(Debian Linux) Script for Signing all DKMS Modules with a Machine Owner Key after Updates

Nov. 5, 2022

In order for some packages to work properly with secure boot enabled on my machine running Debian Linux, I've needed to sign their modules using a Machine Owner Key (MOK) ↗. This works well, but isn't especially robust because the signed modules are replaced with unsigned modules during updates. In order to simplify the process after running `apt upgrade` or similar, I created a script that re-signs all modules with my MOK as needed.

Prerequisites

In order to use this script, you will need to have first generated and enrolled a MOK, by following either the official documentation ↗ or the steps in my post about connecting a DSLR as a webcam. You will also need to have modules that need to be signed, which I assume you do if you're reading this post.

The script

```
#!/bin/bash

# Variables
VERSION="$(uname -r)"
SHORT_VERSION="$(uname -r | cut -d . -f 1-2)"
MODULES_DIR=/lib/modules/$VERSION
KBUILD_DIR=/usr/lib/linux-kbuild-$SHORT_VERSION

# Small helper functions for printing success/error messages
echosuccess() { printf "\e[32m%s\e[0m\n" "$*"; }
echoerr() { printf "\e[01;31m%s\e[0m\n" "$*" >&2; }

# Input the passphrase for the MOK
cd "$MODULES_DIR/updates/dkms"
echo -n "Passphrase for the private key: "  # Give us a friendly
read -s KBUILD_SIGN_PIN  # Store the passphrase for the private
export KBUILD_SIGN_PIN  # Export the variable containing the pas
```

```
# Sign all unsigned modules in the DKMS directory
echo
for module in "$MODULES_DIR/updates/dkms"/*; do
  module_name="$(basename $module .ko)"
  if $(sudo modinfo "$module_name" | grep signature > /dev/null)
  then
    echosuccess "Module '$module_name' already signed"
  else
    echo "Signing module '$module_name'"
    sudo --preserve-env=KBUILD_SIGN_PIN "$KBUILD_DIR"/scripts/si
    sudo modinfo "$module_name" | grep signature > /dev/null &&
  fi
done
```

I have the script saved as `~/bin/sign-modules`; make sure that the script has execution permissions set (e.g. `chmod a+x ~/bin/sign-modules`) and that the script's directory is included in your `$PATH` variable (e.g. by having `PATH="$PATH:~/bin"` in your `~/.bashrc` file).

With all this in place, making sure your modules are signed should be as simple as running `sign-modules` after each time that you run `apt upgrade`.

Comments

You might also like

(Debian Linux) Using a Canon EOS camera as a webcam with secure boot enabled

A guide to connecting a DSLR or other camera to Debian as a webcam while keeping secure boot enabled.

Apple Pie: Baking a MacBook

When display issues killed my 2011 MacBook Pro, I decided the only option left was to toss it in the oven.

Mapping F12 to Play/Pause on a Lenovo Laptop in Windows 10 using Lenovo Vantage

Lenovo laptops have many functions mapped to the function keys on the top row, but (at least on my laptop) media controls are not among them.

Photography ↗