Open Source secure software updates for Linux-based IVI systems

Arthur Taylor, CTO, ATS Advanced Telematic Systems

Abstract

Cyber security and vehicle recalls are two of the hottest topics in automotive software development right now. Over-the-air updates are critical to allowing vehicle and device manufacturers to mitigate security and warranty risks in deploying software to vehicle fleets, but until recently there was no end-to-end open source solution to manage updates. ATS has been working with GENIVI and AGL to implement secure software updates in their development / reference platforms. In this talk we will present a review of existing open source OTA solutions, introduce the open source GENIVI SOTA solution, describe its architecture and security features, and describe the integration of the OTA Client into the GENIVI Development Platform and the AGL Reference Platform. We will demonstrate updates to a running device based on the OpenIVI platform.

Background

- ATS Advanced Telematic Systems GmbH - Berlin start-up, founded 2013
- ATS is AGL and GENIVI member since 2013
- Have worked on client- and server-side software solutions for automotive
  - Ported AGL to Freescale iMX.6-based automotive platform 2015
  - Developed GENIVI SOTA 2015
- Strong focus on and experience with Open Source

Goal

- To OTA enable AGL Reference Platform
- To use end-to-end Open Source tools and software
- To implement an approach which meets the requirements of automotives
- To implement an approach that is accepted by the community
- Build a Proof-of-Concept system to demo here

Background

- Automotive requirements for updates
- Current state of Linux in Automotive

Automotive Requirements for Software Updates

- Atomic updates
- Revert to previous system on update failure
- Update of bootloader, kernel and configuration data, and filesystems
- Support for signing of images and verification of images on installation
- Support trusted boot and execution of software update in a trusted application environment leveraging the platform’s hardware TPM and/or TEE features.
- Enable/disable a specific feature and apply/rollback system updates
  incrementally rather than through a complete OS update that
  replaces the filesystem

Linux-based IVI Systems

- GENIVI
  - Proprietary Linux-based [1]
    - Accenture, ADIT, Aisin, Delphi, Freescale, KPIT, LG, Magnetti, Mentor XSe, Neusoft, NVidia, Continental, Pelagicore, QuEST, Renesas, Harman, TCS, Visteon, Wind River
  - Open Source
    - Tizen
    - GENIVI Development Platform
- Automotive Grade Linux
  - AGL Reference Platform
- Open IVI

[1] https://www.genivi.org/compliant-products

Yocto

- AGL, GENIVI, OpenIVI and many embedded projects use Yocto

	GENIVI / GDP	**AGL Reference Platform**	**Open IVI**
Base Layer	meta-poky	meta-poky	meta-poky
Arch Layer		meta-intel	meta-intel
Middleware Layer	meta-ivi	meta-ivi-common	
Application Layer	meta-genivi-demo-platform	meta-agl	meta-oim

Research and Planning

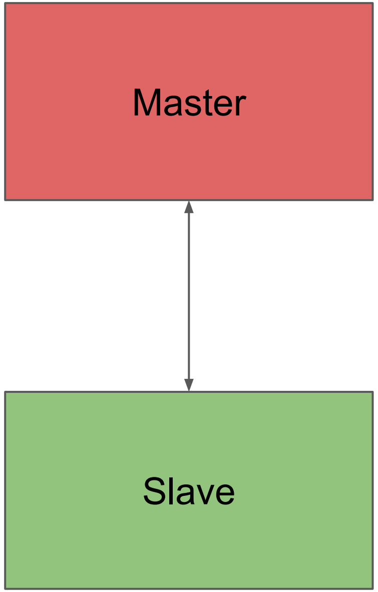
- Investigated existing update tools and approaches for embedded Linux
- 01/05 Commissioned a study from Konsulko team to investigate possibilities
  - SWUpdate, mender.io, resin.io, OSTree, swupd
- 24/05 Published study to AGL mailing list [1]

1. https://lists.linuxfoundation.org/pipermail/automotive-discussions/2016-May/002061.html

