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 94

2010

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, контроль целостности КСЗ, генерация КСЗ, режим киоска.

Во второй части руководства приведено описание тестов КСЗ.

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

1.1. Состав КСЗ

В состав КСЗ входят следующие основные подсистемы:

– модуль подсистемы безопасности, входящий в состав ядра ОС;
– библиотеки;
– утилиты безопасности;
– подсистема протоколирования (регистрации);
– модули аутентификации;
– графическая подсистема;
– консольный вход в систему.

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 соответству-

ющей 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 правил разграничения доступа (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. Расширенные настройки безопасности

Режим «Расширенные настройки безопасности» может устанавливаться как при первоначальной установке ОС, так и в работающей системе (пакет `astra-safepolicy`).

Расширенные настройки безопасности предназначены для определения, настройки и установки общесистемных параметров и их ограничений для следующих подсистем ОС:

– файловой — ограничивается максимальный размер файла и максимальное число открытых файлов;

– управления процессами — определяется максимальное число процессов в системе;

– сетевой — настраивается сетевой фильтр `iptables`;

– безопасности — определяется минимальная длина паролей для пользователей в системе, настраиваются режимы запрета создания исполняемых файлов пользователем (режим `nochmodx`) и безопасного удаления файлов (режим `secrm`).

При установке пакета `astra-safepolicy` автоматически запускается программа конфигурации пакета. Далее приведено краткое описание пунктов с вопросами конфигуратора пакета `astra-safepolicy` и с возможными ответами на них. Пункты, отмеченные звездочкой (*), не показываются при первичной настройке пакета (в т. ч. при первоначальной установке системы), однако показываются при повторной конфигурации пакета:

– «Максимальный размер файла» — устанавливается максимально допустимый размер (в мегабайтах) создаваемого в системе файла. По умолчанию установлено значение 10000;

– «Делитель для расчета максимального числа процессов (*)» — необходимо указать, сколько в среднем требуется памяти (в килобайтах) процессу системы. Общий размер свободной памяти будет поделен на это значение для расчета максимального числа процессов. По умолчанию установлено значение 200;

– «Максимальное число открытых файлов (*)» — по умолчанию установлено значение 4096;

– «Подтверждение согласия на использование установленных системных ограничений» — вопрос «Вы согласны использовать эти системные ограничения?» необходим, поскольку некоторые предыдущие пункты могут отсутствовать (для упрощения первоначального конфигурирования). По умолчанию установлено значение «Нет» (т. е. ограничения не будут использоваться);

– «Минимальная длина пароля» — задается минимальная длина паролей для пользователей. Это значение используется в модуле безопасности `cracklib2`. Минимальная длина пароля должна быть не менее восьми и не более 32 символов. Модуль безопасности автоматически проверяет возможность подбора пароля по словарю. Для того чтобы данное ограничение не накладывалось, следует оставить поле пустым;

– «Точки монтирования, на которых следует включить режим secrm» — при включении режима `secrm` удаление файла, помеченного специальным атрибутом, из данной точки монтирования влечет за собой перезапись некоторым значением (обыч-

но нулем) тех секторов на диске, которые занимал данный файл. Это делается для безопасности, чтобы нельзя было, исследовав при помощи специального ПО содержимое незадействованных секторов жесткого диска, восстановить данный удаленный файл;

– «Запретить создание исполняемых файлов пользователем (режим nochmodx)?» — в результате наложения данного ограничения пользователь (не являющийся суперпользователем) не будет иметь возможности создавать исполняемые файлы. При этом предполагается, что все программы, которые пользователь будет запускать, либо уже содержатся в системе, либо устанавливаются администратором. По умолчанию установлено значение «Нет» (т. е. ограничение не будет использоваться);

– «Уровень безопасности брандмауэра: низкий, средний или высокий» — предоставляется возможность выбрать уровень ограничений на входящий сетевой трафик. Фактически при этом настраивается фильтр сетевых пакетов `iptables`.

Названия уровней безопасности означают следующее:

– «Низкий» — брандмауэр отключен, разрешен любой трафик по любым интерфейсам;

– «Средний» — запрет для всех входящих SYN-пакетов протокола TCP на порты 0–1023, 2049, 6000–6009, 71000, а также запрет для всех входящих UDP-пакетов на порты 0–1023, 2049. Входящий TCP-трафик на порт 22 (SSH) и весь трафик по интерфейсу `loopback` разрешен;

– «Высокий» — запрет всего TCP- и UDP-трафика, за исключением трафика по интерфейсу `loopback` и прохождения UDP-пакетов с адреса сервера имен на порт 52 (`domain`).

1.4. Средства организации единого пространства пользователей

Организация ЕПП обеспечивает:

– сквозную аутентификацию в сети;

– централизацию хранения информации об окружении пользователей;

– централизацию хранения настроек системы защиты информации на сервере;

– централизацию управления серверами DNS и DHCP.

Сетевая аутентификация и централизация хранения информации об окружении пользователя подразумевает использование двух основных механизмов: поддержки кросс-платформенных 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 источ-

ника данных для базовых системных сервисов на базе механизмов NSS и PAM. Подобный подход обеспечивает цнтрализацию хранения информации об окружении пользователей. В том числе информации, предназначенной для обеспечения мнадатного 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;

– членах привелигерованных групп, которые могут получать из базы данных службы каталогов LDAP определенную информацию о пользователях.

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

Более подробное описание ЕПП приведено в Руководстве администратора.

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` и т. д.). При использовании ЕПП аутентификация пользователей осуществляется централизованно по протоколу Kerberos. Переключение режима аутентификации осуществляется автоматически при запуске и остановке сервиса `aldd`.

