Configuring pam_pkcs11 smart card logins for Gentoo 64-bit Linux

Rate this Entry

Posted 07-07-2013 at 12:23 AM by vik
Updated 04-06-2014 at 09:17 PM by vik (Updated howto with pam_pkcs11 support)

If you want to look for compatible smart card readers, check http://wiki.debian.org/Smartcards. For smart cards compatible with opensc, check https://github.com/OpenSC/OpenSC/wik...-USB-tokens%29. You may choose to utilize proprietary smart cards, but this guide only covers opensc-compatible ones.

The majority of the information I used can be found at http://www.gooze.eu and https://github.com/OpenSC/pam_pkcs11/wiki. I ended up purchasing Athena ASECards from http://www.cryptoshop.com/products/s...ypto-card.html and a SCM Microsystems SCR333 smart card reader from https://www.scbsolutions.com/express...4fcf91301737e4. You can also get Athena cards from shop.athena-scs.com but the other place was more responsive to my inquiries. The Athena smart card I purchased is well supported by opensc, https://github.com/OpenSC/OpenSC/wik...ASEPCOS-ASEKey, and the card reader has worked fine so far (supported by kernel ccid driver).

Assumptions: I am not currently utilizing a CRL. Also, this guide utilizes a root CA for certificates, not self-signed certs. You have a few options here:

- create free certificates from http://www.cacert.org
- create your own offline root CA using openssl, and CA.pl.
- create your own offline root CA using TinyCA, GNOMint, etc.

I am not an expert in this area so if you see any glaring security vulnerabilities please let me know.

After installing the card reader, emerge opensc, pcsc-lite, ccid, pam_pkcs11, engine_pkcs11, and gnome-screensaver (unless utilizing a KDE desktop). A brief description of these programs:

- opensc is the interface you will use to communicate with the smart card (pkcs15-init, pkcs15-tool, etc.).
- pcscd is a daemon that allows communication between the smart card reader and smart card.
- ccid is a requirement for the smart card reader driver: it has a userspace component I guess.
- pam_pkcs11 is what allows you to login with your smart card to Linux. An alternative is pam_p11 which has less functionality but is suitable for simpler setups: no CA and no need to lock screen when card is pulled.
- engine_pkcs11 is a library that allows you to generate keys/certs from your smart card in openssl. I'm not using this functionality for this article.
- gnome-screensaver is used to present a lock screen when the card is removed. I tried xscreensaver but the sequence to unlock the screen with a card was too cumbersome (you have to press enter 5 times).

Get pcscd running with

Code:
```
/etc/init.d/pcscd start
```

To start every boot:

Code:
```
rc-update add pcscd default
```

If you have issues, you will get better error messages with:

Code:
```
/etc/init.d/pcscd stop; pcscd -a -d -f
```

. If you insert a card and see "Card ATR: ..." the reader is working. You can also try running

Code:
```
opensc-tool --list-readers
```

and you should get something like this: "SCM Microsystems Inc. SCR33x USB Smart Card Reader [CCID Interface]."

Now that the smart card reader is working, initialize the smart card. There are several ways to do this as documented here http://www.gooze.eu/howto/smartcard-...om-pkcs12-file. As I am utilizing a CA, the CA will create the keys/certs and give me a pkcs12 file. I create 2 different keys/certs, one for a normal user and one for root, then store them on the card. Make sure the CN on the certificate matches the username, or else you will have more configuration to do later.

- erase the card.

  Code:
  ```
  pkcs15-init -E -T
  ```

- initialize the card. If you don't provide your pin and puk on this line it will error out (you can erase your bash history later).

  Code:
  ```
  pkcs15-init -C -T -p pkcs15+onepin --pin <your_pin> --p
  ```

- extract the CA cert:

  Code:
  ```
  openssl pkcs12 -in mycert.p12 -cacerts -nokeys > /etc/p
  ```

- make CA cert world readable and initialize with pam_pkcs11:

  Code:

```
cd /etc/pam_pkcs11/cacerts
chmod a+r cacert.pem
pkcs11_make_hash_link
```

- in your home directory,

  Code:
  ```
  mkdir .eid; cd .eid
  ```

- extract the private key from the .p12 file:

  Code:
  ```
  openssl pkcs12 -in mycert.p12 -nocerts > mykeyenc.pem
  ```

- open mykeyenc.pem and see if it says ENCRYPTED KEY somewhere. if it doesn't, then:

  Code:
  ```
  mv mykeyenc.pem mykey.pem
  ```

- if private key is encrypted, then decrypt with:

  Code:
  ```
  openssl rsa -in mykeyenc.pem -out mykey.pem
  ```

- now extract my cert with the ca chain intact:

  Code:
  ```
  openssl pkcs12 -in mycert.p12 -nokeys > authorized_cert
  ```

- open the authorized_certificates file and see if your certificate has the CA certificate included. if not, combine it with your current certificate, then continue with the next step.
- Figure out what the auth ID is (usually 01):

  Code:
  ```
  pkcs15-tool --list-pins
  ```

- Now store the private key on the card:

  Code:
  ```
  cd ~/.eid
  pkcs15-init --store-private-key mykey.pem --auth-id <au
  ```

- get the key id for the next step (should be a very long string)

  Code:
  ```
  pkcs15-tool --list-keys
  ```

- Store the certificate on the card:

  Code:
  ```
  pkcs15-init --store-certificate authorized_certificates
  ```

- repeat these steps with your root user, starting from extracting the info from the pkcs12 file.

