Авторы:

Шомахов Ислам Аликович – студент 3 курса Российского государственного университета нефти и газа (Национальный исследовательский университет) имени И.М. Губкина;

Каленский Егор Анатольевич – студент 3 курса Российского государственного университета нефти и газа (Национальный исследовательский университет) имени И.М. Губкина;

Копанев Олег Олегович – студент 3 курса Российского государственного университета нефти и газа (Национальный исследовательский университет) имени И.М. Губкина.

Поддержка ролевой модели доступа к системе управления в ОС Альт

Аннотация

В данной статье исследуется применение ролевой модели доступа (RBAC) в операционной системе Альт, рассматриваются теоретические основы RBAC и анализируется ее реализация на уровне ядра и системных вызовов. Описываются механизмы контроля доступа, такие как POSIX ACL, FreeIPA ACL, Capabilities и Control, с примерами их использования. Исследование демонстрирует, как внедрение RBAC повышает безопасность системы и упрощает управление доступом. Работа представляет интерес для специалистов по информационной безопасности и системных администраторов.

Введение

В современных информационных системах безопасность и управление доступом играют ключевую роль. Один из эффективных методов управления доступом – это ролевая модель, или Role-Based Access Control (RBAC). В ОС Альт, как и в других операционных системах, использование RBAC позволяет централизованно управлять доступом пользователей к ресурсам системы, обеспечивая простоту администрирования и высокий уровень безопасности.

Обзор литературы

Обзор существующей литературы позволяет более глубоко понять текущие подходы и методы, применяемые для реализации ролевой модели доступа в операционных системах. Один из таких подходов представлен в работе [1]. В данной работе рассматривается математическая модель, позволяющая выявлять и оценивать угрозы безопасности в контексте использования приложений, работающих под Wine.

Информация про ядро и системные вызовы в контексте управления доступом подробно представлена в работе [2]. В ней изучаются особенности реализации и применения ролевой модели в операционной системе GNU/Linux.

Сравнение политик разграничений доступа рассматривалось в работе [3]. В ней автор рассмотрел принципиальные сходства и различия основных известных политик разграничения доступа, а также способы их реализации.

Общая информация об анализе технологии ACL и ее методах описывается в работе [4]. В ней подробно описывается применение данной технологии в системе сетевой безопасности.

В работе [5] описывается один из подходов к контролю доступа RBAC. В ней рассказывается принцип работы, где применяется данный подход, а также проведено несколько экспериментов, показывающих эффективность данной системы.

Математическое описание RBAC

Для глубокого понимания механизмов работы RBAC в ОС Альт необходимо рассмотреть его математическую модель. Основные компоненты модели включают:

- S = Субъект = Человек или автоматизированный агент (множество пользователей);
- R = Роль = Рабочая функция или название, которое определяется на уровне авторизации (множество ролей);

- P = Разрешения = Утверждения режима доступа к ресурсу (множество прав доступа на объекты системы);
- SE = Сессия = Соответствие между S, R и/или P
- SA = Назначение субъекта

Основные отношения и функции:

- $UA \subseteq S \times R$. Отношение определяет, какие роли назначены каждому пользователю.
- $PA \subseteq P \times R$. Отношение определяет, какие разрешения связаны с каждой ролью.
- $PA: R \rightarrow 2^P \ \forall p \in P, \exists r \in R : p \in PA(r)$. Функция для каждой роли возвращает множество всех назначенных ей разрешений.
- $RH \subseteq R \times R$. Отношение определяет иерархию ролей.
- Функция назначения сессий пользователям: session_users: $SE \rightarrow S$
- Функция назначения ролей сессиям: session_roles: $SE \rightarrow 2^R$. При этом должно выполняться условие: $\forall se \in SE$, session_roles(se) $\subseteq$ {r $\in$ R | (session_users(se), r) $\in$ UA}
- Функция получения разрешений для роли с учетом иерархии: authorized_permissions (r: R) = $\{p \in P \mid \exists r' \leq r, (p, r') \in PA\}$
- Функция проверки наличия разрешения у пользователя в сессии: has_permission(se: SE, p: P) = $\exists r \in$ session_roles(se) : p $\in$ authorized_permissions(r)

Ограничения:

