Установить новое значение атрибута `ac_session_maclabel` можно с помощью команд:

```
SET ac_session_maclabel = новое_значение_метки
SELECT set_config ('ac_session_maclabel', новая_метка, false);
```

В первой форме в качестве нового значения метки можно использовать только явно заданные значения метки, во второй — значение метки может быть любым выражением, возвращающим скалярное значение, приводимое к типу мандатной метки.

4.6.2. Версия СУБД PostgreSQL 9.4

Версия СУБД PostgreSQL 9.4 содержит реализацию ДП-модели управления доступом и информационными потоками, соответствующую текущей версии ОС. Данная ДП-модель описывает все аспекты дискреционного, мандатного и ролевого управления доступом с учетом безопасности информационных потоков.

Согласно ДП-модели в части реализации мандатного разграничения доступа дополнительно к мандатной метке конфиденциальности вводится понятие объектов-контейнеров (объектов, которые могут содержать другие объекты). Для задания способа доступа к объектам внутри контейнеров используется мандатный признак CCR (Container Clearance Required). В случае когда он установлен, доступ к контейнеру и его содержимому определяется его мандатной меткой конфиденциальности, в противном случае доступ к содержимому разрешен без учета уровня конфиденциальности контейнера.

В качестве главного контейнера выбрано табличное пространство `pg_global`, которое создается одно на кластер базы данных. Таким образом, кластер является совокупностью ролей, баз данных и табличных пространств.

ВНИМАНИЕ! ДП-модель накладывает ограничение на мандатную метку конфиденциальности объекта: метка объекта не может превышать метку контейнера, в котором он содержится. Таким образом для назначения меток данным, сначала должны быть последовательно заданы максимальные метки соответствующих контейнеров: кластера, базы данных, табличного пространства, схемы и таблицы.

В реляционной модели в качестве структуры, обладающей меткой, естественно выбрать кортеж, поскольку именно на этом уровне детализации осуществляются операции чтения/записи информации в СУБД. При этом, местом хранения метки может быть выбран только сам кортеж, только так метка будет неразрывно связана с данными, содержащимися в кортеже.

В качестве множества сущностей (объектов и контейнеров) с заданной на нем

иерархической структурой рассматриваются носящие подобный характер объекты реляционных баз данных, применяемые в СУБД PostgreSQL. При этом, поскольку записи базы данных содержат в своем составе мандатный уровень конфиденциальности — они рассматриваются в модели в качестве объектов, а содержащие их таблицы, соответственно, — в качестве контейнеров.

Системный каталог (метаданные) рассматривается как самостоятельная БД, реализованная с помощью средств СУБД. При этом все операции с этой БД осуществляются либо с помощью специальных конструкций языка запросов SQL или привилегированным пользователем в специальном режиме. Таким образом мандатное разграничения доступа применяется ко всем объектам БД. Метки системных объектов располагаются в записях таблиц системного каталога, непосредственно описывающих защищаемый объект.

Примечание. В версии СУБД PostgreSQL 9.2 мандатное разграничения доступа применяется только к видам, таблицам и последовательностям (см. 4.6.1).

Так как мандатный контроль доступа может быть определен только для видов доступа на чтение и на запись информации, все множество операций с данными в защищаемых объектах приводится к ним следующим образом:

– `INSERT` — доступ на запись;

– `UPDATE`, `DELETE` — последовательное выполнение доступа на чтение и запись информации;

– `SELECT` — доступ на чтение.

При обращении пользователя к БД определяются его допустимый диапазон меток и набор специальных мандатных атрибутов. Если пользователю не присвоена метка, то он получает по умолчанию нулевую метку, соответствующую минимальному уровню доступа. Максимальная метка определяется по заданной при регистрации пользователя в ОС. Поскольку сервер БД так же может иметь метку, при превышении метки пользователя метки сервера ему будут разрешены только операции чтения. Текущая метка пользователя определяется по установленному соединению и может быть установлена в пределах его допустимого диапазона мандатных атрибутов при наличии соответствующей привилегии.

Применение мандатных ПРД осуществляется на уровне доступа к объектам БД и на уровне доступа непосредственно к данным (на уровне записей).

Проверка мандатных прав доступа к объектам осуществляется одновременно с проверкой дискреционных прав доступа к ним, после разбора и построения плана запроса, непосредственно перед его выполнением, когда определены все необходимые для проверки данные и проверяемые объекты. Таким образом, доступ предоставляется только при одновременном санкционировании дискреционными ПРД.

Проверка мандатных прав доступа к записям таблиц осуществляется в процессе

выполнения запроса при последовательном или индексном сканировании данных.

Все записи, помещаемые в таблицы, для которых установлена защита на уровне записей, наследуют текущую метку пользователя. Обновляемые записи сохраняют свою метку при изменении. Доступ к существующим записям и возможность их обновления и удаления определяются установленными мандатными правилами.

Для администратора БД предусмотрены системные привилегии игнорирования мандатного разграничения доступа, только таким образом можно производить регламентные работы с БД (например, восстановление резервной копии), т. к. это требует установки меток данных, сохраненных ранее.

Для настройки работы сервера с мандатным разграничением доступа существует ряд конфигурационных параметров, указываемых в конфигурационном файле `postgresql.conf` конкретного кластера данных (таблица 25).

Таблица 25

Параметр	Описание
`ac_ignore_socket_maclabel`	Определяет, будет ли сервер СУБД использовать метку входящего соединения. Если этот параметр установлен в `FALSE`, то метка входящего соединения будет учитываться при определении максимальной доступной метки сессии и после подключения будет доступна только информация с меткой не выше метки входящего соединения. При установке этого параметра в `TRUE` метки сеанса будут определяться максимальной меткой пользователя, полученной из ОС.
`ac_ignore_server_maclabel`	Определяет, будет ли сервер СУБД дополнительно использовать свою метку (метку пользователя `postgres`) при определении прав пользователя на занесение, удаление и модификацию данных или нет. Если этот параметр установлен в `FALSE`, то метка сервера используется для блокирования занесения в БД информации с меткой, превышающей метку сервера. Если этот параметр установлен в `TRUE`, то метка сервера не учитывается.
`ac_enable_trusted_owner`	Определяет, могут ли владельцы объектов назначать права на доступ к ним другим пользователям. Если этот параметр установлен в значение `FALSE`, то право назначать права на доступ к любым объектам БД имеют только суперпользователи. Это предотвращает неконтролируемое распространение прав на доступ к информации. Если этот параметр установлен в `TRUE`, то, кроме суперпользователей, каждый владелец объекта может назначать права на доступ пользователей к «своему» объекту.

Окончание таблицы 25

Параметр	Описание
ac_enable_grant_options	Определяет, могут ли роли передавать права на доступ с опцией WITH GRANT OPTIONS другим ролям. Если ac_allow_grant_options установлен в FALSE, то запрещается использовать команду GRANT с привилегией WITH GRANT OPTION. Если у роли есть привилегия GRANT OPTIONS и ac_allow_grant_options = false, то передача прав доступа другим ролям также запрещается. Изъятие (REVOKE) привилегии GRANT OPTIONS разрешается всегда.
ac_enable_admin_options	Определяет, могут ли роли передавать право членства роли другим ролям. Если ac_allow_admin_options установлен в FALSE, то запрещается использовать GRANT с привилегией WITH ADMIN OPTION. Если у роли есть привилегия ADMIN OPTIONS и ac_allow_admin_options = false, то передача прав членства другим ролям также запрещается. Изъятие (REVOKE) привилегии ADMIN OPTION разрешается всегда.
ac_enable_truncate	Блокирует (FALSE) или разблокирует (TRUE) возможность выполнения команды TRUNCATE.
ac_enable_sequence_ccr	При установке этого параметра в TRUE разрешается ипользование признака CCR для последовательностей аналогично таблицам и видам, в противном случае признак CCR для последовательностей считается всегда установленным.
ac_enable_dblink_mac	Если параметр конфигурации установлен в TRUE, разрешается использование dblink и внешних таблиц FOREIGN TABLE в условиях мандатного разграничения доступа.
ac_auto_adjust_macs	Если параметр конфигурации установлен в TRUE, разрешается автоматическая установка меток контейнеров. Применяется при восстановления резервных копий, созданных в предыдущих версиях СУБД.
ac_enable_copy_to_file	Блокирует (FALSE) или разблокирует (TRUE) возможность выполнения команды COPY с выводом результатов в файл, доступный серверу СУБД.
ac_caps_ttl	Время жизни информации в секундах о привилегиях пользователя подсистемы безопасности PARSEC (определяет время жизни кэшированной информации о PARSEC-привилегиях пользователя; уменьшение значения приводит к увеличению числа обращений сервера СУБД к подсистеме безопасности PARSEC и как следствие — к снижению производительности сервера СУБД).
ac_debug_print	Если установлен в TRUE, добавляет в журнал сервера отладочную информацию о работе механизмов защиты.

Для более гибкой настройки сервера СУБД расширен синтаксис конфигурационного файла pg_hba.conf опцией ignore_socket_maclabel. Данный параметр может быть

указан после метода аутентификации.

Параметр ignore_socket_maclabel=1 позволяет пользователям подключаться к базам данных с игнорированием метки сокета соединения. Если этот параметр не указывается в конфигурационном файле pg_hba.conf или установлен в 0, то игнорирование метки сокета для этого подключения не происходит.

В ОС каждый пользователь может иметь множество меток, которое задается минимальной и максимальной метками диапазона. Чтобы поддержать эту модель в СУБД PostgreSQL каждой сессии пользователя назначаются три метки: максимальная, минимальная и текущая. Их начальная инициализация осуществляется по следующему алгоритму:

1) после прохождения пользователем стандартной процедуры аутентификации сервер считывает из ОС значения максимальной и минимальной меток пользователя и принимает их как максимальную и минимальную метки сессии. При этом, если запись о метках для пользователя не найдена, то максимальная и минимальная метки принимаются равными нулю. Следовательно, пользователи, зарегистрированные только в сервере СУБД PostgreSQL и не имеющие учетной записи в ОС сервера, всегда имеют минимальный уровень доступа к информации;

2) если параметр конфигурации ac_ignore_socket_maclabel установлен в FALSE, считывается метка входящего соединения, и, если она попадает в диапазон меток, считанных из ОС, то максимальная метка сессии устанавливается равной метке входящего соединения. При этом, если минимальная метка такого пользователя в ОС сервера выше нулевой, то он вообще не будет допущен к работе с БД;

3) если параметр конфигурации ac_ignore_server_maclabel установлен в FALSE, то считывается метка серверного процесса и, если она не совместима с максимальной меткой сессии, то процесс аутентификации прерывается;

4) текущей меткой сессии становится максимальная метка сформированного таким образом диапазона.

Если на любом из этих этапов возникает ситуация с несовместимостью меток или выходом за пределы диапазона, то процесс аутентификации клиента прерывается и доступ к БД блокируется.

Если пользователь имеет мандатный атрибут ac_capable_setmac, то он может изменять свою текущую мандатную метку в диапазоне от минимальной до максимальной.

СУБД PostgreSQL предоставляет пользователям возможность создавать функции (и, следовательно, триггеры), указывая при этом, будут ли они выполняться с уровнем доступа пользователя, прямо или косвенно вызвавшего функцию (SECURITY INVOKER), или с уровнем доступа пользователя, создавшего эту функцию (SECURITY DEFINER). При этом

в понятие «уровня доступа» входят как дискреционный уровень доступа, так и мандатный, который в данном случае определяется текущими мандатными атрибутами пользователя СУБД, вызвавшего или создавшего функцию, соответственно. При этом метки текущей сессии пользователя, вызвавшего функцию, не изменяются.

При этом следует учитывать, что:

1) при определении функции как `SECURITY DEFINER` она будет всегда вызываться с переустановкой мандатных атрибутов на атрибуты создавшего ее пользователя;

2) при определении функции как `SECURITY INVOKER` она всегда будет выполняться без изменения текущего значения мандатных атрибутов;

3) при вызове функции в качестве триггера выполняются следующие правила в дополнение к указанным:

– перед вызовом в качестве триггера встроенной в СУБД функции к текущим мандатным атрибутам всегда добавляются флаги `ac_capable_ignmaclvl` и `ac_capable_ignmaccat`, чтобы обеспечить полноценную проверку ссылочной целостности БД;

– перед вызовом в качестве триггера не встроенной функции в качестве текущих мандатных атрибутов всегда устанавливаются мандатные атрибуты пользователя, запустившего данную сессию (соединение) (т. е. пользователя с именем `SESSION_USER`). Это необходимо, чтобы предотвратить получение функцией-триггером пользователя с низким уровнем доступа высоких привилегий в случае каскадного вызова триггеров.

После возврата управления из функции значения текущих мандатных атрибутов всегда восстанавливаются в исходные (до вызова функции) значения.

Функции, написанные на языках низкого уровня, после их подключения имеют полный доступ ко всем внутренним структурам сервера СУБД PostgreSQL и могут произвольно их модифицировать. Кроме этого, поскольку они выполняются в рамках процесса сервера, они имеют соответствующие права доступа к объектам ОС в среде функционирования сервера. Именно поэтому права пользователя `postgres`, под которым запускается сервер, необходимо свести к необходимому минимуму, минуя какой-либо контроль с его стороны (включая текущие мандатные атрибуты).

Поскольку в процессе работы с данными в СУБД возможно изменение организации их хранения путем изменения схемы объектов БД (метаданных), к подобным операциям так же применяются ПРД.

Модификация метаданных возникает каждый раз при изменении структуры БД, что включает в себя создание, модификацию и удаление объектов БД.

Так как некоторые действия над объектами БД могут влиять на хранящихся в них

данных (как правило, модификация или удаление объекта или его части), при использовании мандатного разграничения доступа к данным объекта необходимо разграничивать и доступ к изменению метаданных в части, относящейся к этому объекту.

Аналогично операциям с данными: действия с объектами БД должны быть приведены к видам доступа на чтение и на запись информации для возможности применения к ним мандатных ПРД. Все множество операций с метаданными может быть приведено следующим образом:

– `CREATE`, `ADD` — доступ на запись;

– `ALTER`, `DROP` — последовательное выполнение доступа на чтение и запись информации;

– использование или обращение к объекту в других SQL-командах — доступ на чтение.

Проверка мандатных прав доступа к метаданным осуществляется одновременно с проверкой дискреционных прав доступа к ним после разбора и построения плана запроса непосредственно перед его выполнением, когда определены все необходимые для проверки данные и проверяемые объекты. Таким образом, доступ предоставляется только при одновременном санкционировании дискреционными ПРД.

Некоторые операции над объектами, такие как `DROP` всего объекта или его столбца и `TRUNCATE` влекут за собой удаление данных. В случае защиты метками записей объекта существуют ограничения на выполнение этих операций.

Операции удаления невозможны при наличии разных меток на записях, т. к. операция применяется ко множеству строк. Это связано с тем, что операция удаления интерпретируется как последовательное предоставление доступа на чтение и на запись, что возможно только при равенстве меток субъекта и объекта. В случае, когда строки имеют разные метки, данное условие выполниться не может.

Операция удаления доступна только для администратора и пользователей, обладающих привилегиями игнорирования мандатного разграничения доступа.

4.6.2.1. Средства управления мандатными ПРД к объектам БД

Для управления мандатными ПРД к объектам БД СУБД PostgreSQL используется графическая утилита `pgadmin3` («Средство администрирования СУБД PostgreSQL»).

При создании мандатная метка объекта БД устанавливается равной текущей мандатной метке создавшего его пользователя, мандатный признак CCR при этом выставляется в значение `ON`.

Если пользователь имеет мандатный атрибут `ac_capable_chmac`, то он может менять мандатную метку принадлежащих ему объектов в пределах своего диапазона мандатных меток с помощью следующей команды:

```
MAC LABEL ON тип_объекта имя_объекта IS новое_значение_мандатной_метки;
```

В качестве типа объекта указывается тип объекта БД, например DATABASE, TABLE, FUNCTION.

Аналогичным образом для объектов-контейнеров может быть изменен мандатный признак CCR:

```
MAC CCR ON тип_объекта имя_объекта IS { ON | OFF };
```

Дополнительно введен новый тип объекта — кластер CLUSTER. Для установки метки на кластер используется следующий запрос:

```
MAC LABEL ON CLUSTER IS новое_значение_мандатной_метки;
```

что является синонимом к команде

```
MAC LABEL ON TABLESPACE pg_global IS новое_значение_мандатной_метки;
```

Для изменения мандатного признака CCR кластера используется следующая команда:

```
MAC CCR ON CLUSTER IS { ON | OFF };
```

что является синонимом к команде

```
MAC CCR ON TABLESPACE pg_global IS { ON | OFF };
```

Значения мандатных меток объектов содержатся в полях maclabel таблиц системного каталога, откуда могут быть выбраны соответствующим запросом.

Значения мандатного признака CCR объектов-контейнеров содержатся в следующих полях таблиц системного каталога:

– для баз данных - datmacccr системной таблицы pg_database;

– для схем - nspmacccr системной таблицы pg_namespace;

– для табличных пространств - spcmacccr системной таблицы pg_tablespace;

– для отношений - relmacccr системной таблицы pg_class.

Для просмотра мандатного признака CCR кластера может быть использована следующая команда:

```
SELECT cluster_macccr;
```

По умолчанию записи создаваемых таблиц не защищены мандатными метками. Для того чтобы создать таблицы с защищенными метками записями, следует использовать следующий вариант команды CREATE TABLE:

```
CREATE TABLE имя_таблицы (
  ... -- список_столбцов
) WITH ( MACS = true, ... );
```

При этом все вставляемые записи по умолчанию наследуют текущие мандатные метки создавших их пользователей. Пользователи, имеющие установленный мандатный атрибут ac_capable_chmac, могут явно задать значение мандатной метки вставляемой записи. Задаваемая метка должна быть в пределах диапазона меток пользователя, либо пользователь должен иметь атрибуты игнорирования мандатного контроля

ac_capable_ignmaclvl и ac_capable_ignmaccat с помощью варианта команды INSERT:

```
INSERT INTO имя_отношения (maclabel, ... список_столбцов)
  VALUES (значение_мандатной_метки, ... значения_столбцов)
```

Для защиты записей уже созданных таблиц без мандатных меток следует использовать следующий вариант команды ALTER TABLE:

```
ALTER TABLE имя_таблицы SET WITH MACS;
```

После исполнения этой команды все записи таблицы автоматически получат текущую метку таблицы.

