Программные средства, входящие в состав ОС СН, могут быть поделены на 3 основных группы:

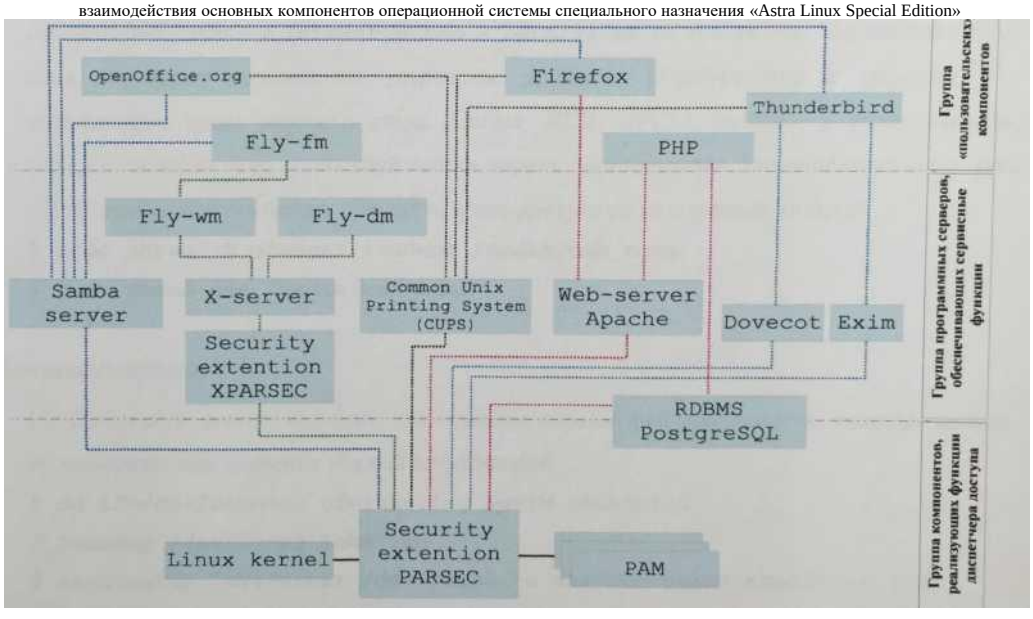
- группа компонентов, реализующих функции диспетчера доступа - программные средства, реализующие диспетчеры доступа (модули ядра, защищенная СУБД);

- группа программных серверов, обеспечивающих сервисные функции - программные средства, реализующие запуск процессов в контексте безопасности пользователя (сервисы, осуществляющие обработку запросов пользователей, поступающих из процессов, запущенных в различных мандатных контекстах, и средства инициализации графической сессии пользователя);

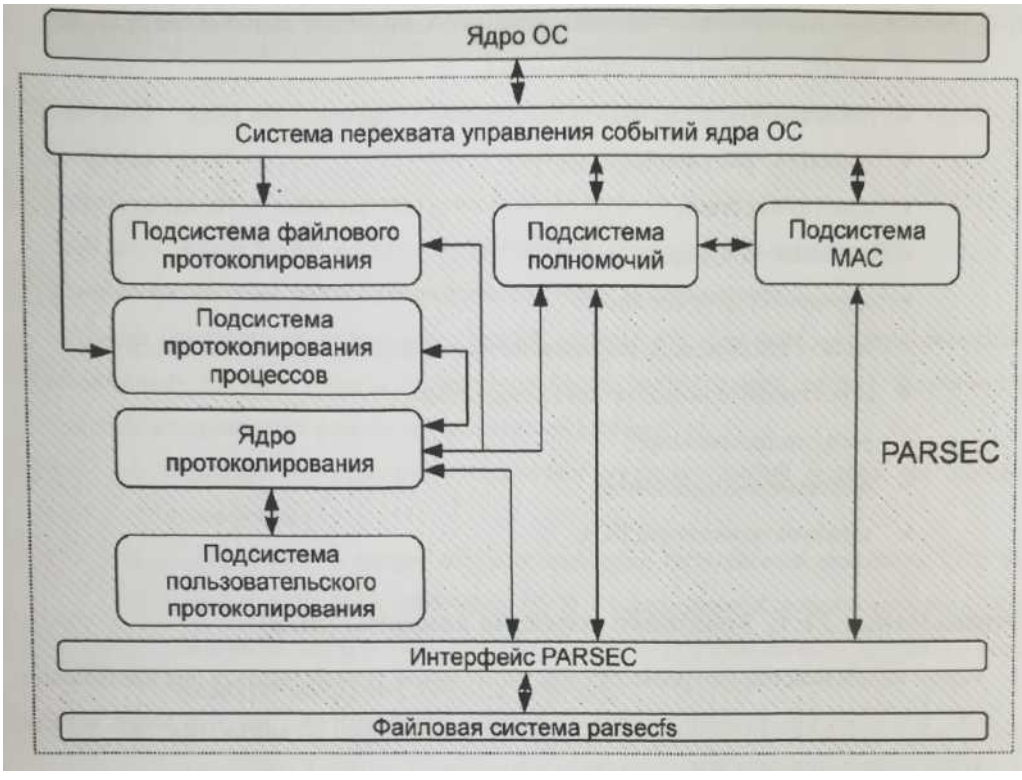
- группа «пользовательских» компонентов - программные средства, реализующие функциональные задачи пользователя (офисные средства, Web-браузер и т.д.).

Классификация программных средств приведена ниже.

ЛОГИЧЕСКАЯ СХЕМА
взаимодействия основных компонентов операционной системы специального назначения «Astra Linux Special Edition»



Логическая схема информационных связей подсистемы безопасности PARSEC показана ниже.