- Взаимоисключающие роли (Mutually Exclusive Roles, MER): MER $\subseteq$ $R \times R \ \forall(r_1, r_2) \in MER, \forall s \in S : (s, r_1) \in UA \Rightarrow (s, r_2) \notin UA$
- Кардинальность ролей: max_cardinality: $R \rightarrow N \ \forall r \in R : |\{s \in S \mid (s, r) \in UA\}| \leq$ max_cardinality(r)

Данные формулы и ограничения предоставляют формальную основу для реализации RBAC в ОС Альт, позволяя точно описать и проверить политики безопасности, реализуемые на уровне ядра и системных вызовов.

Технологическая основа RBAC в ОС Альт

Ролевая модель доступа основывается на назначении прав доступа не отдельным пользователям, а их ролям. Пользователи, в свою очередь, получают доступ к ресурсам в зависимости от их роли в системе.

Стоит затронуть технологии, на основе которых работает ролевая модель доступа.

Ядро операционной системы ОС Альт управляет доступом к оперативной памяти, сети, дисковым пространствам и другим внешним устройствам. Оно инициирует и отслеживает процессы, управляет распределением времени между ними, устанавливает разграничение прав и формирует политику безопасности, которую невозможно обойти без обращения к нему. Ядро работает в режиме "супервизора", это означает, что именно оно предоставляет доступ ко всей оперативной памяти и аппаратной таблице задач. Процессы же работают в "пользовательском режиме". Ядро привязывает их к записям в таблице задач с указанием доступной оперативной памяти. Ядро постоянно находится в памяти, обрабатывая системные вызовы от процессов.

ОС Альт использует модифицированное ядро Linux, которое включает расширенные возможности управления доступом. В частности, ядро поддерживает механизмы LSM (Linux Security Modules), позволяющие реализовать различные модели безопасности, включая RBAC.

Для реализации доступа субъектов (пользователей или процессов/приложений) к объектам в GNU/Linux необходимо использовать системные вызовы на уровне ядра. Они предоставляют файлам доступ (на чтение, запись или выполнение), проводят операции с файлами (создание, переименование или удаление файлов и каталогов), управляют символьными и жесткими ссылками, а также многое другое. Системные вызовы являются единственным способом получить доступ к ресурсам операционной системы из непривилегированного режима. Кроме того, существуют специализированные системные вызовы для операций ввода/вывода,

управления устройствами (например, ioctl), управления файловыми системами (mount, umount, pivot_root, chroot), синхронизации данных на диске, управления временем системы, управления памятью, создания и завершения процессов, загрузки модулей ядра и других аспектов системы.

Так же, существуют специализированные системные вызовы, которые позволяют ядру взаимодействовать с пользовательским пространством. В контексте RBAC ключевыми являются системные вызовы, которые связаны с проверкой прав доступа и управлением привилегиями, такие как capget(), capset(), prctl().

ACL

Как уже было сказано ранее, модель RBAC предполагает доступ к ресурсам на основе ролей, а роли, в свою очередь, имеют наборы разрешений. В подобном контексте разрешения, присваиваемые ролям, могут быть определены с использованием так называемых списков контроля доступа – ACL (Access Control List).

ACL представляет собой набор правил, которые определяют, каким пользователям или системным процессам предоставлен доступ к объектам и какие операции они могут выполнять с этими объектами. Каждый объект в системе имеет связанный с ним атрибут безопасности, который содержит список записей контроля доступа. Эти записи включают в себя разрешения, предоставленные конкретным пользователям или группам пользователей.

Например, для файла можно определить ACL, который предоставляет пользователю "user1" права на чтение (r) и запись (w), а группе "admins" — полный доступ (rwx). Такой подход позволяет гибко настраивать доступ к объектам на уровне отдельных пользователей и операций.