Для того, чтобы убрать защиту записей мандатными метками, следует использовать следующий вариант команды ALTER TABLE:

```
ALTER TABLE имя_таблицы SET WITHOUT MACS;
```

Для изменения мандатных меток существующих записей пользователи с атрибутом ac_capable_chmac могут использовать команду CHMAC:

```
CHMAC имя_отношения SET maclabel = новое_значение_мандатной_метки WHERE ...
```

Примечание. В версии СУБД PostgreSQL 9.2 для изменения мандатных меток существующих записей используется команда UPDATE (см. 4.6.1).

Совокупная метка записей таблицы (максимальная по мандатному уровню и наиболее полная по мандатным категориям) может быть получена с помощью агрегирующей функции supmaclabel:

```
SELECT supmaclabel(maclabel) FROM имя_отношения;
```

Просмотреть значения мандатных меток доступных записей можно с помощью команды SELECT:

```
SELECT maclabel FROM имя_отношения
```

В командах INSERT и CHMAC значения мандатных меток не обязательно должны быть заданы в явном виде. Для задания метки допускается использование любого скалярного выражения, возвращающего результат, приводимый к типу мандатной метки.

Для того чтобы сохранить записи вместе с их метками в архиве и в дальнейшем загрузить их обратно, предусмотрен специальный флаг MACS команды COPY. Вывести доступные пользователю данные вместе с метками может любой пользователь, загрузить же обратно — только пользователь с установленным мандатным атрибутом ac_capabel_chmac. При этом метки загружаемых записей должны находиться в пределах диапазона меток пользователя, либо пользователь должен иметь атрибуты игнорирования мандатного контроля ac_capable_ignmaclvl и ac_capable_ignmaccat. Например, выгрузка и обратная загрузка данных из/в таблицы test может выглядеть так:

```
COPY имя_отношения TO stdout WITH MACS
COPY имя_отношения FROM stdin WITH MACS
```

Использовать команду COPY без указания меток может любой пользователь. Загру-

женные таким образом данные будут иметь метки, равные текущей метке сессии пользователя.

Параметр конфигурации сервера ac_enable_copy_to_file разрешает выполнять команду COPY с выводом результатов в файл, доступный серверу СУБД. Для этого он должен быть установлен в TRUE.

4.6.2.2. Целостность мандатных атрибутов кластера баз данных

В СУБД PostgreSQL версии 9.4 ДП-модель накладывает ограничение на мандатную метку конфиденциальности объекта: метка объекта не может превышать метку контейнера, в котором он содержится(4.6.2).

Для вывода информации о соблюдении ДП-модели между объектами-контейнерами и находящимися в них объектами реализована SQL-функция check_mac_integrity, которая выводит информацию в следующем виде:

– objid — Идентификатор объекта;

– classid — Идентификатор класса объекта;

– cobjid — Идентификатор контейнера, содержащего объект;

– cclassid — Идентификатор класса контейнера, содержащего объект;

– status — Результат проверки. Может принимать следующие значения: OK(модель соблюдается для объекта и контейнера) и FAIL(модель не соблюдается для объекта и контейнера).

Примечание. В СУБД PostgreSQL версии 9.4 информация о соблюдении модели выводится только для отношений(таблиц, представлений, последовательностей), схем, баз данных и табличных пространств.

Для исправления некорректно установленной мандатной метки отношения, например, при восстановлении резервной копии кластера ранних версий, используется SQL функция fix_mac_integrity. Данная функция может быть исполнена только пользователем с правами администратора, а также при установленном параметре ac_auto_adjust_macs = true в конфигурационном файле postgresql.conf.

4.6.2.3. Ссылочная целостность мандатных атрибутов

Средства обеспечения целостности в реляционных БД представляют собой механизмы автоматической поддержки системы правил, определяющих допустимость и корректность обрабатываемых данных, и могут быть разбиты на несколько видов.

– Ограничение целостности полей данных — ограничения, накладываемые на используемые в отношении домены (задание разрешенных диапазонов значений, запрет наличия неопределенных значений NULL, задание значений по умолчанию) или ограничения непосредственно таблицы (уникальность каждой записи, ограничение уникальности по выбранным столбцам, вычисляемые значения столбцов и

условия допустимых сочетаний значений в столбцах).

– Декларативная ссылочная целостность — описание зависимостей между разными отношениями в БД (наличия в одном отношении вторичного ключа, ссылающегося на первичный ключ в другом). Подобные зависимости могут быть выявлены при анализе предметной области между ее сущностями или при проектировании БД в процессе нормализации (в этом случае одна сущность предметной области может состоять из набора отношений). Ссылочная целостность реализуется путем описания ограничений на значения вторичных ключей в одном отношении и правил их обработки в случае изменения первичных ключей в другом.

– Динамическая ссылочная целостность — триггера, назначаемые для выполнения при реализации заданного вида доступа к конкретной таблице. Триггер представляет собой исполняемый код, который может динамически проверить заданные условия корректности выполняемой операции, и при необходимости внести изменения в другие таблицы.

В приведенном определении отсутствуют мандатные атрибуты, так как в общем случае между классификационными метками записей разных таблиц может не быть зависимости. С другой стороны, существует частный случай ссылочной целостности, образованный между таблицами, являющимися частями одной сущности предметной области, что может возникнуть в процессе нормализации.

Для обеспечения целостности мандатных атрибутов список событий для ограничений целостности наряду с существующими событиями `ON SELECT`, `ON INSERT`, `ON UPDATE`, `ON DELETE` расширен событием изменения мандатных атрибутов `ON CHMAC`. Таким образом, возможно создание ограничение ссылочной целостности следующим образом:

```
ALTER TABLE имя_таблицы1 ADD CONSTRAINT имя_ограничения
  FOREIGN KEY (список_полей1) REFERENCES имя_таблицы2 (список_полей2)
    ON CHMAC {NO ACTION | RESTRICT | CASCADE }
```

Механизм действия такого ограничения целостности аналогичен механизму действия подобного ограничения для события `ON UPDATE` для данных: при изменении мандатных атрибутов записи в одной таблице можно указать каскадное изменение мандатных атрибутов связанных записей второй таблицы, либо запрет возможность такого изменения.

Аналогично существует возможность создания триггеров для указанного события изменения мандатных атрибутов (`CHMAC`), например:

```
CREATE TRIGGER имя_триггера
  BEFORE CHMAC ON имя_таблицы
  FOR EACH ROW
  EXECURE PROCEDURE имя_процедуры()
```

Таким же образом могут быть заданы и правила для видов:

```
CREATE RULE имя_правила
  AS ON CHMAC TO имя_вида
  DO ALSO ...
CREATE RULE имя_правила
  AS ON CHMAC TO имя_вида
  DO INSTEAD ...
```

В этом случае при изменении мандатных атрибутов записи будут вызываться соответствующие функции и правила, что дает возможность запрограммировать произвольную логику обеспечения целостности мандатных атрибутов внутри БД.

4.6.2.4. Особенности создания правил и триггеров

В СУБД PostgreSQL версии 9.4 правила (`RULE`) и триггеры (`TRIGGER`) наследуют метку таблицы, для которой они созданы. Эти объекты могут быть созданы пользователем-владельцем таблицы (метка которого будет совпадать с меткой таблицы), либо пользователями с привилегиями игнорирования мандатного доступа. В противном случае генерируется ошибка доступа.

Если триггер использует триггерную функцию, метка которой отлична от `{0,0}`, то при исполнении триггера для таблицы со сброшенным мандатным признаком CCR возможны нарушения в работе триггеров. Это обусловлено тем, что запись триггерной функции может быть не доступна для пользователя в связи с МПРД. В таких случаях будет выведено сообщение:

```
cache lookup failed: внутренняя ошибка или отсутствуют необходимые мандатные
атрибуты
```

Во избежании этого настоятельно рекомендуется устанавливать мандатную метку триггерной функции в значение `{0,0}`.

4.6.2.5. Особенности использования представлений и материализованных представлений

Представления (`VIEW`) и материализованные представления (`MATERIALIZED VIEW`) не могут иметь метки на строках, поскольку выполняют агрегацию данных из других источников данных (таблиц и представлений).

Примечание. Представления создаются с установленным флагом CCR, поэтому при попытке доступа к нему от имени пользователя, мандатная метка которого не превосходит метки представления, это представление не будет «видно» пользователю в силу МПРД. Для того, чтобы пользователь имел доступ к такому представлению, необходимо сбросить мандатный признак CCR представления.

4.6.2.6. Функции сравнения для типа `maclabel`

В СУБД PostgreSQL версии 9.4 изменено поведение следующих функций для типа `maclabel`: `eq`, `ne`, `lt`, `le`, `gt`, `ge`. В указанных функциях вместо индексного сравнения используется сравнение мандатных меток соответствующими операторами =, <>, <, <=, >, >=.

Для проверки на несравнимость мандатных меток используется функция `nc`.

Операторы индексного сравнения переименованы в `maclabel_idx_eq`, `maclabel_idx_ne`, `maclabel_idx_lt`, `maclabel_idx_le`, `maclabel_idx_gt`, `maclabel_idx_ge`.

4.6.2.7. Система привилегий СУБД и управление ими

Система привилегий СУБД PostgreSQL предназначена для передачи отдельным пользователям прав выполнения определенных административных действий. Обычный пользователь системы не имеет дополнительных привилегий.

Привилегии являются подклассом атрибутов пользователя СУБД PostgreSQL.

Привилегии ОС, используемые в СУБД PostgreSQL, кроме атрибута `ac_session_maclabel`, не могут быть изменены с помощью средств СУБД ни пользователями, ни администраторами СУБД:

– `ac_session_maclabel` — текущая мандатная метка сессии пользователя СУБД. Эта метка определяет доступные пользователю объекты БД и является меткой по умолчанию для создаваемых пользователем объектов. При соединении пользователя с СУБД значение этого атрибута устанавливается равным метки соединения или `ac_user_max_maclabel`;

– `ac_user_max_maclabel` — максимально возможное значение для `ac_session_maclabel`;

– `ac_user_min_maclabel` — минимально возможное значение для `ac_session_maclabel`;

– `ac_capable_ignmaclvl` — позволяет пользователю игнорировать мандатный контроль по уровням;

– `ac_capable_ignmaccat` — позволяет пользователю игнорировать мандатный контроль по категориям;

– `ac_capable_mac_readsearch` — позволяет пользователю игнорировать мандатный контроль по уровням и категориям при чтении данных;

– `ac_capable_setmac` — позволяет пользователю изменять текущую метку своей сессии в пределах, заданных ее минимальным и максимальным значением;

– `ac_capable_chmac` — позволяет пользователю изменять метки объектов БД.

В случае, если пользователь СУБД не зарегистрирован в ОС на стороне сервера

СУБД, все его мандатные атрибуты имеют нулевое значение.Администраторам СУБД дополнительно к их атрибутам из ОС всегда добавляются атрибуты `ac_capable_ignmaclvl`, `ac_capable_ignmaccat` и `ac_capable_chmac`.

Для управления привилегиями СУБД PostgreSQL может быть использована графическая утилита `pgadmin3` («Администрирование СУБД PostgreSQL»).

Просмотреть текущие значения привилегий (атрибутов пользователя) можно с помощью команды:

```
SHOW attr_name
```

Установить новое значение атрибута `ac_session_maclabel` можно с помощью команд:

```
SET ac_session_maclabel = новое_значение_метки
SELECT set_config ('ac_session_maclabel', новая_метка, false);
```

В первой форме в качестве нового значения метки можно использовать только явно заданные значения метки, во второй— значение метки может быть любым выражением, возвращающим скалярное значение, приводимое к типу мандатной метки.

4.7. Мандатное разграничение доступа в комплексах программ гипертекстовой обработки данных и электронной почты

Обеспечение мандатного разграничения доступа в комплексах программ гипертекстовой обработки данных и электронной почты реализовано на основе программного интерфейса библиотек подсистемы безопасности PARSEC.

На серверах комплексов программ гипертекстовой обработки данных и электронной почты при обработке запросов на соединение выполняется получение мандатного контекста соединения, унаследованного от субъекта (процесса). Сокет сервера, ожидающий входящих запросов на соединение, работает в контексте процесса, имеющего привилегию для приема соединений с любыми уровнями секретности.

После установки соединения и успешного прохождения процедуры идентификации и аутентификации пользователя процесс сервера, обрабатывающий запросы пользователя, переключается в контекст безопасности пользователя, сбрасывает привилегии, обрабатывает запросы пользователя и завершается.

В комплексе программ гипертекстовой обработки данных пользователь получает доступ к ресурсам, являющихся объектами ФС. Комплекс программ электронной почты использует технологию `maildir`, обеспечивающую хранение почтовых сообщений в виде отдельных объектов ФС. Создаваемые файлы почтовых сообщений маркируются мандатными метками, унаследованными от процесса-создателя. Таким образом, в обоих комплексах программ ресурсы, к которым осуществляется доступ от имени серверных процессов, обрабатывающих запросы пользователей, являются объектами ФС. Следовательно, до-

ступ к защищаемым ресурсам при приеме и обработке запросов пользователей в процессе функционирования серверов комплексов программ гипертекстовой обработки данных и электронной почты подчиняется мандатным ПРД.

5. ОЧИСТКА ПАМЯТИ

Ядро ОС гарантирует, что обычный непривилегированный процесс не получит данные чужого процесса, если это явно не разрешено ПРД. Это означает, что средства IPC контролируются с помощью ПРД, и процесс не может получить неочищенную память (как оперативную, так и дисковую).

В ОС реализован механизм, который очищает неиспользуемые блоки ФС непосредственно при их освобождении. Работа данного механизма снижает скорость выполнения операций удаления и усечения размера файла. Данные любых удаляемых/урезаемых файлов в пределах заданной ФС предварительно очищаются предопределенной или псевдослучайной маскирующей последовательностью. Механизм является настраиваемым и позволяет обеспечить работу ФС ОС (Ext2/Ext3/Ext4) в одном из следующих режимов:

1) Очистка осуществляется посредством перезаписи каждого байта в освобождаемой области посредством четырех сигнатур следующего вида: `11111111`, `01010101`, `10101010`, `00000000`. Использование режима включается параметром `secdel` в конфигурационном файле `/etc/fstab` для раздела ФС, на котором требуется очищать блоки памяти при их освобождении (например, `/dev/sda1`). В список параметров монтирования добавляется параметр `secdel`.

Пример

```
/dev/sda1 /home ext4 acl,defaults,secdel 0 2
```

2) Очистка осуществляется посредством перезаписи каждого байта в освобождаемой области посредством четырех сигнатур следующего вида: `11111111`, `01010101`, `10101010`, `00000000`. Количество перезаписей определяется администратором. Использование режима включается установкой значения параметра `secdel` в конфигурационном файле `/etc/fstab` для раздела ФС, на котором требуется очищать блоки памяти при их освобождении (например, `/dev/sda1`). При установке числа перезаписей больше четырех сигнатуры используются повторно. Например, при установке числа перезаписей, равному `6`, последовательность сигнатур, используемых для перезаписи, имеет следующий вид: `11111111`, `01010101`, `10101010`, `00000000`, `11111111`, `01010101`. В список параметров монтирования добавляется параметр `secdel=6`.

Пример

```
/dev/sda1 /home ext4 acl,defaults,secdel=6 0 2
```

3) Очистка осуществляется посредством перезаписи каждого байта в освобождаемой области посредством четырех псевдослучайных сигнатур. Использование ре-

жима включается параметром secdelrnd в конфигурационном файле /etc/fstab для раздела ФС, на котором требуется очищать блоки памяти при их освобождении (например, /dev/sda1). В список параметров монтирования добавляется параметр secdelrnd.

Пример

```
/dev/sda1 /home ext4 acl,defaults,secdelrnd 0 2
```

4) Очистка осуществляется посредством перезаписи каждого байта в освобождаемой области посредством псевдослучайных сигнатур. Количество перезаписей определяется администратором. Использование режима включается установкой значения параметра secdelrnd в конфигурационном файле /etc/fstab для раздела ФС, на котором требуется очищать блоки памяти при их освобождении (например, /dev/sda1). Например, при установке числа перезаписей, равному 6, в список параметров монтирования добавляется параметр secdelrnd=6.

Пример

```
/dev/sda1 /home ext4 acl,defaults,secdelrnd=6 0 2
```

Установка параметра монтирования для очистки блоков памяти при их освобождении может быть выполнена с использованием графической утилиты fly-admin-smc, запущенной администратором. Более подробное описание утилиты см. в электронной справке.

Для включения очистки активных разделов страничного обмена необходимо установить в конфигурационном файле /etc/parsec/swap_wiper.conf для параметра ENABLE значение Y.

Пример

```
ENABLE=Y
```

Для задания списка разделов страничного обмена, для которых не выполняется очистка, может быть использован параметр IGNORE, значение которого является списком перечисленных через пробел игнорируемых разделов страничного обмена.

Пример

```
IGNORE="/dev/sdz10 /dev/sdz11"
```

Настройка очистки разделов страничного обмена при выключении системы может быть выполнена с использованием графической утилиты fly-admin-smc, запущенной администратором. Более подробное описание утилиты см. в электронной справке.

6. ИЗОЛЯЦИЯ МОДУЛЕЙ

Ядро ОС обеспечивает для каждого процесса в системе собственное изолированное адресное пространство. Данный механизм изоляции основан на страничном механизме защиты памяти, а также механизме трансляции виртуального адреса в физический, поддерживаемый модулем управления памятью. Одни и те же виртуальные адреса (с которыми и работает процессор) преобразуются в разные физические для разных адресных пространств. Процесс не может несанкционированным образом получить доступ к пространству другого процесса, т. к. непривилегированный пользовательский процесс лишен возможности работать с физической памятью напрямую.

Механизм разделяемой памяти является санкционированным способом получить нескольким процессам доступ к одному и тому же участку памяти и находится под контролем дискреционных и мандатных ПРД.

Адресное пространство ядра защищено от прямого воздействия пользовательских процессов с использованием механизма страничной защиты. Страницы пространства ядра являются привилегированными, и доступ к ним из непривилегированного кода вызывает исключение процессора, которое обрабатывается корректным образом ядром ОС. Единственным санкционированным способом доступа к ядру ОС из пользовательской программы является механизм системных вызовов, который гарантирует возможность выполнения пользователем только санкционированных действий.