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 используются следующие графические утилиты:

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

– `fly-admin-ald` («Администрирование ЕПП») — управление БД ALD.

Более подробное описание графических утилит 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 (см. 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.

3. ДИСКРЕЦИОННОЕ РАЗГРАНИЧЕНИЕ ДОСТУПА

3.1. Общие сведения

В ОС реализован механизм дискреционных ПРД, который 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 из девяти пунктов: по три вида доступа для трех групп — 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 текущими правилами (санкционировано).

Кроме общей схемы разграничения доступа, ОС поддерживает также ACL, с помощью которых можно для каждого объекта задавать права всех субъектов на доступ к нему.

Однако на практике ACL целесообразно использовать в исключительных ситуациях: для временного понижения прав или для временного предоставления доступа некоторым пользователям, а также при работе с очень важными файлами.

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.

3.2. Система привилегий

В ОС существует система привилегий, предназначенная для передачи отдельным пользователям прав выполнения определенных административных действий и являющихся стандартными для системы Linux (Linux-привилегии): `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.

Привилегии наследуются процессами от своих родителей. Процессы, запущенные от имени суперпользователя, независимо от наличия у них привилегий имеют возможность осуществлять все перечисленные привилегированные действия.

Кроме того, 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, OpenOffice, fly-scan).

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

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

Для управления Linux-привилегиями пользователей ЕПП используется графическая утилита fly-admin-ald («Управление локальной 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); выполнение (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 выполнение (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:

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

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

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

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

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

Маска эффективных прав.

`[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 пользователя, 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

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 соответственно дискреционные правила разграничения доступа к ней применены быть не могут. В PostgreSQL 8.4 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.

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_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.

3.5. Средства управления дискреционными ПРД к объектам БД СУБД PostgreSQL

Для управления дискреционными ПРД к объектам БД СУБД PostgreSQL используется графическая утилита `fly-admin-postgresql` («Администрирование СУБД PostgreSQL»). Более подробное описание утилиты см. в электронной справке.

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.

Кроме того, при принятии решения могут учитываться полномочия субъекта.

Правила принятия решения могут быть записаны следующим образом. Пусть контекст безопасности субъекта содержит уровень `L0` и категории `C0`, а мандатная метка объекта содержит уровень `L1` 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) категории `C0` меньше категорий `C1` (`C0<C1`), если все биты набора `C0` являются подмножеством набора бит `C1`;
4) категории `C0` равны категориям `C1` (`C0==C1`), если значения `C0` и `C1` совпадают;
5) операция записи разрешена, если `L0==L1` и `C0==C1`;
6) операция чтения разрешена, если `L0>=L1` и `C0>=C1`;
7) операция исполнения разрешена, если `L0>=L1` и `C0>=C1`.

В остальных случаях анализируются полномочия и тип мандатной метки. Тип метки может использоваться для того, чтобы изменять ее эффективное действие. Ненулевой тип метки может быть установлен только привилегированным процессом.

Механизм мандатного разграничения доступа затрагивает следующие подсистемы:

– механизмы IPC;
– стек TCP/IP (IPv4);
– слой ВФС;
– ФС Ext2/Ext3;
– 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.

4.2. Система привилегий

В ОС система привилегий Linux, предназначенная для передачи отдельным пользователям прав выполнения определенных административных действий, расширена следующими привилегиями, относящимися к системе PARSEC и обеспечивающими работу с механизмом мандатного разграничения доступа:

– PARSEC_CAP_SIG — позволяет посылать сигналы процессам, игнорируя дискреционные и мандатные права;

– PARSEC_CAP_SETMAC — разрешает изменить мандатную метку и установить другие привилегии;

– PARSEC_CAP_CHMAC — право менять мандатные метки файлов;

– PARSEC_CAP_AUDIT — право управления политикой аудита;

– PARSEC_CAP_READSEARCH — позволяет игнорировать мандатную политику при чтении и поиске файлов (но не при записи);

– PARSEC_CAP_PRIV_SOCK — позволяет создавать привилегированный сокет и менять его мандатную метку. Привилегированный сокет позволяет осуществлять сетевое взаимодействие, игнорируя мандатную политику;

– PARSEC_CAP_UPDATE_ATIME — право изменять время доступа к файлу;

– PARSEC_CAP_IGNMACLVL — право игнорировать мандатную политику по уровням;

– PARSEC_CAP_IGNMACCAT — право игнорировать мандатную политику по категориям;

– PARSEC_CAP_FILE_CAP — право устанавливать привилегии на файлы;

– PARSEC_CAP_CAP — разрешает устанавливать любой непротиворечивый набор привилегий для другого процесса;

– PARSEC_CAP_MAC_SOCK — возможность смены мандатной точки соединения.

Привилегии наследуются процессами от своих родителей. Процессы, запущенные от имени суперпользователя, независимо от наличия у них привилегий имеют возможность осуществлять все перечисленные привилегированные действия.

Кроме этих привилегий, в ОС имеется набор привилегий, являющихся стандартными для системы Linux (Linux-привилегии) (см. 3.2).

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

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.

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 с 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 (сервер LDAP, сервер DNS, сервер Kerberos и т. д.) необходимо обеспечить возможность их работы с клиентами, имеющими разный мандатный контекст безопасности.

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 локальной политикой безопасности») — управление протоколированием, привилегиями и мандатными атрибутами пользователей, работа с пользователями и группами.

Более подробное описание утилит см. в электронной справке.

Далее рассмотрены средства управления мандатными ПРД в режиме командной строки.

4.4.1. 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 7.

Таблица 7

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.

Пример

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

4.4.2. macid

Синтаксис:

```
macid [опции]
```

Команда `macid` выводит мандатные атрибуты сессии пользователя ОС.

