Enabling Secure Boot on Slackware

On Unified Extensible Firmware Interface (UEFI) based hardware, a system can operate in Secure Boot mode. In Secure Boot mode, only EFI binaries (i.e. boot managers, boot loaders) that are trusted by the platform owner, either explicitly or via a chain of trust, are allowed to run at boot time. This prevents unauthorised EFI binaries and operating systems from running on your system, which can improve security.

This article will teach you:

- About Secure Boot keys and signature databases
- How to enroll Secure Boot keys while booted into Slackware
- How to sign EFI binaries for use in Secure Boot mode.

Make sure you can find and manipulate the Secure Boot settings with your system's UEFI firmware. That way, if you make a mistake, you can simply turn off Secure Boot to have a bootable system again.

Once you have changed your Secure Boot keys, signed your EFI binaries and have tested that Secure Boot is working, you should store your private keys in a safe location until the keys are required again. Anyone with access to your private keys can bypass the protection that Secure Boot offers.

Secure Boot Keys and Signature Databases

Two types of Secure Boot keys are used to create trust relationships:

- Platform Key - This establishes the trust relationship between the platform owner and platform firmware. Only one Platform Key can be stored by the UEFI firmware. The public key is stored in the PK Secure Boot variable. This key allows the platform owner to manipulate all the Secure Boot keys and signature databases.
- Key Exchange Key - This establishes the trust relationship between the operating system and platform firmware. Multiple Key Exchange Keys can be stored by the UEFI firmware. The public keys are stored in the KEK Secure Boot variable. Key Exchange Keys only allow the operating system to manipulate the signature databases.

There are two signature databases for authorising EFI binaries:

- Forbidden signature database - This stores hashes and public signing keys of forbidden EFI binaries. This uses the dbx Secure Boot variable. EFI binaries with hashes present in this database or signatures that can be authenticated using a signing key stored in the database are forbidden from loading.
- Authorised signature database - This stores hashes and public signing keys of trusted EFI binaries. This uses the db Secure Boot variable. EFI binaries with hashes present in this database or signatures that can be authenticated using a signing key stored in the database are

permitted to run if there are no matches with any entries in the Forbidden signature database.

Requirements

You will need the efitools and sbsigntools packages before you begin. Slackbuilds are available at http://slackbuilds.org.

If you do not already have your own Platform Key enrolled in the UEFI firmware, the howto assumes you have Secure Boot off and have cleared the PK variable in the UEFI firmware.

Enrolling Secure Boot Keys and Signature Database Entries

If you do not have an existing Platform Key pair and an EFI binary signing key pair, the easiest method to create the key pairs would be to create self signed keys. It is recommended to create 2048-bit RSA key pairs that use the sha256RSA signature algorithm. To generate self signed keys with the recommended properties, run:

```
openssl req -new -x509 -newkey rsa:2048 -subj "/CN=Platform Key Common
Name/" \
        -keyout PK.priv -out PK.pub -days 3650 -nodes -sha256
openssl req -new -x509 -newkey rsa:2048 -subj "/CN=EFI Binary Signing Key
Common Name/" \
        -keyout db.priv -out db.pub -days 3650 -nodes -sha256
```

which creates private keys with the `.priv` extension and public key certificates with the `.pub` extension. You may wish to adjust the key validity period and choose a different Common Name (CN) to help distinguish your keys.

It is not necessary to create or use your own Key Exchange Key as that is intended for use by operating systems. Key Exchange Key instructions, however, have been provided below so you understand how to enroll Key Exchange Keys for operating systems that require it.

To prepare a new Platform Key for writing to the PK variable:

1. Insert the public Platform Key into an EFI signature list:

   ```
   cert-to-efi-sig-list -g owner_guid PK.pub PK.esl
   ```

   replacing `owner_guid` with a hexadecimal GUID in the format `12345678-1234-1234-123456789abc`. The owner GUID should be the same for all keys that you own. If an operating system cannot add a signature to a signature database due to a lack of resources, it may remove a signature with an owner GUID associated with the operating system.

