presented by

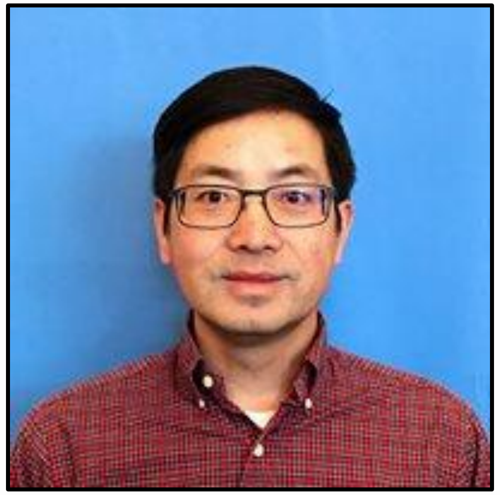
LinuxBoot Integration With UEFI

UEFI 2020 Virtual Plugfest

June 17, 2020

Presented by Jonathan Zhang, Facebook
& Isaac Oram, Intel

Meet the Presenters



Isaac Oram
Principal Engineer
Member Company: Intel

Jonathan Zhang
Software Engineer
Member Company: Facebook

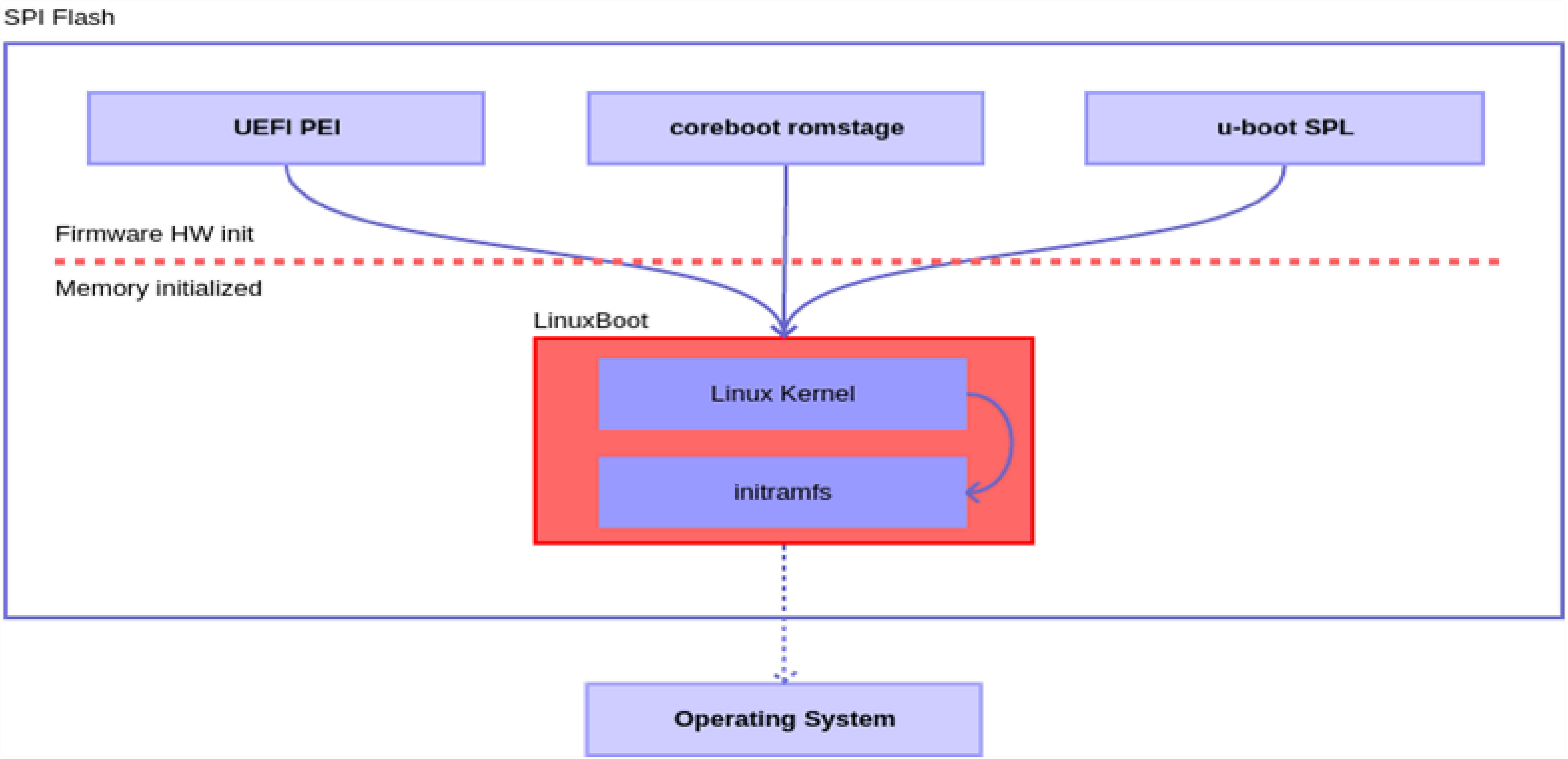
Agenda

- LinuxBoot Introduction
- Proposal
- Call for Action

Why Use Linux to Boot

- Productivity: turn Linux engineer into Firmware engineer
- Manageability: black box → white box
- Update Turn Around Time: Firmware is part of platform ownership
- Security: One set of driver to harden between FW and OS

LinuxBoot Model



What is LinuxBoot?

- Initial bootloader (UEFI, coreboot):
  - Silicon, platform, and board init
  - PCIe enumeration and resource assignment
  - ACPI/SMBIOS
- Linux (kernel/initramfs) in FLASH:
  - Device Drivers (device/networking/file system)
  - Shell
  - Target OS bootloader
- u-root (a type of initramfs):
  - Busybox like
  - Go-lang scripting environment

Why is LinuxBoot Gaining Traction?

- Why now?
- Flash size is less of a problem now
- Linux becomes mature for embedded environment
- Reduce FW/OS duplication
- Hyperscaler scalability challenges

Open System Firmware (OSF)

- Open Compute Project (OCP) incubated OSF in 2018; Formalized in 2019
  - Opencompute.org
- Problem Statements:
  - OCP platform hardware design is open
  - OCP platform adoption increasingly blocked on ability to obtain/customize firmware
  - Hyperscaler scalability challenges increase
- Activities:
  - OCP OSF requirement for OCP platform submission
  - OCP OSF checklist

OCP Open System Firmware Project