7. МАРКИРОВКА ДОКУМЕНТОВ

Защищенный комплекс программ печати и маркировки документов обеспечивает маркировку выводимых на печать документов. Мандатные атрибуты автоматически связываются с заданием для печати на основе мандатного контекста, получаемого с сетевого соединения. Вывод на печать документов без маркировки субъектами доступа, работающими в ненулевом мандатном контексте, невозможен.

Конфигурационные параметры сервера CUPS определены в файле `/etc/cups/cupsd.conf`. Для управления конфигурационными параметрами сервера CUPS используется утилита командной строки `cupsctl`, запускаемая от имени администратора через механизм `sudo`. Описание данной утилиты приведено в `man cupsctl`.

Для разрешения серверу CUPS удаленно принимать задания и команды необходимо от имени учетной записи администратора выполнить следующие команды:

```
sudo cupsctl --remote-admin --remote-printers --remote-any
sudo cupsctl ServerAlias=*
```

Для разрешения серверу CUPS обрабатывать задания печати, формируемые в ненулевом мандатном контексте, необходимо от имени учетной записи администратора выполнить следующие команды:

```
sudo cupsctl MarkerUser=ipp
sudo cupsctl MacEnable=ON
sudo cupsctl DefaultPolicy=default
```

Для разрешения печати на принтере документов пользователей, работающих в ненулевом мандатном контексте, необходимо выполнить от имени учетной записи администратора следующие команды:

```
sudo lpadmin -p <имя_принтера> -o printer-op-policy=parsec
sudo lpadmin -p <имя_принтера> -o mon-printer-mac-max=Lmax:Cmax
sudo lpadmin -p <имя_принтера> -o mon-printer-mac-min=Lmin:Cmin
```

где `Lmin` и `Lmax` — минимальный и максимальный уровни, а `Cmin` и `Cmax` — минимальный и максимальный набор категорий, соответственно, определяющие возможный мандатный контекст, в котором могут формироваться задания для печати на данном принтере.

Если ЕПП не используется, то команды выполняются от имени администратора через механизм `sudo`. Если ЕПП используется, то команды выполняются от имени пользователя, входящего в специальную группу администраторов печати в БД службы каталогов LDAP (см. документ РУСБ.10015-01 95 01-1).

Маркировка осуществляется на основе модифицируемых файлов шаблонов:

– `/usr/share/cups/marker.template` — описание элементов маркера, проставляемых на первой, каждой, последней странице и на обороте последней страницы;

– /usr/share/cups/psmarker/marker.defs — описание положения элементов маркера на странице;

– /usr/share/cups/fonarik/fonarik.defs — описание положения элементов маркера на обороте последей страницы.

Для установки значений атрибутов, определенных в перечисленных выше файлах шаблонов, используется утилита командной строки lpattr. Для применения утилиты без ЕПП необходимо наличие локальной группы lpmac в ОС. Для применения утилиты в ЕПП необходимо наличие группы lpmac в БД службы каталогов LDAP. Запуск утилиты должен осуществляться от имени пользователя, входящего в группу lpmac.

Описание утилит lpadmin и lpattr приведено в man lpadmin и man lpattr, соответственно.

Для управления принтерами используется графическая утилита fly-admin-printer. Подробное описание утилиты см. в электронной справке.

8. ЗАЩИТА ВВОДА-ВЫВОДА ИНФОРМАЦИИ НА ОТЧУЖДАЕМЫЙ ФИЗИЧЕСКИЙ НОСИТЕЛЬ

Отчуждаемые физические носители могут рассматриваться относительно ОС с двух точек зрения:

– как блочные или символьные устройства ввода-вывода;

– как блочное устройство, которое может быть смонтировано.

В первом случае устройство представляет собой специальный файловый объект, доступ к которому контролируется мандатными и дискреционными ПРД обычным образом и, следовательно, ввод-вывод остается в рамках контроля этих правил.

Во втором случае отчуждаемый носитель информации содержит в себе образ ФС, которая и хранит данные. Данный носитель может быть смонтирован в заданный каталог, и при этом ФС носителя становится частью (представленной в виде поддерева) корневой ФС. Доступ к объектам данной ФС подчиняется мандатным и дискреционным ПРД обычным образом и, следовательно, ввод-вывод на отчуждаемый носитель остается в рамках контроля этих правил.

Для ОС возможность санкционированного монтирования конкретным пользователем конкретных носителей с конкретными ФС определяется администратором системы. В ОС реализованы средства разграничения доступа к подключаемым устройствам на основе генерации правил для менеджера устройств `udev`.

Учет носителей и управление их принадлежностью, протоколированием и мандатными атрибутам осуществляется с помощью утилиты `fly-admin-smc` («Управление политикой безопасности»), в том числе и при работе в ЕПП.

Рекомендации по использованию указанных средств приведены в документе РУСБ.10015-01 95 01-1.

9. СОПОСТАВЛЕНИЕ ПОЛЬЗОВАТЕЛЯ С УСТРОЙСТВОМ

ОС обеспечивает ввод-вывод информации на запрошенное пользователем устройство как для произвольно используемых им устройств, так и для идентифицированных (при совпадении маркировки).

ОС включает в себя механизм, обеспечивающий надежное сопоставление мандатного контекста пользователя с мандатным уровнем и категориями, установленными для устройства. Кроме того, механизм сопоставления пользователя с устройством, реализованный в ОС, обеспечивает при проверке совпадения маркировок носителя и пользователя применение дискреционных ПРД.

10. РЕГИСТРАЦИЯ СОБЫТИЙ

В ОС реализована расширенная подсистема протоколирования, осуществляющая регистрацию событий в двоичные файлы с использованием сервиса `parlogd`. Параметры протоколирования могут настраиваться (10.1) для объектов ФС (файловый аудит) и для пользователей (аудит процессов). Применение настроенных параметров аудита процессов осуществляется PAM-модулем `pam_parsec_aud`. По умолчанию регистрация настроенных для пользователя событий аудита процессов включена в PAM-сценарии: `fly-dm`, `fly-dm-np`, `login`, `su`, `sumac`, `sumac.xauth`. Для протоколирования событий аудита процессов пользователя, проходящего аутентификацию через другие PAM-сценарии, необходимо включить в соответствующие сценарии строку следующего вида:

```
session required pam_parsec_aud.so
```

В библиотеках подсистемы безопасности PARSEC реализован программный интерфейс для протоколирования событий с использованием расширенной подсистемы протоколирования, применяемый для регистрации событий в СУБД PostgreSQL и комплексе программ электронной почты.

10.1. Средства управления протоколированием

Для работы с подсистемой протоколированием имеется ряд графических утилит, которые могут быть использованы для настройки параметров регистрации событий и просмотра протоколов:

– `fly-admin-smc` («Управление политикой безопасности») — управление протоколированием, привилегиями и мандатными атрибутами пользователей, работа с пользователями и группами;

– `fly-admin-viewaudit` («Журнал безопасности») — выборочный просмотр протоколов аудита.

Более подробное описание утилит см. в электронной справке.

Далее рассмотрены средства управления протоколированием в режиме командной строки.

10.1.1. getfaud

Синтаксис:

```
getfaud [-d, --default] [-R, --recursive] [-L, --logical] [-P, --physycal]
  [-n, --numeric] [-l, --long] [-p, --absolute-names] [-c, --omit-header]
  [-s, --skip-empty] [-h, --help] [-v, --version] файлы и/или каталоги
```

Команда `getfaud` служит для получения списков правил протоколирования над файловыми объектами. Следующие события доступны для протоколирования:

– `o`, `open` — открытие файла;

– c, create — создание файла;

– x, exec — исполнение файла;

– u, delete — удаление файла (в каталоге);

– r, acl — смена ACL;

– n, chown — смена владельца;

– m, mac — изменение метки;

– y, modify — изменение файла;

– a, audit — изменение списка регистрируемых событий файла;

– d, chmod — изменение прав доступа к файлу.

Информация о списках посылается на стандартный вывод и может являться входными данными для команды setfaud (10.1.2).

Опции приведены в таблице 26.

Таблица 26

Опция	Описание
-d, --default	Работать со списком правил протоколирования по умолчанию
-R, --recursive	Для поддиректорий рекурсивно
-L, --logical	Следовать по символическим ссылкам
-P, --phisical	Не следовать по символическим ссылкам
-n, --numeric	Выводить информацию о флагах регистрации событий в цифровой форме
-l, --long	Выводить флаги регистрации событий в длинной форме
-p, --absolute-names	Абсолютные имена
-c, --omit-header	Не показывать заголовок (имя файла)
-s, --skip-empty	Пропускать файлы с пустыми атрибутами
-h, --help	Вывести справку и выйти
-v, --version	Вывести информацию о версии и выйти

10.1.2. setfaud

Синтаксис:

```
setfaud [-s, --set] [-b, --remove] [-m, --modify] [-d, --default]
  [-S, --set-file] [-X, --remove-all] [-M, --modify-file] [-B, --restore]
  [-R, --recursive] [-L, --logical] [-P, --physical] [-h, --help]
  [-v, --version] [правила протоколирования] файлы...
```

Команда setfaud устанавливает списки правил протоколирования на файлы. Правила протоколирования задаются в виде:

```
[u:<пользователь>:<флаги протоколирования>]
  [,g:<группа>:<флаги протоколирования>][,o:<флаги протоколирования>], ... ,
```

где <пользователь> и <группа> — символические или численные идентификаторы пользователя и группы; u: означает правило для пользователя, g: — для группы, o: — для остальных.

<флаги протоколирования> := <флаги успешных операций>[[:<флаги неуспешных операций>], ...]

При этом флаги операций могут иметь вид:

<+|-><имя протоколируемого события>,...

(например, +exec, -open) или:

[+|-]<число>

или:

<сокращенное имя протоколируемого события#1><сокращенное имя протоколируемого события#2>...

(например, ou — +open, +delete).

Чтобы посмотреть список протоколируемых событий, набрать:

```
setfaud -h
```

Правила задаются или в командной строке (параметры -s, -m), или в файлах (параметры -S, -M, -B). При этом, файлы могут быть сформированы с помощью перенаправления вывода команды getfaud (см. 10.1.1).

Только администратор может изменять списки правил протоколирования у файлов.

Опции приведены в таблице 27.

Таблица 27

Опция	Описание
-s, --set	Установить список протоколирования из командной строки
-b, --remove	Удалить все элементы списка протоколируемых событий
-m, --modify	Изменить или добавить элементы списка из командной строки
-d, --default	Работать со списком протоколирования по умолчанию
-S, --set-file	Установить список протоколируемых событий из файла
-X, --remove-all	Удалить все списки протоколируемых событий
-M, --modify-file	Изменить или добавить элементы списка протоколируемых событий из файла
-B, --restore	Восстановить атрибуты из файла
-R, --recursive	Для поддиректорий рекурсивно
-L, --logical	Следовать по символическим ссылкам
-P, --physical	Не следовать по символическим ссылкам
-h, --help	Вывести справку и выйти
-v, --version	Вывести информацию о версии и выйти

10.1.3. useraud

Синтаксис:

```
useraud [-d, --delete] [-n, --numeric] [-l, --long] [-g, --group]
  [-o, --other] [-m, --modify] [-h,--help] [-v, --version]
  [пользователь/группа]
```

Команда useraud позволяет просматривать и изменять правила протоколирования для пользователей.

```
useraud [-dnghvolm] [имя пользователя(группы)] [флаги протоколирования]
```

где <флаги протоколирования>: = <флаги успешных операций>

```
[[:<флаги неуспешных операций>], ...]
```

При этом флаги операций могут иметь вид:

```
<+|-><имя протоколируемого события>,...
```

(например, +exec, -open) или:

```
[+|-]<число>
```

или:

```
<сокращенное имя протоколируемого события#1><сокращенное имя протоколируемого
  события#2>...
```

(например, ou — +open,+delete).

Список событий можно получить из помощи команды (параметр -h, --help).

Опции приведены в таблице 28.

Таблица 28

Опция	Описание
-d, --delete	Сбросить правила протоколирования
-n, --numeric	Вывести флаги в шестнадцатеричном формате
-l, --long	Длинный формат вывода флагов
-g, --group	Для группы (по умолчанию — для пользователя)
-o, --other	Для остальных (любой пользователь)
-m, --modify	Изменить существующее правило
-h, --help	Вывести справку и выйти
-v, --version	Вывести информацию о версии и выйти

10.1.4. parselog

Синтаксис:

```
parselog [-v, --version] [-h, --help] [-c, --count] [-f, --follow]
  [-l, --syslog] [-s, --silent] [-b, --binary] [-a, --facility] [-e, --events]
  [-t, --time] [-u, --status] [-x, --uids] [-y, --gids] [-z, --euids]
  [-0..F, --arg0 ... --argF] [файл-журнал]
```

Команда `parselog` может быть использована для анализа двоичных файлов аудита, записанных с помощью `parlogd`.

```
parselog [параметры] [двоичный файл аудита]
```

Если в качестве аргумента не указан файл с данными журнала, то данные ожидаются из стандартного ввода, таким образом, совместно с опцией `-b`, позволяя организовывать конвейеры.

Опции приведены в таблице 29.

Таблица 29

Опция	Описание
`-c, --count`	Вывести статистику
`-f, --follow`	Ожидать появления новых записей в файле
`-l, --syslog`	Записывать выходные данные в систему `syslog` (как служба `LOG_LOCAL0`)
`-s, --silent`	Не выводить ничего на консоль
`-b, --binary`	Вывод в двоичном формате (для конвейеризации)
`-h, --help`	Вывести справку и выйти
`-v, --version`	Вывести информацию о версии и выйти

Параметры-фильтры приведены в таблице 30.

Таблица 30

Параметр-фильтр	Описание
`-a, --facility`	Список служб. Доступные службы: `user`, `proc`, `file`, `custom` или десятичное число от 0 до 15
`-e, --events`	Список событий. Для просмотра списка имен событий (зависит от плагинов) использовать `-e help`
`-t, --time`	Временной диапазон в формате: `<от даты>[-до даты]` где формат даты — `%y[%m[%d[%H[%M[%S]]]]]`
`-u, --status`	Статус. Доступные статусы: `success`, `failed`
`-x, --uids`	Список пользователей (в символьном или десятичном формате)
`-y, --gids`	Список групп (в символьном или десятичном формате)
`-z, --euids`	Список эффективных идентификаторов пользователей в символьном или десятичном формате
`-0..F, --arg0 ... --argF`	Поиск аргумента (от 1 до 15) с помощью регулярных выражений. `arg0` всегда соответствует программе-контексту события. Предполагается, что возвращаемое значение — это последний аргумент события

10.1.5. kernlog, userlog

Команды `kernlog` и `userlog` предназначены для анализа двоичных файлов журнала регистрации событий ядра и событий, приходящих от пользователя, соответственно. Обе команды используют `parselog` (см. 10.1.4) и являются надстройками над ней. Команда `parselog` принимает имя обрабатываемого двоичного файла в качестве параметра. Для команд-надстроек имя анализируемого файла предопределено. Для `kernlog` анализируемым файлом является `/var/log/parsec/kernel.mlog`. В этом файле регистрируются события ядра (сервис `parlogd`). Для `userlog` анализируемым файлом является `/var/log/parsec/user.mlog`. В этом файле регистрируются события, приходящие от пользовательских процессов (сервис `parlogd`). В остальном команды `kernlog` и `userlog` аналогичны `parselog` и принимают те же аргументы командной строки (см. 10.1.4).

10.1.6. psaud

Синтаксис:

```
psaud [-d, --delete] [-n, --numeric] [-l, --long] [-h, --help] [--version]
  [правила протоколирования]
```

Команда `psaud` позволяет изменить или считать правила протоколирования выбранного процесса. Если правила протоколирования не указаны в качестве аргумента, то команда выполняет их считывание с процесса, заданного параметром (идентификатор процесса).

Если аргумент правила протоколирования присутствует, то команда устанавливает правила на процесс. Правила задаются в виде:

```
<флаги протоколирования> := <флаги успешных
  операций>[[:<флаги неуспешных операций>], ...]
```

При этом флаги операций могут иметь вид:

```
<+|-><имя протоколируемого события>,...
```

(например, `+exec`, `-open`) или:

```
[+|-]<число>
```

или:

```
<сокращенное имя протоколируемого события#1><сокращенное имя протоколируемого
  события#2>...
```

(например, `ou` — `+open,+delete`).

Список событий можно получить из помощи команды (параметр `-h`, `--help`).

Только администратор может устанавливать и считывать правила протоколирования процессов. Правила протоколирования наследуются порожденными процессами.

Опции приведены в таблице 31.

Таблица 31

Опция	Описание
-d, --delete	Снять все правила протоколирования с процесса
-n, --numeric	Выводить информацию о правилах протоколирования в численном виде
-l, --long	Выводить информацию о правилах протоколирования в длинной форме
-h, --help	Вывести справку и выйти
--version	Вывести информацию о версии и выйти

10.1.7. Дополнительные параметры системы протоколирования событий

Для тестирования ОС на новых аппаратных конфигурациях можно отключить протоколирование набора системных вызовов одним из двух следующих способов:

1) Отключение протоколирования системных вызовов, не используемых для мандатного разграничения доступа. Для этого необходимо выполнить команду:

```
echo 1 > /parsecfs/disable-non-mac-audit
```

Если вывод команды:

```
cat /parsecfs/disable-non-mac-audit
```

равен 1, то протоколирование системных вызовов, не используемых для мандатного разграничения доступа, отключено.

2) Отключение протоколирования всех системных вызовов. Для этого необходимо выполнить команду:

```
echo 1 > /parsecfs/disable-all-audit
```

Если вывод команды:

```
cat /parsecfs/disable-all-audit
```

равен 1, то протоколирование всех системных вызовов отключено.

Примечание. Если необходимо, чтобы отключение происходило при загрузке ОС, то указанные выше команды необходимо поместить в файл /etc/rc.local.

10.2. Регистрация событий в СУБД PostgreSQL

Регистрация событий в СУБД PostgreSQL версиях 9.2 и 9.4 различаются.

В версии СУБД 9.2 для задания списка регистрации событий используется конфигурационный файл pg_audit.conf (10.2.1).

