UEFI and Linux

Matthew Garrett <mjg@redhat.com>

Just what is UEFI?

- Replacement for legacy PC BIOS
- BSD licensed core
- Adds standardised support for new hardware features

Increasingly common

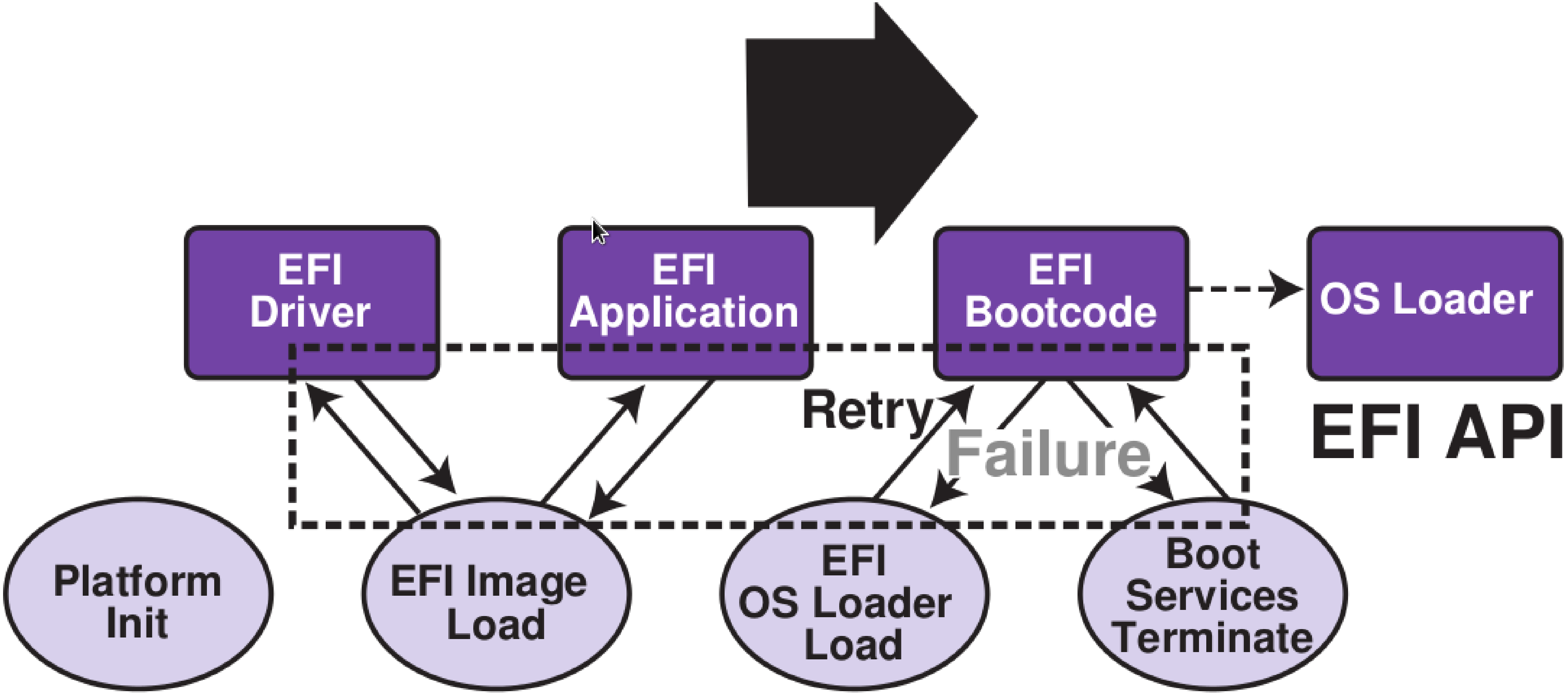
- Most vendors have been shipping UEFI for over a year
- Fairly ubiquitous in consumer hardware
- Spreading through server space
- Required for Windows 8 certification

Convenient timing

- Brings support for >2.2TB disks
- GPT avoids legacy partition table limits
- IPv6 support

How does Linux fit in?

- UEFI is primarily for booting systems
- ...but there's runtime benefits as well



EFI Driver

EFI Application

EFI Bootcode

OS Loader

Retry

Failure

EFI API

Platform Init

EFI Image Load

EFI OS Loader Load

Boot Services Terminate

Standard firmware platform initilization

Drivers and applications loaded iteratively

Boot from ordered list of EFIOS loaders

Operation handed off to OS loader

Booting Linux in a UEFI world

- Firmware reads bootloader off System Partition
- Bootloader has full set of boot services available to it
- Support for native graphics resolutions
- Potential for seamless boot experience

How does the firmware know?

- UEFI boot variables point at each potential boot source
- Firmware can be configured for one-shot booting, and to fall back to other boot targets in the case of failure

Persistent variable storage

- Used for boot variables
- Also available for other services
- Allows Linux to provide crash dumps even on non-enterprise platforms

Standardised firmware updates

- UEFI capsule protocol
- OS passes buffer to firmware
- Firmware updates itself after reboot

That's the good...