https://www.opencompute.org/projects/open-system-firmware

Scope

1. Supports all processor architectures found in the web-scale data center.
2. Support for cloud operating systems
3. Support for compute (GP & AI/ML/FPGA), storage, & network devices.
4. Development and deployment tools
5. Security feature

Regular Project Calls

This project meets every other week on Thursday from 10-11 AM Pacific

- Call Link

Transition schedule for Open System Firmware (OSF) on Open Compute Platform

A proposal from the OSF workstream to the OCP IC, July 26, 2019

Key Objective:

The key objective is to define clear requirements and a timeline for a logo program that ensures OCP has open firmware ready solutions widely available by 2021. The requirements must be clear and testable. The timeline must be reasonable and achievable for hardware vendors and the open communities for OCP, coreboot, LinuxBoot, and TianoCore. The logo program must support the OCP charter and goals.

Background and Introduction:

Open System Firmware (OSF) is a work stream in OCP. While there are many components on OCP systems containing firmware, OSF is directed at the primary CPUs on the board, e.g. the x86 CPUs on x86-based servers and networking equipment. While there may be support CPUs on these boards (e.g. BMC or network card) they are not currently in the scope of OSF.

OCP OSF checklist (project DRAFT)

Summary

This document is in draft status and is being reviewed within the OCP OSF (Open System Firmware) project.

This document captures requirement for the OCP OSF checklist for March, 2021 OCP platform submission related to OSF, according to https://docs.google.com/document/d/1FAFE1apK4J2UVcOAoiJtU0-8MwUNoDnKAopBBF3JChw/edit#heading=h.y6zze7dc2yj9 .

