

Secureboot

Voici une page qui regroupe la gestion du secureboot pour les distributions.

SolusOS

<https://help.getsol.us/docs/user/quick-start/installation/secure-boot/>

CachyOS

https://wiki.cachyos.org/configuration/secure_boot_setup/

LinuxMint

<https://forums.linuxmint.com/viewtopic.php?p=2331461&sid=e4acb5bb9c44ece8314322cc6221d100#p2331461>

Toutes les distributions qui utilisent Mokutils.

Vidéo => <https://youtu.be/yvoTlvJ0kJo>

Wiki Adrien => <https://www.linuxtricks.fr/wiki/fedora-signer-les-modules-noyau-tiers-kmod-pour-le-secure-boot>

Introduction

Fedora Linux utilise des signatures numériques approuvées par l'UEFI Secure Boot. Les binaires du chargeur de démarrage (comme GRUB) et le noyau de Fedora sont signés avec des certificats reconnus par les clés de la plateforme UEFI préinstallées sur la plupart des ordinateurs. Par conséquent, il n'est pas nécessaire de désactiver le Secure Boot pour installer et utiliser Fedora Linux.

Cependant, les modules fournis sous forme d'akmods, livrés dans RPM Fusion par exemple tels que NVidia, Broadcom, VirtualBox, ... sont compilés sur la machine et ne sont pas signés. Ils ne sont pas chargés si le Secure Boot est actif (oui, Secure Boot empêche de charger des modules non signés, c'est son rôle).

Plutôt que de désactiver ce mécanisme de sécurité, nous allons voir comment générer notre certificat et nos clés de signature et les importer dans l'UEFI de la machine.

Tous les modules (de type akmod) seront ensuite automatiquement signés lors de la compilation sans intervention manuelle.

Pour info, akmod signifie Auto-Kernel Module. Akmod compile les modules du noyau au démarrage si un nouveau noyau est détecté et aucun module précompilé n'est disponible. Cela assure que les modules sont toujours disponibles, même après une mise à jour du noyau.

Prérequis

Toutes les manipulations seront réalisées avec les droits root.

Dans un premier temps, on installe les outils nécessaires pour la gestion des clés machines, et des modules akmods :

```
dnf install kmodtool akmods mokutil openssl
```

Une fois fait, vous pouvez vérifier le statut du SecureBoot avec la commande :

```
mokutil --sb-state
```

Sur ma machine, il est actif :

```
SecureBoot enabled
```

Exemple avec un module non signé

Cet exemple est pour la démonstration dans ce tutoriel, ne l'installez pas si vous n'en avez pas le besoin !

Dans cet exemple, j'installe juste le module VirtualBox (qui n'est donc pas signé) :

```
dnf install akmod-VirtualBox
```

Je compile dans la foulée le kmod :

```
akmods
```

Lorsque j'essaie de charger le module :

```
modprobe vboxdrv
```

J'ai une erreur, car le module n'est pas signé et reconnu par ma machine :

```
modprobe: ERROR: could not insert 'vboxdrv': Key was rejected by service
```

Génération des certificats et clés

Dans un premier temps, on va générer un certificat et une paire de clés pour signer les modules :

```
kmodgenca -a
```

L'option **-a** permet de générer avec des valeurs par défaut sans poser de questions.

Deux fichiers sont créés :

/etc/pki/akmods/private/private_key.priv : la clé privée pour signer les modules

/etc/pki/akmods/certs/public_key.der : la clé publique pour vérifier la signature

Ce sont en réalité des liens symboliques vers des noms différents suivant la machine, comme on peut le voir sur ma machine :

```
/etc/pki/akmods/certs/public_key.der -> /etc/pki/akmods/certs/superlinux-788348939.der  
/etc/pki/akmods/private/private_key.priv -> /etc/pki/akmods/private/superlinux-788348939.priv
```

Enregistrer les certificats et clés dans l'UEFI

Lançons l'import dans l'UEFI

On va ensuite enregistrer les éléments précédemment générés dans l'UEFI et le module Secure Boot.