Опции приведены в таблице 8.

Таблица 8

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`.

4.4.3. 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.

4.4.4. psmac

Синтаксис:

```
psmac [-d, --delete] [-n, --numeric] [-h, --help] [--version]
   [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 9.

Таблица 9

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

4.4.5. 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, которые сохраняются в файле `/etc/security/mac`.

Опции приведены в таблице 10.

Таблица 10

Опция	Описание
-d, --delete	Удалить строку пользователя из файла
-z, --zero	Обнулить значения уровней и категорий
-m, --maclabels	Установить допустимые мандатные уровни
-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.

4.4.6. 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 11.

Таблица 11

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

4.4.7. 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 12.

Таблица 12

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

4.4.8. 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).

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 13.

Таблица 13

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

4.4.9. 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 (запускать может пользователь, но только в пределах разрешенных 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.

Если происходит понижение мандатного уровня, то запуск команд с аргументами запрещен. Все аргументы будут игнорироваться при создании процесса. Это необходимо для предотвращения утечки информации на более низкие уровни секретности.

Команда `sumac` порождает новую сессию для текущего пользователя. Если пользователь пытается запустить графическое приложение, то для отображения окна новой сессии (с другими мандатными уровнем и категорией) на текущем рабочем столе `X` необходимо создать определенную запись в файле авторизации `X` (`.Xauthority`). В этом случае следует использовать опцию `-x`. Иначе запись в файле авторизации создана не будет и приложение не сможет отобразить свое окно на текущем рабочем столе `X`. Для консольных приложений опция `-x` не вызывает никакого действия.

Примеры:

1. Предположим существует пользователь, для которого максимально разрешенный мандатный уровень — 2, и он вошел в систему с уровнем 1. Для того чтобы выполнить команду `macid`, сменив мандатный уровень (в пределах максимально разрешенного) для вновь создаваемого процесса, необходимо выполнить команды:

```
$ sumac -o /tmp/macinfo -l 0 macid
$ sumac -o /tmp/macinfo -l 2 macid
```

Первая команда запустит процесс `macid` с мандатным уровнем 0, вторая с мандатным уровнем 2. Стандартный вывод при выполнении команды `macid` будет перенаправлен в файл `/tmp/macinfo`. После выполнения этих команд администратор может убедиться в правильности выполнения команды и в том, что мандатный уровень действительно менялся. Для этого ему необходимо посмотреть содержимое файлов `/var/private/tmp/0/0/macinfo` и `/var/private/tmp/2/0/macinfo`, соответственно, для команды, понижающей мандатный уровень, и команды, повышающей мандатный уровень. Там находится вывод команды `macid`. Так как содержимое общего каталога для хранения временных файлов `/tmp` (и некоторых других общих каталогов) не должно смешиваться для процессов с разной мандатной меткой, то физически временные файлы для процессов с различной мандатной меткой хранятся в `/var/private/уровень/категория` (категория указана в шестнадцатеричном виде). Пользователям нет необходимости знать об этой особенности. Они могут использовать привычное имя `/tmp` для хранения временных файлов. Трансляцию имени `/tmp` в реальное физическое имя используемого каталога система PARSEC произведет автоматически.

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

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

Опции приведены в таблице 14.

Таблица 14

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

4.4.10. 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. 4.4.8).

Опции приведены в таблице 15.

Таблица 15

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

4.5. Средства управления привилегиями пользователей и процессов

Для управления привилегиями файлов и процессов используется графическая утилита `fly-admin-smc` («Управление локальной 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.

```
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 16.

Таблица 16

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

Окончание таблицы 16

Опция	Описание
`-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

Команда `usercaps` должна использоваться только суперпользователем.

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 17.

Таблица 17

Опция	Описание
`-c, --capability (привилегии)`	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 18.

Таблица 18

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 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 использование 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.

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 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 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 8.4 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:

– 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;

– 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_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.

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;

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, обла-

дающих 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. опе-

рация 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.

4.7. Система привилегий СУБД PostgreSQL

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.

4.8. Средства управления мандатными ПРД к объектам БД СУБД PostgreSQL

Для управления мандатными ПРД к объектам БД СУБД PostgreSQL используется графическая утилита fly-admin-postgresql («Администрирование СУБД PostgreSQL»). Более подробное описание утилиты см. в электронной справке.

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 name SET MAC TO новое_значение_мандатной_метки
ALTER TABLE name 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 name ALTER COLUMN column_name
SET MAC TO новое_значение_мандатной_метки
ALTER TABLE name ALTER COLUMN column_name
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 объетка 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 name ENABLE COLUMN MACS
```

Выключается флаг командой:

```
ALTER TABLE name 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 name (
... -- список атрибутов
) 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 name (maclabel, ... список атрибутов)
VALUES (значение_мандатной_метки, ... значения атрибутов)
```

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

```
UPDATE name SET maclabel = новое_значение_мандатной_метки
```

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

```
SELECT maclabel FROM name
```

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 test TO stdout WITH MACS
COPY test 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.

4.9. Средства управления привилегиями СУБД PostgreSQL

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

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.

Кроме атрибутов, определяемых администратором СУБД или конфигурацией ОС, существует большое количество атрибутов, значение которых может установить для себя сам пользователь.

Описание опций приведено в документации на СУБД PostgreSQL: на английском языке в пакете postgresql-doc-8.4, на русском языке — в пакете postgresql-doc-ru-8.4.

4.10. 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, доступ 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 названного механизма снижает скорость выполнения операций удаления и усечения размера файла. Механизм является настраиваемым и позволяет обеспечить работу ФС ОС (Ext2/Ext3) в одном из следующих режимов:

– данные любых удаляемых/урезаемых файлов в пределах заданной ФС предварительно очищаются маскирующей последовательностью;

– данные ФС освобождаются обычным образом (без предварительного маскирования).

Режим работы ФС может быть выбран администратором ОС и задан в виде параметра монтирования ФС (см. руководство `man` по `mount`). Для применения параметров очистки памяти, необходимо в конфигурационном файле `/etc/fstab` для соответствующего раздела ФС на котором требуется очищать блоки памяти при их освобождении (например, `/dev/sda1`) в список параметров монтирования добавить параметр `secdel`.

Пример