Рассмотрим некоторые типы ACL, реализованные в ОС Альт, – они приведены в таблице 1

Таблица 1

Тип ACL	Основные возможности	Утилиты управления	Связанные пакеты	Преимущества
POSIX ACL	Управление правами доступа к файлам и директориям	setfacl, getfacl	acl, attr	Гибкость в настройке доступа, поддержка большинства файловых систем
FreeIPA ACL	Управление доступом к услугам сертификации и профилям	ipa caacl-add, ipa caacl-add-ca, ipa caacl-add-profile и т.д.	freeipa-server / freeipa-server-dns / free-ipa-server-trust-ad	Централизованное управление доступом к сертификатам, интеграция с Kerberos и LDAP
Capabilities	Гранулярное управление привилегиями отдельных процессов	setcap, getcap, capsh	libcap	Уменьшение рисков чрезмерных привилегий, улучшенная безопасность
control	Управление доступом к задачам	new_fmode, new_subst, new_help, new_summary	control	полезен для управления SUID/SGID-бинарниками

Также рассмотрим и примеры использования приведенных типов контроля доступа.

POSIX ACL

Создадим пользователя и зададим пароль для его учетной записи:

```
ALT ~ # useradd test1
ALT ~ # passwd test1
passwd: updating all authentication tokens for user test1.

You can now choose the new password or passphrase.

A valid password should be a mix of upper and lower case letters, digits, and
other characters.  You can use a password containing at least 4 characters
from at least 3 of these 4 classes.
An upper case letter that begins the password and a digit that ends it do not
count towards the number of character classes used.

A passphrase should be of at least 3 words, 6 to 72 characters long, and
contain enough different characters.

Alternatively, if no one else can see your terminal now, you can pick this as
your password: "review2Daisy5hire".

Enter new password:
Weak password: not enough different characters or classes for this length.
Re-type new password:
passwd: all authentication tokens updated successfully.
```

Создаем тестовый файл

```
ALT ~ # touch testfile.txt
```

Смотрим, что предоставляет нам setfacl (смотрим на префиксы), даем пользователя test1 права rwx на файл testfile.txt и убеждаемся, что права действительно назначились:

```
ALT ~ # setfacl -h
setfacl 2.3.1 -- set file access control lists
Usage: setfacl [-bkndRLP] { -m|-M|-x|-X ... } file ...
  -m, --modify=acl        modify the current ACL(s) of file(s)
  -M, --modify-file=file  read ACL entries to modify from file
  -x, --remove=acl        remove entries from the ACL(s) of file(s)
  -X, --remove-file=file  read ACL entries to remove from file
  -b, --remove-all        remove all extended ACL entries
  -k, --remove-default    remove the default ACL
      --set=acl           set the ACL of file(s), replacing the current ACL
      --set-file=file     read ACL entries to set from file
      --mask              do recalculate the effective rights mask
  -n, --no-mask           don't recalculate the effective rights mask
  -d, --default           operations apply to the default ACL
  -R, --recursive         recurse into subdirectories
  -L, --logical           logical walk, follow symbolic links
  -P, --physical          physical walk, do not follow symbolic links
      --restore=file      restore ACLs (inverse of `getfacl -R')
      --test              test mode (ACLs are not modified)
  -v, --version           print version and exit
  -h, --help              this help text
ALT ~ # setfacl -m u:test1:rwx testfile.txt
ALT ~ # getfacl testfile.txt
# file: testfile.txt
# owner: root
# group: root
user::rw-
user:user1:rwx
user:test1:rwx
group::r--
mask::rwx
other::r--
```

FreeIPA ACL

При наличии успешно развернутого сервера FreeIPA имеется возможность подключить модуль, который используется для определения правил, которые управляют тем, какие службы сертификации и профили можно использовать для выдачи сертификатов конкретным учётным записям или группам учётных записей.

Чтобы разрешить запрос на сертификат, учетная запись (или записи), являющаяся субъектом запроса, должна быть включена в область действия ACL службы сертификации, которая также охватывает целевые службы сертификации и профили. ACL службы сертификации может быть связана с одним или несколькими профилями по идентификатору профиля.