В версии СУБД 9.4 для настройки режима работы подсистемы регистрации событий используются конфигурационный параметр ac_audit_mode файла postgresql.conf (10.2.2). Этот параметр может изменен только перезапуском сервера. Он может принимать следующие значения:

– internal — Для настройки регистрации событий используются соответствующие команды SQL, а настройки хранятся в таблице pg_db_role_settings;

– external — Для настройки используется внешний файл pg_audit.conf (аналогично регистрации событий в СУБД версии 9.2 (10.2.1));

– external,internal — Смешанный режим. Настройки берутся сначала из внешнего файла pg_audit.conf, после чего дополняются настройками из таблицы pg_db_role_settings;

– internal,external — Смешанный режим. Настройки берутся сначала из таблицы pg_db_role_settings, после чего дополняются настройками из внешнего файла pg_audit.conf;

– none — Протоколирование в данном режиме отключено.

Примечание. В СУБД PostgreSQL версии 9.4 подсистема регистрации событий по умолчанию работает в режиме internal.

10.2.1. Регистрация событий в СУБД PostgreSQL 9.2

Настройка подсистемы сообщений аудита в СУБД PostgreSQL обеспечивается конфигурационным файлом pg_audit.conf конкретного кластера данных, который имеет следующий формат:

– аудит действий администратора СУБД:

```
success events mask = F00E7 failure events mask = 0 user = postgres
```

– для пользователя sny выполнять регистрацию только неуспешных действий:

```
success events mask = 0 failure events mask = FFFFF user = sny
```

– для всех остальных пользователей выполнять регистрацию всех неуспешных действий и всех успешных действий, кроме доступа к данным:

```
success events mask = F0707 failure events mask = FFFFF
```

В этом конфигурационном файле задаются списки успешных (success events mask) и неуспешных (failure events mask) типов запросов на доступ, которые будут регистрироваться в журнале СУБД и подсистеме аудита ОС для отдельных пользователей и по умолчанию. Списки типов запросов на доступ задаются в виде шестнадцатеричных чисел, в которых каждому типу запроса соответствует установленный (для регистрируемых запросов) или сброшенный (для не регистрируемых запросов) бит (таблица 32).

Таблица 32

Тип запроса	Описание	Бит	Шестнадцатеричное значение
SUBJECT	Добавление/изменение/удаление пользователей и групп	0	1
CONFIGURATION	Изменение конфигурации, влияющей на доступ к данным (запрос на изменение значения переменной ac_session_maclabel)	1	2
RIGHTS	Изменение прав доступа к объектам БД	2	4

Окончание таблицы 32

Тип запроса	Описание	Бит	Шестнадцатеричное значение
CHECK_RIGHTS	Модификация прав доступа к объектам БД	3	8
SELECT	Выборка информации из БД	4	10
INSERT	Добавление информации в БД	5	20
UPDATE	Изменение информации в БД	6	40
DELETE	Удаление информации из БД	7	80
TRUNCATE	Очистка данных	8	100
REFERENCES	Задание столбца таблицы в качестве внешнего ключа	10	400
TRIGGER	Добавление триггера к таблице	11	800
EXECUTE	Запуск хранимой процедуры или триггера	12	1000
USAGE	Использование объекта БД	13	2000
CREATE	Создание объектов в БД	16	10000
CREATE_TEMP	Создание временных объектов в БД	17	20000
DROP	Удаление объектов БД	18	40000
ALTER	Изменение объекта БД	19	80000
CONNECT	Соединение пользователя с БД	30	40000000
DISCONNECT	Разъединение пользователя с БД	31	80000000

Информация о соединении пользователей с БД (CONNECT) и разъединении с ней (DISCONNECT) регистрируется всегда, при условии, что список событий не установлен в 0.

10.2.2. Регистрация событий в СУБД PostgreSQL 9.4

В СУБД PostgreSQL версии 9.4 маска регистрации событий устанавливается в процессе авторизации пользователя согласно выбранному режиму работы подсистемы регистрации событий и находится в атрибуте сессии ac_session_audit.

При этом, реализован следующий порядок применения настроек регистрации событий:

1) Настройки для конкретной роли и конкретной базы данных;
2) Настройки для конкретной роли;
3) Настройки для конкретной базы данных;
4) Для всех остальных.

Маска регистрации событий имеет вид {УСПЕХ:ОТКАЗ}, где УСПЕХ — список успешных событий, ОТКАЗ — список неуспешных событий. Она может быть задана с помощью буквенных кодов или с помощью шестнадцатеричного числа. Вывод маски производится в тестовом виде.

В таблице 33 приведено соответствие между событиями, буквенным и шестнадцатеричным значением маски регистрации событий.

Таблица 33

Событие	Символ	Шестнадцатеричное значение	Описание
SUBJECT	S	1	Добавление/изменение/удаление пользователей и групп
CONFIGURATION	s	2	Изменение конфигурации, влияющей на доступ к данным (запрос на изменение значения переменной ac_session_maclabel)
RIGHTS	R	4	Запрос на модификацию прав доступа к объектам БД
CHECK_RIGHTS	V	8	Проверка прав доступа
SELECT	r	10	Выборка информации из БД
INSERT	a	20	Добавление информации в БД
UPDATE	w	40	Изменение информации в БД
DELETE	d	80	Удаление информации из БД
TRUNCATE	D	100	Очистка данных
REFERENCES	x	400	Задание столбца таблицы в качестве внешнего ключа
TRIGGER	t	800	Добавление триггера к таблице
EXECUTE	X	1000	Запуск хранимой процедуры или триггера
USAGE	U	2000	Использование объекта БД
CREATE	C	10000	Создание объектов в БД
CREATE TEMP	T	20000	Создание временного объекта
DROP	E	40000	Удаление объектов БД
ALTER	M	80000	Изменение объекта БД
CONNECT	c	40000000	Запрос на начало сессии
DISCONNECT	e	80000000	Запрос на окончание сессии
Зарезервированный символ	*	C00F3DFF	Полная маска протоколирования
Зарезервированный символ	O	0	Протоколирование отключено

Атрибут сессии ac_session_audit может быть изменен только администратором с помощью команды SET:

```
SET ac_session_audit TO 'новое_значение';
```

и просмотрен с помощью команды:

```
SHOW ac_session_audit;
```

Для просмотра маски сессии используется следующая команда:

```
SELECT session_audit;
```

Для просмотра текущей маски используется следующая команда:

```
SELECT current_audit;
```

Для конвертации маски протоколирования из текстового (буквенного) в шестнадцатеричное значение и из шестнадцатеричного в текстовое (буквенное) используются SQL-функции `text_to_auditmask(TEXT)` и `auditmask_to_text(TEXT)` соответственно. Например:

Пример

```
SELECT text_to_auditmask('{ace:ce}');
    text_to_auditmask
-------------------------
 {0xC0000020:0xC0000000}
(1 строка)

SELECT auditmask_to_text('{0xC0000020:0xC0000000}');
 auditmask_to_text
-------------------
 {ace:ce}
(1 строка)
```

10.2.2.1. Назначение списков регистрации событий в режиме `internal`

Для назначения маски событий в режиме `internal` используется команда `ALTER ROLE`:

```
ALTER ROLE { ALL | имя_роли } [ IN DATABASE имя_базы_данных ] SET
ac_session_audit TO новое_значение;
```

Для удаления списка регистраций событий используется следующая команда:

```
ALTER ROLE { ALL | имя_роли } [ IN DATABASE имя_базы_данных ] RESET
ac_session_audit;
```

При модификации маски происходит автоматическое обновление атрибута `ac_session_audit`.

Примечание. Для выполнения приведенных команд требуются права администратора.

Примечание. При инициализации кластера баз данных автоматически добавляются следующие правила:

```
ALTER ROLE postgres SET ac_session_audit TO '{SsRawdCTEMce:ce}';
ALTER ROLE ALL SET ac_session_audit TO '{SsRDxCTEMce:SsRVrawdDxtXUCTEMce}';
```

10.2.2.2. Назначение списков регистрации событий в режиме external

Для назначения маски событий в режиме external используется конфигурационный файл pg_audit.conf следующего вида:

```
success events mask = value failure events mask = value user =
имя_пользователя database = имя_базы_данных
success events mask = value failure events mask = value user =
имя_пользователя
success events mask = value failure events mask = value
```

Примечание. Любые изменения этого файла будут применены только при перезапуске сервера.

10.2.2.3. Назначение списков регистрации событий в режимах external,internal и internal,external

Загрузка маски регистрации событий в режиме external,internal двухэтапная: сначала выполняется загрузка маски регистрации событий из файла, после чего дополняется настройками из pg_db_role_settings, если в pg_db_role_settings есть более точные настройки. Для изменения маски регистрации событий сессии могут быть использованы команды из 10.2.2.1.

Аналогично и для режима internal,external.

10.2.2.4. Назначеине списков регистрации событий в режиме none

В режиме none регистрация событий отключена, однако, администратор может изменять маску регистрации событий с помощью команд из 10.2.2.1.

11. НАДЕЖНОЕ ВОССТАНОВЛЕНИЕ

11.1. Восстановление ОС после сбоев и отказов

Основными причинами нарушения процесса функционирования СЗИ ОС являются сбои оборудования, приведшие к различным повреждениям ФС. К таковым относятся: сбои электропитания, повреждения носителей информации (жестких дисков), повреждения соединительных кабелей.

В процессе перезагрузки после сбоя ОС автоматически выполнит программу проверки и восстановления ФС — `fsck`. Если повреждения ФС окажутся незначительными, то ее выполнения достаточно для обеспечения целостности ФС.

В случае обнаружения серьезных повреждений ФС данная программа может предложить перезагрузить компьютер в однопользовательский режим и произвести запуск программы `fsck` вручную. Администратор, контролирующий процесс загрузки ОС, после сбоя должен следовать инструкциям, выдаваемым программой `fsck`. Описание программы приведено в `man fsck`.

После завершения загрузки ОС следует проверить целостность файлов с помощью программы контроля целостности. Если в результате проверки найдутся поврежденные или измененные файлы, особенно в каталоге `/etc` и его подкаталогах, то следует восстановить поврежденные файлы с резервной копии.

Если сбой привел к выходу из строя жестких дисков, то следует заменить вышедшее из строя оборудование и переустановить ОС с DVD-диска с дистрибутивом, а пользовательские данные восстановить с резервной копии.

После серьезного повреждения ФС, когда компьютер невозможно перезагрузить, существует возможность восстановления без переустановки ОС. Для этого следует установить DVD-диск с дистрибутивом ОС в устройство чтения DVD-дисков и начать процедуру переустановки. Дождаться появления на экране монитора: «Информация о документации» и одновременно нажать клавиши **<Alt+F2>**. Произойдет переход в режим командной строки под управлением ядра, загруженного с DVD-диска. Затем ввести команду:

```
fdisk  \-l
```

На экране монитора должна появиться информация о разделах жесткого диска. (Если в результате ввода команды на экране монитора нет информации о разделах диска, то повреждения слишком серьезны и необходима полная переустановка системы.) Определить имя раздела, в который была установлена ОС, и ввести следующую последовательность команд:

```
cd  /mnt/
mkdir  hard
mount  /dev/имя_раздела  /mnt/hard
```

В результате указанный раздел жесткого диска будет смонтирован во вновь созданной ФС. Затем ввести команду:

```
chroot  /mnt/hard
```

После этого можно будет использовать командную оболочку ОС и выполнить необходимые действия по восстановлению (например, редактирование файла `fstab`), после чего ввести команды:

```
exit
umount  /mnt/hard
```

Перезагрузить компьютер.

11.2. Средства резервного копирования и восстановления ОС

Резервное копирование выполняется с целью получения копий данных, сохраняемых на случай их потери или разрушения. Подобные копии должны создаваться периодически, в соответствии с заранее установленным графиком. Схемы резервного копирования изменяются в зависимости от размеров и степени охвата резервным копированием ОС, а также от выдвигаемых требований по надежности сохранения жизнеспособности системы. Элементы системы резервного копирования должны включать необходимое оборудование, носители резервных копий и СПО. В качестве оборудования для резервного копирования в ОС может использоваться достаточно широкий набор аппаратных средств, начиная от USB-накопителя и заканчивая библиотекой ленточных устройств. Тип и количество носителей определяются используемым оборудованием, объемами обрабатываемых данных и выбранной схемой резервирования данных. ПО резервного копирования, включенное в состав ОС, является очень разнородным, начиная от простых команд типа `tar`, `cpio`, `gzip` и заканчивая распределенными системами управления хранилищами данных.

Резервное копирование информации используется для:

– восстановления файлов, случайно удаленных пользователями или утерянных из-за отказов устройств хранения;

– получения периодически создаваемых моментальных снимков (snapshots) состояния данных;

– получения данных для восстановления после аварий.

Система резервного копирования является составной частью плана восстановления системы.

Основная идея резервного копирования — создание копий критической части содержания резервируемой системы. Основными исключениями, как правило, не входящими в процедуру резервного копирования функционирующей ОС, являются каталоги, содержащие служебные данные, меняющиеся в процессе функционирования (`/dev`, `/media`, `/mnt`, `/parsecfs`, `/proc`, `/run`, `/sys`, `/tmp`), а также сетевые каталоги (смонтированная NFS,

Samba и прочие виды сетевых данных).

В состав ОС входит множество средств, обеспечивающих решение различных задача резервного копирования данных. Утилиты `tar`, `cpio`, `gzip` представляют собой традиционные инструменты создания резервных копий и архивирования ФС. При создании архива командами `tar` и `gzip` передается список файлов и каталогов, указываемых как параметры командной строки. Любой указанный каталог просматривается рекурсивно. При создании архива с помощью команды `cpio` ей предоставляется список объектов (имена файлов и каталогов, символические имена любых устройств, гнезда доменов UNIX, поименованные каналы и т. п.). Описание команд приведено в руководстве `man` для команд `tar`, `rsync`, `cpio` и `gzip`.

Утилита `rsync` предоставляет возможности для локального и удаленного копирования (резервного копирования) или синхронизации файлов и каталогов, с минимальными затратами трафика.

Кроме того в состав ОС входит комплекс программ Bacula, предназначенных для решения различных задач резервного копирования и восстановления данных.

Для выполнения операций резервного копирования и восстановления объектов ФС с сохранением и восстановлением мандатных атрибутов и атрибутов аудита в ОС можно использовать комплекс программ Bacula, утилиту `rsync` или утилиту `tar`.

ВНИМАНИЕ! Работа с мандатными атрибутами и атрибутами аудита при использовании различных утилит создания резервных копий требует опций сохранения расширенных атрибутов (как правило вида `–xattrs`).

ВНИМАНИЕ! Для восстановления мандатных атрибутов файлов из резервных копий необходимо от имени учетной записи администратора выполнить команду:

```
sudo echo 1 > /parsecfs/unsecure_setxattr
```

ВНИМАНИЕ! Для восстановления мандатных атрибутов файлов из резервных копий процесс должен иметь PRASEC-привилегию `0x1000`. Привилегия может быть получена с использованием утилиты `execaps`:

```
sudo execaps –c 0x1000 tar .......
```

После восстановления из резервных копий файлов с мандатными атрибутами необходимо от имени учетной записи администратора выполнить команду:

```
sudo echo 0 > /parsecfs/unsecure_setxattr
```

Рассмотрим пример создания и восстановления резервной копии с использованием утилиты `tar`.

ВНИМАНИЕ! Предполагается, что уже создан пользователь `user1`, для которого заданы мандатные атрибуты и пользователь уже выполнял вход в систему.

Создание администратором архива домашнего каталога пользователя может быть выполнено с помощью команды:

```
sudo tar --xattrs --acls -cvzf /opt/home.tgz /home/.pdp/user1
```

Опция `--xattrs` означает включение поддержки расширенных атрибутов. Опция `--acls` означает включение поддержки POSIX ACL. Опции `-cvzf` необходимы для создания архива (`create`), включения режима отображения обрабатываемых файлов (`verboze`), применения метода сжатия (`gzip`), указания файла (`file`) соответственно. Путь `/opt/home.tgz` задает место расположения созданного архива и его имя, путь `/home/.pdp/user1` определяет, что именно будет вложено в архив.

Восстановление выполняется с помощью команды:

```
sudo execaps -c 0x1000 -- tar --xattrs
    --xattrs-include=security.{PDPL,AUDIT,DEF_AUDIT}
    --acls -xvf /opt/home.tgz -C /opt/home2/
```

Опция `--xattrs-include=security.{PDPL,AUDIT,DEF_AUDIT}` определяет подключаемый шаблон восстановления расширенных атрибутов (мандатных атрибутов, атрибутов аудита и атрибутов аудита по умолчанию) для ключа `xattrs`. Опции `-xvf` необходимы для извлечения из архива (`extract`), включения режима отображения обрабатываемых файлов (`verboze`), указания файла (`file`) соответственно.

11.2.1. Комплекс программ Bacula

`Bacula` представляет собой набор программ, позволяющий системному администратору управлять процессами резервного копирования и восстановления данных, а также проверять резервные копии, в т. ч. в гетерогенных сетях.

`Bacula` — это сетевая клиент-серверная система резервного копирования. Программа обладает множеством возможностей, позволяющих легко находить и восстанавливать утраченные или поврежденные файлы. Из-за своей модульной архитектуры `Bacula` может масштабироваться от небольших автономных систем до больших сетей, состоящих из сотен компьютеров.

`Bacula` состоит из следующих составных частей:

– `Bacula Director service` — центральная программа, координирующая все выполняемые операции (функционирует в фоне);

– `Bacula Console services` — программа, позволяющая администратору взаимодействовать с центральной программой;

– `Bacula File services` — клиентская программа, устанавливаемая на каждом обслуживаемом компьютере;

– `Bacula Storage services` — программа, обычно функционирующая на компьютере, к которому присоединены внешние устройства для хранения резервных копий;

– `Catalog services` — программа, отвечающая за индексирование и организа-

цию базы резервных данных.

Программа Bacula обеспечивает поддержку сохранения расширенных атрибутов каталогов и файлов и, при необходимости, их последующее восстановление.

ВНИМАНИЕ! Для восстановления объектов ФС с установленными мандатными атрибутами необходимо запустить консоль управления Bacula с PARSEC-привилегией 0x1000, выполнив команду:

```
sudo execaps -c 0x1000 -- bconsole
```

ВНИМАНИЕ! После восстановления объектов ФС с установленными мандатными атрибутами необходимо выполнить перемонтирование ФС, в которой восстанавливались объекты, или перезагрузить ОС.