```
/dev/sda1 /home ext3 alc,defaults,secdel 0 2
```

Установка параметра монтирования для очистки блоков памяти при их освобождении может быть выполнения с использованием графической утилиты `fly-admin-smc`, запущенной от имени суперпользователя `root`. Более подробное описание утилиты `fly-admin-firewall` приведено 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 мо-

жет быть выполнения с использованием графической утилиты `fly-admin-smc`, запущеной от имени суперпользователя `root`. Более подробное описание утилиты `fly-admin-firewall` приведено 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, поддерживаемым 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

Сервер печати CUPS 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 суперпользователя. Описание данной утилиты приведено в `man cupsctl`.

Для разрешения серверу CUPS удаленно принимать задания и команды, необходимо от имени суперпользователя выполнить следующие команды:

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

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

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

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

```
lpadmin -p <имя_принтера> -o printer-op-policy=parsec
lpadmin -p <имя_принтера> -o mon-printer-mac-max=Lmax:Cmax
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 суперпользователя. Если ЕПП используется, то команды выполняются от имени пользователя, входящего в группу `lpadmin` в БД службы каталогов LDAP.

Маркировка осуществляется на основе модифицируемых файлов шаблонов:

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

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

– /usr/share/cups/fonarik/fonarik.template — описание элементов маркировки, проставляемых на обороте последней страницы при количестве экземпляров меньшем либо равном пяти;

– /usr/share/cups/fonarik/fonarik_gt_5.template — описание элементов маркировки, проставляемых на обороте последней страницы при количестве экземпляров больше пяти;

– /usr/share/cups/fonarik/fonarik.defs — описание положения элементов маркера на странице.

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.

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

8. ЗАЩИТА ВОДА/ВЫВОДА ИНФОРМАЦИИ НА ОТЧУЖДАЕМЫЙ ФИЗИЧЕСКИЙ НОСИТЕЛЬ

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

– как блоковые или символьные устройства ввода/вывода;

– как блоковое 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 процессе создания учетных записей пользователей. ФС Ext2/Ext3 в полной мере поддерживают мандатный механизм разграничения доступа и могут использоваться для отчуждения на внешние носители секретной информации. В остальных случаях мандатными ПРД разрешено отчуждение только несекретной информации.

9. СОПОСТАВЛЕНИЕ ПОЛЬЗОВАТЕЛЯ С УСТРОЙСТВОМ

ОС обеспечивает ввод-вывод информации на запрошенное пользователем устройство как для произвольно используемых пользователем 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`.

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

10.1. Средства управления протоколированием

Для управления протоколированием имеется ряд графических утилит, которые могут быть использованы для настройки и управления СЗИ:

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

– `fly-admin-viewaudit` («События аудита») — 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 19.

Таблица 19

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 20.

Таблица 20

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 21.

Таблица 21

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 22.

Таблица 22

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 23.

Таблица 23

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 24.

Таблица 24

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

10.2. Регистрация событий в СУБД PostgreSQL

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 можно задать 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:

– на добавление/изменение/удаление пользователей и групп (`SUBJECT`) соответствует нулевой бит (шестнадцатеричное значение — 1);

– на изменение конфигурации, влияющей на доступ к данным (запрос на изменение значения переменной `ac_session_maclabel`) (`CONFIGURATION`) соответствует первый бит (шестнадцатеричное значение — 2);

– на изменение прав доступа к объектам БД (`RIGHTS`) соответствует второй бит (шестнадцатеричное значение — 4);

– на выборку информации из БД (`SELECT`) соответствует четвертый бит (шестнадцатеричное значение — 10);

– на добавление информации в БД (`INSERT`) соответствует пятый бит (шестнадцатеричное значение — 20);

– на изменение информации в БД (`UPDATE`) соответствует шестой бит (шестнадцатеричное значение — 40);

– на удаление информации из БД (`DELETE`) соответствует седьмой бит (шестнадцатеричное значение — 80);

– на очистку данных (`TRUNCATE`) соответствует восьмой бит (шестнадцатеричное значение — 100);

– на задание колонки таблицы в качестве внешнего ключа (`REFERENCES`) соответствует десятый бит (шестнадцатеричное значение — 400);

– на добавление триггера к таблице (TRIGGER) соответствует одиннадцатый бит (шестнадцатеричное значение — 800);

– на запуск хранимой процедуры или триггера (EXECUTE) соответствует двенадцатый бит (шестнадцатеричное значение — 1000);

– на использование объекта БД (USAGE) соответствует тринадцатый бит (шестнадцатеричное значение — 2000);

– на создание объектов в БД (CREATE) соответствует шестнадцатый бит (шестнадцатеричное значение — 10000);

– на создание временных объектов в БД (CREATE) соответствует семнадцатый бит (шестнадцатеричное значение — 20000);

– на удаление объектов БД (DROP) соответствует восемнадцатый бит (шестнадцатеричное значение — 40000);

– на изменение объекта БД (ALTER) соответствует девятнадцатый бит (шестнадцатеричное значение — 80000).

Информация о соединении пользователей с БД (CONNECT) и разъединении с ней (DISCONNECT) регистрируется всегда.

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 выполнения программы `fsck` достаточно для обеспечения целостности ФС.