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

В состав КСЗ входят следующие основные подсистемы:

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

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.

Упражнение 21.1. Комплекс средств защиты (КСЗ)

1. Режим «Расширенные настройки безопасности» может устанавливаться при инсталляции ОС или в уже работающей системе. Проверьте, установлен ли у вас этот пакет, если не установлен, установите его.

   `# aptitude search astra-safeplicy`

   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 - 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 secdel).

2. Запустите утилиту Управление политикой безопасности, через меню Пуск -> Настройки
3. Выберите в левом окне открывшейся утилиты Настройки безопасности -> Глобальный профиль. Попробуйте изменить о настройки (все настройки изучались ранее через командную строку)
4. Выберите в левом окне открывшейся утилиты Политики паролей -> Глобальная политика

   Попробуйте изменить о настройки (все настройки изучались ранее через командную строку)

   Выберите в левом окне открывшейся утилиты Группы, а затем Пользователи. Измените настройки

5. Запустите утилиту Сетевой фильтр, через меню Пуск -> Настройки

   Разрешите работу служб, которые Вы настраивали ранее в упражнениях к курсу Фактически при этом настраивается фильтр сетевых пакетов iptables «Уровень безопасности брандмауэра: низкий, средний или высокий» — предоставляется возможность выбрать уровень ограничений на входящий сетевой трафик. Фактически при этом настраивается фильтр сетевых пакетов iptables:

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

   ii. — «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;

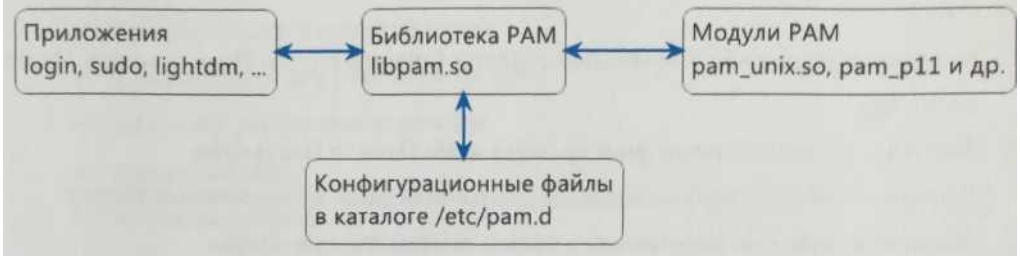
   ш. «Высокий» — запрет всего TCP- и UDP-трафика, за исключением трафика по интерфейсу loopback и прохождения UDP-пакетов с адреса сервера имен на порт 52 (domain).

Модуль 22. Идентификация и аутентификация

Функция идентификации и аутентификации пользователей в ОС основывается на использовании механизма PAM. PAM представляют собой набор разделяемых библиотек - «модулей», с помощью которых системный администратор может организовать процедуру аутентификации пользователей прикладными программами. Каждый модуль реализует свой 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.

Общая схема работы PAM



Сильно упрощенная схема аутентификации в приложении, использующем PAM, выглядит следующим образом:

- Приложение инициализирует библиотеку PAM (libpam.so)
- PAM в соответствии с конфигурационным файлом для приложения обращается к требуемым модулям
- Модули выполняют возложенные на них действия
- Приложению возвращается результат операции

Модули PAM классифицируются по типу модуля. Каждый модуль должен выполнять функции хотя бы одного из четырех типов:

- **Модуль аутентификации** используется для аутентификации пользователей или создания и удаления учетных данных.
- **Модуль управления учетными записями** выполняет действия, связанные с доступом, истечением учетных данных или записей, правилами и ограничениями для паролей и т. д.
- **Модуль управления сеансами** используется для создания и завершения сеансов.
- **Модуль управления паролями** выполняет действия, связанные с изменением и обновлением пароля.

PAM обеспечивает различные функциональные возможности, такие как: аутентификация с однократной регистрацией, управление доступом и другие. Их реализация обеспечивается различными модулями:

pam_access обеспечивает управление входом в систему в виде протоколируемой службы при помощи имени пользователя и домена в зависимости от правил, указанных заранее в файле /etc/security/access.conf.

pam_cracklib проверяет пароли на соответствие правилам для паролей.

pam_env sets/unsets устанавливает и сбрасывает переменные среды из файла /etc/security/pam_env_conf.

pam jichug выполняет отладку PAM. pam_deny блокирует модули PAM. pam_echo выводит сообщения.

раш_схес выполняет внешнюю команду. pam_ftp модуль для анонимного доступа.

pam_localuser проверяет наличие имени пользователя в файле /etc/passwd.