При выполнении в командной строке ipa caacl откроется справка по данному модулю, где можно обнаружить основные команды для управления:

```
Команды темы:
  caacl-add             Создать новую запись CA ACL.
  caacl-add-ca          Добавить CA в CA ACL.
  caacl-add-host        Добавить целевые узлы или группы узлов в CA ACL.
  caacl-add-profile     Добавить профили в CA ACL.
  caacl-add-service     Добавить службы в CA ACL.
  caacl-add-user        Добавить пользователей и группы в CA ACL.
  caacl-del             Удалить CA ACL.
  caacl-disable         Отключить CA ACL.
  caacl-enable          Включить CA ACL.
  caacl-find            Поиск CA ACL.
  caacl-mod             Изменить CA ACL.
  caacl-remove-ca       Удалить CA из CA ACL.
  caacl-remove-host     Удалить целевые узлы или группы узлов из CA ACL.
  caacl-remove-profile  Удалить профили из CA ACL.
  caacl-remove-service  Удалить службы из CA ACL.
  caacl-remove-user     Удалить пользователей и группы из CA ACL.
  caacl-show            Показать свойства CA ACL.
```

Например, создадим ACL службы сертификации “test”, которая предоставляет всем пользователям доступ к профилю “caIPAserviceCert” (для этого сначала выполним kinit):

```
ipa ~ # kinit
Password for root@EXAMPLE.TEST:
ipa ~ # klist
Ticket cache: KEYRING:persistent:0:0
Default principal: root@EXAMPLE.TEST

Valid starting       Expires              Service principal
04.07.2024 00:25:43  05.07.2024 00:25:11  krbtgt/EXAMPLE.TEST@EXAMPLE.TEST
ipa ~ # ipa caacl-add test --usercat=all
-----------------------
Добавлена CA ACL "test"
-----------------------
  Имя ACL: test
  Включено: True
  Категория пользователей: all
ipa ~ #
```

```
ipa ~ # ipa caacl-add-profile test --certprofiles caIPAserviceCert
  Имя ACL: test
  Включено: True
  Категория пользователей: all
  Профили: caIPAserviceCert
-----------------------------------
Количество добавленных участников 1
-----------------------------------
ipa ~ #
```

Просмотрим свойства именованной ACL службы сертификации

```
ipa ~ # ipa caacl-show test
  Имя ACL: test
  Включено: True
  Категория пользователей: all
  Профили: caIPAserviceCert
ipa ~ #
```

Отключим ACL службы сертификации:

```
ipa ~ # ipa caacl-disable test
-----------------------
Отключена CA ACL "test"
-----------------------
ipa ~ #
```

Удалим ACL службы сертификации:

```
ipa ~ # ipa caacl-del test
--------------------
Удалена CA ACL "test"
--------------------
ipa ~ #
```

Capabilities

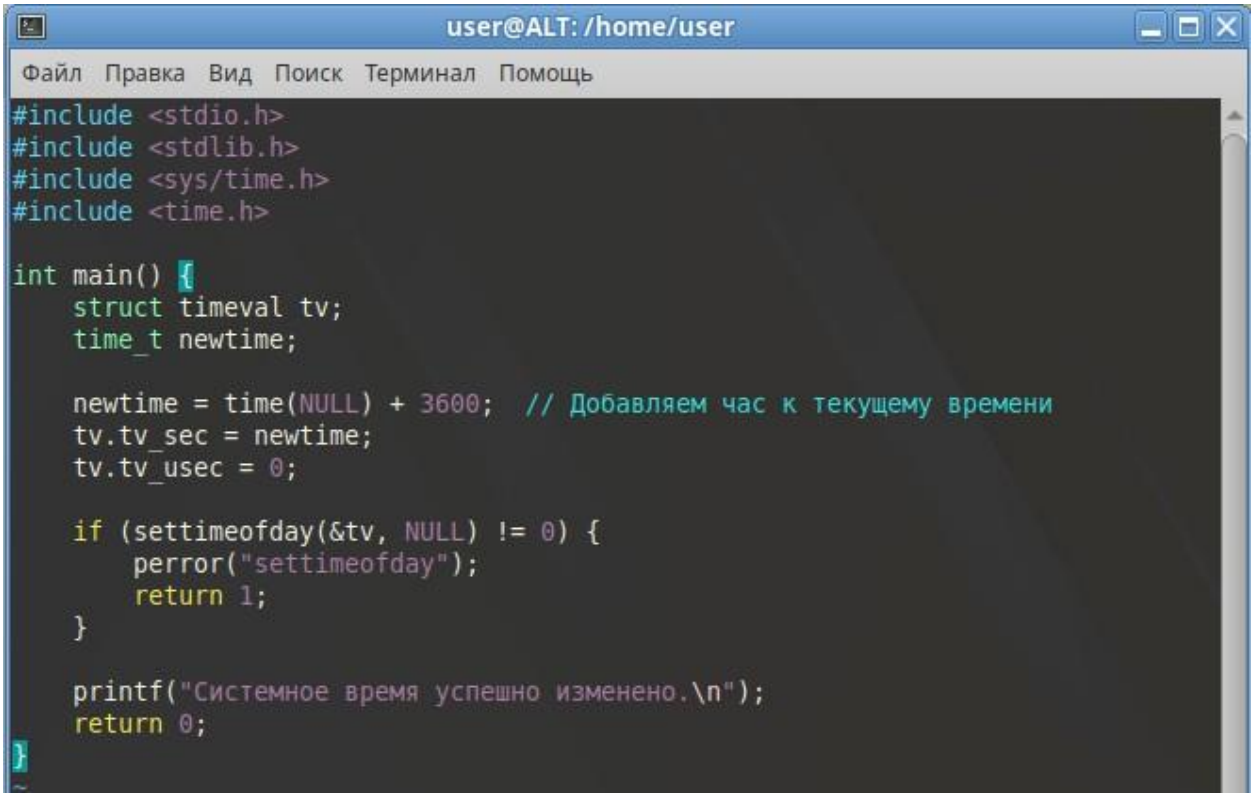
Описание Capabilities можно открыть с помощью команды:

```
ALT ~ # man capabilities
```