В случае обнаружения серьезных повреждений ФС программа `fsck` 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`.

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

```
cd  /mnt/
mkdir  hard
mount  /dev/hxx  /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.

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 операционной системы, а также от выдвигаемых требований по надежности сохранения жизнеспособности системы. Элементы системы резервного копирования должны включать необходимое оборудование, носители резервных копий и специальное программное обеспечение. В качестве оборудования для резервного копирования в ОС может использоваться достаточно широкий набор аппаратных средств, начиная от обычного дисковода 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 организации;

– получения данных для восстановления после аварий.

Система резервного копирования обязательно является составной частью любого продуманного плана восстановления системы. В случае широкомасштабных катастроф данные доставляются из архивов, сохраняемых в отдельном помещении или здании.

Основная идея резервного копирования — создание копий критической части содержания резервируемой системы. Основными исключениями, как правило, не входящими в процедуру резервного копирования функционирующей ОС, являются:

– ФС `/proc`, т. к. она содержит только данные, которые ОС генерирует во время работы;

– ФС `/mnt /media`, поскольку в нее монтируются сменные носители — DVD-диски,

дискеты и т. п.;

– сетевые каталоги — смонтированная NFS, Samba и прочие виды сетевых данных;

– программное обеспечение, которое может быть легко повторно установлено. Здесь надо иметь в виду, что оно может иметь конфигурационные файлы, которые необходимо резервировать, чтобы не выполнять работы по их настройке позже.

11.2. Средства резервного копирования и восстановления ОС

Команды `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.).

11.2.1. Набор программ Bacula

`Bacula` представляет собой набор программ, позволяющий системному администратору управлять процессами резервного копирования и восстановления данных, а также проверять резервные копии, в том числе 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 организацию базы резервных данных.

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

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: на английском языке содержится в пакете `postgresql-doc-8.4`, на русском языке — в пакете `postgresql-doc-ru-8.4`.

11.4. Средства резервного копирования и восстановления СУБД PostgreSQL

11.4.1. 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 используется, 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 25.

Таблица 25

Опция	Описание
`-f, --file=FILENAME`	Имя выходного файла

Окончание таблицы 25

Опция	Описание
`-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

Опции установки соединения приведены в таблице 26.

Таблица 26

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)

Существует большое количество опций, управляющих выводом содержимого, описание которых приведено в документации на СУБД PostgreSQL: на английском языке содержится в пакете `postgresql-doc-8.4`, на русском языке — в пакете `postgresql-doc-ru-8.4`.

11.4.2. 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 27.

Таблица 27

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`.

Существует большое количество опций, управляющих выводом содержимого, описание которых приведено в документации на СУБД PostgreSQL: на английском языке содержится в пакете `postgresql-doc-8.4`, на русском языке — в пакете `postgresql-doc-ru-8.4`.

Если не используется `-f/--file`, SQL-скрипт будет направлен в стандартный вывод.

11.4.3. 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. 11.4.1).

Синтаксис:

```
pg_restore [ОПЦИЯ]... [ФАЙЛ]
```

Опции общего характера приведены в таблице 28.

Таблица 28

Опция	Описание
`-d, --dbname=ИМЯ`	Подсоединиться к указанной БД

Окончание таблицы 28

Опция	Описание
`-f, --file=FILENAME`	Имя выходного 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

Существует большое количество опций, управляющих восстановлением резервной копии, описание которых приведено в документации на СУБД PostgreSQL:на английском языке содержится в пакете `postgresql-doc-8.4`, на русском языке — в пакете `postgresql-doc-ru-8.4`.

12. КОНТРОЛЬ ЦЕЛОСТНОСТИ КСЗ

Для обеспечения контроля целостности (в т. ч. контроля целостности КСЗ) в ОС реализованы:

– средство подсчета контрольных сумм файлов и оптических дисков (см. 12.1);

– средство контроля соответствия дистрибутиву (см. 12.2);

– средства регламентного контроля целостности (см. 12.3);

– средства создания замкнутой программной среды (см. 12.4).

Для решения задач контроля целостности предназначена библиотека `libgost`, в которой для вычисления контрольных сумм реализована функция хэширования в соответствии с ГОСТ Р 34.11-94. Названная библиотека используется в средствах подсчета контрольных сумм файлов и оптических дисков, контроля соответствия дистрибутиву и регламентного контроля целостности.

В ОС реализован механизм, обеспечивающий проверку неизменности и подлинности загружаемых исполняемых файлов формата ELF. Проверка производится на основе контрольных сумм файлов, вычисляемых в соответствии с ГОСТ Р 34.11-94, и ЭЦП, реализованной в соответствии с ГОСТ Р 34.10-2001, 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 запуска.

12.1. 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 [-b buffer_size] [input file name][-o output file name]
    [-d device[iso file]]
```

Опции приведены в таблице 29.

Таблица 29

Опция	Описание
`-b`	Устанавливает размер блоков, которыми будет считываться файл
`input file name`	Задает имя файла для подсчета контрольной суммы (по умолчанию — стандартный поток ввода)
`-o`	Задает имя файла для вывода контрольной суммы (по умолчанию — стандартный поток вывода)

Окончание таблицы 29

Опция	Описание
`-d`	Задает имя файла устройства чтения оптических дисков (файла с образом оптического диска) для подсчета контрольной суммы

Далее приведен пример подсчета контрольной суммы оптического диска:

```
gostsum -d /dev/cdrom
```

12.2. Средство контроля соответствия дистрибутиву

Средство контроля соответствия дистрибутиву предоставляет возможность для контроля соответствия объектов файловой системы ОС дистрибутиву ОС. Для обеспечения контроля целостности объектов ФС ОС (в т. ч. СЗИ) в состав дистрибутива входит файл `gostsums.txt` со списком контрольных сумм 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.