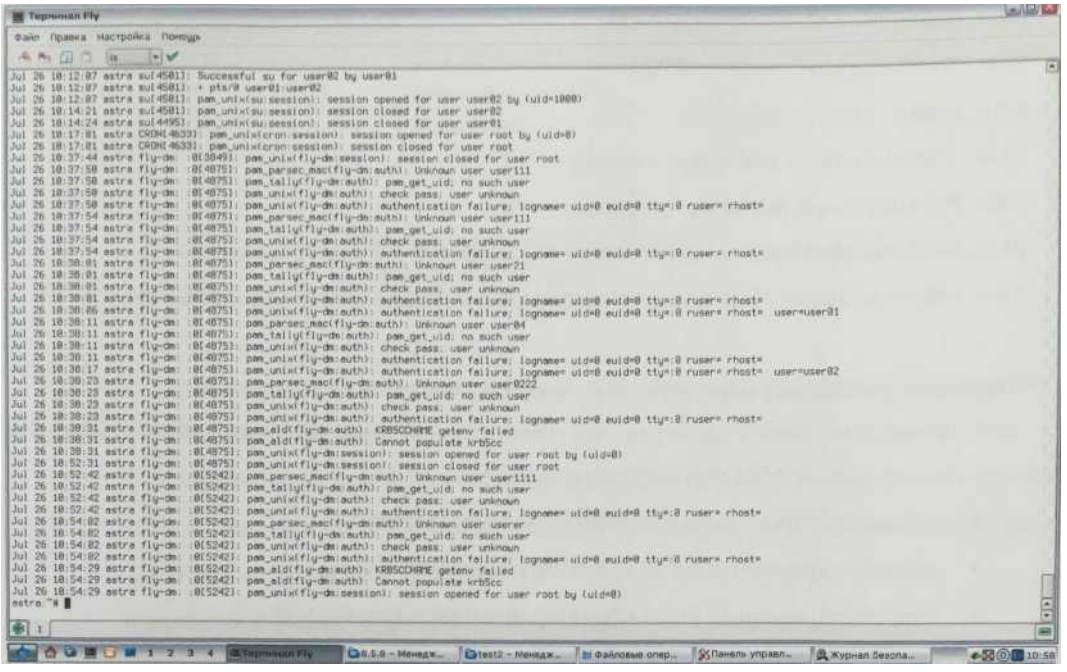
pain unix выполняет обычную аутентификацию на основе пароля из файла /etc/passwd.

Проверка реализации идентификации и аутентификации пользователей Для проведения данной проверки под учетными записями пользователей произведены попытки регистрации на АРМ используя (Рисунок 1.):

- незарегистрированный идентификатор пользователя;
- зарегистрированный идентификатор и неверный пароль;
- зарегистрированный идентификатор пользователя и верный пароль.

При проведении проверки получены следующие результаты:

- после ввода пользователем подлинного (зарегистрированного) идентификатора и пароля ему предоставляется доступ к защищаемым ресурсам в соответствии с установленными правилами разграничения доступа;
- после ввода пользователем незарегистрированных идентификаторов и/или паролей ему отказывается в доступе к защищаемым ресурсам.



Проверка фиксации в регистрационном протоколе всех попытки ложной аутентификации

При проведении проверки по данному пункту были выполнены действия, указанные далее по тексту.

1. Произведена попытка входа на АРМ, используя зарегистрированный идентификатор и неверный пароль.
2. Под учетной записью root выполнена команда userlog. (Рисунок 2.).

```
[u] 'Tue Apr 23 16:20:30 2013' '/usr/bin/fly-dm' <4264,3003,0,0,0> [f] auth("fly-dm","test")
[u] 'Tue Apr 23 16:22:51 2013' '/usr/bin/fly-dm' <4264,3003,0,0,0> [f] auth("fly-dm","test")
[u] 'Tue Apr 23 16:23:04 2013' '/usr/bin/fly-dm' <4264,3003,0,0,0> [s] auth("fly-dm","root")
```

При проведении проверки получены следующие результаты:

- после ввода пользователем подлинного (зарегистрированного) идентификатора и неверного пароля ему отказано в предоставлении доступа к защищаемым ресурсам;
- в регистрационном журнале, вызванном командой userlog, отображаются все попытки неверной аутентификации.

Проверка возможности установки порога на максимальное число неверных попыток аутентификации для локальных пользователей

При проведении проверки по данному пункту были выполнены действия, указанные далее по тексту.

1. Выполнена регистрация на АРМ 1 с использованием учетной записи администратора (root).
2. На рабочем столе выполнен запуск приложения «FLY терминал».
3. Изменен параметр pam.tallyso per user deny=10 на pam.tally_so per_user deny=3 (Рисунок 3).

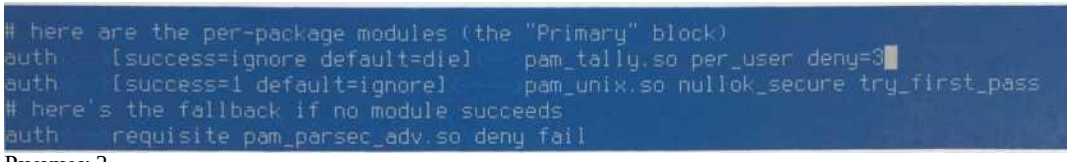


Рисунок 3

4. Произведена попытка аутентификации под учетной записью зарегистрированного пользователя, используя неверный пароль более 3 раз (Рисунок 4).