Update Strategies - Master / Slave full-system

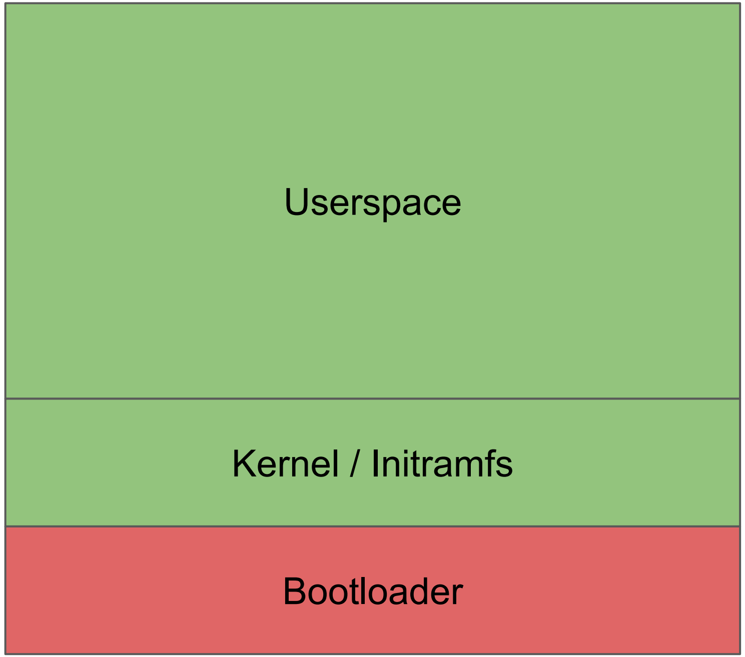
- Master device receives update binaries
- Master device updates Slave device
- Master device validates Slave health
  - Rollback if necessary

- Robust, Simple to implement
- Used in multi-ECU systems
- What to do about the master?



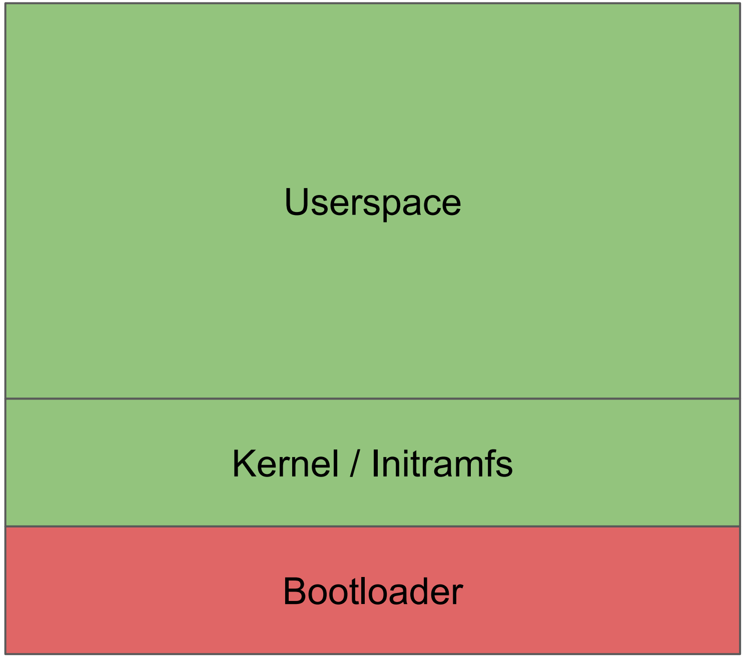
Update Strategies - Full-system by Bootloader

- Userspace downloads binaries
- Reboot bootloader into update mode
- Bootloader flashes new system



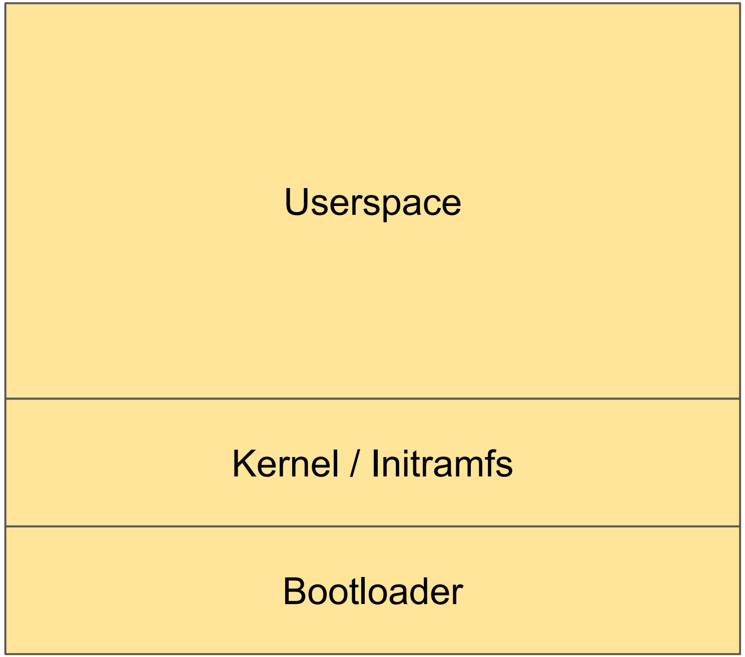
Update Strategies - Full-system by Bootloader

