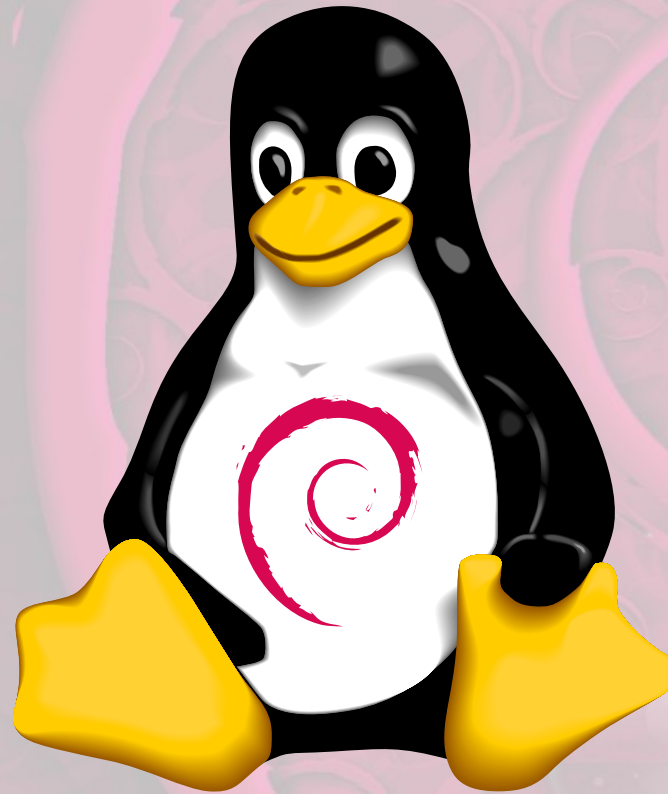




Debian's support for Secure Boot on x86 and ARM



Ben Hutchings
Kernel Recipes, Paris, 2016



Ben Hutchings

- Regular Linux contributor since 2008
- Working on various drivers and kernel code in my day job
- Debian kernel and LTS team member, now doing most of the kernel maintenance aside from ports
- Maintaining Linux 3.2.y and 3.16.y stable update series on kernel.org
- Kernel maintainer for LF Civil Infrastructure Platform, aiming for super-long-term support

Secure Boot

- Optional feature in UEFI - uses certificate store to validate boot loader, UEFI drivers, system firmware updates
- Protects against persistent malware (bootkit / kernel rootkit) if implemented correctly
- Required in 'Designed for Windows' systems since Windows 8 (2012)
- Only common trusted certificates on PCs are for Microsoft signing keys
 - MS *will* sign PC boot loaders for a small fee, and the certificate store is normally editable on PCs
 - ARM-based Windows systems are completely locked down
- HPe shipping ARM64 server systems in SB setup mode, allowing installer to set trusted certificates



GNU/Linux under Secure Boot

- First stage needs MS signature – manual submission process
 - Most distributions introduced 'shim' as first stage boot loader that won't need updating often
- MS expects boot loader and kernel to validate code they load – and it's a good idea anyway
 - For later stages, we control certificates and keys – certificates can be embedded in 'shim'
 - GRUB needs to validate its modules and kernels
 - Linux kernel needs to validate its modules and any other code that runs in kernel mode



Securing the Linux kernel

- Root user, including malware running as root, can modify kernel without using any security bugs
- MS signing requires kernel to validate code it runs – and it's also a good idea in general
- Using Matthew Garrett's patchset to add 'securelevel' feature, activated when booted under SB:
 - Module signatures are mandatory
 - `kexec_load()` is disabled – must use `kexec_file_load()` instead
 - Hibernation is disabled – but images *could* be signed and validated using per-machine key
 - Other kernel APIs that allow peek/poke are disabled
- Force-loading of modules into the wrong kernel version is disabled

The signature problem

- We don't want to expose signing keys to builddds
- Reproducible builds can't depend on anything secret
- So we can't *auto-build* signed binaries in single step

Solution requires an extra source package:

1. Build unsigned binaries from first source package
2. Sign 'offline' and put detached signatures in second source package
3. Build signed binaries from second source package



Introducing linux-signed

- The second source package for linux
- Contains detached signatures for specific kernel version and script to update them for a new kernel version
- linux builds binary packages like `linux-image-4.7.0-1-amd64-unsigned`
- linux-signed clones those packages, attaches signatures, and builds packages like `linux-image-4.7.0-1-amd64`
- Signing currently done with my personal key, which will not be trusted by shim or GRUB