This checklist is intended to enable OCP adopters to use OCP platform with a basically working open system host firmware. This checklist is the initial one, more items/coverages will be added to future checklists, as the industry becomes more mature in terms of OSF.

Explanation/Execution of this checklist is responsible by OCP OSF project leads or their representatives.

What it covers

- It contains minimum items to meet above stated goal.
- It applies to all firmware design approaches, such as open EDKII, or coreboot/linuxboot, hostboot/petitboot, to name a few.
- It applies to all architectures, including x86, ARM, POWER, etc.
- It applies to compute/storage servers and networking servers.
- It applies to host firmware.

Intel/Facebook Joint POC

- POC on Intel® Xeon® Scalable processor
  - Product formerly codenamed Skylake
- POC on first OCP multi-socket server platform
  - TiogaPass (80 threads, 384GB DRAM)
  - Stepping stone for later generations
- Coreboot support for Xeon-SP was enabled first time
  - Code upstreamed
- Coreboot utilizes Intel® Firmware Support Package (FSP)
  - Intel FSP 2.1 API mode

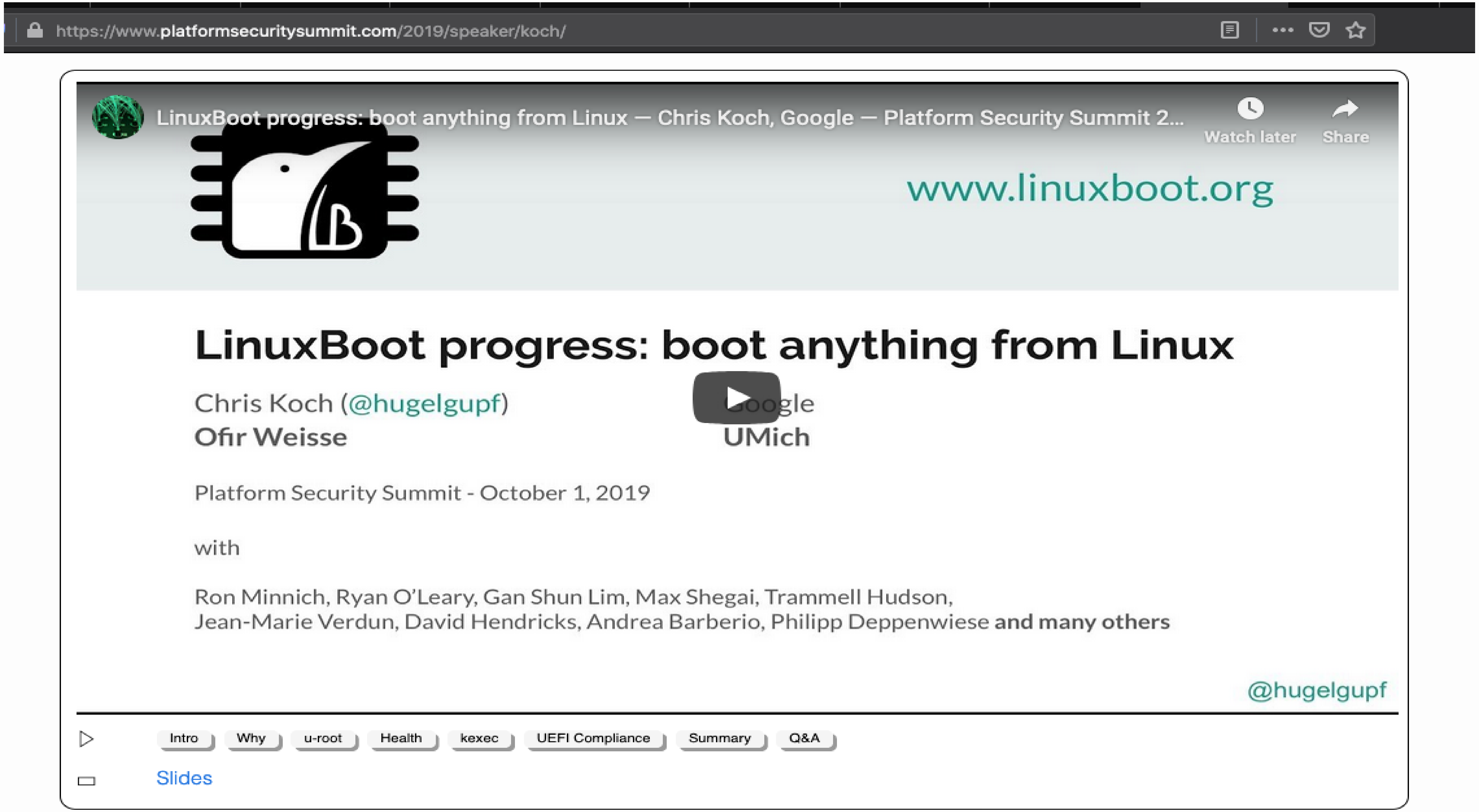
Intel® FSP 2.1 Specification

- Intel® FSP encapsulates basic silicon initialization for use by any bootloader
  - Contents are Intel edk2 silicon initialization PEIM