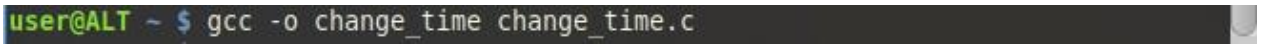
Оно довольно объемное, требует времени и определенных знаний, чтобы разобраться в нем.

Для того чтобы продемонстрировать работу с Capabilities, создадим программу на языке C, которая будет выполнять привилегированную операцию – изменение системного времени.

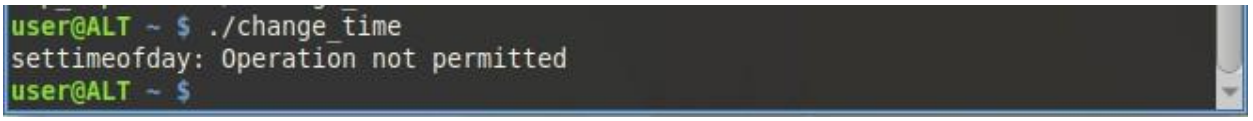
```
user@ALT ~ $ vim change_time.c
```



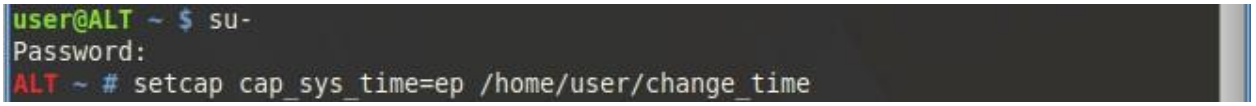
Скомпилируем программу:



Попробуем запустить от обычного пользователя:



Как и ожидалось, операция недоступна. Перейдем в режим суперпользователя и добавим capability:



Просмотрим capability, который мы добавили:

Выйдем из-под рута, выполним программу снова

```
ALT ~ # exit
выход
user@ALT ~ $ ./change_time
Системное время успешно изменено.
```

Системное время успешно изменено.

