

Локальная аутентификация в ALT Linux 6.0-8.0 по Рутокен ЭЦП

Введение

В данной инструкции описывается, как настроить модуль `pam_pkcs11` для работы с библиотекой `librtpkcs11ecp.so`.

Стенд

Нам понадобится токен или смарт-карта семейства Рутокен ЭЦП, отформатированные через Панель управления Рутокен.

Настройки для токена и смарт-карты идентичны.

Проверка модели токена

1. Подключите USB-токен к компьютеру.
2. Для определения названия модели USB-токена откройте **Терминал** и введите команду:

```
$ lsusb
```

В результате в окне Терминала отобразится название модели USB-токена:

```
[dmitrieva@localhost ~]$ lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 004: ID 0a89:0030 Aktiv Rutoken ECP
Bus 002 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 002 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

Убедитесь, что используете: **Aktiv Rutoken ECP**

Если вы используете смарт-карту Рутокен, то проверку проходить не требуется.

Общий порядок действий

1 Устанавливаем необходимые пакеты и их зависимости:

Для этого вы можете воспользоваться Терминалом:

```
$ sudo apt-get install opensc pam_pkcs11 pcsc-lite-ccid openssl-engine_pkcs11
```

Или из меню запустить Приложения - Системные - Программа управления пакетами Synaptic и используя быстрый поиск выбрать для установки пакеты:

- opensc,
- pam_pkcs11,
- pcsc-lite-ccid,
- openssl-engine_pkcs11.

2 Скачиваем и устанавливаем пакет для вашей системы

- [Библиотека rtPKCS11ecp для GNU/Linux RPM 32-bit \(x86\)](#)
- [Библиотека rtPKCS11ecp для GNU/Linux RPM 64-bit \(x86_64\)](#)

Если установка завершилась корректно, то в папке `/usr/lib` (или `/usr/lib64`) появится библиотека `librtpkcs11ecp.so`.

3 Проверяем работу токена или смарт-карты

Подключаем токен или смарт-карту к компьютеру. Запускаем dmesg и убедимся в том, что устройство определилось корректно.

```
usb 2-2.2: new full speed USB device number 6 using uhci_hcd
usb 2-2.2: New USB device found, idVendor=0a89, idProduct=0030
usb 2-2.2: New USB device strings: Mfr=1, Product=2, SerialNumber=0
usb 2-2.2: Product: Rutoken ECP
usb 2-2.2: Manufacturer: Aktiv
usb 2-2.2: configuration #1 chosen from 1 choice
```

Для 32-битной версии используйте команду:

```
$ pkcs11-tool --module /usr/lib/librtpkcs11ecp.so -T
```

Для 64-битной версии используйте команду:

```
$ pkcs11-tool --module /usr/lib64/librtpkcs11ecp.so -T
```

4 Создаем ключевую пару

Если у вас уже имеется выписанная на токен ключевая пара RSA с привязанным к ней сертификатом, то вы можете использовать их для аутентификации.

Рекомендуемая длина ключа RSA - не ниже 2048 бит.

Действуйте по основной инструкции, пропустив шаги 4-8.

Внимание! При выполнении команды запрашивается PIN-код пользователя. Генерация ключевой пары может занять некоторое время.

Для 32-битной версии используйте команду:

```
$ pkcs11-tool --module /usr/lib/librtpkcs11ecp.so --keypairgen --key-type
rsa:2048 -l --id 45
```

Для 64-битной версии используйте команду:

```
$ pkcs11-tool --module /usr/lib64/librtpkcs11ecp.so --keypairgen --key-
type rsa:2048 -l --id 45
```

```
[alt@host-134 ~]$ pkcs11-tool --module /usr/lib/librtpkcs11ecp.so --keypairgen --key-type rsa:2048 -l --id 45
Using slot 0 with a present token (0x0)
Logging in to "Rutoken ECP <no label>".
Please enter User PIN:
Key pair generated:
Private Key Object: RSA
label:
ID: 45
Usage: decrypt, sign, unwrap
Public Key Object: RSA 2048 bits
label:
ID: 45
Usage: encrypt, verify, wrap
```

Утилита `pkcs11-tool` входит в состав `opensc`.

Параметры, задаваемые в этой строке:

<code>--module <arg></code>	путь к библиотеке <code>pkcs11</code> (обязательный параметр)
<code>--keypairgen</code>	генерация ключевой пары
<code>--key-type <arg></code>	задает тип и длину ключа. В нашем случае тип – <code>rsa</code> , длина - 2048 бит (с длиной ключа 1024 бит возникают проблемы)
<code>-l</code>	запрос PIN-кода токена до каких-либо операций с ним (обязательный параметр)

--id <arg>	определяет id создаваемого объекта (понадобится при создании сертификата)
------------	---

5 Создаем сертификат в формате PEM

Запускаем openssl и загружаем модуль поддержки pkcs11:

Для 32-битной версии используйте команду:

```
$ openssl
OpenSSL> engine dynamic -pre SO_PATH:/usr/lib/openssl/engines
/libpkcs11_gost.so -pre ID:pkcs11 -pre LIST_ADD:1 -pre LOAD -pre
MODULE_PATH:/usr/lib/librtpkcs11ecp.so
```

Для 64-битной версии используйте команду:

