

Login und Authentifizierung

Nutzung für ssh

Voraussetzungen

Deaktivierung des gpg-agent

Der gpg-agent ist in der Lage, auch ssh-Authentifizierungen zu erledigen - mit den SSH-Schlüsseln, die im GnuPG-Keyring oder auf einer Karte abgelegt sind. Dafür muss sich der gpg-agent auch um die SSH-Anfragen kümmern. Erledigt wird das über die Umgebungsvariable `SSH_AUTH_SOCK` die dann z.B. folgenden Wert haben muss: `/run/user/1000/gnupg/S.gpg-agent.ssh`.

In diesem Fall darf aber der ssh-agent nicht gestartet werden, da dieser die Umgebungsvariable überschreiben würde.

Leider gibt es keine einheitliche Funktionsweise, wie die Desktop-Umgebungen den ssh-agent starten.

- einige werten die `/etc/X11/Xsession.options` aus - und suchen dort nach der Zeile `use-ssh-agent` (xfce4, frühere Versionen, Mate?, Gnome?)
- andere haben eine eigene Startroutine
- wieder andere nutzen wohl neuerdings auch den `systemd`.

gnome-keyring-manager darf sich weder für ssh noch für gpg verantwortlich fühlen

Noch komplizierter ist die Lage beim gnome-keyring-manager, der seinerseits wiederum auch eine Funktioalität sowohl für den ssh-agent als auch für den gpg-agent mitbringt. Damit die gnupg-card und die Authentifizierung mit ssh klappt, müssen diese Komponenten deaktiviert sein - bzw. der gnome-keyring-manager darf gar nicht laufen.

- <http://www.ozonesolutions.com/programming/2014/04/pgp-smart-card-ssh-login-gpg-agent-ubuntu/>
- <http://www.bootc.net/archives/2013/06/09/my-perfect-gnupg-ssh-agent-setup/>
- https://grepular.com/Smart_Cards_and_SSH_Authentication

Leider hat sich die Interaktion zwischen gnome-keyring-daemon, gpg-agent und ssh-agent in Debian stretch wiederum geändert. Hinweise dazu gibt es hier:



- <https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=850982>
- <https://blog.josefsson.org/2017/06/19/openpgp-smartcard-under-gnome-on-debian-9-0-stretch/>

In Stretch ist jetzt die user-Session von systemd verantwortlich für den Start des gpg-agent. Die verschiedenen Anleitungen, den gnome-keyring-daemon



darán zu hindern, sich für ssh verantwortlich zu fühlen, funktionieren in unseren Setups (unter Mate und Xfce4 allerdings nicht. Der entscheidende Hinweis steht unter `/usr/share/doc/gnupg-agent/README.Debian`

Vorgehen

*ssh-Support im gpg-agent konfigurieren `echo „enable-ssh-support“» ~/.gnupg/gpg-agent.conf` **Gleichzeitig soll verhindert werden, dass der ssh-agent beim Starten einer Desktop-Umgebung startet. Dazu editieren wir die `/etc/X11/Xsession.options` und kommentieren `#use-ssh-agent` aus.** Konflikt mit Gnome-Keyring Manager lösen **In manchen (älteren Umgebungen - z.B. xfce4 bis Debian Jessie richt es, folgende Dateien umzubenennen `/etc/xdg/autostart/gnome-keyring-ssh.desktop` `/etc/xdg/autostart/gnome-keyring-gpg.desktop` jeweils um, in dem wir z.B. ein `.disable` anhängen. Wir könnten sie theoretisch auch löschen. In manchen [Howtos](#) wird empfohlen `apt-get remove libpam-gnome-keyring` zu deinstallieren. Das scheint nicht zwingend notwendig. Auch ein `gconftool-2 -set -t bool /apps/gnome-keyring/daemon-component/ssh false` wird in anderen Howtos genannt, scheint aber nicht in allen Umgebungen nötig zu sein. ==== Einzelne Desktop-Umgebungen ==== XFCE-Desktop in Debian Stretch === Da xfce4 den gnome-keyring-Daemon nicht automatisch startet, gibt es hier keine Probleme. Lediglich der ssh-agent muss vom Starten abgehalten werden. Das lässt sich auf User-Ebene mit folgendem Kommando erreichen: `xfconf-query -c xfce4-session -p /startup/ssh-agent/enabled -n -t bool -s false` Per Default ist diese Eigenschaft nicht gesetzt. Klappt alles nicht, kann ein `SSH_AUTH_SOCK=/run/user/$(id -u)/gnupg/S.gpg-agent.ssh` in die `~/.bashrc` geschrieben werden. Damit wird die Umgebungsvariable gesetzt, mit dem ssh dann den gpg-agent findet, der für die Authentifizierung zuständig sein soll. === Mate-Desktop in Debian Stretch=== Mate scheint den gnome-keyring-manager autonom zu starten - ohne Berücksichtigung der Startskripte unter `/etc/xdg/autostart`. Daher kann Abhilfe auf User-Ebene Abhilfe geschaffen werden: `gsettings set org.mate.session gnome-compat-startup „[‘smproxy’]“` Sieht man sich vorher mit `gsettings get org.mate.session gnome-compat-startup` die Einstellungen an, sieht man, dass nun keyring verschwunden ist. Zusätzlich deaktiviert mensch mit `mate-session-properties` den Autostart vom „GPG-Passwort-Agent“. Nach dem nächsten Neustart sollte der gpg-agent ohne gnome-keyring-manager funktionieren.**