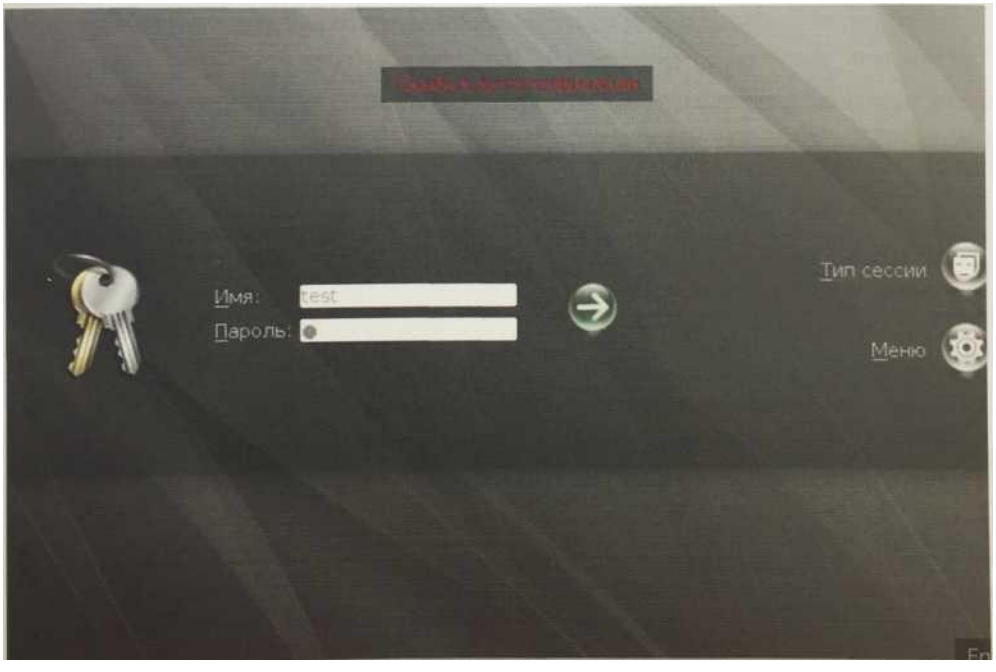


Рисунок 4

□ после 3 попыток аутентификации с использованием неверного пароля учетная запись пользователя блокируется (при вводе логина и верного пароля появляется сообщение «Ошибка аутентификации»).

Проверка возможности установки порога на максимальное число неверных попыток аутентификации для пользователей ALD

При проведении проверки по данному пункту были выполнены действия, указанные далее по тексту.

1. Выполнена регистрация на АРМ 1 с использованием учетной записи администратора (root).
2. На рабочем столе выполнен запуск приложения «FLY терминал».
3. Изменен параметр pam.tally_so per user deny=10 на pam.tally_so per_user deny=3 (Рисунок 5).

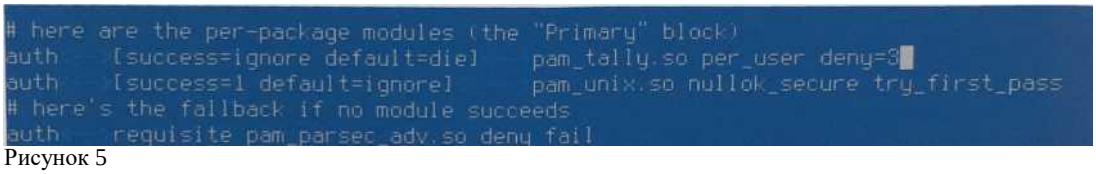


Рисунок 5

4. Произведена попытка аутентификации под учетной записью зарегистрированного пользователя, используя неверный пароль более 3 раз (Рисунок 6).

Рисунок 6

При проведении проверки получены следующие результаты:

□ после 3 попыток аутентификации с использованием неверного пароля учетная запись пользователя блокируется (при вводе логина и верного пароля появляется сообщение «Ошибка аутентификации»).

Проверка автоматического инициирования периодической смены паролей пользователями.

При проведении проверки по данному пункту были выполнены действия, указанные далее по тексту.

1. Выполнена регистрация на АРМ 1 с использованием учетной записи администратора (root).
2. На рабочем столе выполнен запуск приложения «FLY терминал».
3. Изменено число дней работоспособности пароля для тестового пользователя test посредством редактирования файла /etc/shadow.
4. Изменено системное время. Выставлено системное время, при котором пароль пользователя test является просроченным.
5. Выполнена попытка входа в систему с учетной записью пользователя test (Рисунок 7).

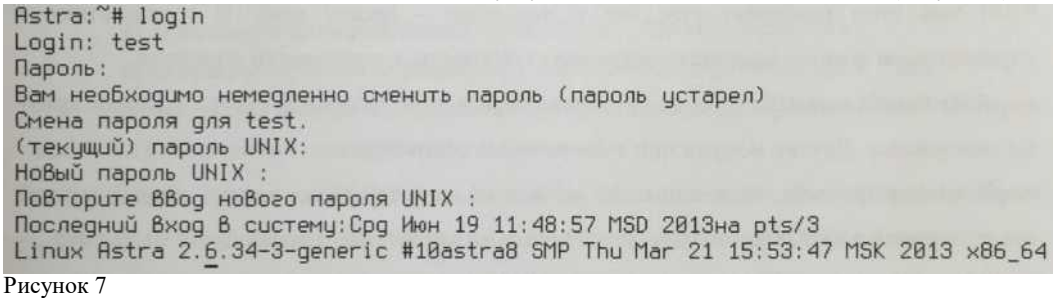


Рисунок 7

При проведении проверки получены следующие результаты:

После прохождения аутентификации пользователем test ОС СН автоматически генерирует запрос о смене пароля пользователя.

Упражнение 22. Идентификация и аутентификация

1. Посмотрите список стандартных модулей в системе:

```
% find / -name pam_*.so
```

2. Для примера, посмотрите на абстрактный файл конфигурации для приложения login. Попробуйте поменять параметры и посмотрите изменения

```
# PAM configuration for login
```

```
auth requisite pam_securetty.so auth required pam_nologin.so auth required
pam_env.so auth required pam_unix.so nullok account required pam_unix.so session
required pam_unix.so session optional pam_lastlog.so

password required pam_unix.so nullok obscure min=:4 max=8
```
Каждая строчка конфига

записывается в виде

```
<тип модуля> <управляющий флаг> <путь к библиотеке> <параметры>
```

Тип модуля соответствует обозначениям самих модулей (т.е.

```
auth/account/session/passwd)
```

Управляющий флаг указывает критичность модуля для успешного выполнения операции. Флаг может принимать следующие значения: requisite (необходимый), required (требуемый), sufficient (достаточный) и optional (необязательный).

Путь к библиотеке задает путь до файла модуля. По умолчанию они ищутся в `/lib/security/`

Параметры задают список аргументов, которые будут переданы модулю.