- Supports two modes
  - API mode for non UEFI bootloaders like coreboot
  - Dispatch mode for UEFI bootloaders like edk2
  - One Intel FSP binary supports both modes
- API mode is simpler
- Dispatch mode is more capable
- Intel planning MinPlatform reference board open implementation that supports both modes
- FB plans to enable coreboot as FSP API mode bootloader

How Target OS is Booted

- How LinuxBoot boots Linux?
  - Kexec
  - CSM mode
  - Google POC
- How could LinuxBoot boot Microsoft Windows?
  - Google POC
  - A small set of requirements are imposed by Windows
- What about ACPI and SMBIOS?
- What about UEFI variables?
- What about other firmware run time services?
- What about SMM?

Google POC


https://www.platformsecuritysummit.com/2019/speaker/koch/
LinuxBoot progress: boot anything from Linux — Chris Koch, Google — Platform Security Summit 2...
Watch later
Share
www.linuxboot.org
LinuxBoot progress: boot anything from Linux
Chris Koch (@hugelgupf)
Google
Ofir Weisse
UMich
Platform Security Summit - October 1, 2019
with
Ron Minnich, Ryan O'Leary, Gan Shun Lim, Max Shegai, Trammell Hudson,
Jean-Marie Verdun, David Hendricks, Andrea Barberio, Philipp Deppenwiese and many others
@hugelgupf
Intro
Why
u-root
Health
kexec
UEFI Compliance
Summary
Q&A
Slides