Signing in dak - background

- Debian archive software (dak) signs archive metadata using keys trusted by Debian systems
 - Separate keys for stable releases, security updates, rolling releases
 - Keys managed by 'FTP' team
- We want dak to also sign code from linux, grub, etc. and publish detached signatures
 - Keys also managed by FTP team, possibly similar split between releases
 - Mustn't publish complete binaries at this stage - need reproducibility, and might have an embargoed fix
- dak supports uploads with non-standard file types (byhand) and configured handlers (auto-byhand!)



Signing in dak - implementation

- Add auto-byhand script that converts tarball of EFI binaries and/or kernel modules to tarball of corresponding signatures
- Enable auto-byhand for all source packages that need code signing
- Include signature tarballs in signed archive metadata
- Change each first source package to include a suitable tarball in uploaded files (dpkg-distaddfile)
- Add preparation script to each second source package to download and unpack published signature tarball



Didn't Ubuntu already do this?

Yes, but:

- Launchpad has a different hook mechanism
- Signing script only covers EFI binaries, not kernel modules
- Signing script produces signed binaries, not detached signatures
- Kernel is only signed on amd64
- linux packages in Debian and Ubuntu have >10 years of divergence

We'll still share shim, grub-signed, sbsigntool, etc.

Secure Boot on ARM?

- Upstream kernel has EFI stub for ARM32 and ARM64
- We have out-of-tree kernel patches for securelevel on ARM64
- sbsigntool works for ARM32 and ARM64 (although the test cases only cover x86)
- GRUB EFI code runs on ARM32 and ARM64
- shim runs on ARM64
- So we should be able to support Secure Boot on ARM64

Securing the signing keys

- Julien Cristau is working on signing with a PKCS#11 hardware security module – some model of Yubikey
- Signing with Yubikey is slow so we plan to use it for EFI binaries only; kernel modules will be signed with a file



Current status

- Bug [#820036](#) tracks work to be done
 - dak support for signed packages – *in progress*
 - shim package – *in review*
 - GRUB signed build and validation of next stage
 - Linux signed build – *ready to go when dak is*
 - fwupdate signed build
 - Signed binaries in installer and live images
- More information at <https://wiki.debian.org/SecureBoot>
- Aiming for stretch release (freeze in Jan 2017)



Credits

- Linux 'Tux' logo © Larry Ewing, Simon Budig.
 - Modified by Ben to add Debian open-ND logo
- Debian open-ND logo © Software in the Public Interest, Inc.
- Debian slide template © Raphaël Hertzog
- Background image © Alexis Younes

Debian's support for Secure Boot on x86 and ARM



Ben Hutchings
Kernel Recipes, Paris, 2016

debian

Ben Hutchings

- Regular Linux contributor since 2008
- Working on various drivers and kernel code in my day job
- Debian kernel and LTS team member, now doing most of the kernel maintenance aside from ports
- Maintaining Linux 3.2.y and 3.16.y stable update series on kernel.org
- Kernel maintainer for LF Civil Infrastructure Platform, aiming for super-long-term support



debian



Secure Boot

- Optional feature in UEFI - uses certificate store to validate boot loader, UEFI drivers, system firmware updates
- Protects against persistent malware (bootkit / kernel rootkit) if implemented correctly
- Required in 'Designed for Windows' systems since Windows 8 (2012)
- Only common trusted certificates on PCs are for Microsoft signing keys
 - MS *will* sign PC boot loaders for a small fee, and the certificate store is normally editable on PCs
 - ARM-based Windows systems are completely locked down
- HPe shipping ARM64 server systems in SB setup mode, allowing installer to set trusted certificates

GNU/Linux under Secure Boot

- First stage needs MS signature – manual submission process
 - Most distributions introduced 'shim' as first stage boot loader that won't need updating often
- MS expects boot loader and kernel to validate code they load – and it's a good idea anyway
 - For later stages, we control certificates and keys – certificates can be embedded in 'shim'
 - GRUB needs to validate its modules and kernels
 - Linux kernel needs to validate its modules and any other code that runs in kernel mode





