Importieren von GPG-Schlüsseln

Der Grund

Am 2008-01-23 hat sich das openSUSE-Projekt entschieden, statt eines gemeinsamen Schlüssels für alle Projekte einen separaten Schlüssel für jedes Projekt zu verwenden. Die Folge für den Anwender ist, dass nach dem ersten Update eines Projekts ein neuer Schlüssel importiert werden muss. Das kann je nach Anzahl der benutzen Installationsquellen langwierig sein...

Der openSUSE-Builder Bernhard Walle hat auf seiner Website unter http://www.bwalle.de/programme/scripts/smart_fetch_keys_buildservice ein Python-Script hinterlegt, das die Keys aller in Smart definierten Repositories auf opensuse.org aktualisiert. Leider wirft das Script bei mir Fehler  – da habe ich ein Shellscript geschrieben, das die selbe Aufgabe lösen soll:

Das Script

```
#!/bin/bash

# new 2008-04-23: added switch between zypper and smart repositories
PARA=$(echo "$1" | tr /A-Z/ /a-z/)
case "$PARA" in
  z|zy|zyp|zypp|zyppe|zypper)
    URLSOURCE="ZYPP"
    ;;
  *)
    URLSOURCE="SMART"
    ;;
esac

TEMPREPO="/tmp/search.repo"
TEMPKEY="/tmp/keyfile"

# the base URL we search on
# new: use more than one base URL for your repositories:
URLLIST="http://download.opensuse.org/repositories/
ftp://ftp5.gwdg.de/pub/opensuse/repositories/"
URLLIST="$URLLIST http://software.opensuse.org/download/"

for SOS_URL in $URLLIST; do
  echo "Looking for ${URLSOURCE}-REPOs on $SOS_URL"
  SOS_LEN=$(expr length "$SOS_URL")
  # only URLs containing $SOS_URL please:
  if [ "$URLSOURCE" = "SMART" ]; then
    URLLIST=$(smart channel --show | grep ^baseurl | cut -d' ' -f3 | grep
"$SOS_URL" | sort)
```

```
  else
    URLLIST=$(grep -r ^baseurl /etc/zypp/repos.d/ | cut -d'=' -f2 | grep
"$SOS_URL" | sort)
  fi

  for URL in $URLLIST; do
    # make sure we have a trailing slash
    echo "$URL" | grep \/$ >/dev/null 2>&1 || URL="$URL/"

    # inside the directory should be a .repo file
    # so we try to find its name
    # substring handling is somewhat #+@%$&# in bash...
    URLAST=${URL#"$SOS_URL"}
    URLAST=$(echo "$URLAST" | rev | cut -d'/' -f 3- | rev | tr -d '/')

    # ...finally...
    rm -f "$TEMPREPO"
    wget -q "${URL}${URLAST}.repo" -O "$TEMPREPO" 2>&1 >/dev/null
    # REPO file exists and is not zero sized?
    if [ ! -f "$TEMPREPO" -o ! -s "$TEMPREPO" ]; then
      echo "Error getting REPO file for $URLAST from $URL"
      continue
    fi

    # now we read the URL of the keyfile from the repo file
    KEYURL=$(grep ^gpgkey "$TEMPREPO" | cut -d'=' -f 2)
    if [ -z "$KEYURL" ]; then
      echo "No key for $URLAST detected"
      continue
    fi

    # download it...
    rm -f "$TEMPKEY"
    wget -q "$KEYURL" -O "$TEMPKEY" 2>&1 >/dev/null
    if [ ! -f "$TEMPKEY" ]; then
      echo "Error getting keyfile $KEYURL for $URLAST"
      continue
    fi

    # identify it, maybe it is already there
    KEYID=$(gpg "$TEMPKEY" | cut -d'/' -f 2 | cut -d' ' -f 1 | tr 'A-Z' 'a-
z')
    # at first, we look inside the rpm database
    RPMINSTALL=0
    LANG=C rpm -q "gpg-pubkey-$KEYID" 2>/dev/null | grep 'is not installed'
>/dev/null 2>&1 && RPMINSTALL=1
    # next, we look at the gpg keyring
    GPGINSTALL=0
    gpg --list-keys "$KEYID" >/dev/null 2>&1 || GPGINSTALL=1
    # so, at the very end, import it - or not :-)
    if [ $RPMINSTALL -eq 1 ]; then
```

```
        echo -n "Importing key $KEYID for $URLAST into RPM database"
        rpm --import "$TEMPKEY"
    else
        echo -n "Key $KEYID for $URLAST already in RPM database"
    fi
    if [ $GPGINSTALL -eq 1 ]; then
        echo ", importing into GPG keyring now"
        gpg --import "$TEMPKEY" >/dev/null 2>&1
    else
        echo " and present in GPG keyring"
    fi
  done
done
```

Hinweise

Der Code hat den Vorteil, dass ein nicht vorhandener Key durch den temporären Key abgefangen wird und nicht wie bei bwalles Script zum Abbruch führt...