Таким образом, мы получаем стек модулей, каждый из которых выполняет свое действие. PAM при этом разбирает стек как и положено - сверху вниз. В соответствии с управляющим флагом задаются следующие требования к успешности операции: requisite (необходимый): если модуль стека вернет отрицательный ответ, то запрос сразу же отвергается. Другие модули при этом не будут выполнены.

required (требуемый): если один или несколько модулей стека вернут отрицательный ответ, все остальные модули будут выполнены, но запрос приложения будет отвергнут, sufficient (достаточный): если модуль помечен как достаточный и перед ним ни один из необходимых или достаточных модулей не возвратил отрицательного ответа, то все оставшиеся модули в стеке игнорируются, и возвращается положительный ответ, optional (дополнительный): если в стеке нет требуемых модулей, и если ни один из достаточных модулей не возвратил положительного ответа, то хотя бы один из дополнительных модулей приложения или службы должен вернуть положительный ответ

Модуль 23. Мандатное разграничение доступа

Механизм контроля мандатного разграничения доступа реализован, как и механизм дискреционного разграничения доступа, в ядре ОС (модуль parsec.ko). 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. При принятии решения учитывается наличие PARSEC-привилегий субъекта.

1) уровень ьо меньше уровня и (ьо<Ы), если численное значение L0 меньше численного значения L1;
2) уровень L0 равен уровню LI (LO—L1), если численные значения L0 и Ы совпадают;
3) уровень целостности iLO меньше уровня iLl (iLOciLl), если численное значение 1L0 меньше численного значения 1Ы;
4) уровень целостности iLO равен уровню целостности iLl (iL0==iLl), если численные значения iLO и iLl совпадают;
5) категории CO меньше категорий C1 (C0<C1), если все биты набора CO являются подмножеством набора бит C1;
6) категории CO равны категориям C1 (C0«»C1), если значения CO и C1 совпадают;
7) операция записи разрешена, если L0=L1, iL0>*=iLl и co=ci;
8) операция чтения разрешена, если L0>=L1, C0>=C1, ViLO, ViLl;
9) операция исполнения разрешена, если L0>=L1 и C0>=C1, ViLO, ViLl.

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

При анализе критических маршрутов было установлено, что с каждым субъектом и объектом связаны мандатный контекст безопасности и мандатная метка, соответственно и механизм мандатного разграничения доступа затрагивает следующие подсистемы:

— механизмы IPC;
— стек TCP/IP;
— ФС Ext2/Ext3;
— сетевая ФС CIFS;
— ФС proc, tmpfs.

Модуль 31. Основы работы с СУБД PostgreSQL

PostgreSQL - свободная объектно-реляционная система управления базами данных (СУБД). PostgreSQL базируется на языке SQL и поддерживает многие из возможностей стандарта SQL:2003 (ISOAEC 9075).

Основные плюсы PostgreSQL:

- поддержка БД практически неограниченного размера;
- мощные и надёжные механизмы транзакций и репликации;
- расширяемая система встроенных языков программирования: в стандартной поставке поддерживаются PL/pgSQL, PL/Perl, PL/Python и PL/Tcl; дополнительно можно использовать PL/Java, PL/PHP, PL/Py, PL/R, PL/Ruby, PL/Scheme и PL/sh, а также имеется поддержка загрузки C-совместимых модулей;
- наследование;
- легкая расширяемость.

Входящий в состав СУБД PostgreSQL набор программных средств можно разделить на следующие классы:

— управление БД;

— выполнение запросов пользователя;

— оптимизация производительности;

— обеспечение средств копирования и восстановления.

Работа с СУБД требует установки соединения с сервером БД, что при использовании клиентских утилит командной строки обеспечивается заданием свойств соединения с помощью аргументов (ОПЦИИ) командной строки:

-h, —host=HOSTNAME - указывает имя сервера БД или каталог сокетов UNIX, если начинается с символа «/»

-p, —port=PORT - указывает порт сервера БД или расширение имени сокета UNIX, по которым сервер принимает соединения

-и, —username=usERNAME - указывает имя пользователя для установки соединения **-w, —no-password** - подавление запроса пароля пользователя. В случае, когда установка соединения с сервером требует ввода пароля, а пароль недоступен, например из файла .pgpass, попытка установки соединения завершается ошибкой. Опция полезна при выполнении пакетов заданий или скриптов

-W, —password - принудительный запрос пароля при установке соединения.

При отсутствии перечисленных аргументов используются переменные окружения (PGDATABASE, PGHOST, PGPORT, PGUSER), определяющие параметры соединения по умолчанию.

Управление базами данных: создание и удаление БД, управление пользователями и процедурными языками.

Создание кластера БД состоит из создания каталогов для хранения данных БД, создания разделяемых таблиц

системного каталога (таблиц, относящихся ко всему кластеру БД, а не к конкретной БД), и создания БД template 1 и postgres. При создании в дальнейшем новых БД в них копируется содержимое БД template 1. (все, что установлено в БД template 1, автоматически будет скопировано в каждую создаваемую в дальнейшем БД.) БД postgres является БД по умолчанию для использования пользователями, утилитами и сторонними приложениями. Создание кластера выполняется администратором на сервере с помощью утилиты **initdb.**

Создание и удаление баз данных. Для создания новой БД используется утилита **createdb,** для удаления — утилита **dropdb.** По умолчанию владельцем новой БД становится пользователь, выполняющий команду. В тоже время в качестве владельца новой БД может быть указан другой пользователь с помощью опции -o, если выполняющий команду пользователь обладает соответствующими привилегиями. При этом удаление может выполнить только суперпользователь или владелец БД.

