

Installer rtl8812au dans DKMS

Sébastien Millet, mars 2018

Il s'agit ici de la version 4.2.2 du driver de la clé WIFI USB Realtek. Sa version exacte (`lsusb -v`) est :

Bus 001 Device 009: ID 0bda:0129 Realtek Semiconductor Corp. RTS5129 Card Reader Controller

Fait sur Kubuntu 17.10.

- Important

Cette note s'attache, en plus d'activer DKMS pour ce driver, à automatiser la signature du driver pour un PC sur lequel SecureBoot est activé. Cela suppose d'avoir à disposition une paire (clé publique, clé privée), le certificat étant enregistré dans le SecureBoot.

C'est parti !

1. Copie des sources

Copier le contenu de `rtl8812au-master` dans `/usr/src/8812au-4.2.2`.

`rtl8812au-master` est le répertoire du driver sur le mini-CD fourni avec la clé.

DKMS fonctionne typiquement en repérant chaque module par son nom, et son numéro de version. Le nom de module de la clé étant `8812au.ko` et la version (mars 2018) étant 4.2.2, la source doit être nommée `8812au-4.2.2` pour suivre le standard DKMS.

2. Fichier `dkms.conf`

Le driver `8812au` contient un fichier `dkms.conf` qui est suffisant, pas besoin de le modifier. Pour information, voici ce fichier :

```
PACKAGE_NAME=8812au
PACKAGE_VERSION=4.2.2

DEST_MODULE_LOCATION=/kernel/drivers/net/wireless
BUILT_MODULE_NAME=8812au

MAKE="'make' all"
CLEAN="'make' clean"
AUTOINSTALL="yes"
```

3. Installation dans DKMS

```
root # dkms add -m 8812au -v 4.2.2
```

```
Creating symlink /var/lib/dkms/8812au/4.2.2/source ->
/usr/src/8812au-4.2.2
```

DKMS: add completed.

4. Compilation par DKMS

```
root # dkms build -m 8812au -v 4.2.2
```

Kernel preparation unnecessary for this kernel. Skipping...

```
Building module:
cleaning build area...
'make' all.....
cleaning build area...
```

DKMS: build completed.

5. Installation par DKMS

À noter que si on lance l'installation du module alors qu'il n'est pas encore compilé, DKMS s'en charge.

Mais dans ce document, j'ai préféré détailler ces étapes une par une.

```
dkms install -m 8812au -v 4.2.2
```

```
8812au:
Running module version sanity check.
```

```
Good news! Module version v4.2.2_7502.20130517 for 8812au.ko
exactly matches what is already found in kernel 4.13.0-37-generic.
DKMS will not replace this module.
You may override by specifying --force.
```

```
depmod...
```

DKMS: install completed.

6. Faire signer le module par DKMS

Ici la difficulté est que l'exécution d'un script POST_BUILD, comme documenté sur Internet, ne fonctionne pas. Après exécution de POST_BUILD le module est "stripé", ce qui supprime la signature. Il faut le signer dans le script PRE_INSTALL.

- **Note**

Modifier le fichier Makefile du source pour y ajouter la signature du module ne fonctionne pas car :

- Si l'on ajoute la signature après le build (cible 'modules' du Makefile), le strip du module fait par DKMS élimine la signature.
- Si l'on ajoute la signature avant ou après l'installation (cible 'install' du Makefile), rien n'est changé : DKMS n'installe pas le module en exécutant un make install avec le Makefile du source. DKMS utilise son propre code.

Cela étant précisé, la documentation sur Internet est bien pratique : <https://computerlinguist.org/make-dkms-sign-kernel-modules-for-secure-boot-on-ubuntu-1604.html>

- **Note (2)**

Pour savoir si un module est signé, il faut en afficher le contenu binaire et regarder si la fin du fichier contient le texte ~Module signature appended~.