- Implemented in **SWUpdate**
- Require temp storage for update files
- Have to hope that updated system boots
- Bootloader may not be capable of rollback
- How to update bootloader?



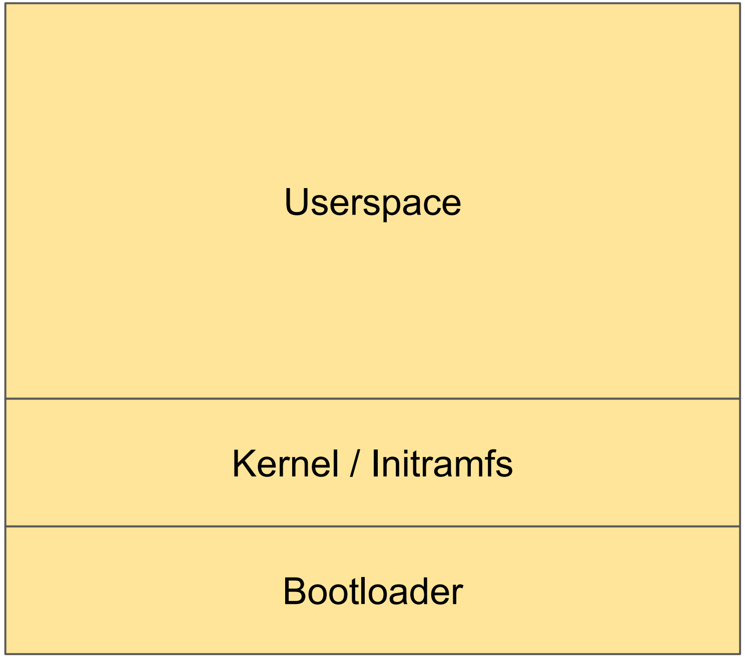
Update Strategies - Full-system by Userspace

- Kill unnecessary processes
- Remount the filesystem readonly
- mlock the update process in place
- Pray
- Stream the new binary directly to flash under the running system
- Pray
- Reboot
- Pray



Update Strategies - Full-system by Userspace

- Implemented by at least one CE device :)
- No additional flash storage required!
- Can update entire filesystem
- Supports secure boot
- Kinda risky...



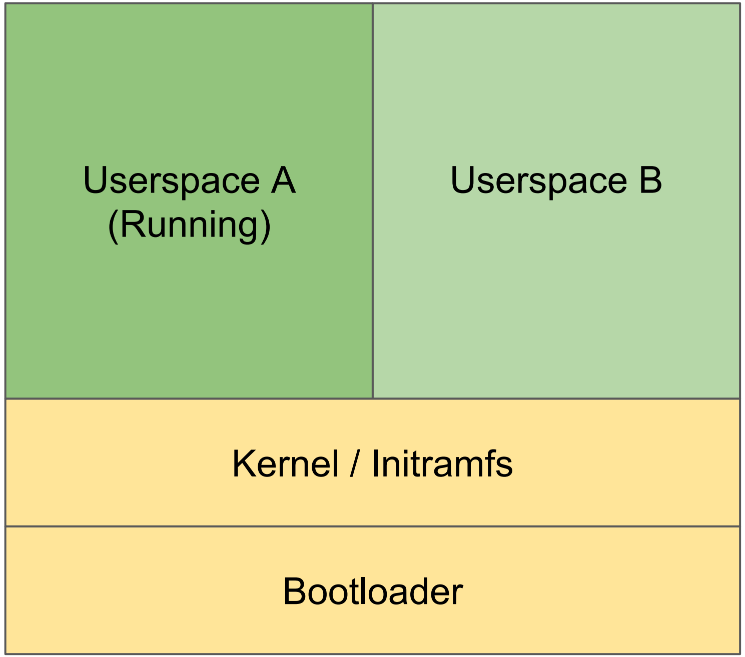
Userspace

Kernel / Initramfs

Bootloader

Update Strategies - A/B full-system

- Userspace A receives update binaries
- Userspace A flashes Userspace B
- Userspace A validates Userspace B
- Userspace A notifies Bootloader
- Bootloader attempts to boot Userspace B
- Revert to Userspace A on error



