EMBEDDED LINUX CONFERENCE

@

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
EUROPE

Tales from the Crypt:

Implementing Secure Boot and Disk Encryption on Tegra Platforms

Tim Orling, Konsulko Group

#ossummit @moto_timo

Konsulko Group

THE LINUX FOUNDATION

Konsulko Group

- Services company specializing in Embedded Linux and Open Source Software
- Hardware/software build, design, development, and training services
- Based in San Jose, CA with an engineering presence worldwide

- https://konsulko.com/

Abstract

“Secure boot” is not one size fits all, but rather there are different implementations on different platforms. For Tegra platforms, secure boot involves a one-time only burning of keys into the on-device fuses. We’ll share the lessons learned from turning a board into a lovely paperweight as well as the reliable approach we used to confidently secure boot into the vendor’s Ubuntu based OS before creating our own Yocto Project built OS.

For disk encryption with LUKS and dm-crypt, we extended our approach of testing the vendor’s OS before moving on to creating our own. The added complexity of unique passphrases derived from disk UUIDs and per-device HW-derived keys was an interesting challenge. We attempted to stay as close to the vendor's tools (luks-srv and luks-srv-app) and design as we could, to hopefully future proof the implementation for newer releases of Linux for Tegra. Extending to A/B flashing for OTA updates (e.g. rauc or mender) added additional challenges, especially when trying to generalize the approach for the meta-tegra community. The end solution must address the bootloader, initramfs, kernel command line, /etc/crypttab, /etc/fstab and more. Add in the complexity of the partition table layout and flashing tools for Tegra platforms and you are in for a wild ride.

Agenda

- Secure Boot
  - Caveats
  - NVidia Secure Boot Implementation Primer
  - Fuses on Tegra Platforms
  - A Lovely Paperweight
  - secureboot-tegra
  - meta-tegra
- Disk Encryption
  - NVidia's Implementation
  - Dead End (It seemed like a good idea)
  - Another "Good" Idea
  - Yocto Project/OpenEmbedded Friendlier Approach
  - tegra-test-distro
  - eks.img
  - A/B OTA Updates
- Future Work
  - Jetpack 5.0.2+ (OP-TEE, UEFI and 5.10.y Kernel)
  - tegra-demo-distro

Secure Boot

Secure Boot: NVidia Implementation Primer

- Keys (or hashes of keys) are burned into fuses
- Keys stored on “disk” in Encrypted Key Blob (EKB)
- Initial boot with Trusted Operating System (TOS)
- Client Applications in `initrd` can retrieve keys from Trusted Applications (`hwkey-agent` and `luks-srv`)
- Multiple levels possible:
  - Public Key Cryptography (PKC)
  - + Secure Boot Key (SBK) => “SBKPKC”
  - + user_key (EKB and eks.img)

Secure Boot: Caveats

- Secure Boot is not one-size fits all
  - Each platform can have different implementations and nuances
  - This work specifically targeted the Jetson AGX Xavier platform (T194)
    - Developed on:
      - Jetpack 4.6.1/L4T 32.7.1
      - Yocto Project ‘dunfell’ (3.1.y) release
    - Forward ported to:
      - Jetpack 4.6.2/L4T 32.7.2
      - Yocto Project ‘kirkstone’ (4.0.y) release

Secure Boot: Fuses on Tegra Platforms

Software and configuration fuses related to Secure Boot for Jetson Xavier NX series and Jetson AGX Xavier series (T194) and Jetson TX2 series (T186)		
Bit Size	Name	Description
1	odm_production_mode	0x1 for “production”
256	public_key_hash	RSA public key hash.
128	secure_boot_key	Secure Boot Key (SBK): AES encryption key for encrypting bootloader.
128	KEK0	Four 32□bit registers named KEK00 through KEK03.
128	KEK1	Four 32□bit registers named KEK10 through KEK13.
256	KEK256	Not a distinct fuse; addresses KEK0 and KEK1 as a single 256□bit fuse.
128	KEK2	Four 32□bit registers named KEK20 through KEK23.