Now for pam_pkcs11 configuration. Open /etc/pam_pkcs11/pam_pkcs11.conf and change these sections:

Code:

```
pkcs11_module opensc {
...
cert_policy = ca,signature
...
}

# use_mappers = opensc, openssh, digest, cn, pwent, uid, ma
use_mappers = opensc, openssh, null;
```

Now we need to tell pam to start using smart card certificates. This will vary per distro as Debian uses common-auth, Gentoo uses /etc/pam.d/system-auth, etc.:

Code:
```
auth required pam_env.so
auth [success=1 default=ignore] pam_pkcs11.so
auth required pam_unix.so try_first_pass likeauth nullok
...
```

To test, make sure this prompts for your smartcard pin:

Code:
```
su
```

If it doesn't work correctly, add debug info to /etc/pam_pkcs11/pam_pkcs11.conf:

Code:
```
pam_pkcs11 {
...
debug = true;
...
}

mapper opensc {
debug = true;
...
}
```

Run su again. You should see "certificate is valid and matches the user", "signature is valid", and "pam_sm_setcred() called."

Cleanup:

- for each user, remove private keys laying around:

  Code:
  ```
  cd ~/.eid
  rm mykey.pem mykeyenc.pem
  ```

- for each user, change certs to read-only

  Code:
  ```
  chmod 400 authorized_certificates
  ```

- for each user, cleanup bash history:

  Code:
  ```
  history -c
  ```

- remove debug from /etc/pam_pkcs11/pam_pkcs11.conf:

  Code:

```
pam_pkcs11 {
...
debug = false;
...
}

mapper opensc {
debug = false;
...
}
```

For card screensaver lock:

- edit /etc/pam_pkcs11/pkcs11_eventmgr.conf

  Code:
  ```
  pkcs11_eventmgr {
          # Run in background? Implies debug=false if t
          daemon = true;

          # show debug messages?
          debug = false;

          # polling time in seconds
          polling_time = 3;

          # expire time in seconds
          # default = 0 ( no expire )
          expire_time = 0;

          # pkcs11 module to use
          pkcs11_module = /usr/lib64/opensc-pkcs11.so;

          #
          # list of events and actions

          # Card inserted
          event card_insert {
                  # what to do if an action fail?
                  # ignore  : continue to next action
                  # return  : end action sequence
                  # quit    : end program
                  on_error = ignore ;

                  # You can enter several, comma-separa
                  # they will be executed in turn
                  action = "gnome-screensaver-command -
          }
  ```

- Autostart pkcs11_eventmgr so it will sense smart card insert/removal. Create /etc/xdg/autostart/smartcard-screensaver.desktop:

  Code:
  ```
  [Desktop Entry]
  Type=Application
  Name=Smart Card Screensaver
  Comment=Application to lock screen on smart card remova
  Exec=/usr/bin/pkcs11_eventmgr daemon
  ```

- for LXDE, make sure the script is executed by running lxsession-edit and check the box. Also, check that PolicyKit Authentication Agent (needed by gnome-screensaver) and Screensaver are checked.

Logout and log back in. When your login manager comes up insert your smart card and enter your pin in the password field. For su enter your pin (it should ask for a passcode not a password). If it all works then feel free to change passwords on your accounts to something long and random.

SSH:
You don't have to install anything special on the SSH server like installing pam_p11 or opensc. SSH already supports key exchanges so just do this:

- Code:
  ```
  cd ~/.ssh
  ```

- find the ID that matches up with the ID you generated for your user account earlier.

  Code:
  ```
  pkcs15-tool --list-public-keys
  pkcs15-tool --read-ssh-key <id of this user's key> > au
  ```

- sftp into the server and put the authorized_keys file in ~/.ssh. chmod 600 authorized_keys
- This should permit you to ssh into this computer and authenticate with the key stored on your smart card:

  Code:
  ```
  ssh -I /usr/lib64/opensc-pkcs11.so -v <username>@<host>
  ```

  You should see something like this:
  debug1: Authentications that can continue: publickey, password
  debug1: Next authentication method: publickey
  debug1: Offering public key: /usr/lib/opensc-pkcs11.so
  debug1: Server accepts key:
- If the server won't accept your key, chances are you did this earlier: pkcs15-tool --read-public-key instead of pkcs15-tool --read-ssh-key.
- To make permanent, add this line to your /etc/ssh/ssh_config:

  Code:
  ```
  PKCS11Provider /usr/lib64/opensc-pkcs11.so
  ```

Firefox:
Unfortunately, very few sites let you utilize a certificate for login purposes. This article will help you configure Firefox to work with them: https://www.opensc-project.org/opensc/wiki/MozillaSteps. Substitute /usr/lib64/opensc-pkcs11.so in the path to the .so file and you can then bring up the certificates on your smart card.

Possibly Helpful Tips:

- Unlock a card if you forget the PIN but not the PUK PIN:

  Code:
  ```
  pkcs15-tool -u
  ```

- Figure out which certificate ID corresponds to each user on the card:

  Code:
  ```
  pkcs15-tool --list-certificates
  pkcs15-tool -r <id_of_certificate> | openssl x509 -text
  ```

- If you are in a desktop environment, ctrl-alt-f1 to a terminal, and try to login but get "Wrong smartcard PIN": chances are you don't have numlock enabled in the terminal. To check, try typing in numbers in the login prompt on the numpad.

Posted in Uncategorized Views 2412 Comments 0

« Prev Main Next »

Comments

Total Comments 0