Обобщенный способ вызова заключается в передаче опций и имени БД. При этом используются правила установки соединения, рассмотренные выше: **createdb [ОПЦИИ]… [БАЗА_ДАННЫХ] [ОПИСАНИЕ] dropdb [ОПЦИИ]… [БАЗА_ДАННЫХ]**

Управление пользователями. В СУБД PostgreSQL для управления правами на доступ к БД используется концепция ролей. Под ролью понимается пользователь или группа пользователей БД, в зависимости от параметров роли. Роли могут являться владельцами объектов БД (например, таблиц) и могут назначать привилегии на управление объектами для других ролей, имеющих доступ к данным объектам. Кроме того, существует возможность предоставления членства в роли для другой роли, что позволяет членам роли использовать привилегии, назначенные роли, членами которой они являются. Таким образом, концепция ролей объединяет концепции «пользователи» и «группы».

Корректная работа с СУБД предполагает использование механизма ЕПП, что подразумевает использование в качестве пользователей СУБД пользователей домена ЕПП.

Для создания нового пользователя или роли используется утилита **createuser,** для удаления — **dropuser.**

Только суперпользователи и пользователи с привилегией

CREATEROLE могут создавать и удалять пользователей и роли. Удалять суперпользователя может только суперпользователь.

Синтаксис:

createuser [ОПЦИИ]…

[РОЛЬ] dropuser [ОПЦИИ]…

[РОЛЬ]

Использование процедурных языков. СУБД PostgreSQL предоставляет пользователям возможность создавать хранимые процедуры (функции) и триггеры для обработки данных, хранящихся в БД. Для этого могут использоваться следующие процедурные языки: PL/Perl,

PL/pgSQL, PL/Python и PL/Tcl.

Для возможности использования конкретного процедурного языка его необходимо установить в конкретную БД.

Для установки поддержки процедурного языка в БД используется утилита **createlang**, для удаления поддержки языка из БД — **droplang.**

Синтаксис:

createlang [ОПЦИИ] . . . ЯЗЫК

[БАЗА_ДАННЫХ] droplang [ОПЦИИ]... ЯЗЫК

[БАЗА_ДАННЫХ]

Несмотря на то, что поддержка процедурного языка может быть выполнена непосредственно некоторыми SQL-командами (например, DROP LANGUAGE), рекомендуется использовать данные утилиты, т. к. они осуществляют необходимые проверки.

Выполнение запросов. Взаимодействие пользователя с СУБД в основном осуществляется с помощью прикладного ПО, созданного для решения конкретных прикладных задач.

В то же время в состав СУБД входят средства интерактивного взаимодействия с пользователем. Для этого предлагается консольная утилита **psql** (интерактивный терминал) и утилита администрирования с визуальным пользовательским интерфейсом **fly-admin- postgres.**

Интерактивный терминал. Утилита **psql** является интерактивным клиентом PostgreSQL и позволяет интерактивно набирать запросы, отправлять их серверу и получать результаты. Так же ввод может осуществляться из файла. В дополнение утилита поддерживает метакоманды и некоторые возможности командной оболочки для облегчения создания скриптов и автоматизации широкого круга задач. Синтаксис: **psql [ОПЦИИ]... [БАЗА_ДАННЫХ [ПОЛЬЗОВАТЕЛЬ]]**

Утилита **psql** является клиентским приложением PostgreSQL. Для установки соединения требуется указание БД, имя и номер порта сервера и имя пользователя, под которым устанавливается соединение. Существует возможность указать эти параметры с помощью аргументов командной строки **-d, -h, -p** и **-и,** соответственно. Если аргумент не соответствует ни одной из опций, он воспринимается как имя БД (или имя пользователя, если имя БД уже было получено).

Если значения по умолчанию не верны, существует возможность их переопределения установкой переменных окружения PGDATABASE, PGHOST, PGPORT и/или PGUSER в соответствующие значения. Так же удобно использовать файл ~/ **.pgpass** для устранения необходимости регулярного ввода пароля.

Альтернативным путем задания параметров соединения является строка соединения, используемая вместо имени БД. Этот механизм предоставляет широкие возможности по управлению установкой соединения. Например:

$ psql "service=myservice sslmode=require"

При невозможности установки соединения в силу тех или иных причин (например, недостаток прав доступа, сервер не запущен, и т.п.) утилита **psql** возвращает ошибку и завершает работу.

При нормальном функционировании **psql** выводит приглашение с именем БД, с которой в настоящее время установлено соединение, за которым следует =>. Например:

$ psql

testdb psql

(x.x.O)

Наберите "help" для справки.

testcLb=>

После приглашения пользователь имеет возможность ввода SQL-команд. Обычно введенный запрос отсылается серверу после ввода завершающего символа «;». Перевод строки не завершает команду. Команда может быть записана в несколько строк для лучшего восприятия. Если команда была отослана серверу и выполнена без ошибок, на экран выводится результат ее выполнения.

В случае ввода строки, начинающейся с не заключенного в кавычки символа \, она воспринимается как метакоманда и обрабатывается непосредственно утилитой **psql.** Подобные команды делают утилиту более удобной для администрирования и создания скриптов.