Tegra Linux Driver Package Development Guide/bootloader_secure_boot.html

Secure Boot: Fuses on Tegra Platforms (cont'd)

- Fuses can only be changed from a "0" to a "1".
- NVidia recommends "burning" all the fuses at once. Plan ahead.
  - In practice*:
    - PKC
    - SBK
    - KEK{0,1}|256
    - KEK2

Keep your keys in ESCROW!

*development stage (for production also burn odm_production_mode fuse, etc.)

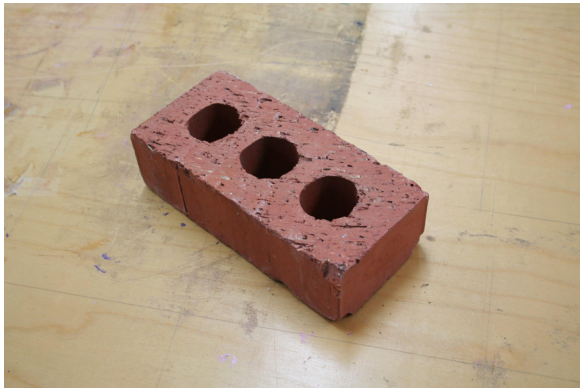
Image Source: https://pixabay.com/vectors/sign-caution-warning-danger-safety-304093/

Konsulko Group

THE LINUX FOUNDATION

Secure Boot: A Lovely Paperweight

- Despite reading the documentation multiple times.
- Despite “following all the instructions”.



https://forums.developer.nvidia.com/t/postmortem-jetson-xavier-agx-will-not-get-past-boot-rom-after-burning-pkc-sbk-kek256/208426

Image Source: https://en.wikipedia.org/wiki/Brick#/media/File:Brick.jpg

Konsulko Group

THE LINUX FOUNDATION

Secure Boot: secureboot-tegra

- Known good steps to:
  - Generate keys
  - Burn fuses
  - Flash NVidia Ubuntu-based OS to target
  - Prove Secure Boot is working!

Keep your keys in ESCROW!

- Paranoia?
  - Use only bare-metal Ubuntu 18.04 host

https://github.com/konsulko/secureboot-tegra (for Jetson AGX Xavier and L4T 32.7.1)

Forked from: https://github.com/Trellis-Logic/secureboot-tegra

Konsulko Group

THE LINUX FOUNDATION

Secure Boot: meta-tegra

https://github.com/OE4T/meta-tegra

- For Yocto Project/OpenEmbedded builds
- Well maintained and tested
- Secure Boot Support
  - https://github.com/OE4T/meta-tegra/wiki/Secure-Boot-Support

```
TEGRA_SIGNING_ARGS = "-u /path/to/rsa_priv.pem -v /path/to/sbk_hex_file"
```

The resulting `<image>-<machine>.tegraflash.tgz` contains signed/encrypted artifacts, but is installed as normal (`sudo ./doflash.sh`)

Disk Encryption

Disk Encryption: NVidia's Implementation

- LUKS (Linux Unified Key Setup)
- Uses `dm-crypt`
- Scripts encrypt Ubuntu-based OS in chroot (in-place)
  - Unencrypted "/boot" partition
  - Encrypted "rootfs" (excluding /boot)
- Resulting images are then signed for Secure Boot and ready to "flash"
- LUKS passphrase is derived from keys in fuses, unique device id (ECID) and keys stored in EKB partition.
- Decryption is done by the `initrd`
  - The `luks-srv-app` Client Application queries `luks-srv` Trusted Application on boot.
  - Unlocks `device-mapper` devices.
  - Switches to full rootfs.