Control

В ОС Альт control является механизмом переключения между неким набором фиксированных состояний для задач, допускающих такой набор. Чтобы просмотреть управляемые утилитой control команды, выполним:

```
ALT ~ # control
at              public          (public restricted atdaemon)
autofs-browse-mode disabled        (disabled enabled default)
bind-caps       disabled        (disabled enabled)
bind-chroot     disabled        (disabled enabled)
bind-debug      disabled        (enabled disabled)
bind-slave      disabled        (enabled disabled)
card-actions    default         (default terminate)
cgi-bin_printenv symlink_root_noexec (none file_root_noexec file_root_exec file_
webmaster_noexec file_webmaster_exec symlink_root_noexec symlink_root_exec symli
nk_webmaster_noexec symlink_webmaster_exec)
cgi-bin_test-cgi symlink_root_noexec (none file_root_noexec file_root_exec file_
webmaster_noexec file_webmaster_exec symlink_root_noexec symlink_root_exec symli
nk_webmaster_noexec symlink_webmaster_exec)
chage           restricted      (public restricted)
chfn            restricted      (public restricted)
chrony          server          (server client)
chsh            restricted      (public restricted)
consolehelper   public          (public wheelonly restricted)
crontab         public          (public restricted)
cups            server          (server local)
dvd-ram-control public          (public restricted legacy)
dvd+rw-booktype public          (public restricted legacy)
dvd+rw-format   public          (public restricted legacy)
dvd+rw-mediainfo public          (public restricted legacy)
```

Перечень команд довольной обширный, мы остановимся на “su”. Создадим пользователя testuser, зададим ему пароль:

```
ALT ~ # useradd testuser
ALT ~ # passwd testuser
passwd: updating all authentication tokens for user testuser.

You can now choose the new password or passphrase.

A valid password should be a mix of upper and lower case letters, digits, and
other characters.  You can use a password containing at least 4 characters
from at least 3 of these 4 classes.
An upper case letter that begins the password and a digit that ends it do not
count towards the number of character classes used.

A passphrase should be of at least 3 words, 6 to 72 characters long, and
contain enough different characters.

Alternatively, if no one else can see your terminal now, you can pick this as
your password: "Groin5Symbol4Smelly".

Enter new password:
Weak password: not enough different characters or classes for this length.
Re-type new password:
passwd: all authentication tokens updated successfully.
ALT ~ #
```

Проверим текущее состояние control для su:

```
ALT ~ # control su
wheelonly
```

Это означает, что только пользователи группы wheel могут выполнять команду su. Убедимся, что пользователя testuser нет в группе wheel, войдем в учетную запись пользователя testuser и попытаемся выполнить команду su:

```
ALT ~ # getent group wheel
wheel:x:10:root,user
ALT ~ # su- testuser
testuser@ALT ~ $ su-
-bash: /bin/su: Отказано в доступе
testuser@ALT ~ $
```

Далее вернемся к суперпользователю и поменяем политику su на public

```
testuser@ALT ~ $ exit
выход
ALT ~ # control su public
ALT ~ # control su
public
ALT ~ #
```

Снова зайдем на учетную запись пользователя testuser и убедимся, что команда su теперь работает:

Заключение

В данной статье проведено исследование ролевой модели доступа (RBAC) в операционной системе Альт. Был рассмотрен теоретический аспект модели, включая математическое описание основных компонентов и функций, а также технологическая реализация RBAC в ОС Альт на уровне ядра и системных вызовов. Исследование показало, что применение RBAC в ОС Альт обеспечивает централизованное управление доступом, упрощает администрирование и повышает уровень безопасности системы. Внедрение и настройка RBAC с использованием технологий, таких как ACL, Capabilities и Control позволяет эффективно контролировать доступ к ресурсам системы и минимизировать риски, связанные с чрезмерными привилегиями.

Список источников.

1. Уймин, А. Г. Оценка безопасности Wine с использованием методологии STRIDE: математическая модель / А. Г. Уймин, И. М. Морозов // Современная наука: актуальные проблемы теории и практики. Серия: Естественные и технические науки. – 2023. – № 6-2. – С. 164-170. – DOI 10.37882/2223-2982.2023.6-2.40. – EDN HYCKNP;
2. Диссертация Каннера А.М. «Модель и алгоритмы обеспечения безопасности управления доступом в операционных системах семейства Linux»;
3. Статья Марочкина А.А. «Анализ современных способов разграничения доступа»;

4. Статья Azmi, F., Kalsum, T. U., Alamsyah, H. «Analysis and Application of Access Control List (ACL) Methods on Computer Networks»
5. Статья K. Rajesh Rao «Role recommender-RBAC: Optimizing user-role assignments in RBAC».
6. Альт Рабочая Станция. Документация. Руководство пользователя // Редакция – март, 2024 // [Электронный ресурс]: https://docs.altlinux.org/ru-RU/alt-workstation/10.2/html/alt-workstation/index.html