Cela permettra au noyau Linux de faire confiance aux pilotes signés avec notre clé :

```
mokutil --import /etc/pki/akmods/certs/public_key.der
```

On va devoir entrer un mot de passe.

Il n'a pas besoin d'être très long ou complexe. On en aura juste besoin pour importer la clé dans le secure boot.

Attention, on sera en qwerty pour la saisie, donc on fera simple !

Une fois fait, on va redémarrer le système :

```
reboot
```

Validons l'import côté UEFI

Lorsque l'UEFI s'initialise, on est intercepté pour ajouter la clé en attente.

secureboot-mok1

On valide par Entrée

Ensuite :

secureboot-mok2

Sélectionner **Enroll MOK**

secureboot-mok3

Puis **Continue**

secureboot-mok4

Puis **Yes**

secureboot-mok5

On saisit le mot de passe renseigné précédemment (rappel on est en qwerty)

secureboot-mok6

Puis **Reboot**

Installer ou réinstaller des modules

Installer de nouveaux modules

Maintenant, tous les modules "akmods" seront compilés et signés automatiquement avec notre clé, que SecureBoot connaît.

Réinstaller des modules existants

Si on avait déjà des modules installés, il suffit de forcer leur recompilation :

```
akmods --rebuild --force
```

Dans l'exemple de ce tuto avec le module VirtualBox :

```
Checking kmods exist for 6.9.8-200.fc40.x86_64      [ OK ]  
Building and installing VirtualBox-kmod             [ OK ]
```

On régénère aussi l'image initrd avec :

```
dracut --force
```

Et enfin on reboot :

```
reboot
```

Vérifier que les modules sont chargés

Si notre module est signé correctement, alors il est chargé par le système.

On pourra le constater avec la fameuse commande lsmod :

```
lsmod | grep vbox
```

Ici, le module est bien chargé :

```
vboxdrv          704512  1
```

Signer des modules installés manuellement

Si vous n'avez pas de modules installés sous forme de "akmod", il est possible de signer les modules "manuellement".

Je prends dans cet exemple VirtualBox, mais installé avec le .rpm du site Oracle.

Après avoir généré et importé les certificats et clés, on recherche le chemin du module :

```
modinfo vboxdrv
```

Cela me renvoie :

```
filename:    /lib/modules/6.9.8-200.fc40.x86_64/misc/vboxdrv.ko
version:     7.0.18 r162988 (0x00330004)
license:     GPL
description:  Oracle VM VirtualBox Support Driver
author:      Oracle and/or its affiliates
```

```
srcversion: 9CBC44140E50A5E89770210
depends:
retpoline: Y
name: vboxdrv
vermagic: 6.9.8-200.fc40.x86_64 SMP preempt mod_unload
parm: disabled:Disable automatic module loading (int)
parm: force_async_tsc:force the asynchronous TSC mode (int)
```

Il existe un script pour signer les modules, qui est installé grâce au paquet kernel-devel nommé **sign-file**.

Son chemin est le suivant :

```
/usr/src/kernels/$(uname -r)/scripts/sign-file
```

Pour signer notre module, on utilisera le script, avec les clés privées et publiques créées par la commande **kmmodgenca** :

```
/usr/src/kernels/$(uname -r)/scripts/sign-file sha256 /etc/pki/akmods/private/private_key.priv /etc/pki/akmods/certs
```

Pour mon exemple :

```
/usr/src/kernels/$(uname -r)/scripts/sign-file sha256 /etc/pki/akmods/private/private_key.priv /etc/pki/akmods/certs
```

Évidemment, ce sont des actions manuelles à faire à chaque mise à jour du noyau et recompilation du module !