12.3. 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 программных средств 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 34.11-94.

12.3.1. 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 34.11-94 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
/lib/modules      PARSEC
/sbin             PARSEC
/lib64/security   PARSEC
/lib/security     PARSEC
/usr/sbin         PARSEC
/etc/security     PARSEC
/etc/pam.d        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.

12.4. 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 (12.4.2).

Механизм контроля целостности исполняемых файлов и разделяемых библиотек формата ELF при запуске программы на выполнение реализован в модуле ядра ОС digsig_verif.

Модуль digsig_verif является невыгружаемым модулем ядра Linux, который может функционировать в одном из следующих режимов:

– исполняемым файлам и разделяемым библиотекам с неверной ЭЦП, а также без ЭЦП загрузка на исполнение запрещается (штатный режим функционирования);

– 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);

– ЭЦП при загрузке исполняемых файлов и разделяемых библиотек не проверяется (отладочный режим для тестирования СПО).

12.4.1. Настройка модуля digsig_verif

Для изменения режима функционирования модуля digsig_verif необходимо отредактировать файл /etc/digsig/digsig_initramfs.conf.

Для использования отладочного режима для тестирования СПО необходимо установить для параметра DIGSIG_LOAD_KEYS значение 0:

```
DIGSIG_LOAD_KEYS=0
```

Для использования режима для проверки ЭЦП в СПО необходимо установить для параметра DIGSIG_LOAD_KEYS значение 1:

```
DIGSIG_LOAD_KEYS=1
DIGSIG_ENFORCE=0
```

Для использования штатного режима функционирования необходимо установить следующие значения параметров:

```
DIGSIG_LOAD_KEYS=1
DIGSIG_ENFORCE=1
```

Управление модулем digsig_verif осуществляется через интерфейс sysfs с ис-

пользованием файлов:

– /sys/digsig/enforce — проверка и переключение режима работы;

– /sys/digsig/key — файл загрузки главного ключа;

– /sys/digsig/additional — файл загрузки дополнительных ключей.

Каждый дополнительный ключ, использованный для подписывания СПО (12.4.2), необходимо скопировать в каталог /etc/digsig/, например:

```
# cp /<каталог>/<файл ключа> /etc/digsig/
```

Для загрузки дополнительного ключа модулем digsig_verif необходимо отредактировать файл скрипта /etc/digsig/digsig_initramfs, добавив в него после строки:

```
cat /etc/digsig/key_for_signing.gpg > /sys/digsig/additional 2>/dev/null
```

строку следующего вида:

```
cat /etc/digsig/<файл ключа> >> /sys/digsig/additional 2>/dev/null
```

Проверка режима работы выполняется командой:

```
#cat /sys/digsig/enforce
```

12.4.2. Подписывание СПО

В модуле ядра digsig_verif реализован механизм, позволяющий использовать несколько ключей при подписывании файлов формата ELF.

Порядок использования ключей для digsig_verif:

1) главный ключ записывается в /sys/digsig/key;

2) дополнительные ключи записываются в /sys/digsig/additional.

Все дополнительные ключи должны быть подписаны главным ключом.

Для создания дополнительных ключей используется GNU Pivacy Guard. Модифицированный GnuPG выводит ГОСТ в списке доступных алгоритмов. Для получения списка доступных алгоритмов необходимо выполнить команду:

```
# gpg --version
gpg (GnuPG) 1.4.9
Copyright (C) 2008 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
Supported algorithms:
Pubkey: RSA, RSA-E, RSA-S, ELG-E, DSA, GOST_R34.10-2001
Cipher: 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH
Hash: MD5, SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224,
GOST_R34.11-94
Compression: Uncompressed, ZIP, ZLIB, BZIP2
```

Далее приведен пример создания дополнительного ключа и его использования для подписывания СПО.

Пример

1) создается ключевая пара ГОСТ Р 34.10-2001 и сохранятся в каталоге ~/.gnupg. Для создания ключевой пары необходимо выполнить приведенную далее команду с последующим выбором в меню gpg алгоритма ГОСТ.

```
keys@debian:~$ gpg --gen-key
gpg (GnuPG) 1.4.9; Copyright (C) 2008 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: directory ‘/home/keys/.gnupg’ created
gpg: new configuration file ‘/home/keys/.gnupg/gpg.conf’ created
gpg: WARNING: options in ‘/home/keys/.gnupg/gpg.conf’ are not yet active /
   during this run
gpg: keyring ‘/home/keys/.gnupg/secring.gpg’ created
gpg: keyring ‘/home/keys/.gnupg/pubring.gpg’ created
Please select what kind of key you want:
(1) DSA and Elgamal (default)
(2) DSA (sign only)
(5) RSA (sign only)
(10) GOST R34.10-2001
Your selection? 10
GOST keypair will have 256 bits.
Please specify how long the key should be valid.
0 = key does not expire
<n> = key expires in n days
<n>w = key expires in n weeks
<n>m = key expires in n months
<n>y = key expires in n years
Key is valid for? (0) 0
Key does not expire at all
Is this correct? (y/N) y

You need a user ID to identify your key; the software constructs the user ID
from the Real Name, Comment and Email Address in this form:
"Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: Test GOST R 34.10-2001 Secondary Key
Email address: test@gost.secondary.key
Comment:
You selected this USER-ID:
```

```
"Test GOST R 34.10-2001 Secondary Key <test@gost.secondary.key>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
You need a Passphrase to protect your secret key.

You don’t want a passphrase - this is probably a *bad* idea!
I will do it anyway. You can change your passphrase at any time,
using this program with the option "--edit-key".

gpg: /home/keys/.gnupg/trustdb.gpg: trustdb created
gpg: key BA703834 marked as ultimately trusted
public and secret key created and signed.
gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u
pub 256E/BA703834 2009-06-18
Key fingerprint = F802 B362 47F8 268D 6679 6E1E F69A 9FA1 BA70 3834
uid Test GOST R 34.10-2001 Secondary Key <test@gost.secondary.key>

keys@debian:$ gpg --export "Test GOST R 34.10-2001 Secondary Key
<test@gost.secondary.key>" > /tmp/secondary_gost_key.gpg
```