В процессе запуска **psql** пытается прочитать и выполнить команды из общесистемного файла **psqlrc** и пользовательского файла ~/ **.psqlrc** (см. . . . **. /share/psqlrc. sample** для примера общесистемного файла). Он может быть использован для настройки параметров клиента или сервера (используя команды **set** или **SET) .**

История команд сохраняется в файле ~/ **.psql_history.**

Примеры: 1. Разбиение команды при вводе в несколько строк (следует обратить внимание на изменение приглашения при этом)

```
^escaJD=> CREATE TABLE my_table ( testdb(>
first integer not null default 0, testdb(>
second text) testdb-> ;
CREATE TABLE
```

2. Просмотр определения таблицы `testdb=> \d my_table`

```
                Таблица "my_table"
 Атрибут | Тип |            Модификатор
----------------+------------ +----------------------------
 first      |  integer |  not null default 0
 second     |     text |
```

3. Просмотр содержимого таблицы

```
peterdlocalhost testdb=> SELECT * FROM my_table;
 first | second
----------- +----------
       1  | one
```

```
 2  | two
 3  | three
 4  | four
(4 строк)
```

Утилита администрирования с визуальным пользовательским интерфейсом. Утилита **fly-admin-postgres** предназначена для администрирования БД СУБД PostgreSQL и **позволяет:**

- просматривать иерархическую структуру БД;
- удаленно редактировать конфигурационные файлы СУБД и загружать их на сервер;
- управлять пользователями и группами СУБД;
- управлять дискреционным и мандатным доступом к объектам;
- выполнять SQL-запросы;
- создавать, изменять и удалять различные объекты БД;
- просматривать и редактировать данные таблиц.

Системные операции. Существует необходимость осуществлять ряд системных операций как для оптимизации работы СУБД, так и в качестве регламентных работ по обеспечению отказоустойчивости и возможности восстановления после сбоев.

Оптимизация баз данных. С целью оптимизации работы СУБД для увеличения производительности используются как архитектурные способы при разработке конкретной схемы БД, так и применение различных способов индексирования информации. При этом может возникать необходимость перестройки индексов в процессе изменения большого количества данных. Для пересоздания индексов в БД PostgreSQL используется утилита **reindexdb,** а для кластеризации (оптимизации индексов) ранее кластеризованных таблиц в БД используется утилита **clusterdb.** Она находит таблицы, которые были ранее кластеризованы, и кластеризует их заново по тем же индексам, которые были указаны до этого. Таблицы, которые до этого не были кластеризованы, не затрагиваются. Так же существует понятие сборки мусора, т. е. очистки таблиц от ранее удаленных записей.

Для сборки «мусора» и сбора статистики, необходимой для работы оптимизатора запросов,

БД PostgreSQL используется утилита **vacuumdb.**

```
reindexdb [ОПЦИИ]... [БАЗА_ДАННЫХ]
clusterdb [ОПЦИИ]... [БАЗА_ДАННЫХ]
vacuumdb [ОПЦИИ]... [БАЗА_ДАННЫХ]
```

Резервное копирование и восстановление. Для создания резервной копии БД в виде файла в текстовом или других форматах используется утилита **pg_dump.** Утилита 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.**

Альтернативные форматы упакованного файла могут быть использованы утилитой **pg_restore** для пересоздания **БД.** Они позволяют выбирать, что именно восстанавливать, или даже менять порядок элементов перед восстановлением.

Утилита **pg_dump** предоставляет гибкий механизм архивирования и переноса при использовании одного из форматов упаковки файла и комбинирования с **pg_restore.** Например, может быть выполнено создание резервной копии всей БД, после чего может быть использована утилита **pg_restore** для просмотра и/или выбора частей резервной копии для восстановления. **pg_dump [ОПЦИИ]... [БАЗА_ДАННЫХ]**

Созданный утилитой **pg_dump** архивный файл не содержит информации о статистике, которую использует оптимизатор запросов, таким образом рекомендуется выполнять команду ANALYZE после восстановления из резервной копии для достижения лучшей

производительности. Архивный файл так же не содержит команд `ALTER DATABASE ...` **SET,** эти установки архивируются утилитой **pg_dumpall** вместе с информацией о пользователях и других глобальных параметров установки.

Примеры:

Создание резервной копии БД mydb в виде SQL-скрипта **$ pg_dump mydb > db.sql** Загрузка подобного скрипта в новую БД newdb **$ psql -d newdb -f db.sql**

Создание резервной копии всех схем, начинающихся с east или west и заканчивающихся на gsm, исключая все схемы, содержащие слово test

$ pg_dump -n 'east*gsm' -n 'west*gsm' -N '*test*' mydb > db.sql Утилита **pg_dump** создает за раз дамп только одной БД, при этом информация о ролях или табличных пространствах не сохраняется (эта информация относится ко всему кластеру, а не к каждой отдельной БД). Для обеспечения удобного сохранения дампа всего содержимого кластера предназначена утилита **pg_dumpall.** Она создает резервную копию каждой БД кластера, а также сохраняет информацию о кластере, такую как определения ролей и табличных пространств.

В качестве стартовой БД возможно указание любого имени, но при загрузке данных в пустой кластер, как правило требуется указание **postgres.** При восстановлении дампа, полученного с помощью **pg_dumpall,** необходимо обладать правами суперпользователя БД, поскольку они требуются для восстановления информации о ролях и табличных пространствах. При использовании табличных пространств следует убедиться, что пути табличных пространств из дампа подходили для новой конфигурации.

Утилита **pg_dumpall** сначала выполняет команды для воссоздания ролей, табличных пространств и пустых БД, и лишь затем запускает **pg_dump** для каждой БД. Это означает, что хотя каждая БД будет обладать внутренней целостностью, «снимки» различных БД могут не быть полностью синхронизированы.

Примеры:

. Создание резервной копии всех БД **$ pg_dumpall > db.out** Восстановление сохраненных БД **$ psql -f db.out postgres**

Не имеет значения, с какой БД было осуществлено соединение, т.к. созданный с помощью **pg_dumpall** скрипт содержит соответствующие команды для создания и соединения для указанных БД.

Для восстановления БД из архивов, созданных утилитой **pg_dump** в одном из нетекстовых форматов, предназначена утилита **pg_restore.** Она осуществляет 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.