Securing the Linux kernel

- Root user, including malware running as root, can modify kernel without using any security bugs
- MS signing requires kernel to validate code it runs – and it's also a good idea in general
- Using Matthew Garrett's patchset to add 'securelevel' feature, activated when booted under SB:
 - Module signatures are mandatory
 - `kexec_load()` is disabled – must use `kexec_file_load()` instead
 - Hibernation is disabled – but images *could* be signed and validated using per-machine key
 - Other kernel APIs that allow peek/poke are disabled
- Force-loading of modules into the wrong kernel version is disabled

The signature problem

- We don't want to expose signing keys to builds
- Reproducible builds can't depend on anything secret
- So we can't *auto-build* signed binaries in single step

Solution requires an extra source package:

1. Build unsigned binaries from first source package
2. Sign 'offline' and put detached signatures in second source package
3. Build signed binaries from second source package



Introducing linux-signed

- The second source package for linux
- Contains detached signatures for specific kernel version and script to update them for a new kernel version
- linux builds binary packages like linux-image-4.7.0-1-amd64-unsigned
- linux-signed clones those packages, attaches signatures, and builds packages like linux-image-4.7.0-1-amd64
- Signing currently done with my personal key, which will not be trusted by shim or GRUB





Signing in dak - background

- Debian archive software (dak) signs archive metadata using keys trusted by Debian systems
 - Separate keys for stable releases, security updates, rolling releases
 - Keys managed by 'FTP' team
- We want dak to also sign code from linux, grub, etc. and publish detached signatures
 - Keys also managed by FTP team, possibly similar split between releases
 - Mustn't publish complete binaries at this stage - need reproducibility, and might have an embargoed fix
- dak supports uploads with non-standard file types (byhand) and configured handlers (auto-byhand!)

debian

Signing in dak - implementation

- Add auto-byhand script that converts tarball of EFI binaries and/or kernel modules to tarball of corresponding signatures
- Enable auto-byhand for all source packages that need code signing
- Include signature tarballs in signed archive metadata
- Change each first source package to include a suitable tarball in uploaded files (dpkg-distaddfile)
- Add preparation script to each second source package to download and unpack published signature tarball



Didn't Ubuntu already do this?

Yes, but:

- Launchpad has a different hook mechanism
- Signing script only covers EFI binaries, not kernel modules
- Signing script produces signed binaries, not detached signatures
- Kernel is only signed on amd64
- linux packages in Debian and Ubuntu have >10 years of divergence

We'll still share shim, grub-signed, sbsigntool, etc.



debian



Secure Boot on ARM?

- Upstream kernel has EFI stub for ARM32 and ARM64
- We have out-of-tree kernel patches for securelevel on ARM64
- sbsigntool works for ARM32 and ARM64 (although the test cases only cover x86)
- GRUB EFI code runs on ARM32 and ARM64
- shim runs on ARM64
- So we should be able to support Secure Boot on ARM64



Securing the signing keys

- Julien Cristau is working on signing with a PKCS#11 hardware security module – some model of Yubikey
- Signing with Yubikey is slow so we plan to use it for EFI binaries only; kernel modules will be signed with a file



Current status

- Bug [#820036](#) tracks work to be done
 - dak support for signed packages – *in progress*
 - shim package – *in review*
 - GRUB signed build and validation of next stage
 - Linux signed build – *ready to go when dak is*
 - fwupdate signed build
 - Signed binaries in installer and live images
- More information at <https://wiki.debian.org/SecureBoot>
- Aiming for stretch release (freeze in Jan 2017)

debian

Credits

- Linux 'Tux' logo © Larry Ewing, Simon Budig.
 - Modified by Ben to add Debian open-ND logo
- Debian open-ND logo © Software in the Public Interest, Inc.
- Debian slide template © Raphaël Hertzog
- Background image © Alexis Younes

Linux 'Tux' logo © Larry Ewing, Simon Budig.

Redistribution is free but has to include this notice.
Modified by Ben to add Debian open-ND logo.

Debian open-ND logo © Software in the Public Interest, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

OpenOffice.org template by Raphaël Hertzog
<http://raphaelhertzog.com/go/ooo-template>
License: GPL-2+

Background image by Alexis Younes "ayo"
<http://www.73lab.com/>
License: GPL-2+

'Access denied' image by Mike Licht
<http://NotionsCapital.com>
License: CC-BY-2.0 - <https://creativecommons.org/licenses/by/2.0/>