Secure Boot
Security, tooling and the future

Steve McIntyre <93sam@debian.org>

21st July 2026

Agenda

- What is UEFI Secure Boot?
- Is it worthwhile?
- Key Management and updates
- Packaging and tools
- Stuff coming

What is UEFI?

- Unified Extensible Firmware Interface
  - Rich(er) set of APIs than BIOS
  - UEFI Boot Services
  - UEFI Runtime Services
  - ExitBootServices() changes the world

- EDK2 reference implementation
  - Started by Intel, now maintained in the Tianocore project

What is Secure Boot?

- A way to protect against persistent boot-time malware
- Software is verified by signature
- (Typically) each step in the boot chain verifies the next
- Just(!) a problem of key management…
- Can act as a base for further security solutions
  - Measured Boot
  - Locked-down kiosk systems
  - Etc.

What is Secure Boot **not**?

- A way to lock people out of their own machines
  - Enrol your own keys
  - Or turn off SB
- A way to stop people using Linux and other Free Software
  - Microsoft and Linux folks talk regularly
  - The point is to add security for users **on both sides**

Is it Worthwhile?

- **Yes**, for **most** people
  - Persistent boot-time malware **is** a real problem
- It does make some things harder
  - Hibernation
  - Loading third-party kernel modules
  - Kexec
  - Direct access to memory and I/O ports

Misunderstandings

(and misinformation?)

- “Secure Boot is just designed to lock you into Windows”
- “Secure Boot adds more vulnerabilities”
- “Secure Boot doesn’t work if you’re using Testing”
- “Secure Boot doesn’t protect the average user”
- …

Key Management

- Microsoft keys included with most (x86) PCs
- Logistics of being a CA
- Arm-based machines too
  - And likely further architectures
- Multiple keyrings defined
  - Platform Key (PK)
  - Key Exchange Key (KEK)
  - **Authenticated Variable Database (DB)**
  - Revocation Database (DBX)
  - *later: MOK via shim*
- On most machines you can modify the list of trusted keys

DB Key updates

- Two root CA keys initially defined in DB:
  - `Microsoft Windows Production PCA 2011`
    - expires October 2026
  - `Microsoft Corporation UEFI CA 2011`
    - **EXPIRED** June 2026

- Three new CA keys added:
  - `Windows UEFI CA 2023`
  - `Microsoft Option ROM UEFI CA 2023`
  - `Windows UEFI CA 2023`
- Key expiry should **not** break boot
  - But will stop signing of future binaries

DBX updates

- Happen regularly
- Revocations for security
  - Block older keys
  - Block older binaries
- Keep applying these!
  - Just like security updates elsewhere

Fwupd and LVFS

- The easy way to manage UEFI keys
  - (and other firmware updates!)
- Needs 2.0.20 or newer for DB updates
  - Forky, Trixie, Bookworm
- `# fwupdtool refresh`
- `# fwupdtool get-updates`
- `# fwupdtool update`
- `# mokutil --db –short` (to list the DB keys)

Other ways

- Vendor firmware updates
- `efivar`
  - Possible, but risky
- Windows updates
- Safety checks
  - Don’t break boot!

KEK updates

- PK signs KEK
- KEK signs DB and DBX
- Existing KEK keys also expired:
  - Microsoft KEK CA 2011 (June 2026)
- Also update KEK on your machines
  - Fwupd again

Key management is **hard**

- But it’s necessary
- CA rollover was **late**
- More **will** be coming
- Multiple signatures should work
  - Except firmware bugs

shim

- Small primary loader
  - Permissive licence
  - Signed by Microsoft **after review**
- Embed your own key(s)
  - We embed the Debian SecureBoot CA
- Adds another trusted keyring:
  - Machine Owner Key (MOK)
  - Useful for DKMS etc.

shim-review

- Team of cross-distro volunteers
  - Supporting users with FLOSS boot chain
- Check over shim builds for security
- Review the boot chain
  - Shim, GRUB, Linux
- Validate identity
- Lots of questions to answer
- Lots of sanity checks

shim-signed

- Packaging for the signed shim
  - The end of a long pipeline
- We **currently** have dual-signed shims
  - Signed with both of the UEFI CAs
  - Verify and combine sigs during build
- But the next shim **won't** be dual-signed
- Preinst checks for boot safety:
  - Will this shim boot on this machine?

Cross-distro discussions

- Regular discussions with
  - Shim maintainers
  - GRUB maintainers
  - Fwupd maintainers
  - Microsoft included

Problems seen with SB

- Limited variable storage space
  - DBX design doesn’t scale
  - SBAT mostly fixes this
- Buggy firmware implementations
  - Dual-signed shim problems
  - CA expiry tracking
  - Inclusion of extra keys
- Vendors leaking keys

SBAT

- Secure Boot Advanced Targeting
- Generation-based revocations approach
  - Much more efficient than DBX
- `SbatLevel` variable decides if things are allowed to run
- CSV data embedded in signed application binaries – see example

SbatLevel

- Base set at shim build time
- Can be advanced further
  - So you can add stricter blocking
- Currently we set

```
shim,4
grub,5
```

SBAT example

```
sbat,1,SBAT Version,sbat,1,https://github.com/rhboot/shim/blob/main/SBAT.md

grub,5,Free Software Foundation,grub,2.14,https://www.gnu.org/software/grub/

grub.debian,5,Debian,grub2,2.14-2,https://tracker.debian.org/pkg/grub2

grub.debian14,1,Debian,grub2,2.14-2,https://tracker.debian.org/pkg/grub2

grub.peimage,2,Canonical,grub2,2.14-2,https://salsa.debian.org/grub-team/
grub/-/blob/master/debian/patches/secure-boot/efi-peimage.patch
```

- Bump the number (SBAT level) for each app each time there is a security event
  - grub.5 → grub.6
- Bump the app.vendor number if the there is a vendor-specific security event
  - grub.debian,5 → grub.debian,6

Tools

- FLOSS tools **can** be great, but...
- Security tools are often not
- Not many people using these tools, so...
  - Lots of sharp edges
  - Minimal documentation

efivar / efibootmgr

- `efivar` gives low-level access to EFI variable space
  - `/sys/firmware/efi/efivars/`
  - Directly view (and maybe edit!)
- `efibootmgr` handles EFI boot variables
  - Boot entries, Boot order, etc.

mokutil

- Manipulates EFI variables specifically relating to Secure Boot and shim
  - `mokutil --db`
  - `mokutil --kek`
  - `mokutil --sb-state`
  - `mokutil --list-sbat-revocations`
  - `mokutil --import`
  - `mokutil --set-verbosity`
  - `etc...`

sbsigntools

- Suite of tools for managing signatures
  - `sbsign`
  - `sbattach`
  - `sbverify`
  - `etc...`

pesign

- **Another** suite of tools for managing signatures
  - `pesign`
  - `efikeygen`
  - `pesigcheck`
  - `etc...`

What’s coming?

- More CA and key rollovers
  - Post-quantum crypto requirements
- More architectures
  - riscv support in next shim release
- More bugfixes and hardening
  - NX happened
- More revocations
  - Because…

Questions?

More information:

`https://wiki.debian.org/SecureBoot`