Exemples :

```
## Cas 1 : module non signé
# hexdump -C /lib/modules/4.13.0-37-generic/updates/dkms/8812au.ko | tail
001b2a90 08 00 00 00 00 00 00 00 18 00 00 00 00 00 00 00 |.....|
001b2aa0 09 00 00 00 03 00 00 00 00 00 00 00 00 00 00 00 |.....|
001b2ab0 00 00 00 00 00 00 00 00 78 4b 0e 00 00 00 00 00 |.....xK.....|
001b2ac0 57 ce 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |W.....|
001b2ad0 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
001b2ae0 11 00 00 00 03 00 00 00 00 00 00 00 00 00 00 00 |.....|
001b2af0 00 00 00 00 00 00 00 00 b0 21 1b 00 00 00 00 00 |.....!.....|
001b2b00 2c 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |,.....|
001b2b10 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
001b2b20

## Cas 2 : module signé
# hexdump -C /lib/modules/4.13.0-37-generic/updates/dkms/8812au.ko | tail
001b2c40 bc fb c4 b6 68 34 c7 d3 8a 20 28 47 ba 87 a7 4a |...h4... (G...J|
001b2c50 5d c9 b1 07 16 fb 95 8a bc f5 a0 fe a0 46 f3 d4 |].....F..|
001b2c60 2b 4e 29 4b 60 5d e8 11 cf 19 cc 69 af f2 7d 3c |+N)K'.....i..}<|
001b2c70 21 5b 66 3a a6 2c eb b2 46 fc 03 d9 b9 6e 7a 74 |![f:.,..F....nzt|
001b2c80 24 ac 5d 79 54 ad 03 1d 16 d8 fa 0b 93 8c ba cc |$.]yT.....|
```

```

001b2c90 b5 2d 57 43 9f d3 44 b5 6f c8 86 14 8e b8 c0 dc |.-WC...D.o.....|
001b2ca0 e4 00 00 02 00 00 00 00 00 00 00 01 81 7e 4d 6f |.....~Mo|
001b2cb0 64 75 6c 65 20 73 69 67 6e 61 74 75 72 65 20 61 |dule signature a|
001b2cc0 70 70 65 6e 64 65 64 7e 0a |ppended~.|
001b2cc9

```

- **Apparté sur les clés de signature**

Cette solution suppose que la clé et le certificat pour signer le module se trouvent dans le répertoire /root/efikeys, et ont pour nom db.key et db.cer. En l'occurrence, le certificat est enregistré dans le variable EFI 'db'.

Voilà le contenu de 'db' et de la clé :

```

# efi-readvar -v db
[...]
db: List 5, type X509
  Signature 0, size 777, owner eee29bf3-fbc6-4c38-96df-bfe181770f39
  Subject:
    CN=SMT db
  Issuer:
    CN=SMT db
[...]
# openssl x509 -inform der -in /root/efikeys/db.cer -noout -text
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 11380626299561560859 (0x9df01b5282d5cb1b)
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: CN=SMT db
    Validity
      Not Before: Dec 15 19:30:46 2017 GMT
      Not After : Dec 13 19:30:46 2027 GMT
    Subject: CN=SMT db
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      Public-Key: (2048 bit)
      Modulus:
        00:a4:8a:15:5e:19:51:f3:f1:9e:61:bd:1e:89:55:
        [...]
        2c:1b
      Exponent: 65537 (0x10001)
    X509v3 extensions:
      X509v3 Subject Key Identifier:
        4B:E3:36:82:65:86:6F:8D:9D:36:99:7E:BF:69:C8:46:F8:D1:12:49
      X509v3 Authority Key Identifier:
        keyid:4B:E3:36:82:65:86:6F:8D:9D:36:99:7E:BF:69:C8:46:F8:D1:12:49
      X509v3 Basic Constraints:

```

```
CA:TRUE
Signature Algorithm: sha256WithRSAEncryption
83:3f:1e:c2:73:63:3c:4e:e9:79:d2:73:ef:67:a5:4d:2d:51:
[...]
6e:63:a4:2d
```

1. Fichier `sign-module.sh`

Le fichier `sign-module.sh` est à placer dans la racine des sources du module, dans notre cas, `/usr/src/8812au-4.2.2`.

Il doit être exécutable.

