UEFI Secure Boot

Vojtech Pavlik
SUSE, Director SUSE Labs
vojtech@suse.com

Udo Seidel
Linux Strategy at Amadeus
useidel@amadeus.com

Olaf Kirch
SUSE, Director SUSE Linux Enterprise
okir@suse.com

Agenda

- Introduction to Secure Boot
- Why this is a challenge for Open Source
- Secure Boot: SUSE® solution

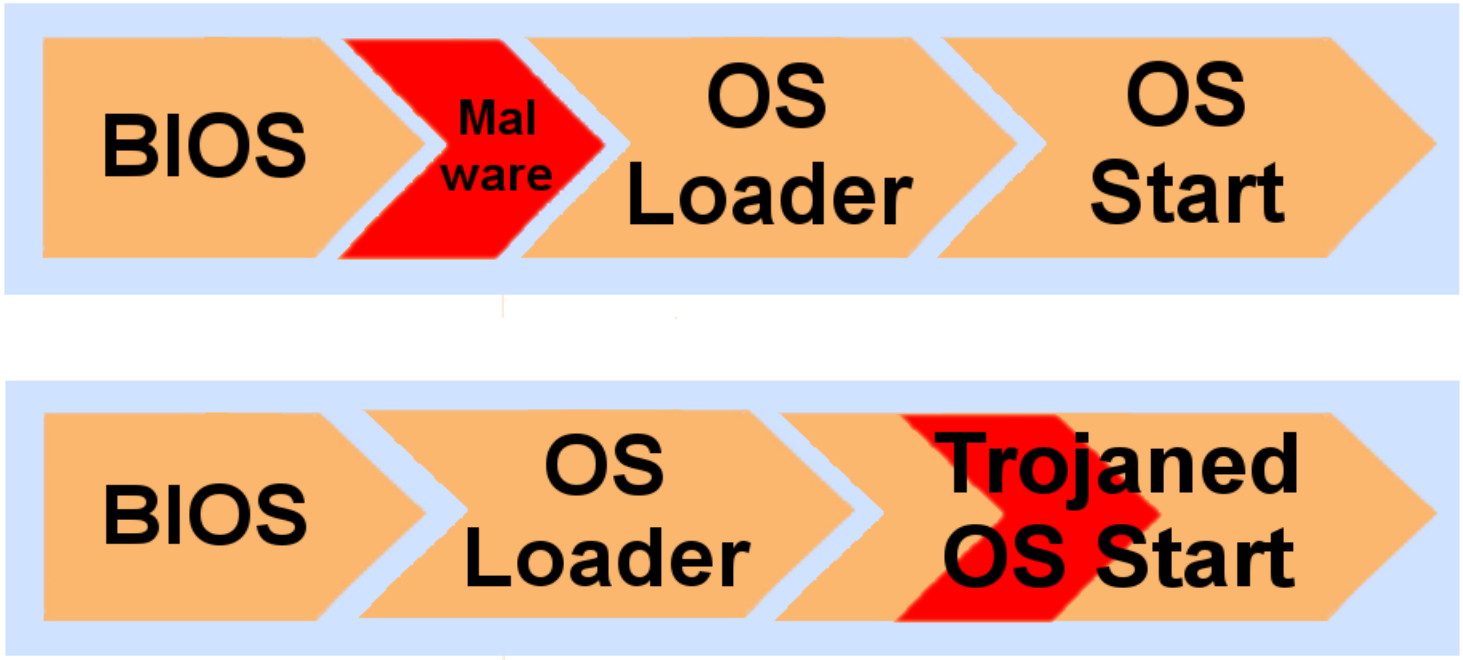
Secure Boot in 60 Seconds

BIOS: Basic Input Output System

- Has been around for a while
- Will boot anything you give it
  - The Operating System you installed
  - A trojaned copy of your operating system
  - A virus written to the Master Boot Record of your disk
- Malware writers have become very good at concealing their steps
  - You can no longer trust that the OS you run is the OS you wanted to run