- What about the ugly?

Quality of code

- UEFI is becoming near-ubiquitous
- But low consumer adoption means relatively little testing
- Several significant UEFI bugs, including some that can cripple hardware

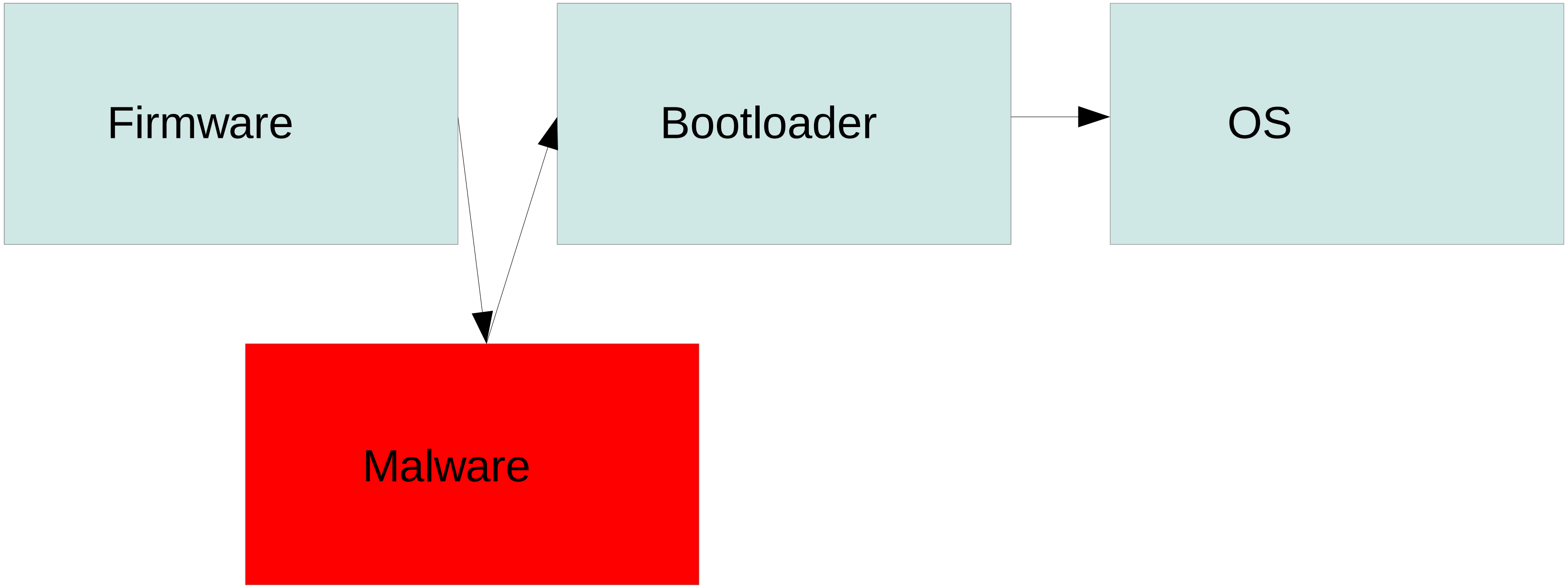
Complex specification

- 2214 pages (2.3.1A)
- Different vendors have different interpretations
- Kernel workarounds required to ensure compatibility