Userspace A (Running)

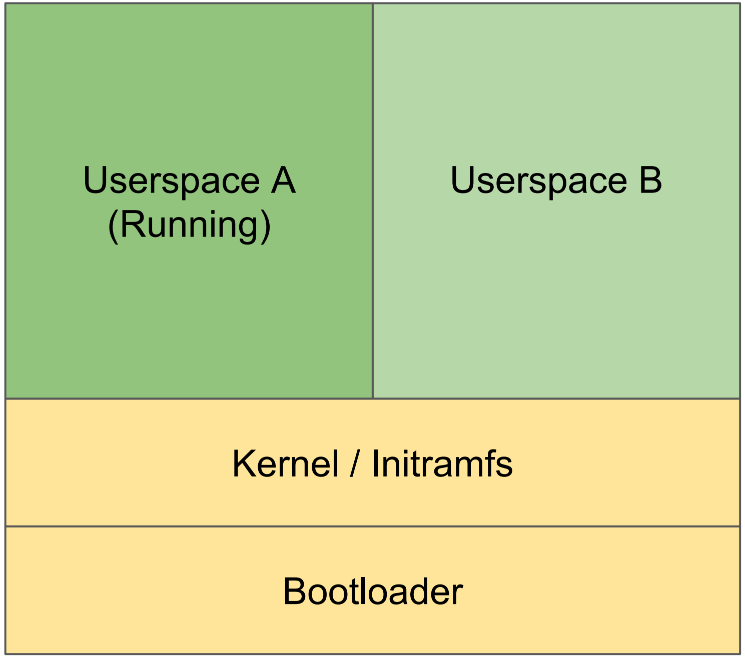
Userspace B

Kernel / Initramfs

Bootloader

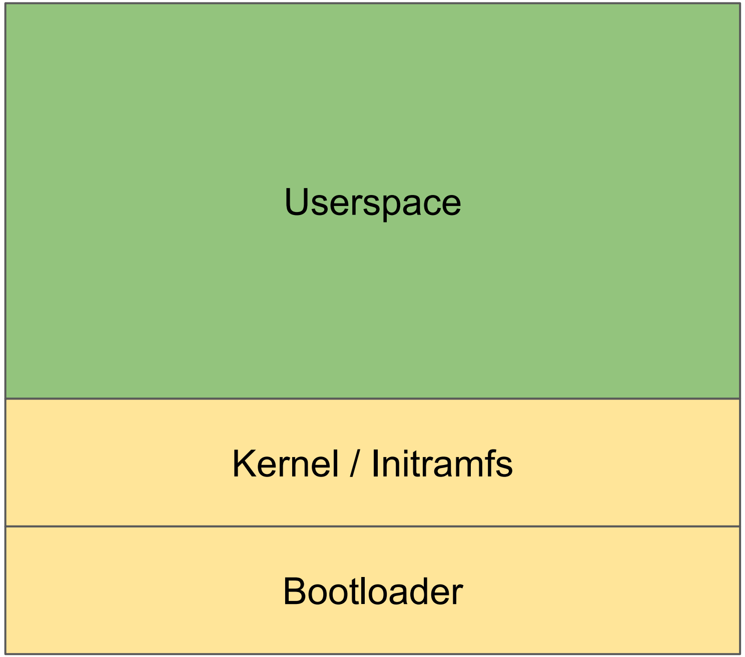
Update Strategies - A/B full-system

- Implemented in **SWUpdate, Mender.io**
- Similar approach in **CoreOS, Resin.io**
- Atomic, with rollback support
- Very robust against unbootable updates
- Supports secure boot
- Requires 2 x Storage
- Bootloader / kernel updates must be done 'blind' (hard to test before reboot)
- All user settings on a separate partition
  - /home easy. /etc on overlayfs?