Disk Encryption: Dead End (It seemed like a good idea)

- Create an image class to perform the LUKS encryption
  - BUT
    - BitBake tasks run in “pseudo” and real “sudo/root” privileges are not possible
      - Can’t communicate with `device-mapper` to format LUKS container (create the LUKS header)
        - » Not gonna work (ok, maybe could use QEMU?)

https://github.com/moto-timo/meta-tegra/tree/deadend-dunfell-luks

Konsulko Group

THE LINUX FOUNDATION

Disk Encryption: Another “Good” Idea

- Keep per partition passphrase (like Nvidia implemention)
- Create another image class to generate unencrypted “/boot” partition
- (Ab)Use “var flags” to store per partition information
- At one point used `xmllint` and `xmlstarlet` to query and make changes to “flash.xml”
  - BUT
    - Over-engineered
    - Need changes in “/etc/crypttab” per partition
    - Nested ${} variable expansion hell
    - Still haven’t addressed actual LUKS encryption

https://github.com/moto-timo/meta-tegra/commits/wip-dunfell-luks
https://github.com/moto-timo/tegra-demo-distro/tree/wip-dunfell-luks

Disk Encryption: Yocto Project/OpenEmbedded Friendlier Approach

- What if we?:
  - Don't rip apart the rootfs
  - Perform disk encryption *on-device*
  - Sign for Secure Boot independently of LUKS Disk Encryption
- BONUS: More production friendly
  - Images can be signed on a dedicated secure machine.
  - Images can be installed on the factory floor or in the field.
  - A/B OTA updates can be performed in the field.

Disk Encryption: tegra-test-distro

- Two stage approach:
  - first boots into a system installer which:
    - creates the proper partitions (from “flash.xml”) with the LUKS headers
    - writes the real rootfs (from a bundled tarball) into the now encrypted partition
    - Installs bootloader update payload (BUP) with real initramfs
- Decouples secureboot signing from LUKS disk encryption
  - the actual encryption is performed on the device which has already been booted securely.

Disk Encryption: tegra-test-distro (cont'd)

- Enable LUKS disk encryption:

```
MACHINEOVERRIDES =. "cryptparts:"
IMAGE_FSTYPES = "tar.gz tegraflash"
```

- Prefix encrypted partitions in flash.xml with "`crypt-`"
  - Use "`id=`<number>" to set the order of partitions
- Build installer image:

```
bitbake tegra-sysinstall
```

- Functional branch(es) for Jetson AGX Xavier:
  - https://github.com/moto-timo/tegra-test-distro/tree/kirkstone-agx
  - https://github.com/madisongh/tegra-test-distro/pull/11

Konsulko Group

THE LINUX FOUNDATION

Disk Encryption: eks.img

- Stock fuses are all zeroes
  - KEK2
- Stock eks.img is also “zeroes”
  - user_key
- Generate your own eks.img
  - Extended secureboot-tegra

    https://github.com/moto-timo/secureboot-tegra/tree/jetson-agx-xavier-devkit-luks

Disk Encryption: A/B OTA Updates

- Because the disk encryption is performed on the device, you can build update bundles the same as normal.
- They should be installable on the running device as per normal procedures.
- It JustWorks™

Future Work

Future Work: Jetpack 5.0.2+ (OP-TEE, UEFI and 5.10.y Kernel)

- The GA release has support for Secure Boot and Disk Encryption
- Changed from ‘trusty’ to ‘OP-TEE’
  - Ported `luks-srv/luks-srv-app`
  - Ported `hwkey-agent/hwkey-app`
- Changed from ‘cboot’ to ‘UEFI’
- Changed from 4.9.y to 5.10.y kernel
- Supports Ubuntu 20.04 host
- Jetson AGX Xavier, Jetson Xavier NX and Jetson AGX Orin platforms only

Future Work: tegra-demo-distro

- Integrate components from tegra-test-distro fork into community supported tegra-demo-distro
  - Add tegra-sysinstall
  - Add crypt initramfs
  - CI and tests!
  - Short-term target is Jetpack 4.6.2/Linux4Tegra 32.7.2 and ‘kirkstone’
  - Upgrade to Jetpack 5.0.2+ (and ‘langsdale’)

Konsulko Group

THE LINUX FOUNDATION

Thank You!