2) владелец главного ключа заверяет ключ пользователя.

```
#
# gpg --import /tmp/secondary_gost_key.gpg
gpg: key BA703834: public key "Test GOST R 34.10-2001 Secondary Key
<test@gost.secondary.key>" imported
gpg: Total number processed: 1
gpg: imported: 1
# gpg --sign-key "Test GOST R 34.10-2001 Secondary Key
<test@gost.secondary.key>"

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0 valid: 1 signed: 4 trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: depth: 1 valid: 4 signed: 0 trust: 4-, 0q, 0n, 0m, 0f, 0u
pub 256E/BA703834 created: 2009-06-18 expires: never
usage: SC
trust: unknown
validity: unknown
[ unknown] (1). Test GOST R 34.10-2001 Secondary Key
<test@gost.secondary.key>
pub 256E/BA703834 created: 2009-06-18 expires: never
usage: SC
```

```
trust: unknown
validity: unknown
Primary key fingerprint: F802 B362 47F8 268D 6679 6E1E F69A 9FA1 BA70 3834

Test GOST R 34.10-2001 Secondary Key <test@gost.secondary.key>

Are you sure that you want to sign this key with your
key "Test GOST <test@gost.key>" (C0585133)

Really sign? (y/N) y

# gpg --export "Test GOST R 34.10-2001 Secondary Key
<test@gost.secondary.key>" > /tmp/secondary_gost_key_signed.gpg
```

3) пользователь подписывает на данном ключе некоторый файл формата ELF. Для этого используется утилита bsign:

```
keys@debian:~$ bsign --sign test_elf
0l 52-> 52 0l 52-> 52 header
52 l 256-> 308 52 l 256-> 308 entry program
308 l 3356-> 3664 from file
3664 l 327-> 3991 3664 l 340-> 4004 section names
3991 l 1-> 4005 from file
3992 l 1480-> 5472 4005 l 1520-> 5525 entry section
5472 l 2142-> 7667 from file
7614 l 0-> 7614 7667 l 512-> 8179 signature

Enter pass phrase:
```

4) для проверки правильности ЭЦП файла формата ELF используется утилита bsign:

```
keys@debian:~$ bsign -w test_elf
version: 1
id: bsign v1.0
hash: cf37 e2f4 6999 28d6 d486 67f6 1ba6 92ed 5173 570c 6b05 66c7 1605 7a16
5b8d 3df8
signature_size: 96
signature:
88 5e 04 00 22 5e 00 06 05 02 4a 3b 20 ec 00 0a
09 10 f6 9a 9f a1 ba 70 38 34 eb 33 00 ff 5b 29
aa fb 48 4f c8 86 f1 74 c0 9a c9 dc 4d 64 5b 92
c5 cb 77 c8 82 df 33 16 f7 19 7e 7d 29 94 00 ff
52 99 5a 33 86 f1 3f ed 98 3a c9 38 96 be ca dd
f9 c1 64 eb 69 06 4f 8c 81 f2 9e 2a 76 ec e7 8f
signer: BA703834EB3300FF
```

```
timestamp: 19 Jun 2009 09:23:56 (1245389036)
bsign: good hash found in 'test_elf'.
```

5) дополнительный ключ пользователя, подписанный на главном ключе, копируется в каталог /etc/digsig/, под управлением которого будет функционировать СПО пользователя. Для загрузки дополнительного ключа пользователя модулем digsig_verif необходимо отредактировать файл скрипта /etc/digsig/digsig_initramfs (см. 12.4.1);

6) подписанный файл формата ELF может выполняться:

```
keys@debian:~$ ./test_elf
hello world!
keys@debian:~$
```

13. ГЕНЕРАЦИЯ КСЗ

Генерация КСЗ осуществляется в одном из двух следующих режимов:

– режим ЕПП;

– локальный режим.

Для использования в ОС 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 соответсвующие действия по настройке (см. документ РУСБ.10015-01 95 01 "Операционная система специального назначения "Astra Linux Special Edition. Руководство администратора").

Переключение в режим ЕПП осуществляется командой:

```
ald-client start
```

Переключение в локальный режим осуществляется командой:

```
ald-client stop
```

После определения режима работы КСЗ ОС для завершения процедуры генерации КСЗ необходимо выполнить следующие действия:

– создать набор мандатных уровней;

– создать набор мандатных категорий;

– создать набор служебных пользователей, наличие которых неободимо для функционирования защищенных комплексов программ СУБД, гипертекстовой обработки данных, электронной почты и программ маркировки и учета печатных документов из состава ОС;

– установить для привилегии для служебных пользователей;

– установить параметры аудита (протоколирования событий) в ОС;

После выпонения 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-ald` («Администрирование ЕПП») — управление БД ALD (управление протоколированием, привилегиями и мандатными атрибутами пользователей, работа с пользователями и группами и т.д в ЕПП) или при помощи утилиты командой строки `ald-admin` (см. `man ald-admin`).

В локальном режиме указанные действия выполняются при помощи графической утилиты `fly-admin-smc` («Управление локальной политикой безопасности») — управление протоколированием, привилегиями и мандатными атрибутами пользователей, работа с пользователями и группами и т.д в ОС или при помощи соответсвующих утилит командой строки (см. 4.4).