```
$ openssl
OpenSSL> engine dynamic -pre SO_PATH:/usr/lib64/openssl/engines
/libpkcs11_gost.so -pre ID:pkcs11 -pre LIST_ADD:1 -pre LOAD -pre
MODULE_PATH:/usr/lib64/librtpkcs11ecp.so
```

```
[alt@host-134 ~]$ openssl
OpenSSL> engine dynamic -pre SO_PATH:/usr/lib/openssl/engines/engine_pkcs11.so -pre ID:pkcs11 -pre
LIST_ADD:1 -pre LOAD -pre MODULE_PATH:/usr/lib/librtpkcs11ecp.so
(dynamic) Dynamic engine loading support
[Success]: SO_PATH:/usr/lib/openssl/engines/engine_pkcs11.so
[Success]: ID:pkcs11
[Success]: LIST_ADD:1
[Success]: LOAD
[Success]: MODULE_PATH:/usr/lib/librtpkcs11ecp.so
Loaded: (pkcs11) pkcs11 engine
OpenSSL>
```

Создаем сертификат в PEM-формате. Внимание! При выполнении этой команды запрашивается PIN-код пользователя.

```
OpenSSL> req -engine pkcs11 -new -key 0:45 -keyform engine -x509 -out
cert.pem -text
```

```
OpenSSL> req -engine pkcs11 -new -key 0:45 -keyform engine -x509 -out cert.pem -text
engine "pkcs11" set.
PKCS#11 token PIN:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value.
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [RU]:
State or Province Name (full name) []:Moscow
Locality Name (eg. city) []:Moscow
Organization Name (eg. company) []:Aktiv
Organizational Unit Name (eg. section) []:Aktiv
Common Name (e.g. your name or your server's hostname) []:alt
Email Address []:alt@mail.ru
```

Здесь:

-key	указывает закрытый ключ (в нашем случае 0:45 – слот:ID ключа)
-x509	выдает самоподписанный сертификат

6 Конвертируем сертификат из формата PEM в формат CRT

```
OpenSSL> x509 -in cert.pem -out cert.crt -outform DER
```

7 Сохраняем сертификат на аутентифицирующий носитель

Закрываем openssl (exit). Сохраняем сертификат CRT на Рутокен. Внимание! При выполнении этой команды запрашивается PIN-код пользователя.

Для 32-битной версии используйте команду:

```
$ pkcs11-tool --module /usr/lib/librtpkcs11ecp.so -l -y cert -w cert.crt --id 45
```

Для 64-битной версии используйте команду:

```
$ pkcs11-tool --module /usr/lib64/librtpkcs11ecp.so -l -y cert -w cert.crt --id 45
```

```
[alt@host-134 ~]$ pkcs11-tool --module /usr/lib/librtpkcs11ecp.so -l -y cert -w cert.crt --id 45
Using slot 0 with a present token (0x0)
Logging in to "Rutoken ECP <no label>".
Please enter User PIN:
Created certificate:
Certificate Object, type = X.509 cert
label:
ID: 45
```

Здесь:

-y <arg>	тип объекта (может быть cert, privkey, pubkey, data)
-w <arg>	записать объект на токен

8 Проверяем, что на токене есть всё, что необходимо

Внимание! При выполнении команды запрашивается PIN-код пользователя.

Для 32-битной версии используйте команду:

```
$ pkcs11-tool --module /usr/lib/librtpkcs11ecp.so -0 -l
```

Для 64-битной версии используйте команду:

```
$ pkcs11-tool --module /usr/lib64/librtpkcs11ecp.so -0 -l
```

9 Создаем файлы конфигурации pam_pkcs11

Воспользуемся примерами, которые предоставляют разработчики pam_pkcs11.

Потребуются права суперпользователя:

```
$ su
Password:
#
```

Для ALT Linux версии 6.0 и 7.0 используйте команду:

```
# cp /usr/share/pam_pkcs11/pam_pkcs11.conf.example /etc/security/pam_pkcs11/pam_pkcs11.conf
# cp /usr/share/pam_pkcs11/subject_mapping.example /etc/security/pam_pkcs11/subject_mapping
```

Для ALT Linux версии 8 используйте команду:

```
# cp /usr/share/doc/pam_pkcs11/pam_pkcs11.conf.example /etc/security/pam_pkcs11/pam_pkcs11.conf
# cp /usr/share/doc/pam_pkcs11/subject_mapping.example /etc/security/pam_pkcs11/subject_mapping
```

на вопрос о переписывании файла следует ответить "y"

10 Включаем аутентификацию по внешнему носителю

```
# rm /etc/pam.d/system-auth
# ln -s /etc/pam.d/system-auth-pkcs11 /etc/pam.d/system-auth
```

на вопрос об удалении ссылки следует ответить "y"

11 Редактируем конфигурацию аутентификации в системе

Отредактируем **вторую строку** файла конфигурации /etc/pam.d/system-auth.

Внимание, приведенная ниже конфигурация является примером, а не эталоном настройки системы.

Для редактирования можно воспользоваться редактором mcedit