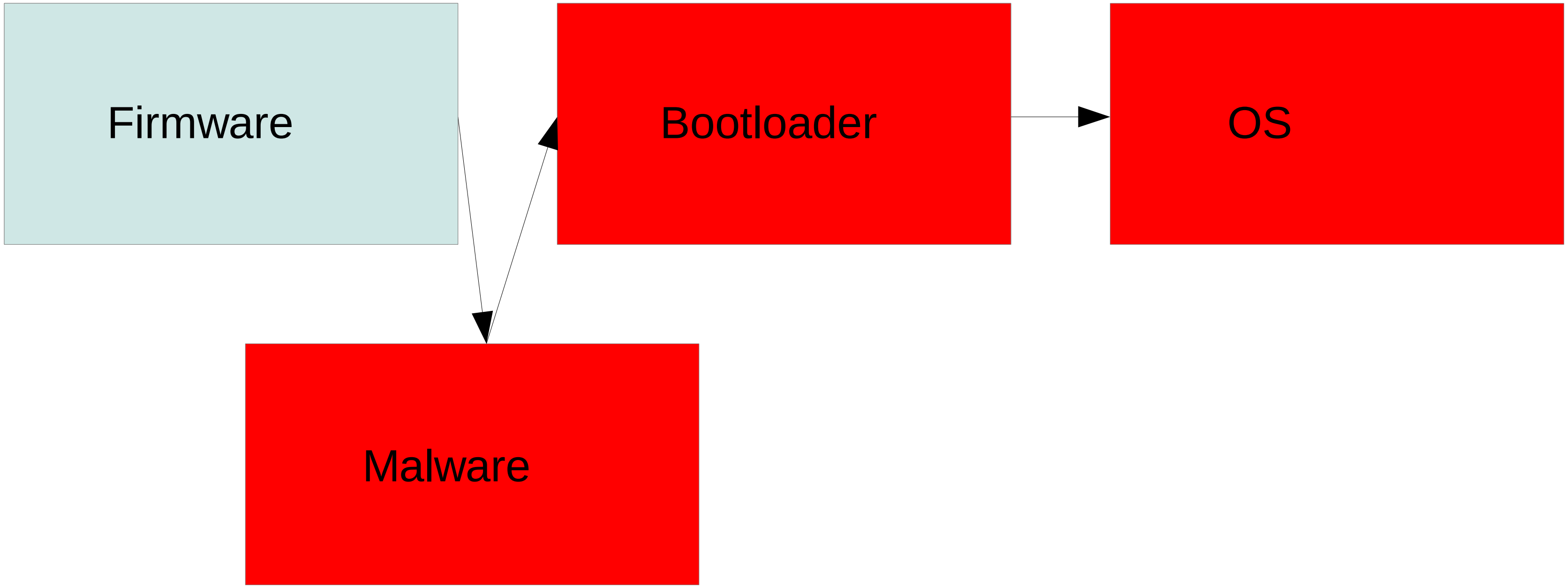
Secure Boot

- Firmware will only execute objects with appropriate signatures
- Public keys must be present in the system firmware
- Control of keys in the hands of platform vendor

Bootkits



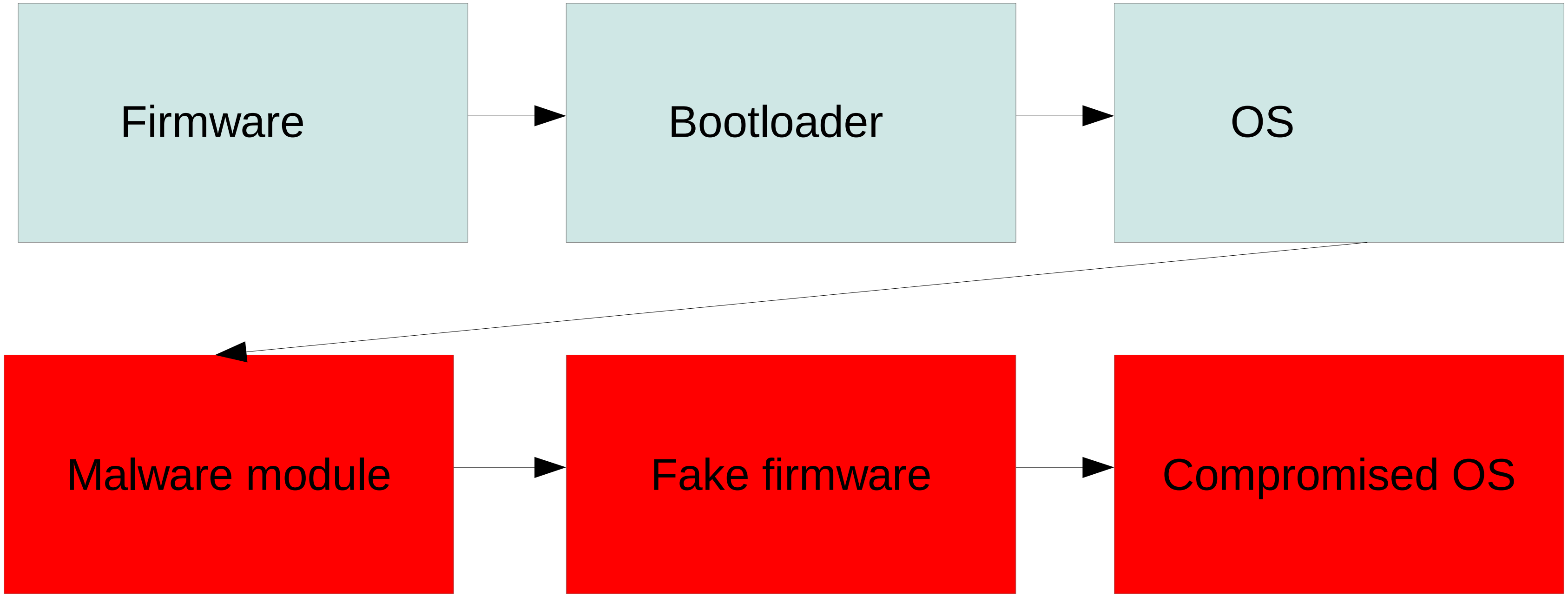
Bootkits



Handling this in Linux

- License concerns (GPLv3)
- Significant quantity of code to write
- Getting anything wrong is a serious problem

Kernel based attack



Secure boot has widespread implications for Linux

- Kernel must be heavily locked down
- No support for unsigned modules
- No direct hardware access from userspace

Questions?