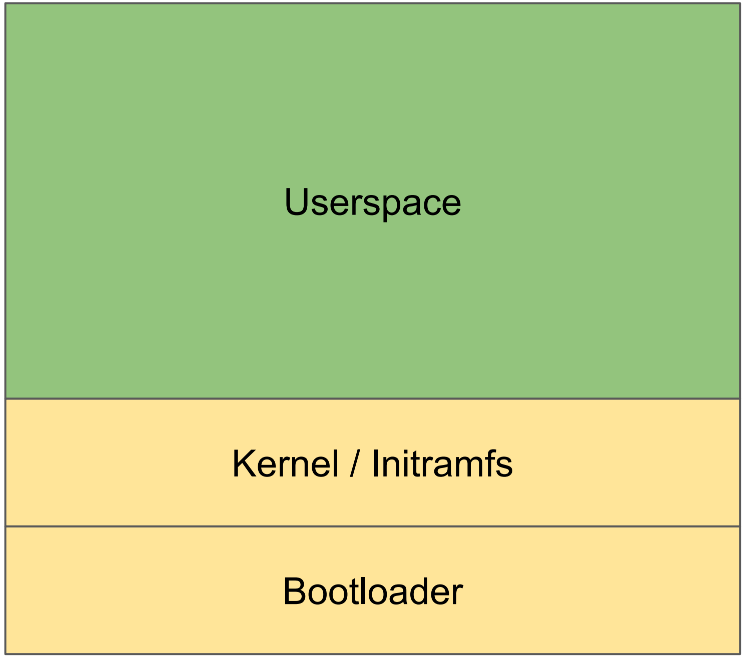
Update Strategies - OverlayFS

- Have a fixed base system
  - Or manage base system with another approach
- Make progressive updates to OverlayFS



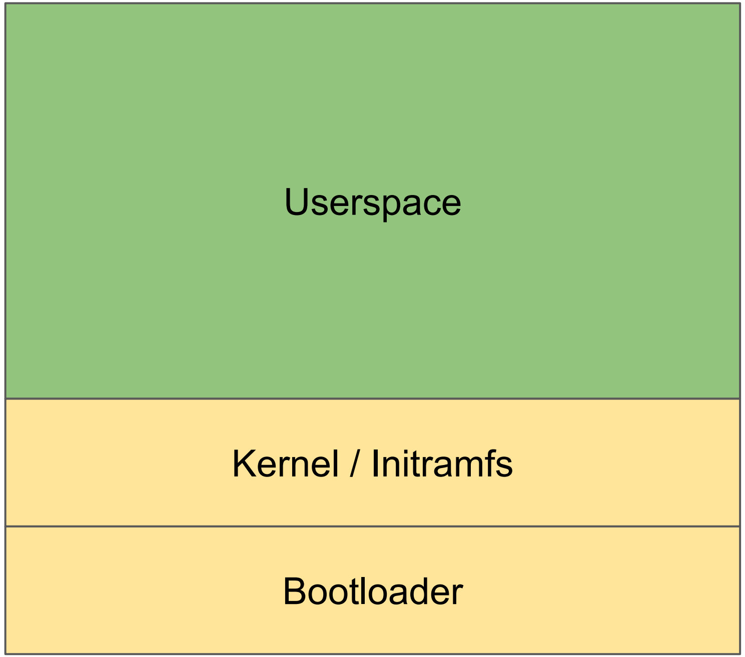
Update Strategies - OverlayFS

- Good support for rollback
- Easy to implement
  - Can be used with any existing packaging system
- What happens if base system changes?



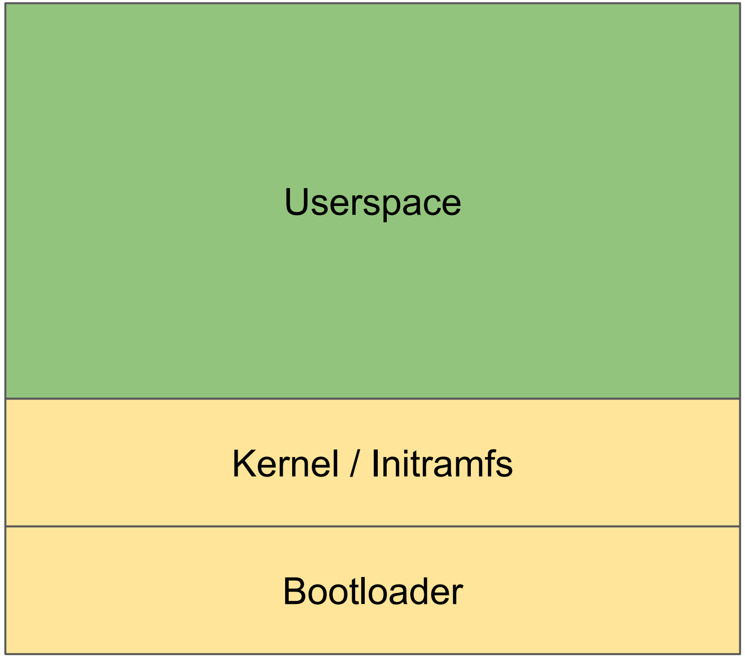
Update Strategies - Progressive by Userspace