Примеры:

1. Создание резервной копии БД mydb в формате «custom»
$ pg_dump -Fc mydb > db.dump

3. Удаление БД и воссоздание ее из резервной

копии **$ dropdb mydb**

$ pg_restore -C -d postgres db.dump

БД, указанной в опции **-d,** может быть любая **БД** кластера. **pg_restore** используется только для выполнения команды CREATE DATABASE. С опцией **-c** данные всегда восстанавливаются в **БД,** указанную в резервной копии.

4. Загрузка резервной копии в новую БД

newdb **$ createdb -T templateO newdb**

$ pg_restore -d newdb db.dump

Необходимо отметить, что опция -c не была использована, вместо этого осуществлялось подключение непосредственно к восстанавливаемой БД. Новая БД была создана из шаблона templateO, а не template 1, для обеспечения первоначальной чистоты базы.

Совместная работа сервера СУБД PostgreSQL с ALD

Для работы СУБД PostgreSQL с ALD необходимо выполнение следующих условий:

1) наличие в системах, на которых функционируют сервер и клиенты СУБД PostgreSQL, установленного пакета клиента ALD — **aid-client;**

2) разрешение имен должно быть настроено таким образом, чтобы имя системы разрешалось, в первую очередь, как полное имя (например, **myserver. example. ru);**

3) клиент ALD должен быть настроен на используемый ALD домен.

Для проведения операций по настройке ALD и администрированию Kerberos необходимо знание паролей администраторов ALD и Kerberos.

Для обеспечения совместной работы сервера СУБД PostgreSQL с ALD необходимо, чтобы сервер СУБД PostgreSQL функционировал как сервис Kerberos. Выполнение данного условия

требует наличия в БД Kerberos принципала для сервера СУБД PostgreSQL, имя которого задается в формате: servicename/hostname@realm где имя сервиса servicename соответствует имени учетной записи пользователя, от которой осуществляется функционирование сервера СУБД PostgreSQL (по умолчанию — `postgres)`, и указывается в конфигурационном файле сервера PostgreSQL как значение параметра `krb srvname`. В качестве значения hostname указывается полное доменное имя системы, на которой функционирует сервер СУБД PostgreSQL, а в качестве значения realm — имя домена ALD.

Таким образом, для обеспечения совместной работы сервера СУБД PostgreSQL с ALD необходимо:

1) создать в БД ALD с помощью утилиты администрирования ALD принципала, соответствующего устанавливаемому серверу PostgreSQL. Принципал создается с автоматически сгенерированным случайным ключом:

```
# ald-admin service-add postgres/server.my_domain.org
```

2) ввести созданного принципала в группу сервисов mac, используя следующую команду:

```
# ald-admin sgroup-svc-add postgres/server .my_domain.org —sgroup=mac
```

3) создать файл ключа Kerberos для сервера СУБД PostgreSQL с помощью утилиты администрирования ALD aid-client, используя следующую команду (пример приведен для кластера БД по умолчанию):

```
# aid-client update-svc-keytab postgres/server .my_domain.org \
        —ktfile="/etc/postgresql/x.x/main/krb5.keytab"
```

Полученный файл должен быть доступен серверу СУБД PostgreSQL по пути, указанному в конфигурационном параметре `krb_server_keyfile` (в данном случае — /etc/postgresql/x.x/main/krb5. `keytab)`. Права доступа к этому файлу должны позволять читать его пользователю, от имени которого работает сервер СУБД PostgreSQL (как правило, владельцем файла назначается пользователь postgres);

4) сменить владельца, полученного на предыдущем шаге, файла `krb5. keytab` на пользователя posgtres, выполнив следующую команду:

```
# chown postgres /etc/postgresql/x.x/main/krb5.keytab
```

5) задать в конфигурационном файле сервера СУБД PostgreSQL /etc/postgreslq/x.x/main/postgresql. conf для приведенных ниже параметров соответствующие значения:

```
krb_server_keyf ile = ' /etc/postgresql/x. x/main/krb5. keytab' krb srvname
= fpostgres'
```

6) указать для внешних соединений в конфигурационном файле сервера СУБД PostgreSQL /etc/postgreslq/x 1 x/main/pg_hba. conf метод аутентификации gss.

Пример

```
host all all 192.168.32.0/24 gss
```

Общие условия, при которых обеспечивается совместное функционирование клиентов СУБД

PostgreSQL с ALD. Кроме того, сервер СУБД PostgreSQL должен быть также настроен соответствующим образом. Для настройки клиента СУБД PostgreSQL необходимо:

1) создать в БД ALD учетную запись пользователя, зарегистрированного в СУБД PostgreSQL (например, pgusemame). Дополнительная информация приведена в руководстве man на ALD;

2) задать в качестве значения параметра соединения krbsrvname имя сервиса servicename, используемое при создании принципала сервера СУБД PostgreSQL. Имя принципала сервера СУБД PostgreSQL задается в формате: servicename/hostname@realm где servicename — обычно имя учетной записи сервера СУБД PostgreSQL, используемое при создании принципала сервера СУБД PostgreSQL (по умолчанию — postgres), а hostname — полное доменное имя системы, на которой функционирует сервер СУБД PostgreSQL.

Мандатное разграничение доступа СУБД **PostgreSQL.** В Astra Linux доступно расширение команд SQL при мандатном разграничении в СУБД. К примеру, при необходимости обеспечения сквозной аутентификации из скриптов, запускаемых на WEB-сервере, которые должны взаимодействовать с другими службами в режиме ЕПП, например, с защищенным сервером СУБД, в конфигурационном файле для виртуального хоста следует дополнительно прописать:

```
KrbSaveCredentials on
```

В настройках вашего браузера необходимо задать в качестве значений параметра network.negotiate-auth.delegation-uris, маски доменов которым можно передавать данные для сквозной аутентификации.