Подпись пакетов с помощью GPG и ключей хранящихся на токене

Описание проблемы

В настоящий момент GPG напрямую не поддерживает работу с любым видом смарт-карт и может работать только с теми, что поддерживают интерфейс openPGP, что накладывает ограничения работы с остальными смарт-картами. Но для карт, которые поддерживают интерфейс pkcs11, есть решение, заключающееся в использовании демона pkcs11 смарт-карт, вместо стандартного демона scdaemon. Таким образом, при установке и правильной настройке, мы можем сделать возможно использование в GPG ключей и сертификатов, хранящихся, например, на Рутокен ЭЦП 2.0.

Ниже представлено описание настройки и использование этих ключей для создания цифровой подписи для пакетов.

Настройка GPG и gnupg-pkcs11-scd

Установите необходимые пакеты (используйте строчку в зависимости от системы, в которой вы работаете. Первая -- для red hat, вторая -- для debian)

```
sudo yum instal gnupg2 gnupg-pkcs11-scd

sudo apt-get install gpg
```

Проверьте, что версия gpg >= 2.1.19.

Для debian может потребоваться скачать и установить последнюю релизную версию демона pkcs11 смарт-карт gnupg-pkcs11-scd из репозитория.

Установите librtpkcs11ecp.so из пакета с нашего сайта.

Проверьте, что на токене находятся только RSA ключи, так как наличие GOST ключей сделает невозможным работу с токеном.
Лучшим вариантом будет удалить все ключи с токена и создать только одну пару RSA ключей и сертификат для него (самоподписанный или подписанный другим сертификатом).

В файле ~/.gnupg/gpg-agent.conf запишите:

```
scdaemon-program /usr/bin/gnupg-pkcs11-scd
```

Во втором файле ~/.gnupg/gnupg-pkcs11-scd.conf напишите:

```
providers rutoken

provider-rutoken-library /usr/lib64/librtpkcs11ecp.so
```

Отключите gpg-agent:

```
sudo killall gpg-agent
```

Проверьте, что токен откликается:

```
gpg --card-status
```

Правильный вывод должен выглядеть примерно так:

```
[lo1ol@localhost .gnupg]$ gpg --card-status

Application ID ...: D2760001240111503131CAE8D55A1111

Version ..........: 11.50

Manufacturer .....: unknown

Serial number ....: CAE8D55A

Name of cardholder: [not set]

Language prefs ...: [not set]

Sex ..............: unspecified

URL of public key : [not set]

Login data .......: [not set]

Signature PIN ....: forced

Key attributes ...: 1R 1R 1R

Max. PIN lengths .: 0 0 0

PIN retry counter : 0 0 0

Signature counter : 0

Signature key ....: [none]

Encryption key....: [none]

Authentication key: [none]

General key info..: [none]
```

Если отклика не происходит, попробуйте обратиться к демону через gpg-agent. Запустите gpg-agent сервер и подключитесь к демону:

```
gpg-agent --server
SCD LEARN
```

Если вы получили ошибку "Bad certificate", скорее всего, на токене лежат сертификаты, не поддерживаемые GPG.

Правильный вывод выглядит следующим образом:

```
[lo1ol@localhost .gnupg]$ gpg-agent --server

OK Pleased to meet you

SCD LEARN

....

S KEYPAIRINFO 892E053AE031FC23F3E7CCC73BC60859F11F6B90 Aktiv\x20Co\x2E/Rutoken\x20ECP/3ac67ae9
/Rutoken\x20ECP\x20\x3Cno\x20label\x3E/45

OK
```

Регистрация ключей с токена в GPG

Узнаем хэш сертификата на токене, его можно узнать запустив:

```
gpg-agent --server
SCD LEARN
```

В одной из выведенных строк, начинающейся на S KEYPAIRINFO, сразу напротив нее должен отобразиться хэш. Например, в строчке:

```
S KEYPAIRINFO 892E053AE031FC23F3E7CCC73BC60859F11F6B90 Aktiv\x20Co\x2E/Rutoken\x20ECP/3ac67ae9
/Rutoken\x20ECP\x20\x3Cno\x20label\x3E/45
```

Значение хеша будет **892E053AE031FC23F3E7CCC73BC60859F11F6B90**. Запомним его. Теперь зарегистрируем сертификат и его ключи в GPG.

