Linuxkernelmodules/Treiber SSL signieren für EFI und Secureboot

Bei neuerer Hardware wird es immer mehr interessanter Treiber auch digital zu signieren. Gerade unter EFI Bios mit oder auch ohne Secureboot wird genau das erwünscht.

Wie sehe ich nun das ich meine Module signieren sollte?

Mal kurz in's Bootlog geschaut:

```
journalctl -r -p3
...
PKCS#7 signature not signed with a trusted key
...
```

Dies beschreibt ein Modul das nicht signiert, also nicht vertraut wird. Dies sind meist Module von proprietären Anbietern wie Grafikartenhersteller, oder auch Oracle Virtualbox.

Signierprozess

Dank eines tollen Projektes auf Github, ist das sehr einfach. Einfach das Script in das eigene Home Downloaden:

create-efi-keys.sh

```
# VERY IMPORTANT! After each kernel update or dkms rebuild the modules
must be signed again with the script
# ~/.ssl/sign-all-modules.sh

# Place all files in ~/.ssl folder
mkdir ~/.ssl
cd ~/.ssl

# Generate custom keys with openssl
openssl req -new -x509 -newkey rsa:2048 -keyout MOK.priv -outform DER -
out MOK.der -nodes  -subj "/CN=Owner/"

# Set more restrictive permisions as these are private keys
chmod 600 MOK.*

# Add the sign-all-modules script to the .ssl folder
cat <<EOT > sign-all-modules.sh
#!/bin/bash

sudo -v
```

```
echo "Signing the following modules"

for filename in /lib/modules/$(uname -r)/updates/*.ko; do
    sudo /usr/src/linux-headers-$(uname -r)/scripts/sign-file sha256
~/.ssl/MOK.priv ~/.ssl/MOK.der $filename

    echo "$filename"
done

for filename in /lib/modules/$(uname -r)/kernel/drivers/char/drm/*.ko;
do
    sudo /usr/src/linux-headers-$(uname -r)/scripts/sign-file sha256
~/.ssl/MOK.priv ~/.ssl/MOK.der $filename

    echo "$filename"
done
EOT

chmod +x ~/.ssl/sign-all-modules.sh

#Run the script
~/.ssl/sign-all-modules.sh

#Add the key to the trusted keys database
sudo apt-get install mokutil
sudo mokutil --import ~/.ssl/MOK.der

cd ~
#Reboot and in the boot screen select add/import key
```

Beim Ersten Aufruf werden die Schlüssel generiert und alle nötgen Kernelmodule signiert. Damit die Zertifikate auch ins BIOS wandern bootet der Computer beim Nächsten start in die Keychain. Hier rollt man das Zertifikate einfach ins BIOS aus :) Danach sind alle Module signiert und der Fehler im Log ist weg.

Kernelupdate, Treiberupdate was tun?

Nach einem Update sind die neuen Module lediglich neu zu signieren. Hiefür führt man das zuvor automatisch erstellte Script `sign-all-modules.sh` im Home unter `~.ssl` aus. Das Script wird anscheinend nicht richtig generiert. Hier ist das funktionierende:

sign-all-modules.sh

```
#!/bin/bash

sudo -v
```

```
echo "Signing the following modules"

for filename in /lib/modules/$(uname -r)/updates/*.ko; do
    sudo /usr/src/linux-headers-$(uname -r)/scripts/sign-file sha256
~/.ssl/MOK.priv ~/.ssl/MOK.der $filename

    echo "$filename"
done

for filename in /lib/modules/$(uname -r)/kernel/drivers/char/drm/*.ko;
do
    sudo /usr/src/linux-headers-$(uname -r)/scripts/sign-file sha256
~/.ssl/MOK.priv ~/.ssl/MOK.der $filename

    echo "$filename"
done
```