2. Sign the EFI signature list. In Setup mode (Secure Boot off) the private half of the inserted key

should sign the signature list. In User mode (Secure Boot on) the private key of the current Platform key should sign the signature list:

```
sign-efi-sig-list -k PK.priv -c PK.pub PK PK.esl PK.signed
```

A similar procedure applies for preparing a Key Exchange Key or a signature database entry for writing to the KEK, db, or dbx variables. Key Exchange Keys must be signed by the private half of the Platform Key:

```
cert-to-efi-sig-list -g owner_guid KEK.pub KEK.esl
sign-efi-sig-list -a -k PK.priv -c PK.pub KEK KEK.esl KEK.signed
```

And signature database entries must be signed by the private half of the Platform Key or any of the Key Exchange Keys:

```
cert-to-efi-sig-list -g owner_guid db.pub db.esl
sign-efi-sig-list -a -k PK.priv -c PK.pub db db.esl db.signed
```

Note that the `-a` option was used to prepare for an append write.

To update the Secure Boot variables you must have root privileges. You will need to load the efivarfs kernel module and mount the efivarfs filesystem beforehand if it has not been taken care of already:

```
modprobe efivarfs
mount -t efivarfs efivarfs /sys/firmware/efi/efivars
```

To enroll the Platform Key, run:

```
efi-updatevar -f PK.esl.signed PK
```

If the system was in Setup mode it will now be in User mode.

To add keys to the KEK, db or dbx variables, run (as appropriate):

```
efi-updatevar -a -f KEK.signed KEK
```

```
efi-updatevar -a -f db.signed db
```

```
efi-updatevar -a -f dbx.signed dbx
```

You can check that your keys have been properly enrolled using `efi-readvar`.

Signing EFI Binaries

My recommendation (at the time of writing) is that you either use a boot manager with an EFI stub kernel, or directly boot an EFI stub kernel. ELILO, efilinux and syslinux (and possibly GRUB but I do not know for sure) will allow unsigned kernels to run (or at least it does on my hardware and VM), which defeats the purpose of Secure Boot. If you do follow my recommendation, make sure you sign your

kernel every time you change it.

You will need to sign all EFI binaries, up to, and including your bootloader and/or EFI stub kernel. To sign an binary, run:

```
sbsign --key db.priv --cert db.pub --output signed_binary.efi binary.efi
```

An example of how to add an EFI stub kernel entry using efibootmgr is:

```
efibootmgr -c -L SlackSecureBoot -l '\EFI\Slackware\vmlinuz-signed.efi' -u
'root=/dev/sda3'
```

If you see `warning: gap in section table` when signing an EFI binary (see below), the binary will probably not work in Secure Boot mode. This warning appears for EFI binaries built against earlier gnu-efi library versions. If you plan to use ELILO you will need to recompile it yourself, the version shipped with Slackware will not work.

```
warning: gap in section table:
    .text   : 0x00000400 - 0x00017c00,
    .reloc  : 0x00017ca1 - 0x000180a1,
warning: gap in section table:
    .reloc  : 0x00017ca1 - 0x000180a1,
    .data   : 0x00018000 - 0x00033000,
gaps in the section table may result in different checksums
warning: data remaining[225792 vs 242346]: gaps between PE/COFF
sections?
```

Disabling Secure Boot

If you want to remove all Secure Boot keys and revert to Setup mode, the easiest way to do so is to sign an empty file with your Platform Key and write the signed file to all the Secure Boot variables:

```
touch empty
sign-efi-sig-list -k PK.priv -c PK.pub PK empty empty.signed
efi-updatevar -f empty.signed PK
efi-updatevar -f empty.signed KEK
efi-updatevar -f empty.signed db
efi-updatevar -f empty.signed dbx
```

Dual/Multi-booting with Windows

If Windows is one of your boot options, you will need the Microsoft KEK and db certificates. The certificates can be found at https://technet.microsoft.com/en-us/library/dn747883.aspx and will need to be converted from DER format to PEM format:

```
openssl x509 -in certificate.der -inform DER -out certificate.pem
```

The instructions that were provided above can then be used to enroll the certificates. The owner GUID you should use for Microsoft keys is 77fa9abd-0359-4d32-bd60-28f4e78f784b.

Sources

- Originally written by turtleli

More information can be found at:

- http://uefi.org
- http://www.rodsbooks.com/efi-bootloaders/secureboot.html
- http://blog.hansenpartnership.com/
- https://technet.microsoft.com/en-us/library/dn747883.aspx

howtos, security, secure boot, uefi, author turtleli