Для этого введем команду:

```
gpg --expert --full-generate-key
```

Далее нам будет предложена какой тип ключа мы хотим получить. Выбираем импорт существующего RSA ключа (13 опция):

```
lo1ol@lo1ol-VirtualBox:~$ gpg --expert --full-generate-key

gpg (GnuPG) 2.2.4; Copyright (C) 2017 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.

Please select what kind of key you want:

   (1) RSA and RSA (default)

   (2) DSA and Elgamal

   (3) DSA (sign only)

   (4) RSA (sign only)

   (7) DSA (set your own capabilities)

   (8) RSA (set your own capabilities)

   (9) ECC and ECC

  (10) ECC (sign only)

  (11) ECC (set your own capabilities)

  (13) Existing key

Your selection? 13
```

Теперь нас запросят ввести хеш сертификата данного ключа:

```
Enter the keygrip: 892E053AE031FC23F3E7CCC73BC60859F11F6B90
```

Далее нам предложат выбрать опции ключа и указать логин и email пользователя ключа. Выберите опции ключа как указано в примере, а логин и e-mail, естественно, используйте свои:

```
Possible actions for a RSA key: Sign Certify Encrypt Authenticate

Current allowed actions: Sign Certify Encrypt

   (S) Toggle the sign capability

   (E) Toggle the encrypt capability

   (A) Toggle the authenticate capability

   (Q) Finished

Your selection?

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)

Key does not expire at all

Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.

Real name: lo1ol

Email address: lo1ol@example.com

Comment:

You selected this USER-ID:

    "lo1ol <lo1ol@example.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O

       .

gpg: /home/lo1ol/.gnupg/trustdb.gpg: trustdb created

gpg: key 676E42AAAFBCF227 marked as ultimately trusted

gpg: directory '/home/lo1ol/.gnupg/openpgp-revocs.d' created

gpg: revocation certificate stored as '/home/lo1ol/.gnupg/openpgp-revocs.d
/0CD2B9CEE398990609D6C164676E42AAAFBCF227.rev'

public and secret key created and signed.

pub   rsa2048 2019-10-25 [SCE]

      0CD2B9CEE398990609D6C164676E42AAAFBCF227

uid                      lo1ol <lo1ol@example.com>
```

Подпись deb пакетов

Скачиваем пакет для создания и верификации подписей dpkg-sig:

```
sudo apt-get install dpkg-sig
```

После этого выбираете любой пакет, который вы хотите подписать, возьмем для примера пакет с библиотекой librtpkcs11ecp.so.

Приступим, узнаем id нашей подписи в gpg. Для этого введем команду:

```
gpg --list-sigs
```

Вывод будет примерно следующим:

```
lo1ol@lo1ol-VirtualBox:~/Downloads$ gpg --list-sigs

/home/lo1ol/.gnupg/pubring.kbx

------------------------------

pub   rsa2048 2019-10-25 [SCE]

      0CD2B9CEE398990609D6C164676E42AAAFBCF227

uid           [ultimate] lo1ol <lo1ol@example.com>

sig 3        676E42AAAFBCF227 2019-10-25  lo1ol <lo1ol@example.com>
```

Из вывода мы видим, что идентификатор подписи **676E42AAAFBCF227**. Теперь приступим к подписи пакета. Для этого введем команду:

```
dpkg-sig -k 676E42AAAFBCF227 --sign builder librtpkcs11ecp_1.9.15.0-1_amd64.deb
```

Как видно, после опции -k указывается id подписи. Опция --sign -- указывает роль того, кто сделал подпись, в нашем случае, это тот, кто собирал пакет. Проверить подпись можно с помощью команды:

```
dpkg-sig --verify librtpkcs11ecp_1.9.15.0-1_amd64.deb
```

Подпись rpm пакетов

Скачиваем пакет для создания и верификации подписей dpkg-sig:

```
sudo apt-get install rpm-sign
```

После этого выбираете любой пакет, который вы хотите подписать. Возьмем для примера пакет с библиотекой librtpkcs11ecp.so:

Приступим, узнаем id нашей подписи в gpg. Для этого введем команду:

```
gpg --list-sigs
```

Вывод будет примерно следующим:

```
lo1ol@lo1ol-VirtualBox:~/Downloads$ gpg --list-sigs

/home/lo1ol/.gnupg/pubring.kbx

------------------------------

pub   rsa2048 2019-10-25 [SCE]

      0CD2B9CEE398990609D6C164676E42AAAFBCF227

uid           [ultimate] lo1ol <lo1ol@example.com>

sig 3        676E42AAAFBCF227 2019-10-25  lo1ol <lo1ol@example.com>
```

Из вывода мы видим, что идентификатор подписи **676E42AAAFBCF227**. Теперь приступим к подписи пакета. Для этого создадим файл конфигурации .rpmmacros для создании подписи со следующим содержимым:

```
%_signature gpg
%_gpg_name 676E42AAAFBCF227
```

Кстати, в поле %_gpg_name можно записать не id подписи, а юзернейм создателя ключа.

И введем команду:

```
rpm --addsign librtpkcs11ecp-1.9.15.0-1.x86_64.rpm
```

Чтобы проверить файл, экспортируем наш публичный gpg ключ в список доверенных:

```
gpg -a -o ~/RPM-GPG-KEY-test --export 676E42AAAFBCF227
sudo rpm --import ~/RPM-GPG-KEY-test
```

Теперь приступим к самой проверке. Для этого введем команду:

```
rpm -K librtpkcs11ecp-1.9.15.0-1.x86_64.rpm
```

Вывод должен быть следующим:

```
librtpkcs11ecp-1.9.15.0-1.x86_64.rpm: digests signatures
```