Более подробное описание графических утилит см. в электронной справке.

14. РЕЖИМ КИОСКА

14.1. Общие сведения

Режим киоска служит для ограничения прав пользователей в системе. Степень этих ограничений 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.

14.2. 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 выполнение 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` (14.3).

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 30.

Таблица 30

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--"
```

14.3. 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. 14.2).

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. И поэтому для того, чтобы облегчить администрирование, есть возможность использования профилей 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 хра-

нятся в каталоге /etc/parsec/kiosk и заполняются вручную на основе готовых профилей 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. 14.2).

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 31.

Таблица 31

Параметр	Описание
-h, --help	Вывести справку и выйти

Окончание таблицы 31

Параметр	Описание
`-s, --silent`	Не выводить информационные сообщения
`-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"
```

14.4. 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 приведено в электронной справке.

15. ЗАПУСК ОС

15.1. Запуск

При запуске ОС появляется окно, вид которого представлен на рис. 1.

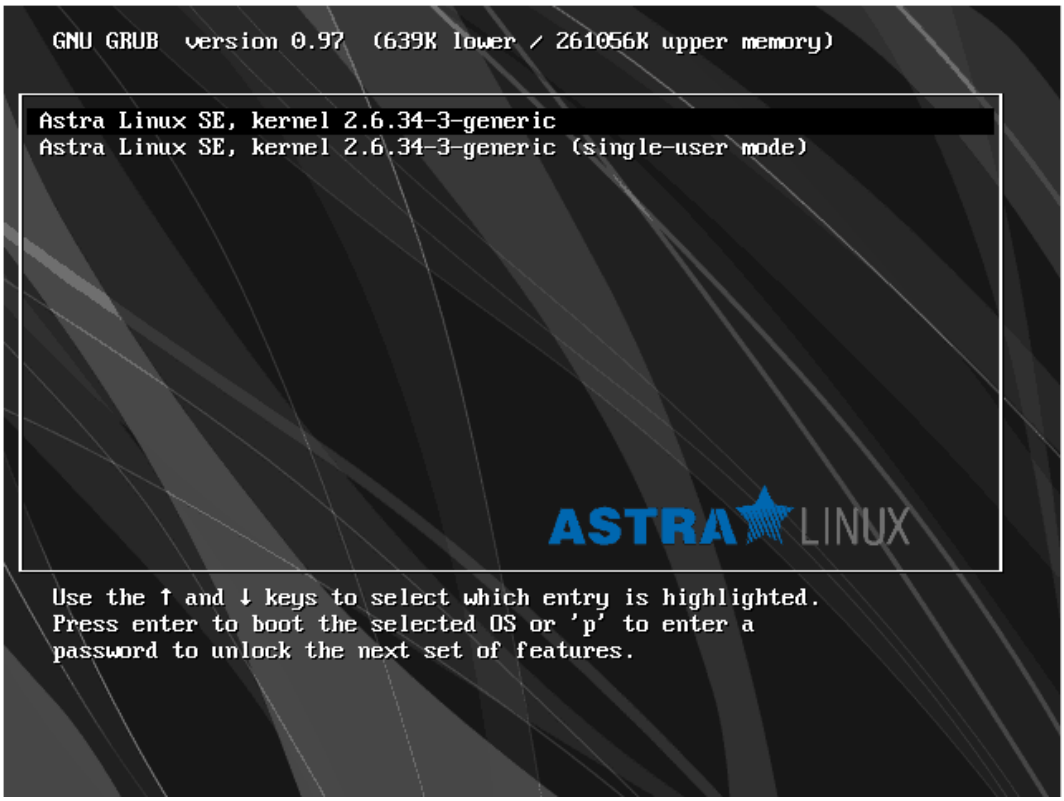

Рис. 1 – Окно запуска ОС

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. При установленном пакете `ald-client` в настройках, выполненных запуском скрипта `ald-client` с параметром `start`, запуск СЗИ будет выполнен для работы в режиме ЕПП. В противном случае запуск КСЗ будет осуществлен в штатном режиме.

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.

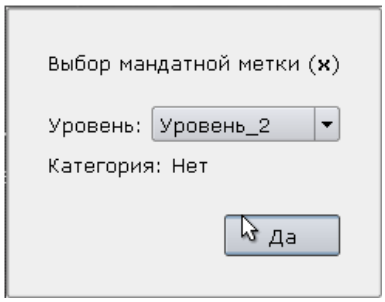

Рис. 2 – Окно выбора мандатной метки

15.2. 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. 12.4);

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 (например, графической утилиты `fly-admin-runlevel`) должна быть отключена служба `gpm` для поддержки мыши в консольном режиме.

15.3. Проверка правильности запуска

В случае успешного входа пользователя в систему при наведении мыши на индикатор мандатных атрибутов в панели задач (рис. 3), появляется всплывающее окно, которое отражает текущие мандатные атрибуты сеанса пользователя. При правильном запуске СЗИ данные атрибуты должны совпадать с введенными пользователем при входе в ОС.

Рис. 3

Кроме того, для получения текущих мандатных атрибутов пользователя может быть использована консольная утилита `macid`.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

БД — база данных
ВФС — виртуальная файловая система
ЕПП — единое пространство пользователей
КСЗ — комплекс средств защиты
НСД — несанкционированный доступ
ОП — оперативная память
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)
PAM — Pluggable Authentication Modules (подгружаемые аутентификационные модули)
PID — Process Identifier (идентификатор процесса)
RFC — Request for Comments (документ, содержащий технические спецификации и стан-

дарты, 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					