Fichier `sign-module.sh` :

```
#!/bin/sh

cd ../$kernelver/$arch/module
echo Signing module
kmodsign SHA256 /root/efikeys/db.key /root/efikeys/db.cer 8812au.ko
```

2. Fichier `/etc/dkms/8812au.conf`

Ce fichier indique à DKMS d'exécuter `sign-module.sh` juste avant d'installer le module. Comme indiqué précédemment, si on exécute le script dans le `POST_BUILD`, la signature est éliminée durant le `strip`.

`/etc/dkms/8812au.conf` :

```
PRE_INSTALL=sign-module.sh
```

7. Modification de `dkms`

L'un des soucis quand on configure `dkms` est la difficulté à prendre en compte les répertoires relatifs. En effet `dkms` n'accepte pas de chemin absolu, et jongler avec les répertoires relatifs peut être ardu.

Par défaut, si `dkms` ne peut pas exécuter le fichier de script, il affiche une erreur comme quoi le fichier n'est pas exécutable. Ce qui peut être le cas, l'ennui est que `dkms` affiche cette erreur y compris si le fichier n'est pas trouvé.

Avec cette modification :

- Si le fichier est trouvé mais n'est pas exécutable, le message d'erreur habituel est affiché.
- Si le fichier n'est pas trouvé, `dkms` le dit et donne le chemin du script.

Dans le fichier /usr/sbin/dkms version 2.2.1.0, chercher la ligne qui contient

```
warn $"The $1 script is not executable."
```

Et écrire à la place :

```
if [[ -e ${run%% *} ]]; then
    warn $"The $1 script is not executable."
else
    warn $"The $1 script is not found, file: $run"
fi
```

8. Résumé

- Note

À tout moment, il est possible de savoir où en est DKMS avec la commande `dkms status`.

Ci-dessous, on part d'un DKMS ne contenant pas le module pour la version de noyau en cours d'exécution. On commence par le compiler puis l'installer, et on termine avec un `modprobe` pour le charger.

```
# uname -r -i
4.13.0-37-generic x86_64
# dkms status
8812au, 4.2.2, 4.13.0-36-generic, x86_64: installed
# lsmod | grep 8812au
# dkms build -m 8812au -v 4.2.2
```

Kernel preparation unnecessary for this kernel. Skipping...

```
Building module:
cleaning build area...
'make' all.....
cleaning build area...
```

```
DKMS: build completed.
# dkms status
8812au, 4.2.2, 4.13.0-36-generic, x86_64: installed
8812au, 4.2.2, 4.13.0-37-generic, x86_64: built
# dkms install -m 8812au -v 4.2.2
```

```
8812au:
Running module version sanity check.
```

```
Running the pre_install script:
Signing module
```

- Original module
 - No original module exists within this kernel
- Installation
 - Installing to /lib/modules/4.13.0-37-generic/updates/dkms/

depmod...

```
DKMS: install completed.
# dkms status
8812au, 4.2.2, 4.13.0-36-generic, x86_64: installed
8812au, 4.2.2, 4.13.0-37-generic, x86_64: installed
# modprobe 8812au
# lsmod | grep 8812au
8812au                999424  0
```

9. Post Scriptum

La compilation suite à l'installation d'un nouveau noyau ne fonctionne pas.

Erreur qui s'est produite :

```
8812au: version magic '4.13.0-37-generic SMP mod_unload ' should be
      '4.13.0-38-generic SMP mod_unload '
```

Il semble donc que la compilation ait été faite en fonction du noyau actuel, alors qu'il fallait prendre en compte le noyau *à venir*.

Le post suivant sur stackexchange donne une solution :

<https://elementaryos.stackexchange.com/questions/4767/dkms-not-running-correctly-on-system-update>

Solution : modifier le Makefile dans le source du module.

Avant :

```
$(MAKE) KERNELRELEASE=$(kernelver) ARCH=$(ARCH) CROSS_COMPILE=$(CROSS_COMPILE) \
      -C $(KSRC) M=$(shell pwd) modules
```

Après :

```
$(MAKE) ARCH=$(ARCH) CROSS_COMPILE=$(CROSS_COMPILE) \
      -C $(KSRC) M=$(shell pwd) modules
```

À suivre (non encore testé)...