On peut charger le module avec succès et vérifier que le module est signé :

```
modinfo vboxdrv
```

```
filename: /lib/modules/6.9.8-200.fc40.x86_64/misc/vboxdrv.ko
version: 7.0.18 r162988 (0x00330004)
license: GPL
description: Oracle VM VirtualBox Support Driver
author: Oracle and/or its affiliates
srcversion: 9CBC44140E50A5E89770210
depends:
retpoline: Y
name: vboxdrv
vermagic: 6.9.8-200.fc40.x86_64 SMP preempt mod_unload
sig_id: PKCS#7
signer: fedora-4164085003
sig_key: 4B:2F:C9:65:78:87:ED:2D:8B:C0:03:C1:87:54:1B:1A:6F:9F:B7:6B
sig_hashalgo: sha256
signature: 91:EE:D8:DB:64:16:CE:40:C8:0F:BC:D9:A3:0F:7D:A5:57:4D:06:C7:
```

E9:9D:01:94:92:C3:AE:0F:72:BF:23:B5:73:2C:DC:B3:43:8B:2A:7F:
07:EA:72:6A:F7:38:CC:F5:72:79:02:05:27:EE:C9:83:14:CB:10:5D:
C2:2E:AD:A8:E9:F2:B4:02:7F:72:B8:75:F4:85:F0:14:0E:67:B1:CE:
AF:E3:35:6A:58:03:74:9E:BE:BC:05:D8:39:18:53:9C:DD:9F:9E:F0:
DD:00:B0:80:05:32:D9:2B:ED:96:8D:D2:36:E8:8A:03:EA:00:7E:5D:
6E:3A:1F:05:B4:F7:41:65:6D:33:EA:53:88:89:16:89:5D:AC:AC:59:
B5:78:74:0C:7A:00:F4:09:18:DE:96:2F:8B:B8:60:6E:90:57:77:2A:
A2:97:9C:C2:EE:78:6C:7B:50:16:62:AC:43:71:70:E2:15:E5:CE:40:
EF:DF:50:93:08:33:EB:9F:59:AD:33:D0:3E:97:94:4D:4B:50:C3:FE:
18:14:09:D9:6C:7A:33:90:48:DC:75:2D:CF:03:58:41:02:BC:BD:2F:
D7:36:42:AC:34:D0:26:65:60:12:02:DA:D3:37:92:A4:3B:9E:40:B0:
52:0F:F1:53:E6:48:BB:01:70:A5:0F:D0:81:29:E1:78:6B:53:97:A3:
BC:36:F2:5F:F2:DD:E6:4F:53:AC:CE:4A:59:B6:86:D7:FA:68:A2:DA:
F9:66:5D:53:21:40:EB:79:D7:B9:A9:D8:F3:22:6B:E1:98:DD:95:1D:
AB:AD:71:E8:4A:BE:B7:2F:67:DB:95:A4:05:BD:3E:C0:FB:42:0D:1F:
B5:34:2F:1E:7E:43:25:15:1B:DC:AE:60:24:BB:33:0D:52:C7:56:C4:
A2:76:6E:6D:FA:12:F5:BA:D2:E4:AD:4F:69:F0:B4:EB:E6:11:CB:17:
47:4E:6B:DA:22:6F:82:E6:BB:5C:CE:3B:EC:9F:BF:91:BF:84:A3:FB:
89:0B:0D:8F:6F:05:3E:B9:17:42:5D:11:4D:D2:59:24:3D:14:8F:A6:
92:64:47:D0:57:F3:59:D7:07:AF:51:31:80:CE:9E:5A:11:52:F1:91:
1B:0F:4F:4F:39:9B:82:BA:25:DE:46:B9:A4:19:90:8C:60:0A:44:4B:
E7:CA:49:D8:2D:78:4D:A0:71:E3:03:56:BC:41:31:47:0D:41:17:30:
D8:D0:AA:D4:EE:50:C6:DB:92:E9:B0:25:02:77:72:61:31:A6:AD:55:
16:9A:B1:AB:37:0D:93:D8:08:C1:A3:3C:04:4A:B5:80:12:B2:41:53:
E6:B5:CB:38:71:05:32:F7:06:D3:A9:B0

parm: disabled:Disable automatic module loading (int)

parm: force_async_tsc:force the asynchronous TSC mode (int)

Revision #1

Created 9 January 2025 16:12:23 by Julien

Updated 9 January 2025 16:37:12 by Julien