Описание установки и настройки Bacula приведено в документе РУСБ.10015-01 95 01-1.

11.3. Восстановление СУБД PostgreSQL после сбоев и отказов

Во избежание потерь данных БД PostgreSQL должны регулярно архивироваться.

В случае возникновения ошибок в хранящихся данных, нарушению целостности или в случае программного и/или аппаратного сбоя сервера БД необходимо проведение процедуры восстановления БД. При этом, в зависимости от тяжести повреждений может осуществляться как сохранение существующего кластера БД, с последующим его восстановлением, так и восстановление из резервных копий, созданных в процессе регулярного проведения регламентных работ.

В PostgreSQL существуют три фундаментально отличающихся подхода к резервному копированию данных:

– SQL-дамп;

– резервное копирование на уровне ФС;

– непрерывное архивирование.

Более подробное описание этих методов и процедур копирования и восстановления приведено в документации на СУБД PostgreSQL в пакете postgresql-doc-x.x.

СУБД PostgreSQL содержит ряд стандартных средств резервного копирования и восстановления БД. К ним относятся утилиты pg_dump (11.3.2), pg_dumpall (11.3.3), pg_restore (11.3.4) и, в том числе, интерактивный терминал psql, с помощью которого могут быть восстановлены резервные копии, сохраненные в виде скрипта SQL.

11.3.1. Создание и восстановление резервных копий баз данных с мандатными атрибутами

Для создания и восстановления резервных копий баз данных с мандатными атрибутами необходимо, чтобы пользователь имел привилегии parsec_cap_setmac,

parsec_cap_chmac.

В случае создания резервной копии необходимо назначить максимальную метку на каталог, в который будет выгружена базы данных (для назначения метки на файл копии).

В случае восстановления помимо указанных привилегий требуется права администратора в базе данных (для создания объектов и назначения мандатных атрибутов).

Для восстановления копии базы данных с мандатными атрибутами в другой базе данных, необходимо:

1) Назначить максимальную метку на каталог, в который будет проводиться выгрузка резервной копии;

2) Создать резервную копию исходной базы данных от имени пользователя с привилегиями parsec_cap_setmac и parsec_cap_chmac.

3) На целевом кластере назначить мандатные атрибуты на кластер;

4) Восстановить резервную копию базы данных с помощью клиента psql (если резервная копия сделана в текстовом виде) или с помощью pg_restore (если резервная копия сделана в бинарном виде) от имени пользователя администратора базы данных с привилегиями parsec_cap_setmac и parsec_cap_chmac.

Примечания:

1. Утилита pg_dump, поставляемая вместе с СУБД версии 9.4 выгружает команды по установке комментариев, меток безопасности и назначению мандатных атрибутов в следующем виде:

```
COMMENT ON DATABASE CURRENT_DATABASE IS 'комментарий';
SECURITY LABEL ON DATABASE CURRENT_DATABASE IS '...';
MAC LABEL ON DATABASE CURRENT_DATABASE IS '...';
MAC CCR ON DATABASE CURRENT_DATABASE IS '...';
```

Такая резервная копия может быть восстановлена с установкой комментариев, меток безопасности и мандатных атрибутов в желаемую базу данных.

2. Для восстановления резервных копий баз данных, сделанных в предыдущих версиях, в СУБД версии 9.4 необходимо установить параметр ac_auto_adjust_macs = true;

3. Для переноса кластера с предыдущих версий СУБД на 9.4 необходимо воспользоваться утилитами pg_upgradecluster или pg_upgrade (см. документ «Операционная система специального назначения
«Astra Linux Special Edition» Руководство администратора. Часть 2»).

11.3.2. pg_dump

Для создания резервной копии БД в виде файла в текстовом или других форматах используется утилита pg_dump, которая создает согласованную копию, даже если БД ис-

пользуется, при этом доступ к ней других пользователей (как читающих, так и пишущих) не блокируется.

Резервная копия может создаваться в виде скрипта или форматах упакованного файла. Скрипт резервной копии представляет собой текст, содержащий последовательность SQL-команд, необходимых для воссоздания БД до состояния, в котором она была сохранена. Для восстановления из скрипта он подается на вход утилиты `psql`. Скрипт может быть использован для воссоздания БД даже на другом сервере или архитектуре и с небольшими изменениями на других СУБД.

Синтаксис:

```
pg_dump [OPTION]... [DBNAME]
```

Опции общего характера приведены в таблице 34.

Таблица 34

Опция	Описание
`-f, --file=FILENAME`	Имя выходного файла
`-F, --format=c|t|p`	Формат выходного файла (пользовательский, `tar`, текстовый)
`-v, --verbose`	Режим вывода всех сообщений
`-Z, --compress=0-9`	Уровень сжатия для форматов сжатия
`--lock-wait-timeout=TIMEOUT`	Завершение ошибкой после ожидания `TIMEOUT` для блокировки таблицы
`--help`	Вывести справку и выйти
`--version`	Вывести информацию о версии и выйти

Если не используется `-f/--file`, SQL-скрипт будет направлен в стандартный вывод.

Опции установки соединения приведены в таблице 35.

Таблица 35

Опция	Описание
`-h, --host=HOSTNAME`	Имя сервера БД или каталог сокетов
`-l, --database=DBNAME`	Указать альтернативную БД — шаблон
`-p, --port=PORT`	Номер порта сервера БД
`-U, --username=NAME`	Соединиться как указанный пользователь
`-w, --no-password`	Не запрашивать пароль
`-W, --password`	Принудительный запрос пароля (должен происходить автоматически)

Для получения полной информации о способах использования утилиты `pg_dump`

см. руководство man для утилит pg_dump и psql.

11.3.3. pg_dumpall

Утилита pg_dumpall используется для создания резервной копии всего кластера в виде скрипта.

Скрипт содержит SQL-команды и может быть подан в дальнейшем на вход утилиты psql для восстановления. Операция осуществляется последовательным вызовом утилиты pg_dump для каждой БД кластера. Кроме этого, pg_dumpall сохраняет глобальные объекты, единые для всех БД (pg_dump подобные объекты не сохраняет). Данные объекты включают в себя информацию о пользователях и группах и такие свойства, как: права доступа, применяемые для всех БД в целом.

Синтаксис:

```
pg_dumpall [OPTION]...
```

Опции общего характера приведены в таблице 36.

Таблица 36

Опция	Описание
-f, --file=FILENAME	Имя выходного файла
--lock-wait-timeout=TIMEOUT	Завершение ошибкой после ожидания TIMEOUT для блокировки таблицы
--help	Вывести справку и выйти
--version	Вывести информацию о версии и выйти

Опции установки соединения аналогичны команде pg_dump (см. таблицу 35).

Для получения полной информации о способах использования утилиты pg_dumpall см. руководство man для утилит pg_dumpall и psql.

11.3.4. pg_restore

Для восстановления архивов резервных копий БД, полученных с помощью утилиты pg_dump, используется утилита pg_restore. Она выполняет команды, необходимые для воссоздания БД до состояния на момент времени создания резервной копии. Архивные файлы так же позволяют выбирать с помощью утилиты pg_restore, что именно восстанавливать, и даже менять порядок восстанавливаемых элементов. Файлы архивов разработаны переносимыми между разными архитектурами.

pg_restore может функционировать в двух режимах. При указании БД архив восстанавливается непосредственно в нее. В другом случае, скрипт, содержащий необходимые для пересоздания БД SQL-команды, создается и выводится в файл или стандартный поток вывода. Результирующий скрипт эквивалентен формату текстового вывода утилиты pg_dump. Вследствие этого некоторые опции, управляющие выводом, аналогичны опциям

pg_dump (см. 11.3.2).

Синтаксис:

```
pg_restore [ОПЦИЯ]... [ФАЙЛ]
```

Опции общего характера приведены в таблице 37.

Таблица 37

Опция	Описание
-d, --dbname=ИМЯ	Подсоединиться к указанной БД
-f, --file=FILENAME	Имя входного файла
-F, --format=c|t	Формат файла резервной копии (должно быть автоматически)
-l, --list	Напечатать итоговое оглавление архива
-v, --verbose	Режим вывода всех сообщений
--help	Вывести справку и выйти
--version	Вывести информацию о версии и выйти

Для получения полной информации о способах использования утилиты pg_restore см. руководство man для утилиты pg_restore.

12. СРЕДСТВА ОГРАНИЧЕНИЯ ПРАВ ДОСТУПА К СТРАНИЦАМ ПАМЯТИ

Средства ограничения прав доступа к страницам памяти реализованы на основе набора изменений PaX для ядра ОС, который обеспечивает предоставление наименьших привилегий для процессов при доступе к сегментам памяти в собственном адресном пространстве. Главной функциональной возможностью набора изменений PaX для ядра ОС является защита исполняемого кода в адресном пространстве. Эта защита использует преимущества аппаратной реализации в процессоре неисполняемого бита (NX-бит) для предотвращения выполнения произвольного кода.

Средства ограничения прав доступа к страницам памяти обеспечивают:

– запрет записи в область памяти, помеченную как исполняемая;
– запрет создания исполняемых областей памяти;
– запрет перемещения сегмента кода;
– запрет создания исполняемого стека;
– рандомизацию адресного пространства процесса.

Набор изменений PaX предотвращает выполнение произвольного кода на основе контроля доступа к сегментам памяти следующих типов: чтение, запись, исполнение или их комбинации. Комбинация «запись и исполнение» запрещена.

Набор изменений PaX устанавливает для сегментов данных процессов атрибуты, обеспечивающие невозможность их исполнения, а для сегментов кода программ — атрибуты, обеспечивающие невозможность записи в них. При этом применяется механизм `PAGEEXEC`, который использует эмуляцию или аппаратную реализацию NX-бита. При наличии аппаратной реализации NX-бита механизм `PAGEEXEC` использует ее вместо эмуляции, обеспечивая отсутствие снижения производительности. Набор изменений PaX гарантирует, что адреса с произвольным доступом не будут одновременно доступны на запись и выполнение. Гарантия реализуется использованием в функции `mprotect()` безопасного механизма защиты памяти `MPROTECT`.

Кроме того, набор изменений PaX обеспечивает случайный характер (рандомизацию) смещений сегментов кода и данных (в том числе стека и кучи) при использовании системного вызова отображения в память `mmap()`.

Ядро ОС имеет архитектуру, обеспечивающую невозможность его перемещения в физическом адресном пространстве. Невозможность перемещения сегмента кода в адресном пространстве процесса обеспечивается при использовании PaX установкой соответствующих атрибутов доступа на сегмент кода.

При использовании ПО с открытыми исходными текстами в качестве основы при разработке ПО, обладающего заданными функциональными возможностями возможно воз-

никновение ситуаций, при которых ядро ОС (набор изменений PaX) останавливает исполнение ELF-файла из-за нарушения правил PaX для доступа к страницам памяти в адресном пространстве процесса. Для запускаемого исполняемого модуля формата ELF могут быть установлены специальные атрибуты PaX, разрешающие процессу выполнять определенные действия, запрещенные по умолчанию (см. 12.1).

12.1. Средство управления атрибутами PaX

Для управления атрибутами PaX для запускаемого исполняемого модуля формата ELF используется утилита командной строки `paxctl`, которая обеспечивает отображение и установку специальных атрибутов PaX, разрешающих процессу выполнять определенные запрещенные по умолчанию действия со страницами памяти в своем адресном пространстве.

ВНИМАНИЕ! Не следует использовать утилиту `paxctl` для установки атрибутов библиотек формата ELF.

Синтаксис:

```
paxctl <опции> <файлы>
```

Опции приведены в таблице 38.

Таблица 38

Опция	Описание
`-P`	Включить использование механизмом страничного обмена неисполняемого бита (`PAGEEXEC`)
`-p`	Отключить использование механизмом страничного обмена неисполняемого бита (`NOPAGEEXEC`)
`-E`	Эмулировать трамплины (`EMUTRAMP`, не используется в ОС)
`-e`	Не эмулировать трамилины (`NOEMUTRAMP`, не используется в ОС)
`-M`	Использовать безопасные механизмы защиты памяти (`MROTECT`)
`-m`	Не использовать безопасные механизмы защиты памяти (`NOMROTECT`)
`-R`	Использовать случайное распределение адресов в областях памяти (`RANDMAP`)
`-r`	Не использовать случайное распределение адресов в областях памяти (`NORANDMAP`)
`-X`	Случайным образом определять базовый адрес нормального (`ET_EXEC`) исполняемого файла (`RANDEXEC`)
`-x`	Не определять случайным образом базовый адрес нормального (`ET_EXEC`) исполняемого файла (`RANDEXEC`)
`-S`	Включить сегментирование (`SEGMEXEC`), основанное на использовании механизмом страничного обмена неисполняемого бита (не используется в ОС)
`-S`	Отключить сегментирование (`NOSEGMEXEC`), основанное на использовании механизмом страничного обмена неисполняемого бита (не используется в ОС)
`-v`	Просмотр флагов

Окончание таблицы 38

Опция	Описание
`-z`	Сброс всех флагов
`-c`	Создать в исполняемом файле заголовок `PT_PAX_FLAGS` (если он не существует) путем конвертирования заголовка `PT_GNU_STAK` (если он существует)
`-C`	Создать в исполняемом файле заголовок `PT_PAX_FLAGS` (если он не существует) путем добавления нового заголовка (если возможно)
`-q`	Подавлять сообщения об ошибках
`-Q`	Выводить флаги в коротком формате

13. КОНТРОЛЬ ЦЕЛОСТНОСТИ КСЗ

Для обеспечения контроля целостности (в т. ч. контроля целостности КСЗ) в ОС реализованы:

– средство подсчета контрольных сумм файлов и оптических дисков (13.1);

– средство подсчета контрольных сумм файлов в deb-пакетах (13.2;

– средство контроля соответствия дистрибутиву (13.3);

– средства регламентного контроля целостности (13.4);

– средства создания замкнутой программной среды (13.5).

Для решения задач контроля целостности предназначена библиотека `libgost`, в которой для вычисления контрольных сумм реализованы функции хэширования в соответствии с ГОСТ Р 34.11-94, ГОСТ Р 34.11-2012 с длиной хэш-кода 256 бит и ГОСТ Р 34.11-2012 с длиной хэш-кода 512 бит. Названная библиотека используется в средствах подсчета контрольных сумм файлов и оптических дисков, контроля соответствия дистрибутиву и регламентного контроля целостности, модулях аутентификации.

В ОС реализован механизм, обеспечивающий проверку неизменности и подлинности загружаемых исполняемых файлов формата ELF. Проверка производится на основе контрольных сумм, вычисляемых в соответствии с ГОСТ Р 34.11-94 и ГОСТ Р 34.11-2012 с длиной хэш-кода 256 бит, и ЭЦП, реализованной в соответствии с ГОСТ Р 34.10-2001 и ГОСТ Р 34.10-2012, которые внедрены в исполняемые файлы формата ELF в процессе сборки ОС. Данный механизм предназначен для выявления фактов несанкционированного изменения исполняемых файлов формата ELF (в т. ч. относящихся к КСЗ) и предотвращения их загрузки.

В ОС реализован механизм, обеспечивающий проверку неизменности и подлинности файлов. Проверка производится на основе контрольных сумм, вычисляемых в соответствии с ГОСТ Р 34.11-94 и ГОСТ Р 34.11-2012 с длиной хэш-кода 256 бит, и ЭЦП, реализованной в соответствии с ГОСТ Р 34.10-2001 и ГОСТ Р 34.10-2012, которые внедряются в расширенные атрибуты файловой системы. Данный механизм предназначен для выявления фактов несанкционированного изменения исполняемых файлов и предотвращения их открытия.

13.1. Средство подсчета контрольных сумм файлов и оптических дисков

Для подсчета контрольных сумм файлов и оптических дисков в состав ОС включена утилита командной строки `gostsum`. Для вывода информации о синтаксисе утилиты `gostsum` необходимо выполнить команду:

```
gostsum -h
```

Синтаксис:

```
gostsum [КЛЮЧ] ... [ФАЙЛ]
```

Опции приведены в таблице 39.

Таблица 39

Опция	Описание
`--gost-94`	Устанавливает, что будет использован алгоритм ГОСТ Р 34.11-94
`--gost-2012`	Устанавливает, что будет использован алгоритм ГОСТ Р 34.11-2012 с длиной хэш-кода 256 бит (по умолчанию)
`--gost-2012-512`	Устанавливает, что будет использован алгоритм ГОСТ Р 34.11-2012 с длиной хэш-кода 512 бит
`-b`	Устанавливает размер блоков, которыми будет считываться файл
`-o`	Задает имя файла для вывода контрольной суммы (по умолчанию — стандартный поток вывода)
`-d`	Задает имя файла устройства чтения оптических дисков (файла с образом оптического диска) для подсчета контрольной суммы
`-t`	Тестирование алгоритмов подсчета контрольных сумм
`-p`	Тестирование алгоритмов подсчета контрольных сумм в многопоточной среде
`-h [--help]`	показать эту справку и выйти

Далее приведен пример подсчета контрольной суммы оптического диска:

```
gostsum -d /dev/cdrom
```

13.2. Средство подсчета контрольных сумм файлов в deb-пакетах

Для подсчета контрольных сумм файлов в deb-пакетах в состав ОС включена утилита командной строки `gostsum_from_deb`. Для вывода информации о синтаксисе утилиты `gostsum_from_deb` необходимо выполнить команду:

```
gostsum_from_deb -h
```

Синтаксис:

```
gostsum_from_deb [gostsum аргументы] [-d директория] [-p deb-пакет]
```

Опции приведены в таблице 40.

Таблица 40

Опция	Описание
`--gost-94`	Устанавливает, что будет использован алгоритм ГОСТ Р 34.11-94
`--gost-2012`	Устанавливает, что будет использован алгоритм ГОСТ Р 34.11-2012 с длиной хэш-кода 256 бит (по умолчанию)
`--gost-2012-512`	Устанавливает, что будет использован алгоритм ГОСТ Р 34.11-2012 с длиной хэш-кода 512 бит
`gostsum arguments`	аргументы утилиты `gostsum`

Окончание таблицы 40

Опция	Описание
`-d директория`	Задает имя каталога, содержащего deb-пакеты, для файлов в которых вычисляются контрольные суммы
`-p deb-пакет`	Задает имя deb-пакета, для файлов которого вычисляются контрольные суммы

13.3. Средство контроля соответствия дистрибутиву

Средство контроля соответствия дистрибутиву предоставляет возможность для контроля соответствия объектов ФС ОС дистрибутиву. Для обеспечения контроля целостности объектов ФС ОС (в т. ч. СЗИ) в состав дистрибутива входит файл `gostsums.txt` со списком контрольных сумм по ГОСТ Р 34.11-2012 с длиной хэш-кода 256 бит для всех файлов, входящих в пакеты программ дистрибутива. Используя графическую утилиту `fly-admin-int-check`, можно провести вычисление контрольных сумм файлов системы и проверку соответствия полученных контрольных сумм файлов системы эталонным контрольным суммам. Более подробное описание утилиты см. в электронной справке.