- Matt Madison
- Ilies Chergui
- OE4T community

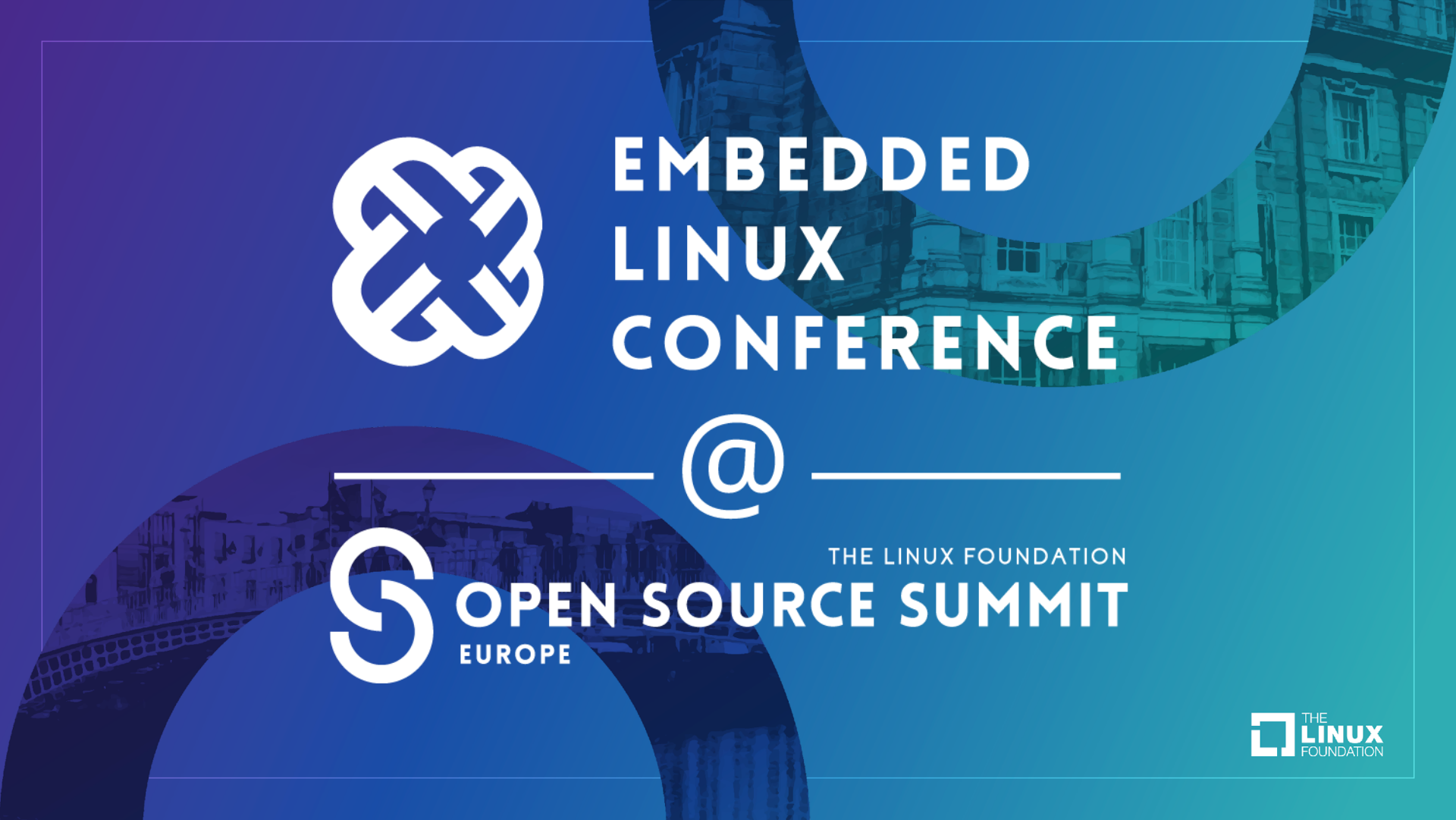
Konsulko Group

THE LINUX FOUNDATION

Questions?

Konsulko Group

THE LINUX FOUNDATION


EMBEDDED
LINUX
CONFERENCE
@
THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
EUROPE
THE
LINUX
FOUNDATION