Dieses Script bearbeitet alle Repositories, die in Smart definiert sind (unabhängig davon, ob sie als `disabled` markiert sind) und in deren `baseurl` eine bestimmte URL vorkommt. Ursprünglich wurde SOS_URL direkt gesetzt, aber jetzt habe ich eine Schleife über mehrere URLs eingebaut. Die Basis-URLs werden am Anfang in der Variablen URLLIST gespeichert und dann nacheinander abgearbeitet. Wer mehr URLs unterbringen will als in eine Zeile passen, kann die Variable ja „zeilenweise bauen“:

```
URLLIST="http://software.opensuse.org/repositories/"
URLLIST="$URLLIST http://ftp5.gwdg.de/pub/opensuse/repositories/"
URLLIST="$URLLIST ftp://ftp5.gwdg.de/pub/opensuse/repositories/"
```

usw. Gemäß den Regeln für for-Schleifen der Bash  werden die Einträge in URLLIST durch Leerzeichen getrennt.

Achtung: weil auch deaktivierte Repositories von Smart ausgegeben werden, kann es hier zu Fehlermeldungen kommen – das Script läuft allerdings trotzdem durch, dann halt mit dem nächsten Repository  .

Änderung 2008-04-19: die Keys werden jetzt parallel im GPG-Keyring installiert. Das weiß man zu schätzen, wenn man auch apt benutzt 

Änderung 2008-04-23: wird ein Parameter mitgegeben, der eindeutig auf „zypper“ schließen lässt  (siehe erste Abfrage im Script), werden die zu prüfenden Repositories aus dem Bestand von zypper geholt anstelle von Smart. Das dürfte erst ab openSUSE 10.3 möglich sein, weil die Daten unter 10.1/10.2 unter /var/lib/zypp liegen. Evtl. hilft es schon, die URL anzupassen – ich habe keine Maschine mit diesen OSsen mehr.

Diese Seite ist auch in english verfügbar

2008-06-22 Unter der kurzen, leicht zu merkenden URL http://en.opensuse.org/SDB:YOU_or_RPM_Report_Problems_Verifying_Package_Signatures habe ich gefunden, dass es evtl. mit einem einfachen `gpg --import` nicht getan ist – dort findet sich etwa `gpg --no-options --no-default-keyring --keyring /usr/lib/rpm/gnupg/pubring.gpg --import /mnt/pubring.gpg`, um den Key zu importieren. Nun, wir werden beobachten...

2008-06-27 So, nach etlicher Beobachtung habe ich den Schluss des Codes etwas umgestaltet:

```
# identify it, maybe it is already there
KEYID=$(gpg "$TEMPKEY" | cut -d'/' -f 2 | cut -d' ' -f 1 | tr 'A-Z' 'a-z')
INSTALLEDKEYS=$(LANG=C rpm -q "gpg-pubkey-$KEYID" >/dev/null 2>&1)
RPMINSTALL=0
echo $INSTALLEDKEYS | grep 'is not installed' && RPMINSTALL=1
# look at PGP/GPG keys here
GPGINST1=0
gpg --list-keys "$KEYID" >/dev/null 2>&1 || GPGINST1=1
GPGINST2=0
if [ -f "$OTHERKEYRING" ]; then
    gpg --list-keys --no-default-keyring --keyring "$OTHERKEYRING" "$KEYID"
>/dev/null 2>&1 || GPGINST2=1
else
    GPGINST=5
fi
# so, at the very end, import it - or not :-)
if [ $RPMINSTALL -eq 1 ]; then
    echo -n "Importing key $KEYID for $URLAST into RPM database"
    rpm --import "$TEMPKEY"
else
    echo -n "Key $KEYID for $URLAST already in RPM database"
fi
if [ $GPGINST1 -eq 1 ]; then
    echo -n ", importing into default GPG keyring now"
    gpg --import "$TEMPKEY" > /dev/null 2>&1
else
    echo -n " and present in default GPG keyring"
fi
if [ $GPGINST2 -eq 1 ]; then
    echo ", importing into rpm keyring now."
    gpg --no-options --no-default-keyring --keyring "$OTHERKEYRING" --import
"$TEMPKEY" > /dev/null 2>&1
elif [ $GPGINST2 -ne 5 ]; then
    echo ", present in rpm keyring."
else
    echo "."
fi
```

Außenrum sind natürlich noch die beiden do...done-Schleifen! Und bitte nicht vergessen, am Scriptanfang die Variable `OTHERKEYRING=/usr/lib/rpm/gnupg/pubring.gpg` zu initialisieren!

Damit werden die Keys in beide Keyrings gefüttert, sogar nachträglich. Und in einem der beiden nützen sie sogar 🙂