Локальная аутентификация в ALT Linux 9.0 и новее 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.

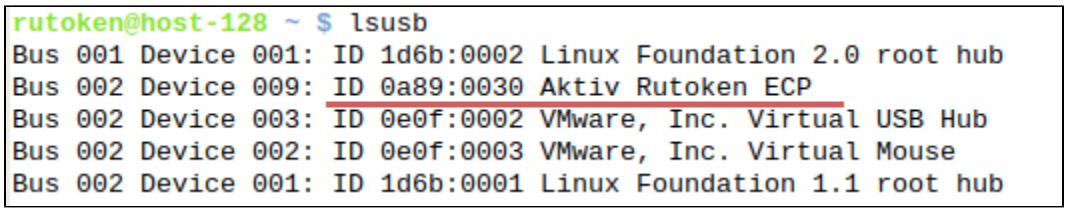
Настройки для токена и смарт-карты идентичны.

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:



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:

```
$ su

# 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
```

```
host-128 ~ # pkcs11-tool --module /usr/lib64/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>".
WARNING: user PIN to be changed
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:

`--module <arg>`	путь к библиотеке pkcs11 (обязательный параметр)
`--keypairgen`	генерация ключевой пары
`-- key-type <arg>`	задает тип и длину ключа. В нашем случае тип – rsa, длина - 2048 бит (с длиной ключа 1024 бит возникают проблемы)
`-l`	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 подгружаем модуль поддержки pkcs11:

Для 32-битной версии используйте команду:

```
$ openssl
OpenSSL>  engine dynamic -pre SO_PATH:/usr/lib64/openssl/engines-1.1/pkcs11.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-1.1/pkcs11.so -pre ID:pkcs11 -pre LIST_ADD:1 -
pre LOAD -pre MODULE_PATH:/usr/lib64/librtpkcs11ecp.so
```

```
host-128 ~ # openssl
OpenSSL> engine dynamic -pre SO_PATH:/usr/lib64/openssl/engines-1.1/pkcs11.so -pre ID:pkcs11 -pre LIST_ADD:1 -pre LOAD -pre MODULE_PATH:/usr/lib64/librtpkcs11ecp.so
(dynamic) Dynamic engine loading support
[Success]: SO_PATH:/usr/lib64/openssl/engines-1.1/pkcs11.so
[Success]: ID:pkcs11
[Success]: LIST_ADD:1
[Success]: LOAD
[Success]: MODULE_PATH:/usr/lib64/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
```

```
host-128 ~ # openssl
OpenSSL> engine dynamic -pre SO_PATH:/usr/lib64/openssl/engines-1.1/pkcs11.so -pre ID:pkcs11 -pre LIST_ADD:1 -pre LOAD -pre MODULE_PATH:/usr/lib64/librtpkcs11ecp.so
(dynamic) Dynamic engine loading support
[Success]: SO_PATH:/usr/lib64/openssl/engines-1.1/pkcs11.so
[Success]: ID:pkcs11
[Success]: LIST_ADD:1
[Success]: LOAD
[Success]: MODULE_PATH:/usr/lib64/librtpkcs11ecp.so
Loaded: (pkcs11) pkcs11 engine
OpenSSL> req -engine pkcs11 -new -key 0:45 -keyform engine -x509 -out cert.pem
engine "pkcs11" set.
Enter PKCS#11 token PIN for Rutoken ECP <no label>:
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]: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
OpenSSL>
```

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 (DER)

```
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
```

```
host-128 ~ # pkcs11-tool --module /usr/lib64/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>".
WARNING: user PIN to be changed
Please enter User PIN:
Created certificate:
Certificate Object; type = X.509 cert
  label:
  subject:    DN: C=RU, ST=Moscow, L=Moscow, O=Aktiv, OU=Aktiv, CN=alt/emailAddress=alt@mail.ru
  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 -O -l
```

Для 64-битной версии используйте команду:

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

```
rutoken@host-128 ~ $ pkcs11-tool --module /usr/lib64/librtpkcs11ecp.so -O -l
Using slot 0 with a present token (0x0)
Logging in to "Rutoken ECP <no label>".
WARNING: user PIN to be changed
Please enter User PIN:
Public Key Object; RSA 2048 bits
  label:
  ID:         45
  Usage:      encrypt, verify, wrap
Private Key Object; RSA
  label:
  ID:         45
  Usage:      decrypt, sign, unwrap
Certificate Object; type = X.509 cert
  label:
  subject:    DN: C=RU, ST=Moscow, L=Moscow, O=Aktiv, OU=Aktiv, CN=alt/emailAddress=alt@mail.ru
  ID:         45
```

9 Включаем аутентификацию по внешнему носителю

Потребуются права суперпользователя:

```
$ su

Password:
#

# 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"

10 Редактируем конфигурацию аутентификации в системе

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

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

Для редактирования можно воспользоваться редактором pluma

```
# pluma /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
```

11 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 pluma

```
# pluma /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/security/pam_pkcs11/cacerts;
    crl_dir = /etc/security/pam_pkcs11/crls;
    cert_policy = signature;
  }

  use_mappers = opensc;
  mapper_search_path = /lib/pam_pkcs11;

# Search certificates from $HOME/.eid/authorized_certificates to match users
mapper opensc {
debug = false;
module = /lib64/pam_pkcs11/opensc_mapper.so;
}
}
```

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;

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

Добавляем сертификат в список доверенных сертификатов

```
$ mkdir ~/.eid
$ chmod 0755 ~/.eid
$ cat cert.pem >> ~/.eid/authorized_certificates
$ chmod 0644 ~/.eid/authorized_certificates
```

13 Проверяем 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.

```
rutoken@host-128 ~ $ sudo login
Добро пожаловать, Rutoken!
Введите PIN-код токена Rutoken:
```

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

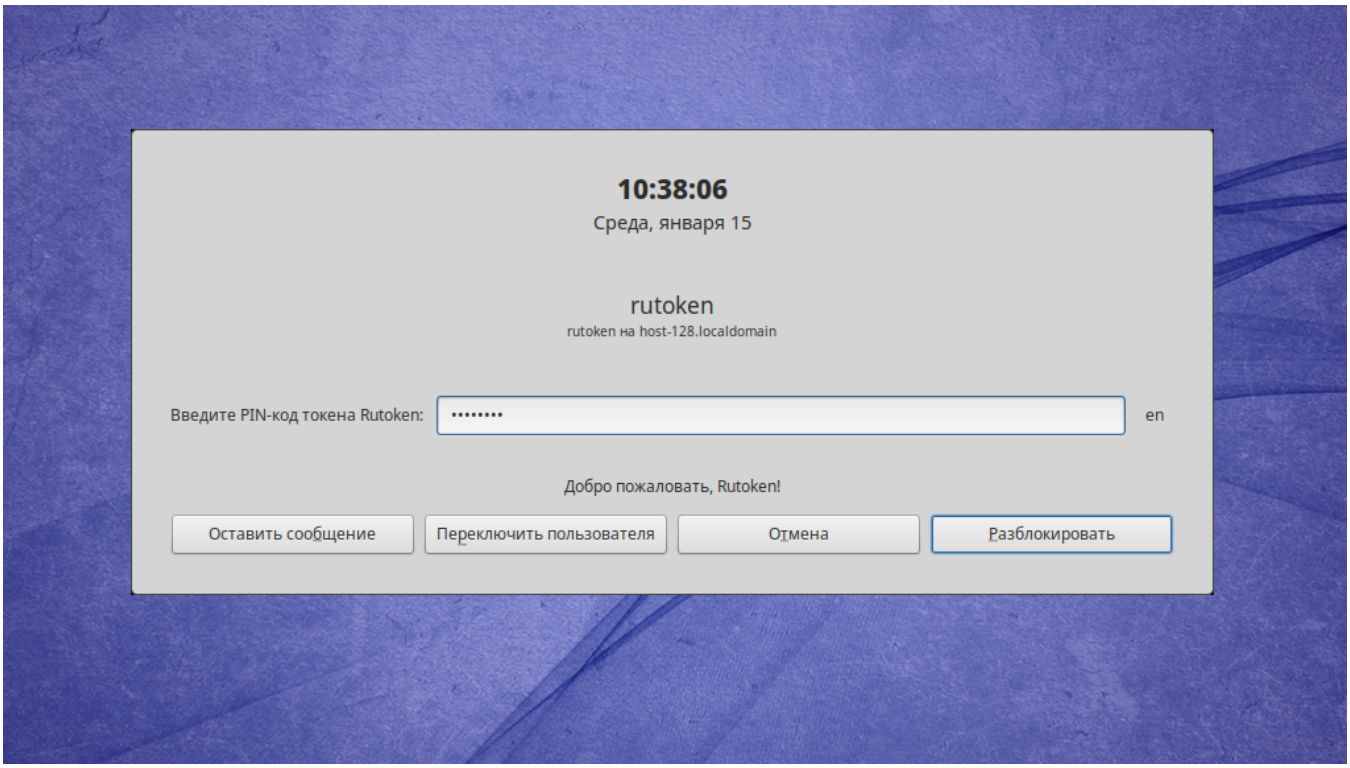
Для этого:

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.

14 Настройка завершена!

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



15 Другие пользователи

При необходимости добавить вход по токену 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.