13.4. Средства регламентного контроля целостности

Организация регламентного контроля целостности ОС, прикладного ПО и СЗИ обеспечивается набором программных средств на основе «Another File Integrity Checker». В указанном наборе реализована возможность для проведения периодического (с использованием системного планировщика заданий `cron`) вычисления контрольных сумм файлов и соответствующих им атрибутов расширенной подсистемы безопасности PARSEC (мандатных атрибутов и атрибутов расширенной подсистемы протоколирования) с последующим сравнением вычисленных значений с эталонными. В указанном наборе программных средств реализовано использование библиотеки `libgost`, обеспечивающей подсчет контрольных сумм в соответствии с ГОСТ Р 34.11-94.

Эталонные значения контрольных сумм и атрибутов фалов хранятся в БД. База контрольных сумм и атрибутов может быть создана при помощи команды:

```
afick -i
```

Для вычисления контрольных сумм могут использоваться алгоритмы: MD5-Digest, SHA1 и ГОСТ Р 34.11-2012 с длиной хэш-кода 256 бит.

13.4.1. Настройка

Для настройки достаточно параметров, которые указаны в конфигурационном файле по умолчанию (`etc/afick.conf`). Кроме различных путей, например, к файлам БД:

```
database:=/var/lib/afick/afick
```

где содержится указание о том, какие файлы/каталоги подвергаются контролю целостности и с какими правилами.

Правило PARSEC выглядит следующим образом:

```
PARSEC = p+d+i+n+u+g+s+b+md5+m+e+t
```

где p+d+i+n+u+g+s+b+md5+m означает слежение за всеми стандартными атрибутами файла и использование хэш-функции MD5-Digest для слежения за целостностью содержимого файлов. +e+t означает контроль расширенных атрибутов: мандатной метки и флагов аудита, соответственно. Контроль ACL осуществляется при установке флага +g.

Правило GOST выглядит следующим образом:

```
GOST = p+d+i+n+u+g+s+b+gost+m+e+t
```

где p+d+i+n+u+g+s+b+gost+m означает слежение за всеми стандартными атрибутами файла и использование хэш-функции ГОСТ Р 34.11-2012 с длиной хэш-кода 256 бит для слежения за целостностью содержимого файлов. +e+t означает контроль расширенных атрибутов: мандатной метки и флагов аудита, соответственно. Контроль ACL осуществляется при установке флага +g.

Правило для каталогов:

```
DIR = p+i+n+u+g
```

Правило означает слежение за правами доступа, метаданными, количеством ссылок и другими стандартными атрибутами (подробнее см. /etc/afick.conf).

В файле конфигурации задаются пути к файлам и каталогам, контролируемых afick, например:

```
/boot             GOST
/bin        GOST
/etc/security     PARSEC
/etc/pam.d        PARSEC
/etc/fatab        PARSEC
/lib/modules      PARSEC
/lib64/security   PARSEC
/lib/security     PARSEC
/sbin             PARSEC
/usr/bin   PARSEC
/usr/lib   PARSEC
/usr/sbin         PARSEC
```

Кроме того, на выбор администратора представлен ряд дополнительных путей с правилами. Соответствующие строки помечены знаком комментария # и могут быть активированы снятием этого знака.

При запуске afick с параметром -i:

```
afick -i
```

будет создан файл /var/lib/afick/afick. Это и есть БД формата ndbm. Если посмотреть ее содержимое, то можно обнаружить набор строк, каждая из которых — имя файла и далее через пробел его атрибуты и сигнатуры.

БД защищается системой разграничения доступа.

При запуске AFICK автоматически установит ежедневное задание для CRON. Файл с заданием находится в /etc/cron.daily/afick_cron.

Параметр report_url:=stdout задает местоположение файла-отчета.

В конфигурационном файле есть простой язык макросов, который используется при определении переменных для заданий системного планировщика заданий cron.

Необходимо обеспечить с использованием аппаратно-программных модулей доверенной загрузки и утилиты afick контроль целостности следующих объектов:

1) файлы образов ядра ОС:

```
/boot/vmlinuz-*
```

2) файлы образов временной файловой системы, используемой ядром ОС при начальной загрузке (добавляются в список контроля целостности после создания после выполнения всех необходимых операций по настройке, требующих обновления образов временной файловой системы):

```
/boot/initrd.img-*
```

3) конфигурационный файл меню загрузчика grub:

```
/boot/grub/menu.lst
```

4) кофигурационный файл, определяющий используемый по умолчанию графический дисплейный менеджер:

```
/etc/X11/default-display-manager
```

5) конфигурационный файл настройки файловых системы, доступных для монтирования через NFS:

```
/etc/exports
```

6) конфигурационный файл, содержащий информацию о различных устройствах хранения и файловых системах:

```
/etc/fstab
```

7) файл, содержащий перечень локальных групп ОС (добавляется в список контроля целостности после создания всех необходимых групп):

```
/etc/group
```

8) скрипты для запуска сервисов в каталоге

```
/etc/init.d/
```

9) конфигурационный файл, определяющий параметры работы первого процесса пользовательского режима init:

/etc/inittab

10) конфигурационные файлы, определяющие порядок работы PAM-модулей:

/etc/pam.conf

/etc/pam.d/chfn

/etc/pam.d/chsh

/etc/pam.d/common-account

/etc/pam.d/common-account.pam-old

/etc/pam.d/common-auth

/etc/pam.d/common-auth.pam-old

/etc/pam.d/common-password

/etc/pam.d/common-password.pam-old

/etc/pam.d/common-session

/etc/pam.d/common-session.pam-old

/etc/pam.d/cron

/etc/pam.d/cups

/etc/pam.d/cvs

/etc/pam.d/dovecot

/etc/pam.d/fly-dm

/etc/pam.d/fly-dm-np

/etc/pam.d/login

/etc/pam.d/other

/etc/pam.d/passwd

/etc/pam.d/polkit

/etc/pam.d/samba

/etc/pam.d/sshd

/etc/pam.d/su

/etc/pam.d/sumac.xauth

11) файл, содержащий перечень локальных пользователей ОС (добавляется в список контроля целостности после создания всех необходимых пользователей):

/etc/passwd

12) символические ссылки на скрипты для запуска сервисов в каталогах:

/etc/rc*

13) конфигурационный файл, определяющий перечень терминалов, с которых суперпользователь root может регистрироваться в системе:

/etc/securetty

14) конфигурационный файл, определяющий перечень регистрируемых оболочек в ОС:

/etc/shells

15) конфигурационный файл, содержащий значения параметров ядра:

`/etc/sysctl.conf`

16) модули ядра, входящие в подсистему безопасности PARSEC:

`/lib/modules/*/misc/digsig_verif.ko`

`/lib/modules/*/misc/parsec.ko`

`/lib/modules/*/misc/parsec-cifs.ko`

С помощью команд `lsmod` и `modinfo` можно определить перечень модулей ядра, подлежащих контролю целостности средствами АПМДЗ.

17) PAM-модули в каталоге:

`/lib/security/pam`

18) системные исполняемые файлы в каталоге:

`/sbin/`

19) исполняемые файлы в каталогах:

`/bin/`

`/sbin/`

`/usr/bin/`

`/usr/sbin/`

13.5. Средства создания замкнутой программной среды

Средства создания замкнутой программной среды предоставляют возможность внедрения цифровой подписи в исполняемые файлы формата ELF, входящие в состав устанавливаемого СПО (13.5.2), и в расширенные атрибуты файловой системы.

Механизм контроля целостности исполняемых файлов и разделяемых библиотек формата ELF при запуске программы на выполнение реализован в модуле ядра ОС `digsig_verif`, который является не выгружаемым модулем ядра Linux, и может функционировать в одном из следующих режимов:

1) исполняемым файлам и разделяемым библиотекам с неверной ЭЦП, а также без ЭЦП загрузка на исполнение запрещается (штатный режим функционирования);

2) исполняемым файлам и разделяемым библиотекам с неверной ЭЦП, а также без ЭЦП загрузка на исполнение разрешается, при этом выдается сообщение об ошибке проверки ЭЦП (режим для проверки ЭЦП в СПО);

3) ЭЦП при загрузке исполняемых файлов и разделяемых библиотек не проверяется (отладочный режим для тестирования СПО).

Механизм контроля целостности файлов при их открытии на основе ЭЦП в расширенных атрибутах файловой системы также реализован в модуле ядра ОС `digsig_verif` и может функционировать в одном из следующих режимов:

1) запрещается открытие файлов, поставленных на контроль, с неверной ЭЦП или

без ЭЦП;

2) открытие файлов, поставленных на контроль, с неверной ЭЦП или без ЭЦП разрешается, при этом выдается сообщение об ошибке проверки ЭЦП (режим для проверки ЭЦП в расширенных атрибутах файловой системы);

3) ЭЦП при открытии файлов не проверяется.

13.5.1. Настройка модуля digsig_verif

Для изменения режима функционирования модуля `digsig_verif` необходимо отредактировать файл `/etc/digsig/digsig_initramfs.conf`.

Настройка режима функционирования механизма контроля целостности исполняемых файлов и разделяемых библиотек формата ELF при запуске программы на выполнение осуществляется следующим образом:

1) Для использования отладочного режима для тестирования СПО необходимо установить для параметра `DIGSIG_LOAD_KEYS` значение `0`:

```
DIGSIG_LOAD_KEYS=0
```

2) Для использования режима для проверки ЭЦП в СПО необходимо установить для параметра `DIGSIG_LOAD_KEYS` значение `1`:

```
DIGSIG_LOAD_KEYS=1
```

3) Для использования штатного режима функционирования необходимо установить следующие значения параметров:

```
DIGSIG_LOAD_KEYS=1
DIGSIG_ENFORCE=1
```

Каждый дополнительный ключ, использованный для подписывания СПО (13.5.2), необходимо скопировать в каталог `/etc/digsig/keys/`, например, с использованием команды:

```
cp /<каталог>/<файл ключа> /etc/digsig/keys/
```

В каталоге `/etc/digsig/keys/` может располагаться иерархическая структура дополнительных ключей для контроля целостности исполняемых файлов и разделяемых библиотек формата ELF. В указанной структуре одни дополнительные ключи могут быть подписаны на других дополнительных ключах. При этом дополнительные ключи должны располагаться в подкаталогах таким образом, чтобы при их загрузке не нарушалась цепочка проверки подписей.

Пример

```
/etc/digsig/keys/key1.gpg - публичный ключ 1, подписанный
на первичном ключе ОАО <<НПО РусБИТех>>
/etc/digsig/keys/key2.gpg - публичный ключ 2, подписанный
на первичном ключе ОАО <<НПО РусБИТех>>
/etc/digsig/keys/key1/key1-1.gpg - публичный ключ, подписанный на ключе 1
```

```
/etc/digsig/keys/key1/key1-2.gpg - публичный ключ, подписанный на ключе 1
/etc/digsig/keys/key2/key2-1.gpg - публичный ключ, подписанный на ключе 2
/etc/digsig/keys/key2/key2-2.gpg - публичный ключ, подписанный на ключе 2
```

Для проверки использования дополнительных ключей для контроля целостности исполняемых файлов и разделяемых библиотек формата ELF (до перезагрузки ОС) можно от имени учетной записи администратора через механизм `sudo` выполнить команды:

```
cat /etc/digsig/key_for_signing.gpg > /sys/digsig/keys
cat /etc/digsig/keys/key1.gpg >> /sys/digsig/keys
cat /etc/digsig/keys/key2.gpg >> /sys/digsig/keys
cat /etc/digsig/keys/key1/key1-1.gpg >> /sys/digsig/keys
cat /etc/digsig/keys/key1/key1-2.gpg >> /sys/digsig/keys
cat /etc/digsig/keys/key2/key2-1.gpg >> /sys/digsig/keys
cat /etc/digsig/keys/key2/key2-2.gpg >> /sys/digsig/keys
```

ВНИМАНИЕ! При проверке ЭЦП в расширенных атрибутах файловой системы установка для параметра `DIGSIG_ENFORCE` значения `1` запрещает открытие поставленных на контроль файлов с неверной ЭЦП или без ЭЦП.

Настройка режима функционирования механизма контроля целостности файлов при их открытии на основе ЭЦП в расширенных атрибутах файловой системы осуществляется следующим образом:

1) Для включения механизма необходимо установить для параметра `DIGSIG_USE_XATTR` значение `1`:

```
DIGSIG_USE_XATTR=1
```

2) Для загрузки дополнительных ключей, используемых только при проверке ЭЦП в расширенных атрибутах файловой, системы необходимо установить для параметра `DIGSIG_LOAD_XATTR_KEYS` значение `1`:

```
DIGSIG_LOAD_XATTR_KEYS=1
```

3) Для игнорирования дополнительных ключей, используемых только при проверке ЭЦП в расширенных атрибутах файловой, необходимо установить для параметра `DIGSIG_IGNORE_XATTR_KEYS` значение `1`:

```
DIGSIG_IGNORE_XATTR_KEYS=1
```

4) Для настройки шаблонов имен, используемых при проверке ЭЦП в расширенных атрибутах ФС, необходимо в файле `/etc/digsig/xattr_control` задать их список. Каждая строка задает свой шаблон в виде маски полного пути. Например, следующий шаблон определяет, что будет проверяться ЭЦП всех файлов в каталоге `/bin`, имя которых начинается на `lo`:

```
\bin\lo
```

Каждый дополнительный ключ, использованный для подписывания файлов в расширенных атрибутах, необходимо скопировать в каталог `/etc/digsig/xattr_keys/`, на-

пример, с использованием команды:

```
cp /<каталог>/<файл ключа> /etc/digsig/xattr_keys/
```

В каталоге /etc/digsig/xattr_keys/ может располагаться иерархическая структура дополнительных ключей для контроля целостности файлов. В указанной структуре одни дополнительные ключи могут быть подписаны на других дополнительных ключах. При этом дополнительные ключи должны располагаться в подкаталогах таким образом, чтобы при их загрузке не нарушалась цепочка проверки подписей.

Пример

```
/etc/digsig/xattr_keys/key1.gpg - публичный ключ 1
/etc/digsig/xattr_keys/key2.gpg - публичный ключ 2
/etc/digsig/xattr_keys/key1/key1-1.gpg - публичный ключ, подписанный на ключе 1
/etc/digsig/xattr_keys/key1/key1-2.gpg - публичный ключ, подписанный на ключе 1
/etc/digsig/xattr_keys/key2/key2-1.gpg - публичный ключ, подписанный на ключе 2
/etc/digsig/xattr_keys/key2/key2-2.gpg - публичный ключ, подписанный на ключе 2
```

Для проверки использования дополнительных ключей для контроля целостности файлов (до перезагрузки ОС) можно от имени учетной записи администратора через механизм sudo выполнить команды:

```
cat /etc/digsig/xattr_keys/key1.gpg >> /sys/digsig/xattr_keys
cat /etc/digsig/xattr_keys/key2.gpg >> /sys/digsig/xattr_keys
cat /etc/digsig/xattr_keys/key1/key1-1.gpg >> /sys/digsig/xattr_keys
cat /etc/digsig/xattr_keys/key1/key1-2.gpg >> /sys/digsig/xattr_keys
cat /etc/digsig/xattr_keys/key2/key2-1.gpg >> /sys/digsig/xattr_keys
cat /etc/digsig/xattr_keys/key2/key2-2.gpg >> /sys/digsig/xattr_keys
```

Управление модулем digsig_verif осуществляется через интерфейс sysfs с использованием файлов:

– /sys/digsig/enforce — проверка и переключение режима работы;

– /sys/digsig/use_xattr — включить проверку ЭЦП в расширенных атрибутах ФС;

– /sys/digsig/keys — файл загрузки дополнительных ключей для проверки ЭЦП исполняемых файлов формата ELF и ЭЦП в расширенных атрибутах ФС;

– /sys/digsig/ignore_gost2001 — отключение проверки ЭЦП по ГОСТ Р 34.10-2001;

– /sys/digsig/ignore_xattr_keys — 1;

– /sys/digsig/xattr_control — список шаблонов имен, используемых при проверке ЭЦП в расширенных атрибутах ФС;

– /sys/digsig/xattr_keys — файл загрузки дополнительных ключей, используемых только при проверке ЭЦП в расширенных атрибутах ФС.

Проверка режима работы выполняется командой:

```
cat /sys/digsig/enforce
```

ВНИМАНИЕ! Для отключения проверки ЭЦП по ГОСТ Р 34.10-2001 необходимо в конфигурационном файле `/etc/digsig/digsig_initramfs.conf` установить следующее значение параметра:

```
DIGSIG_IGNORE_GOST2001=1
```

ВНИМАНИЕ! После внесения изменений в конфигурационный файл `/etc/digsig/digsig_initramfs.conf` и для загрузки модулем `digsig_verif` ключей после их размещения его в каталогах `/etc/digsig/keys/` и `/etc/digsig/xattr_keys/` необходимо от имени учетной записи администратора через механизм `sudo` выполнить команду:

```
sudo update-initramfs -u -k all
```

13.5.2. Подписывание

В модуле ядра `digsig_verif` реализован механизм, позволяющий использовать несколько ключей при подписывании файлов формата ELF.

Порядок использования ключей для `digsig_verif`: дополнительные ключи записываются в `/sys/digsig/keys` или `/sys/digsig/xattr_keys` в иерархической последовательности цепочек подписей.

Все дополнительные ключи должны быть подписаны главным ключом или другим дополнительным ключом, подпись которого может быть проверена (за исключением первого ключа в каталоге `/sys/digsig/xattr_keys`).