```
# mcedit /etc/pam.d/system-auth
```

Для 32-битной версии используйте строку:

```
auth [success=1 default=ignore] pam_pkcs11.so
pkcs11_module=/usr/lib/librtpkcs11ecp.so
```

Для 64-битной версии используйте строку:

```
auth [success=1 default=ignore] pam_pkcs11.so
pkcs11_module=/usr/lib64/librtpkcs11ecp.so
```

12 Редактируем конфигурацию pam_pkcs11

Отредактируем файл /etc/security/pam_pkcs11/pam_pkcs11.conf

Внимание, приведенная ниже конфигурация является примером, а не эталоном настройки системы.

Для редактирования можно воспользоваться редактором mcedit

```
# mcedit /etc/security/pam_pkcs11/pam_pkcs11.conf
```

Для 32-битной версии используйте:

```

pam_pkcs11 {
    nullok = false;
    debug = false;
    use_first_pass = false;
    use_authtok = false;
    card_only = false;
    wait_for_card = false;
    use_pkcs11_module = rutokenecp;

    # Aktiv Rutoken ECP
    pkcs11_module rutokenecp {
        module = /usr/lib/librtpkcs11ecp.so
        slot_num = 0;
        support_thread = true;
        ca_dir = /etc/pam_pkcs11/cacerts;
        crl_dir = /etc/pam_pkcs11/crls;
        cert_policy = signature;
    }

    use_mappers = subject;
    mapper_search_path = /lib/pam_pkcs11;

    mapper subject {
        debug = false;
        module = internal;
        ignorecase = false;
        mapfile = file:///etc/security/pam_pkcs11/subject_mapping;
    }
}

```

Для 64-битной версии замените строку

`module = /usr/lib/librtpkcs11ecp.so` на строку `module = /usr/lib64/librtpkcs11ecp.so`

и строку

`mapper_search_path = /lib/pam_pkcs11;` на строку `mapper_search_path = /lib64/pam_pkcs11;`

13 Добавляем связку сертификата на токене с пользователем системы ALT Linux.

Для этого выполняем команду `pkcs11_inspect`

```
# pkcs11_inspect > /etc/security/pam_pkcs11/subject_mapping
```

Используя текстовый редактор прокомментируем или удаляем ненужные строчки

Для этого можно использовать редактор `mcedit`

```
# mcedit /etc/security/pam_pkcs11/subject_mapping
```

```

# Printing data for mapper subject:
/C=RU/ST=Moscow/L=Moscow/O=Aktiv/OU=Aktiv/CN=alt/emailAddress=alt@mail.ru -
> alt

```

Внимание! Вместо **alt** нужно установить имя пользователя в вашей системе

Если вы не знаете имя пользователя запустите команду `whoami` (без прав суперпользователя)

```
$ whoami
```

14 Проверям выполненные настройки

Проверьте, что настройка была выполнена верно, используя команду login. **Не завершайте свою сессию, пока не убедитесь в том, что все работает корректно.**

Если команда login выполняется успешно, то вы можете завершать свою сессию и использовать аутентификацию по токенам и смарт-картам Rutoken.

```
[alt@host-134 ~]$ sudo login
Found the Smart card.
Welcome Rutoken ECP <no label>!
Smart card PIN:
[alt@host-134 ~]$
```

В случае возникновения ошибок еще раз проверьте все настройки. Для выявления проблемы вы так же можете включить вывод дополнительной информации при аутентификации.

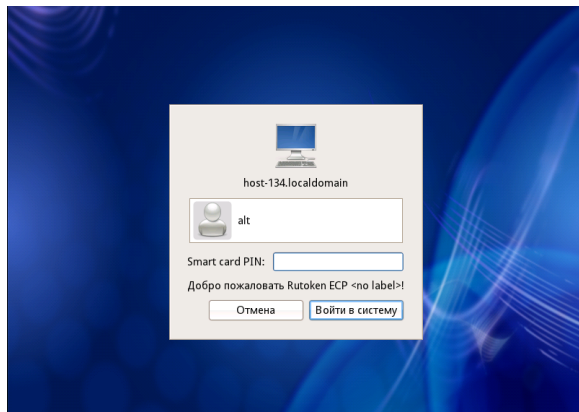
Для этого:

1. В файле pam_pkcs11.conf исправьте все строки вида "debug = false;", на строки "debug = true;".
2. В конец **второй строки** файла конфигурации /etc/pam.d/system-auth добавьте слово "debug".

Не забудьте отключить вывод дополнительной информации после настройки системы.

15 Настройка завершена!

На этом настройка закончена. После перезапуска ОС окно входа в систему будет выглядеть так:



16 Другие пользователи

При необходимости добавить вход по токenu для других пользователей следует:

- 1) Настроить другие токены аналогичным образом. Это рекомендуемый способ, так как политика "один токен - один пользователь", является предпочтительной.
- 2) Выписать другую пару ключей и сертификат на тот же токен. (иногда бывает удобно для периодической работы из под суперпользователя)

В обоих случаях в файле subject_mapping должно оказаться две (или несколько) записей