- Download updated binaries
- Install them into the running system



Update Strategies - Progressive by Userspace

- Implemented in desktop / server Linux systems
  - .deb, .rpm, etc.
- Similar approach in **OSTree, swupd**
  - OSTree - Git-like tree of hard-links
  - swupd - software "bundles"
- Support for file-level updates
- Package approaches widely used
  - Easy to package and distribute updates
  - Update binaries highly portable
  - Rollback can be tricky
- OSTree approach has good traction
  - Used in GnomeContinuous
  - Allows atomic updates and rollback



Implementation

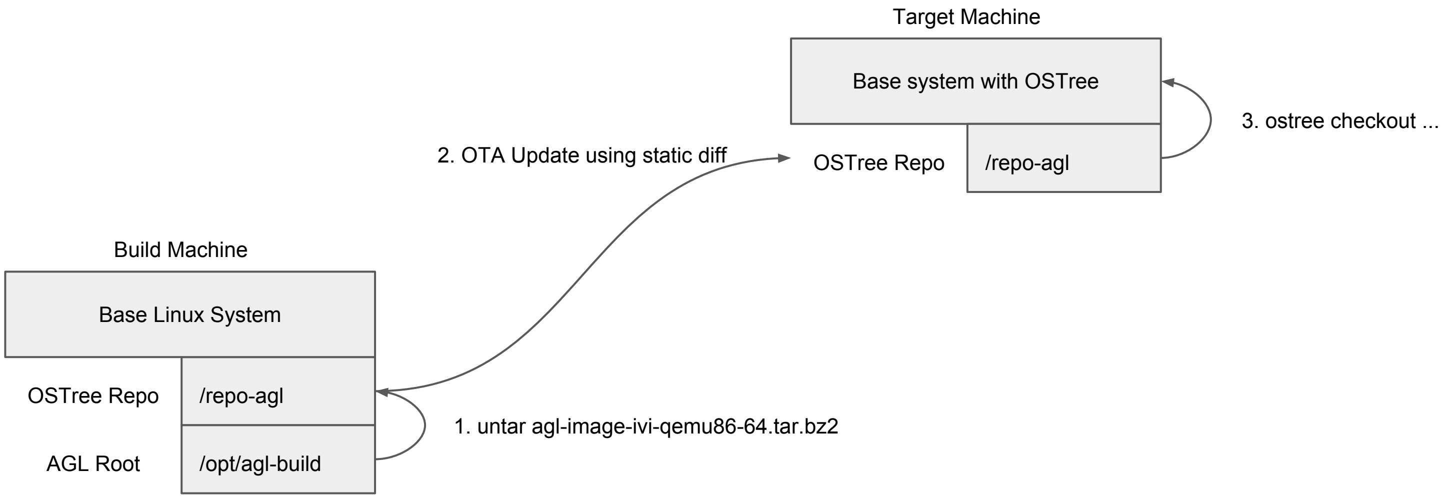
- 30/05 Agreed with AGL team during Vannes F2F on OSTree approach
- 01/06 Analyse technical risk of implementation

1. https://lists.linuxfoundation.org/pipermail/automotive-discussions/2016-May/002061.html

Challenges

- Trusted Execution Environment
  - Only basic features available
  - Any OS interaction would require implementation of eMMC and Filesystem drivers
  - Build a chain of trust some other way
    - Integrity checks from TEE?
    - OSTree metadata in TEE?
- OSTree
  - Needs integration with target OS
  - Requires modification of bootloader / initramfs for full root-fs integration
  - What to do about modified files?
    - Configuration (/etc)
    - App working data (/var)
    - User data (/home)

Chosen Approach


Target Machine
Base system with OSTree
OSTree Repo
/repo-agl
3. ostree checkout ...
2. OTA Update using static diff
Build Machine
Base Linux System
OSTree Repo
/repo-agl
AGL Root
/opt/agl-build
1. untar agl-image-ivi-qemu86-64.tar.bz2