Для создания дополнительных ключей используется GNU Privacy Guard. Модифицированный GnuPG выводит ГОСТ Р 34.11-94 в списке доступных алгоритмов. Для получения списка доступных алгоритмов необходимо выполнить команду:

```
# gpg --version
gpg (GnuPG) 1.4.18
Copyright (C) 2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Home: ~/.gnupg
Поддерживаются следующие алгоритмы:
С открытым ключом: RSA, RSA-E, RSA-S, ELG-E, DSA,
           GOST_R 34.10-2001,GOST_R 34.10-2012
Симметричные шифры: 3DES, CAST5, BLOWFISH,
           AES, AES192, AES256, TWOFISH,
           CAMELLIA128, CAMELLIA192, CAMELLIA256
Хэш-функции: MD5, SHA1, RIPEMD160, SHA256, SHA384, SHA512,
```

```
          SHA224, GOST_R34.11-2012, GOST_R34.11-94
Алгоритмы сжатия: Без сжатия, ZIP, ZLIB,
          BZIP2
```

Далее приведен пример создания дополнительного ключа и его использования для подписывания СПО.

Пример

1) создается ключевая пара GOST R 34.10-2001 и сохраняется в каталоге ~/.gnupg. Для создания ключевой пары необходимо выполнить приведенную далее команду с последующим выбором в меню gpg алгоритма ГОСТ Р 34.10-2001:

```
keys@debian:~$ gpg --gen-key
gpg (GnuPG) 1.4.18; Copyright (C) 2014 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: создан каталог '/home/keys/.gnupg'
gpg: создан новый файл настоек '/home/keys/.gnupg/gpg.conf'
gpg: ВНИМАНИЕ: параметры в '/home/keys/.gnupg/gpg.conf' еще не активны при /
   этом запуске
gpg: создана таблица ключей '/home/keys/.gnupg/secring.gpg'
gpg: создана таблица ключей '/home/keys/.gnupg/pubring.gpg'
Выберите тип ключа:
(1) RSA и RSA (по умолчанию)
(2) DSA и Elgamal
(3) DSA (только для подписи)
(4) RSA (только для подписи)
(10) GOST R 34.10-2001 (только для подписи) устаревший
(12) GOST R 34.10-2012 (только для подписи)
Ваш выбор? 12
GOST keypair will have 256 bits.
Выборите срок действия ключа.
0 = без ограничения срока действия
<n> = срок действия - n дней
<n>w = срок действия - n недель
<n>m = срок действия - n месяцев
<n>y = срок действия - n лет
Срок действия ключа? (0) 0
Срок действия ключа не ограничен
Все верно? (y/N) y

Для идентификации Вашего ключа необходим ID пользователя. Программа создаст /
   его
из Вашего имени, комментария и адреса электронной почты в виде:
```

```
"Baba Yaga (pensioner) <yaga@deepforest.ru>"

Ваше настоящее имя: Test GOST R 34.10-2012 Secondary Key
Адрес электронной почты: test@gost.secondary.key
Комментарий:
Вы выбрали следующий ID пользователя:
"Test GOST R 34.10-2001 Secondary Key <test@gost.secondary.key>"

Сменить (N)Имя, (C)Комментарий, (E)адрес или (O)Принять/(Q)Выход? O
Для защиты закрытого ключа необходим пароль.
Введите пароль: ПАРОЛЬ

Повторите пароль: ПАРОЛЬ
```

2) экспорт ключа в файл осуществляется командой:

```
keys@debian:$ gpg --export "Test GOST R 34.10-2012 Secondary Key
<test@gost.secondary.key>" > /tmp/secondary_gost_key.gpg
```

3) ключ пользователя может быть заверен с использованием команд:

```
gpg --import /tmp/secondary_gost_key.gpg

gpg --sign-key "Test GOST R 34.10-2012 Secondary Key
<test@gost.secondary.key>"

gpg --export "Test GOST R 34.10-2001 Secondary Key
<test@gost.secondary.key>" > /tmp/secondary_gost_key_signed.gpg
```

4) пользователь подписывает на данном ключе некоторый файл формата ELF с использованием утилиты `bsign` (по умолчанию подпись внедряется в том числе и в расширенные атрибуты файла):

```
keys@debian:~$ bsign --sign test_elf
```

5) пользователь подписывает утилитой `bsign` на данном ключе произвольный файл с внедрением подписи только в расширенные атрибуты:

```
keys@debian:~$ bsign --sign --xattr test_elf
```

Дополнительная информация приведена в справке по утилите `bsign`;

6) для проверки правильности ЭЦП файла формата ELF используется утилита `bsign`:

```
keys@debian:~$ bsign -w test_elf
```

7) дополнительный ключ пользователя, подписанный на главном ключе, копируется в каталог `/etc/digsig/` (см. 13.5.1);

8) подписанный файл формата ELF может выполняться:

```
keys@debian:~$ ./test_elf
hello world!
keys@debian:~$
```

14. ГЕНЕРАЦИЯ КСЗ

Генерация КСЗ осуществляется в одном из двух следующих режимов:

– режим ЕПП;

– локальный режим.

Для использования режима ЕПП необходимо наличие в сети установленного и настроенного сервера ALD. На рабочих местах пользователей в ОС должен быть установлен пакет `ald-client` и выполнены соответствующие действия по настройке (см. документ РУСБ.10015-01 95 01-1).

После определения режима работы КСЗ ОС для завершения процедуры генерации КСЗ необходимо выполнить следующие действия:

1) создать набор мандатных уровней;

2) создать набор мандатных категорий;

3) создать набор служебных пользователей, наличие которых необходимо для функционирования защищенных комплексов программ СУБД, гипертекстовой обработки данных, электронной почты и программ маркировки и учета печатных документов из состава ОС;

4) установить привилегии для служебных пользователей;

5) установить параметры аудита (протоколирования событий) в ОС.

ВНИМАНИЕ! Для изменения максимального мандатного контекста объектов (см. 4.1) необходимо внести изменения в сценарий `pdp-init-fs`, размещенный в каталоге `/usr/sbin`, переопределив значения переменных:

– `sysmaxlev` – максимальный мандатный уровнь;

– `sysmaxilev` – максимальный уровнь целостности;

– `sysmaxcat` – максимальный набор мандатных категорий.

После выполнения перечисленного набора действий осуществляется создание пользователей, установка для них разрешенных мандатных уровней и категорий, а в случае необходимости, параметров протоколирования.

В режиме ЕПП указанные действия выполняются при помощи графической утилиты `fly-admin-smc` или при помощи утилиты командной строки `ald-admin` (см. `man ald-admin`).

В локальном режиме указанные действия выполняются при помощи графической утилиты `fly-admin-smc` или при помощи соответствующих утилит командной строки (см. 4.4) от имени учетной записи администратора (см. раздел 1) через механизм `sudo`. Более подробное описание графических утилит см. в электронной справке.

15. РЕЖИМ КИОСКА

15.1. Общие сведения

Режим киоска служит для ограничения прав пользователей в системе. Для использования функциональных возможностей данного режима необходимо установить пакет `parsec-cups`.

Степень этих ограничений прав пользователей задается маской киоска. Ее действие аналогично действию маски `umask` с тем отличием, что если `umask` накладывается при создании новых объектов ФС, то маска киоска накладывается на права доступа к файлу при любой попытке пользователя получить доступ. Маска киоска задается в конфигурационном файле `/etc/parsec/kiosk_mask` и по умолчанию равна `0000` (режим киоска выключен). Если маска равна `0003` (типичное значение при включенном режиме киоска), для пользователя блокируется доступ по записи и исполнению ко всем файлам, не принадлежащим ему, либо группе, в которую он входит. При маске, равной `0000`, поведение системы остается стандартным и на права доступа пользователя не накладывается никаких ограничений. Получить текущую маску киоска можно, прочитав содержимое файла `/parsecfs/mode_mask`. Маска киоска применяется только к обычным файлам. К каталогам, сокетам и т. д. маска не применяется.

В режиме киоска (маска по умолчанию) пользователь не имеет возможности запустить ни одну системную программу, т. к. эти действия замаскированы.

15.2. mkiosk

Команда `mkiosk` позволяет задавать права доступа пользователя к конкретным файлам. Эти права доступа не подвержены действию маски и реализованы в виде ACL на специальные виртуальные файлы ФС `parsec`, являющиеся ссылками на реальные файлы. После перезагрузки все установленные ACL будут утеряны.

Права доступа к файлу задаются в виде:

```
<абсолютный_путь_к_файлу>
```

Так, например, права только на чтение для файла `/etc/hosts` можно задать в виде:

```
/etc/hosts r--
```

Права на чтение, запись и исполнение для файла `/usr/bin/example.sh` задаются в виде:

```
/usr/bin/example.sh rwx
```

Так как задавать все права доступа в командной строке было бы крайне неудобно, существует система профилей. Это файлы с готовыми наборами прав доступа для запуска каких-либо программ. Например, есть профиль для запуска `bash`, профиль для запуска `ls`

и т. д. Эти профили хранятся в каталоге `/etc/parsec/kiosk-profiles`. Вместо прав доступа к конкретным файлам, в командной строке команды `mkiosk` можно указать готовый профиль. Она отличает задание файлов от задания профиля по наличию первого символа `/` в имени файла. Имена профилей задаются без указания полного пути к ним. В общем случае профиль может содержать в себе права доступа на более сложные действия, чем запуск одной программы. Существует также профиль с именем `default`, который используется автоматически при каждом запуске `mkiosk`. В нем содержатся права доступа, которые необходимы всегда. Это, например, право на использование динамического линковщика. Для создания профилей можно использовать команду `otrace` (15.3).

Для автоматизации процесса установки прав доступа для каждого пользователя существует конфигурационный файл, хранящийся в каталоге `/etc/parsec/kiosk` и содержащий все необходимые права доступа. По сути это такой же профиль, как и в случае профилей программ, но относящийся к конкретному пользователю. Этот файл может содержать как явное задание прав доступа к конкретным файлам, так и ссылки на профили программ. При входе пользователя в систему права доступа из конфигурационного файла будут установлены автоматически при помощи специального PAM-модуля.

Параметры команды приведены в таблице 41.

Таблица 41

Параметр	Описание
`-h, --help`	Вывести справку и выйти
`-u , --user=`	Установить права доступа для пользователя
`-w, --without-profile`	Не использовать профиль указанного пользователя для установки прав доступа. Будут использованы только права доступа из командной строки
`-e, --mask`	Указать маску киоска. Права доступа на файлы для пользователя будут устанавливаться только в том случае, когда необходимые биты маскируются указанной маской. По умолчанию используется текущая маска киоска из файла `/parsecfs/mode_mask`

Примеры:

1. Установить права доступа для пользователя `ttt`, взятые из его профиля `/etc/parsec/kiosk/ttt`

```
mkiosk -u ttt
```

2. Установить права на чтение файла `/etc/passwd` для пользователя `ttt`. При этом не учитывать профиль пользователя. Предполагается, что системная маска киоска равна `0003` и, соответственно, замаскированы права на запись и выполнение файлов

```
mkiosk -u ttt --mask=3 --without-profile "/etc/passwd r--"
```

15.3. otrace

Команда `otrace` предназначена для трассировки процессов относительно системных вызовов `open()` и `execve()`.

Синтаксис:

```
otrace [-h, --help] [-s, --silent] [-o, --output=] [-k, --kiosk-dir=]
   [-p, --pid=] [-u, --user=] [-t, --trace] [-a, --audit-trace]
   [-m, --merge] [-f, --trace-failed] [-e, --mask=] [command]
```

`otrace` используется в процессе конфигурирования режима киоска и служит для автоматизации создания профилей прав доступа.

В режиме киоска (стандартные настройки) пользователю запрещены запись и выполнение файлов, не принадлежащих ему, либо группе, в которую он входит. Чтобы пользователь имел возможность хотя бы войти в систему, необходимо явно указать права доступа ко всем файлам, прямо или косвенно участвующим при этой операции. Права доступа к файлам задаются с помощью установки ACL на специальные файлы-ссылки в ФС `parsec`. При этом права доступа на реальные файлы не изменяются. При перезагрузке системы все ACL на специальные файлы-ссылки будут утеряны.

Чтобы облегчить задачу установки пользователям прав доступа, используются профили прав доступа. Системные профили хранятся в каталоге `/etc/parsec/kiosk-profiles`. Далее приведен пример профиля, позволяющий запустить команду `ls`:

```
/bin/ls r-x
/etc/group r--
/etc/ld.so.cache r--
/etc/ld.so.preload r--
/etc/localtime r--
/etc/nsswitch.conf r--
/etc/passwd r--
/etc/selinux/config r--
/lib/libacl.so.1 r--
/lib/libattr.so.1 r--
/lib/libc.so.6 r--
/lib/libdl.so.2 r--
/lib/libnsl.so.1 r--
/lib/libnss_compat.so.2 r--
/lib/libnss_files.so.2 r--
/lib/libnss_nis.so.2 r--
/lib/libpthread.so.0 r--
```

```
/lib/librt.so.1 r--
/lib/libselinux.so.1 r--
/proc/mounts r--
/usr/lib/gconv/gconv-modules.cache r--
/usr/lib/gconv/KOI8-R.so r--
/usr/lib/locale/locale-archive r--
/usr/share/locale/locale.alias r--
/usr/share/locale/ru/LC_MESSAGES/coreutils.mo r--
/usr/share/locale/ru/LC_TIME/coreutils.mo r--
/usr/share/locale/ru_RU/LC_MESSAGES/coreutils.mo r--
/usr/share/locale/ru_RU/LC_TIME/coreutils.mo r--
/usr/share/locale/ru_RU.utf8/LC_MESSAGES/coreutils.mo r--
/usr/share/locale/ru_RU.UTF-8/LC_MESSAGES/coreutils.mo r--
/usr/share/locale/ru_RU.utf8/LC_TIME/coreutils.mo r--
/usr/share/locale/ru_RU.UTF-8/LC_TIME/coreutils.mo r--
/usr/share/locale/ru.utf8/LC_MESSAGES/coreutils.mo r--
/usr/share/locale/ru.UTF-8/LC_MESSAGES/coreutils.mo r--
/usr/share/locale/ru.utf8/LC_TIME/coreutils.mo r--
/usr/share/locale/ru.UTF-8/LC_TIME/coreutils.mo r--
```

Для применения профиля к конкретному пользователю используется команда `mkiosk` (см. 15.2).

Если профиль служит для запуска программы, как это было в примере, он должен содержать права доступа не только к исполняемому файлу, но и ко всем используемым библиотекам и всем файлам, которые открывает программа в процессе исполнения. Также необходимы права доступа на динамический линковщик, которые не попадут автоматически в профиль, созданный командой `otrace`. Минимальные права доступа, которые требуются всегда, содержатся в специальном профиле `/etc/parsec/kiosk-profile/default` и добавляются туда вручную.

Профили могут описывать права доступа для более сложных задач, чем запуск одной программы. Чтобы облегчить администрирование, можно использовать профили внутри других профилей. Если внутри профиля встречается строка с именем другого профиля, то содержимое указанного профиля полностью объединяется с содержимым текущего профиля. Объединение профилей осуществляется рекурсивно. Если строка в файле профиля начинается не с символа `/`, то она рассматривается как имя профиля. Команда `otrace` также занимается объединением профилей (см. опцию `--merge`).

Наконец, существуют профили, относящиеся к отдельным пользователям. Они хранятся в каталоге `/etc/parsec/kiosk` и заполняются вручную на основе готовых про-

филей либо стандартных строк, описывающих права доступа к конкретному файлу. При включенном режиме киоска профили пользователей автоматически применяются при входе пользователя в систему. Это реализовано при помощи специального PAM-модуля и команды `mkiosk` (см. 15.2).

`otrace` может использовать два различных механизма для трассировки процессов. Первый — это программа `strace` (опция `--trace`). Второй — подсистема аудита PARSEC (опция `--audit-trace`). Программа `strace` имеет некоторые ограничения по использованию, поэтому применять механизм `--` следует только для трассировки довольно простых, не SUID-программ.

В режиме `--trace` цель трассировки может быть задана либо в командной строке команды `otrace` в качестве аргумента (тогда указанная программа будет запущена), либо может быть задан PID уже существующего процесса (опция `--pid`).

В режиме `--audit-trace` цель трассировки задается так же, как и в режиме `--trace`. Но кроме описанных, есть еще дополнительный способ задания цели трассировки — опция `--user`. При этом всем существующим процессам, принадлежащим указанному пользователю, будут выставлены соответствующие флаги аудита (см. 10.1.6). Таким образом, будут трассироваться все действия пользователя. Режим полезен, когда необходимо создать профиль, разрешающий пользователю выполнять целый набор сложных действий и трассировать каждую программу в отдельности затруднительно.

Если используется режим `--audit-trace`, то в системе не должно быть сторонних процессов, на которых установлены флаги аудита. Иначе в профиль может попасть информация, порожденная сторонними процессами. В режиме трассировки всех процессов указанного пользователя достаточно, чтобы в системе не было других процессов с установленными флагами аудита и принадлежащих этому пользователю.

`otrace` по умолчанию записывает в профиль информацию только о тех действиях процесса (`open()`, `execve()`), которые прошли успешно. Однако для большей универсальности можно использовать опцию `--trace-failed`, которая позволит записать в профиль также информацию и о неудачных попытках. Это полезно, например, если процесс пытается открывать конфигурационные файлы. В момент трассировки файл может не существовать, но впоследствии он может появиться.

Параметры команды приведены в таблице 42.