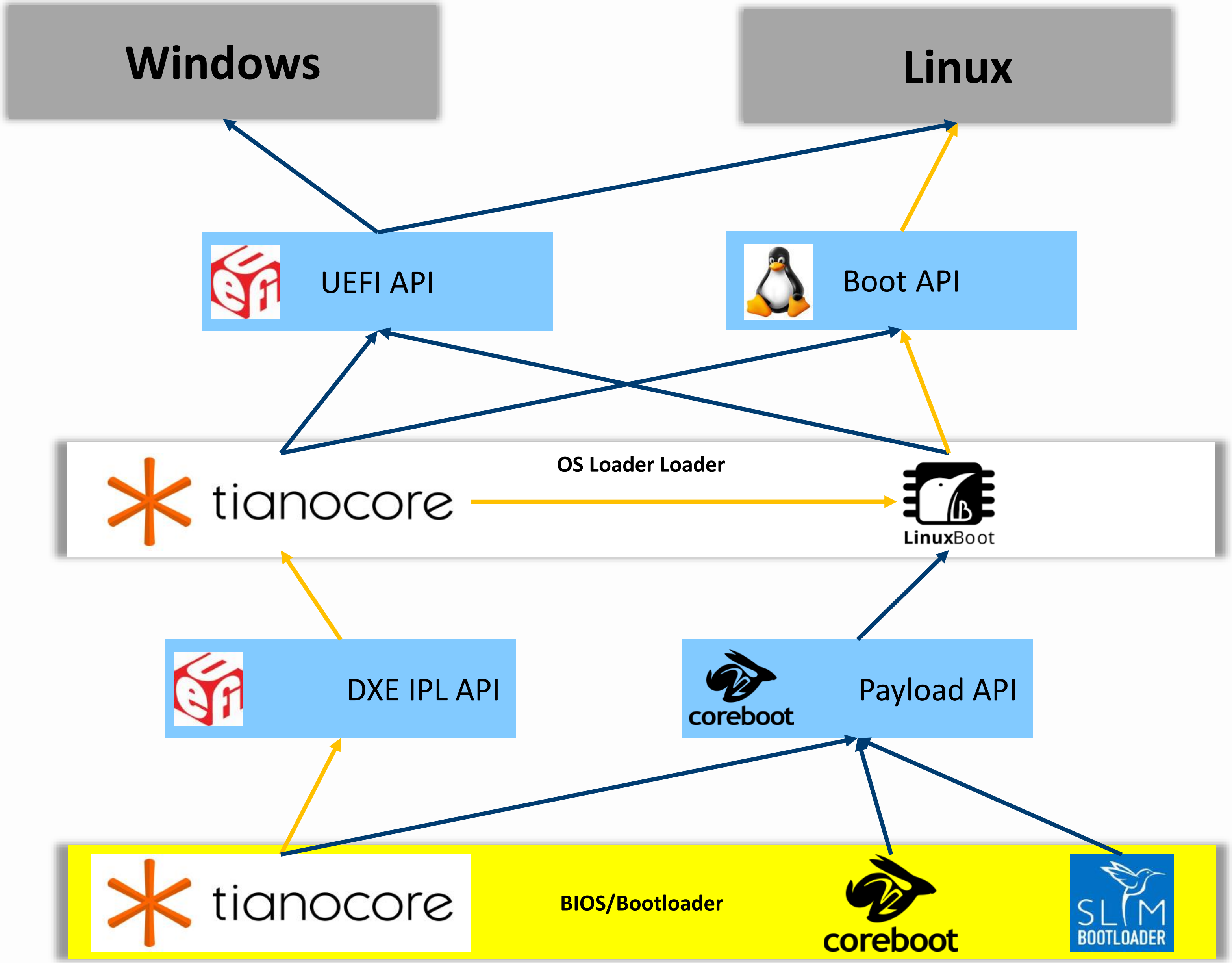
Google POC Findings

- Discovered minimum requirements imposed by the Windows boot manager
- 8 services
- 6 protocols

Table 13. Required UEFI Implementation Elements

Element	Description
EFI_SYSTEM_TABLE	Provides access to UEFI Boot Services, UEFI Runtime Services, consoles, firmware vendor information, and the system configuration tables.
EFI_BOOT_SERVICES	All functions defined as boot services.
EFI_RUNTIME_SERVICES	All functions defined as runtime services.
EFI_LOADED_IMAGE_PROTOCOL	Provides information on the image.
EFI_LOADED_IMAGE_DEVICE_PATH_PROTOCOL	Specifies the device path that was used when a PE/COFF image was loaded through the EFI Boot Service LoadImage().
EFI_DEVICE_PATH_PROTOCOL	Provides the location of the device.

Bootloader and Payload Flexibility



- More choices emerging
- UEFI plays a vital role
- Linux will play a vital role in the future

Proposal

- LinuxBoot and UEFI: two communities working together, could and should
- ACPI/PI: serve both
- UEFI: reduce minimum requirements (UEFI spec 2.8 section 2.6)
- Linux: Upstream support for such minimum requirements
- End result: LinuxBoot becomes UEFI system

UEFI Proposal

- Linux OS boot is very flexible
- Linux UEFI needs can be smaller than UEFI 2.6 minimum requirements
- Proposal: reduce minimum requirements (UEFI spec 2.8 section 2.6)
  - Considering an embedded OS loader target
  - Reduce driver model requirements
    - No console or boot devices required
  - No User Interface, localization, setup application

Call for Action

- Compatible solutions with same standard vs. competing solutions
- Work together with LinuxBoot community to enable new technology
- View open source as an opportunity

Questions?

Thanks for attending the UEFI 2020 Virtual Plugfest

For more information on UEFI Forum and UEFI Specifications, visit http://www.uefi.org

presented by