<http://william.shallum.net/random-notes/disabling-gnome-keyring-daemon-ssh-agent-on-mate-desktop> ==== ssh über mehrere Server (ForwardAgent) ==== *

<http://livecipher.blogspot.co.uk/2013/02/ssh-agent-forwarding.html> Damit das funktioniert, muss der ssh-client so konfiguriert werden, dass er „agent-forwarding“ unterstützt: `* /etc/ssh/ssh_config (systemweit) * /home/user/.ssh/config (per user) <code> Host * ForwardAgent yes </code>` Eine ausführliche Beschreibung findet sich hier: *

<http://www.unixwiz.net/techtips/ssh-agent-forwarding.html> ==== Innerhalb von screen-sessions ==== Hier steht was dazu: *

<http://www.bootc.net/archives/2013/06/09/my-perfect-gnupg-ssh-agent-setup/> Aber ab debian-wheezy sollte das out of the box funktionieren. ==== ssh über mehrere Server (ForwardAgent) ==== *

<http://livecipher.blogspot.co.uk/2013/02/ssh-agent-forwarding.html> Damit das funktioniert, muss der ssh-client so konfiguriert werden, dass er „agent-forwarding“

unterstützt: `*/etc/ssh/ssh_config (systemweit) */home/user/.ssh/config (per user) <code> Host * ForwardAgent yes </code>` Eine ausführliche Beschreibung findet sich hier: <http://www.unixwiz.net/techtips/ssh-agent-forwarding.html> ==== Innerhalb von screen-sessions ==== Hier steht was dazu: <http://www.bootc.net/archives/2013/06/09/my-perfect-gnupg-ssh-agent-setup/> Aber mit debian-wheezy sollte das out of the box funktionieren. ===== Login in lokalen Linux-Rechner ===== Die Linux-Pam und die verschiedenen Session-Manager können mit GnuPG-Card umgehen. <http://www.schiessle.org/howto/poldi.php> Die dort verwendeten Optionen für poldi-ctrl scheinen in der wheezy-Version noch nicht zu existieren. Folgendes Information hilft weiter: <http://walter.silvergeeks.com/rechner/howto/how-to-anmeldung-mit-smart-card-oder-usb-t-oken-am-lokalen-linux-system/> ===== Einrichten des Poldi-Moduls: ===== (alle Kommandos als Root oder mit sudo) Das Packet muss installiert werden: `<code> apt-get install libpam-poldi </code>`

Unter Debian Jessie findet poldi das Binary von sddaemon nicht mehr. Ein



```
ln -s /usr/lib/gnupg2/sddaemon
/usr/bin/sddaemon
```

als Root schafft einfache Abhilfe.

Ein `<code> poldi-ctrl -d </code>` testet, ob die Karte erkannt wird. Mit `<code> poldi-ctrl -s </code>` finden wir die Seriennummer der Karte heraus (gpg `-card -status` würde auch funktionieren). Jetzt müssen wir unter `/etc/poldi/localdb/users` eine Kartei der User anlegen, für die die Authentifizierung mit der jeweiligen Karte eingerichtet werden soll. Z.B. mit: `<code> echo „`poldi-ctrl -s` USERNAME“ | tee -a /etc/poldi/localdb/users </code>` Unter `/etc/poldi/localdb/keys/` muss jetzt für jede Karte eine Datei mit der Seriennummer als Dateiname angelegt werden, die den public Key der Karte enthält. Das geht einfach mit: `<code> poldi-ctrl -s> serialno.txt poldi-ctrl -k | tee -a /etc/poldi/localdb/keys/$(cat serialno.txt) </code>` Damit sollte Poldi fertig konfiguriert sein. ===== Einrichten der Authentifizierung ===== Nach der grundlegenden Einrichtung der Smartcard, kann das Pam-Poldi-Modul zur `/etc/pam.d/common-auth` hinzugefügt werden. Alle Authentifikationen, die auf die `common-auth` zurückgreifen (u.a. `xscreensaver`, `sudo`) können dann auch via smartcard authentifizieren. Wenn das Modul am Anfang der `common-auth` hinzugefügt wird und alle anderen Einträge bleiben, ist die GnuPG Card ein zusätzliches** Authentifizierungsmittel. Bitte immer testen, so lange noch eine offene Konsole verfügbar ist, um die Änderungen ggf. zu korrigieren. Sonst sperrt mensch sich leicht aus.

```
--- a/pam.d/common-auth
+++ b/pam.d/common-auth
@@ -13,6 +13,8 @@
 # pam-auth-update to manage selection of other modules. See
 # pam-auth-update(8) for details.

+# with smartcard
+auth sufficient pam_poldi.so
 # here are the per-package modules (the "Primary" block)
```

```
auth [success=1 default=ignore] pam_unix.so nullok_secure  
# here's the fallback if no module succeeds
```

From:

<https://wiki.datenkollektiv.net/> - **datenkollektiv.net**

Permanent link:

https://wiki.datenkollektiv.net/public/gnupg/gnupg-card_auth?rev=1505378984

Last update: **2017/09/14 10:49**