Таблица 42

Параметр	Описание
`-h, --help`	Вывести справку и выйти
`-s, --silent`	Не выводить информационные сообщения

Окончание таблицы 42

Параметр	Описание
`-o , --output=`	Записать результаты трассировки в указанный файл. По умолчанию — `stdout`
`-k , --kiosk-dir=`	Указать путь к каталогу с профилями киоска. Используется в операции `--`. По умолчанию — `/etc/parsec/kiosk-profiles`
`-p , --pid=`	Трассировать процесс с указанным идентификатором, а также все порожденные им процессы
`-u , --user=`	Указать имя пользователя. Используется совместно с `--audit-trace` или `--merge`
`-t, --trace`	Использовать для трассировки процессов команду `strace`. Не может быть использована совместно с `--audit-trace`
`-a, --audit-trace`	Использовать для трассировки процессов подсистему аудита PARSEC. Не может быть использована совместно с `--trace`
`-m, --merge`	Объединять все права доступа, указанные каким-либо способом в единый поток с уникальными записями. Права доступа могут быть указаны в явном виде, в виде профилей, в виде профиля пользователя и профиля, используемого по умолчанию (`/etc/parsec/kiosk-profiles/default`)
`-f, --trace-failed`	Учитывать неудачные попытки вызова `open()` и `execve()`. Права доступа к этим файлам будут заданы согласно параметрам, с которыми процесс пытается получить доступ к ним
`-e , --mask=`	Указать маску киоска. Это позволяет обрабатывать данные согласно этой маске и учитывать только те файлы, права доступа к которым будут действительно замаскированы. По умолчанию маска равна 7

Примеры:

1. Запустить и трассировать процесс `ls /` с помощью команды `strace`. Записать результат в файл `/tmp/ls_trace`

```
otrace --trace -o /tmp/ls_trace ls /
```

2. Трассировать запущенные процессы пользователя `ttt` и все вновь порожденные ими процессы с помощью подсистемы аудита PARSEC. Отслеживать также информацию о неудачных попытках открытия файлов и запуска процессов. Результаты трассировки вывести в `stdout`

```
otrace --audit-trace -u ttt -f
```

3. Объединить содержимое профиля пользователя `ttt`, профиля с именем `ls` из каталога `/etc/parsec/kiosk-profiles` и добавить права на чтение и выполнение файла `/usr/bin/example`. Предполагать, что системная маска киоска равна 3 и, соответственно, учитывать только те файлы, для которых права доступа должны включать права на запись или выполнение

```
otrace --merge --mask=3 -u ttt ls "/usr/bin/example r-x"
```

15.4. fly-admin-kiosk

Кроме средств для работы в режиме командной строки, в распоряжении администратора имеется графическая утилита `fly-admin-kiosk`, которая может быть использована для настройки и управления режимом киоска.

Описание утилиты см. в электронной справке.

ВНИМАНИЕ! Перед использованием утилиты необходимо сделать резервную копию всех системных профилей, находящихся в каталоге `/etc/parsec/kiosk-profiles`, и системного профиля `fly-dm` в каталоге `/etc/parsec/kiosk`. Системные профили, устанавливаемые по умолчанию, находятся в пакете `parsec-kiosk`.

Переместить системный профиль `fly-dm` из каталога `/etc/parsec/kiosk` в каталог `/etc/parsec/kiosk-profiles`, выполнив команду:

```
mv /etc/parsec/kiosk/fly-dm /etc/parsec/kiosk-profiles/fly-dm
```

Создать пользовательский профиль `fly-dm` в каталоге `/etc/parsec/kiosk`, выполнив команду:

```
echo fly-dm > /etc/parsec/kiosk/fly-dm
```

Отредактировать содержимое файла `/etc/parsec/kiosk-profiles`, заключив в кавычки имена файлов:

```
"/lib64/ld-linux-x86-64.so.2" r-x
"/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2" r-x
```

Далее можно использовать утилиту `fly-admin-kiosk`.

15.5. Настройка режима киоска для пользователя

15.5.1. Для разрешения входа пользователя в режиме текстовый консоли необходимо, используя механизм `sudo`, от имени пользователя `root` создать файл профиля для пользователя `имя_пользователя` в каталоге `/etc/parsec/kiosk` и добавить в файл `имя_пользователя` перечень профилей, необходимых для разрешения входа пользователя, выполнив команды:

```
echo default > /etc/parsec/kiosk/имя_пользователя
echo bash >> /etc/parsec/kiosk/имя_пользователя
```

15.5.2. Для добавления разрешенных для пользователя команд необходимо:

1) войти в систему от имени учетной записи пользователя на одной консоли (например, `tty1`);

2) войти в систему от имени учетной записи администратора на другой консоли (например, `tty2`), используя команду `sudo -s`, переключиться в сессию суперпользователя `root` и запустить протоколирование действий пользователя, выполнив команду:

```
otrace -a -o /etc/parsec/kiosk-profile/новый_файл_профиля -f --mask=3
```

```
    -u имя_пользователя
```

3) дождаться перезапуска сервиса `parlogd`;

4) перейти на консоль пользователя и выполнить все команды, которые должны быть разрешены в дальнейшем;

5) завершить сеанс пользователя;

6) в консоли пользователя `root` остановить трассировку, нажав клавишу **<Enter>**;

7) подключить полученный новый профиль к пользователю, выполнив команду:

```
echo новый_файл_профиля >> /etc/parsec/kiosk/имя_пользователя
```

8) в консоли пользователя `root` включить режим киоска и перезагрузить ОС, выполнив команды:

```
echo 0003 > /etc/parsec/kiosk_mask
reboot
```

15.5.3. Для разрешения входа пользователя в графическом режиме необходимо:

1) проверить наличие профиля `fly-dm` в файле `/etc/parsec/kiosk/fly-dm`;

2) подключить профиль `fly` к профилю пользователя `имя_пользователя`, выполнив от имени пользователя `root` команду:

```
echo fly >> /etc/parsec/kiosk/имя_пользователя
```

15.5.4. Для создания профиля `fly` необходимо:

1) войти в систему в режиме текстовой консоли от имени администратора, используя команду `sudo -s`, переключиться в сессию суперпользователя `root` и остановить работу `fly-dm`, выполнив команду:

```
service fly-dm stop
```

2) проверить наличие логической ссылки `/usr/bin/x-session-manager`, выполнив команду:

```
ls -l /usr/bin/x-session-manager
```

3) удалить указанную логическую ссылку и создать другую с суффиксом `.orig`, выполнив команды:

```
rm /usr/bin/x-session-manager
ln -s /etc/alternatives/x-session-manager
    /usr/bin/x-session-manager.orig
```

4) создать текстовый файл `/usr/bin/x-session-manager` со следующим содержимым:

```
#!/bin/bash
/usr/sbin/otrace -t -f -o /test/fly --mask=3
    /usr/bin/x-session-manager.orig $@ &
sleep 20
/usr/bin/fly-wmfunc FLYWM_EXIT
```

5) изменить права на созданный файл, выполнив команду:

```
chmod 777 /usr/bin/x-session-manager
```

6) установить пакет strace, выполнив команду:

```
apt-get install starce
```

7) создать каталог /test с правами 777, выполнив команду:

```
mkdir -m 777 /test
```

8) запустить fly-dm, выполнив команду:

```
service fly-dm start
```

9) выполнить вход пользователя в графическом режиме и подождать 20 с до автоматического завершения сессии пользователя (можно открыть стартовую меню-панель Fly, терминал fly-term);

10) в сессии суперпользователя root проверить наличие профиля fly в каталоге /test, выполнив команду:

```
ls -l /test/fly
```

11) удалить файл /usr/bin/x-session-manager, выполнив команду:

```
rm /usr/bin/x-session-manager
```

12) создать логическую ссылку /usr/bin/x-session-manager, выполнив команду:

```
ln -s /etc/alternatives/x-session-manager /usr/bin/x-session-manager
```

13) открыть полученный профиль /test/fly в текстовом редакторе, удалить строки вида ... resumed ..., строки для /proc и строки, содержащие пути, начинающиеся с ./;

14) перенести профиль в каталог профилей, выполнив команду:

```
cp /test/fly /etc/parsec/kiosk-profiles/fly
```

15) подключить профиль fly к профилю пользователя, выполнив команду:

```
echo fly >> /etc/parsec/kiosk/имя_пользователя
```

16) включить режим киоска и перезагрузить ОС, выполнив команды:

```
echo 0003 > /etc/parsec/kiosk_mask
reboot
```

17) выполнить в графическом режиме вход в систему от имени пользователя;

18) если сессия не открывается, то проверить в консоли пользователя root содержимое файла /home/имя_пользователя/.xsession-errors, содержащего файлы, права на которые не выставлены в профиле fly, и добавить разрешение вручную. Пример строки из файла /home/имя_пользователя/.xsession-errors:

```
/etc/X11/fly-dm/Xsession : line 55 /bin/df: отказано в доступе
```

Пример добавления строки в профиль fly:

```
echo '"/bin/df" r-x' >> /etc/parsec/kiosk-profiles/fly
```

19) после изменений применить профиль к пользователю, выполнив команду:

```
mkiosk -u user
```

20) при наличии в файле /home/имя_пользователя/.xsession-errors строки, содержащей:

```
unable to create error file
```

проверить наличие в профиле fly команд chmod, rm, cp и файлов XErrorDB, /usr/lib/parsec/bin/x-session-manager и, при необходимости, добавить в профиль полные пути к ним.

15.5.5. Для добавления пользователю разрешения на выполнение конкретных программ (например, firefox) необходимо:

1) выключить режим киоска, выполнив команды:

```
echo 0000 > /etc/parsec/kiosk_mask
reboot
```

2) выполнить в графическом режиме вход в систему от имени пользователя;

3) войти в систему в режиме текстовой консоли от имени администратора, используя команду sudo -s, переключиться в сессию root и запустить протоколирование действий пользователя, выполнив команду:

```
otrace -a -o /etc/parsec/kiosk-profile/firefox -f --mask=3
    -u имя_пользователя
```

4) дождаться перезапуска сервиса parlogd;

5) перейти в графический интерфейс пользователя и запустить/завершить firefox;

6) завершить сеанс пользователя;

7) в консоли пользователя root остановить трассировку, нажав клавишу **<Enter>**;

8) подключить полученный профиль firefox для пользователя, выполнив команду:

```
echo firefox >> /etc/parsec/kiosk/имя_пользователя
```

9) в консоли пользователя root включить режим киоска и перезагрузить ОС, выполнив команды:

```
echo 0003 > /etc/parsec/kiosk_mask
reboot
```

15.6. Киоск Fly

Для ограничения возможности запуска программ локальным пользователям администратор может воспользоваться утилитой fly-admin-smc. Флаг «Режим киоска Fly» включается во вкладке «Киоск Fly» утилиты fly-admin-smc. Флаг включает кнопки-переключатели: «С рабочим столом» — режим киоска включается при работе с Рабочим столом, «Только с программой» — режим киоска включается при работе с выбранной про-

граммой.

Кроме графической утилиты можно в режиме командной строки перевести пользователя в режим киоск. Для этого необходимо хотя бы один раз авторизоваться этим пользователем.

Если пользователю предоставлен доступ к мандатным уровням и категориям отличным от 0, то надо хотя бы один раз авторизоваться на каждом уровне доступа. Далее необходимо запустить `fly-kiosk` для каждого уровня с соответствующим значением `--home` и без `--home`. Например, для пользователя user с уровнями 1 и 2 нужно запустить:

```
fly-kiosk -u john --action lock
fly-kiosk -u john --action lock --home /home/.pdp/user/l1i0c0x0t0x0
fly-kiosk -u john --action lock --home /home/.pdp/user/l2i0c0x0t0x0
```

Аналогично для выхода из режима киоска:

```
fly-kiosk -u john --action unlock
fly-kiosk -u john --action unlock --home /home/.pdp/user/l1i0c0x0t0x0
fly-kiosk -u john --action unlock --home /home/.pdp/user/l2i0c0x0t0x0
```

ВНИМАНИЕ! Настройка режима киоска с помощью программы `fly-admin-smc` возможно только для пользователя который не имеет доступа к мандатным уровням и категориям отличным от 0.

16. РЕЖИМ ЗАПРЕТА УСТАНОВКИ ИСПОЛНЯЕМОГО БИТА

В ОС реализован режим запрета установки исполняемого бита, обеспечивающий предотвращение несанкционированного создания пользователями или непреднамеренного создания администратором исполняемых сценариев для командной оболочки.

Включение режима запрета установки исполняемого бита осуществляется от имени учетной записи администратора через мезханизм `sudo` с использованием следующей команды:

```
sudo echo 1 > /parsecfs/nochmodx
```

Отключение режима запрета установки исполняемого бита осуществляется от имени учетной записи администратора через мезханизм `sudo` с использованием следующей команды:

```
sudo echo 0 > /parsecfs/nochmodx
```

ВНИМАНИЕ! В режиме запрета установки исполняемого бита данная операция запрещена для всех пользователей ОС, включая администратора и суперпользователя `root`. Установка пакетов программ, создающих в ФС файлы с исполняемым битом будет завершаться с ошибкой.

17. ЗАПУСК ОС

17.1. Запуск

При запуске ОС появляется окно, вид которого представлен на рис. 1.

Рис. 1 – Окно запуска ОС

В окне приведен перечень режимов загрузки ОС. Для загрузки в штатном режиме необходимо в перечне выбрать режим `generic`. Для загрузки в целях восстановления работоспособности ОС может использоваться режим `generic (single-user mode)`.

В процессе загрузки ОС осуществляется инициализация модуля ядра и запуск сервисов системы безопасности информации PARSEC.

По завершении загрузки ОС появляется окно, содержащее приглашение для ввода имени и пароля пользователя. В случае успешного прохождения идентификации и аутентификации пользователю будет предложено выбрать мандатный уровень и категории, которые будут использованы при создании сеанса пользователя в ОС. Выбор мандатной метки осуществляется в соответствии с текущими настройками максимального и минимального уровней и максимального и минимального наборов категорий, установленных для данного

пользователя.

Вид окна для выбора мандатной метки (уровня и категорий) представлен на рис. 2.

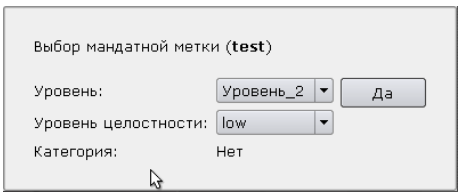

Рис. 2 – Окно выбора мандатной метки

17.2. Настройка параметров, необходимых для эксплуатации ОС

Перед началом эксплуатации ОС администратор безопасности должен обеспечить выполнение следующих условий:

1) механизм замкнутой программной среды должен быть настроен для работы в штатном режиме (см. 13.5);

2) с использованием средств управления дискреционными ПРД (см. 3.3) пользователям должен быть запрещен доступ к библиотеке `libpcprofile.so`;

3) с использованием средств управления мандатными ПРД (см. 4.4) всем отчуждаемым носителям, используемым на объекте эксплуатации, должны быть присвоены мандатные метки, соответствующие грифу обрабатываемой информации. Все отчуждаемые носители должны быть учтены режимно-секретным отделом организации, эксплуатирующей автоматизированную систему. Использование неучтенных отчуждаемых носителей должно быть запрещено;

4) с использованием средств управления дискреционными ПРД (см. 3.3) пользователям, не обладающим привилегиями администратора, должен быть запрещен запуск (использование) средств создания символических ссылок;

5) с использованием средств управления запуском сервисов должна быть отключена служба `gpm` для поддержки мыши в консольном режиме;

6) в случае разрешения интерактивного входа суперпользователя `root` для предотвращения подбора его пароля необходимо заблокировать возможность его удаленного входа в ОС посредством включения PAM-модуля `pam_securetty` в файл сценария `/etc/pam.d/common-auth`. Для этого необходимо в «Primary block» в указанном файле первой строкой добавить:

```
auth required pam_securetty.so
```

17.3. Проверка правильности запуска

В случае успешного входа пользователя в систему при наведении мыши на индикатор мандатных атрибутов в области уведомлений на панели задач (рис. 3) открывается всплывающее окно, которое отражает текущие мандатные атрибуты сеанса пользователя.

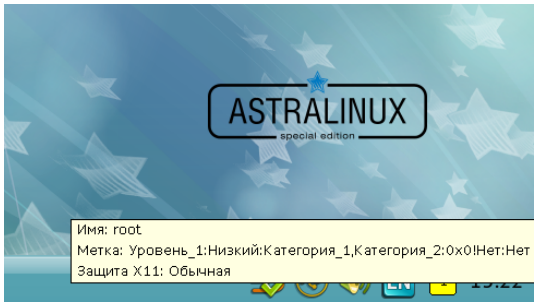

Рис. 3

При правильном запуске СЗИ данные атрибуты должны совпадать с введенными пользователем при входе в ОС.

Кроме того, для получения текущих мандатных атрибутов пользователя может быть использована консольная утилита `pdp-id`, описание которой приведено в `man pdp-id`.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

БД — база данных

ВФС — виртуальная файловая система

ЕПП — единое пространство пользователей

КСЗ — комплекс средств защиты

НСД — несанкционированный доступ

ОС — операционная система

ПО — программное обеспечение

ПРД — правила разграничения доступа

СЗИ — средства защиты информации

СЗФС — сетевая защищенная файловая система

СПО — специальное программное обеспечение

СУБД — система управления базами данных

ФС — файловая система

ЭЦП — электронная цифровая подпись

ACL — Access Control List (список контроля доступа)

ALD — Astra Linux Directory (единое пространство пользователей)

BIND — Berkley Internet Name Domain (служба доменных имен в сети Интернет)

CIFS — Common Internet File System (общий протокол доступа к файлам Интернет)

DAC — Discretionary Access Control (дискреционное управление доступом)

DHCP — Dynamic Host Configuration Protocol (протокол динамической конфигурации хоста)

DNS — Domain Name System (служба доменных имен)

FTP — File Transfer Protocol (протокол передачи файлов)

GID — Group Identifier (идентификатор группы)

IP — Internet Protocol (протокол Интернет)

IPC — InterProcess Communication (межпроцессное взаимодействие)

LDAP — Lightweight Directory Access Protocol (легковесный протокол доступа к сервисам каталогов)

MAC — Mandatory Access Control (мандатное управление доступом)

NFS — Network File System (сетевая файловая система)

NIS — Network Information Service (сетевая информационная служба)

NSS — Name Service Switch (служба для организации унифицированного доступа к информации о пользователях, группах, сетевых именах, службах и т.п. на основе конфигурации различных источников, таких как: локальные файлы, различные

средства сетевой идентификации (DNS, NIS), а также LDAP)

PAM — Pluggable Authentication Modules (подгружаемые аутентификационные модули)

PID — Process Identifier (идентификатор процесса)

RFC — Request for Comments (документ, содержащий технические спецификации и стандарты, применяемые в сети Интернет)

RSA — Rivest Shamir Adelman (алгоритм шифрования по схеме открытого ключа)

SQL — Structured Query Language (язык структурированных запросов)

TCP — Transmission Control Protocol (протокол передачи данных)

UDP — User Datagram Protocol (протокол пользовательских дейтаграмм)

UID — User Identifier (идентификатор пользователя)

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ документа	Входящий № сопроводительного докум. и дата	Подп.	Дата
	измененных	замененных	новых	аннулированных					