BIOS: What It's Supposed To Do

BIOS: What You Can Make It Do



All that's needed for the attacker is root access to your hard disk

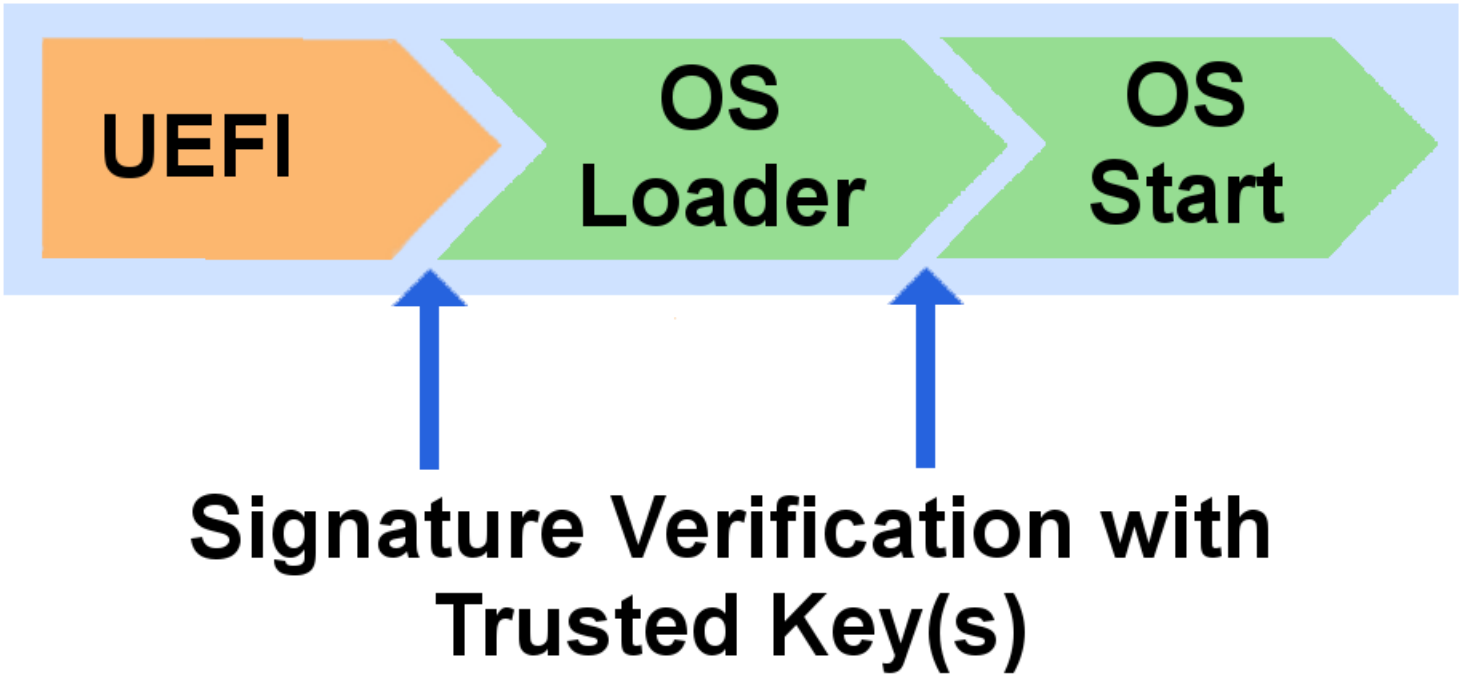
But... Does It Have To Be This Way?

- Wouldn't it be nice if you could trust your laptop to not boot a trojaned kernel?
  - This is what Secure Boot is supposed to help with

Enter UEFI

- UEFI is the *Unified Extensible Firmware Interface*
  - Based on an older standard called EFI
  - This will replace legacy BIOS
- Secure Boot
  - The purpose is to *prevent execution* of Malware OS
  - This is just one aspect of UEFI
  - Specified in version 2.3.1c of the standard
- This is very different from Trusted Computing
  - Which was about attestation of integrity

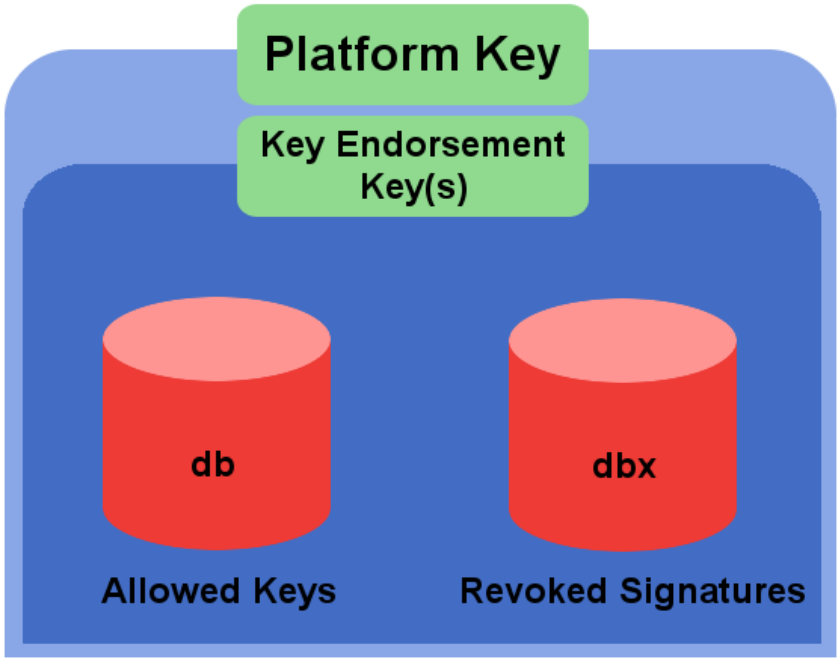
How Secure Boot Changes the Picture



Trust Model

- Secure Boot comes with several databases of public keys embedded in the firmware
  - Anything the UEFI firmware will load *must* be signed by one of these keys
    - the signature is supposed to express a “trust” relationship
    - this includes the OS loader, UEFI modules, firmware updates
  - Anything without a valid signature is not trusted, and hence is never loaded
- There is also a database containing hashes of known malware code
  - This is *not* a virus scanner

Secure Boot Key Hierarchy



- Platform key owned by IHV
- Signs one or more Key Endorsement Keys (such as The Big Microsoft Key)
- Two key databases included in BIOS
- The user is not be allowed to add arbitrary keys. Firmware may allow adding keys signed by KEK
- Hardware vendor populates key databases

Physical Presence

- The specification allows a firmware to disable all these checks *if the user is physically present*
  - Which means it is permitted to offer a setup option to disable Secure Boot completely
- Which also means Secure Boot doesn't protect you from attackers with physical access to the machine

Opting Out?

- Most UEFI implementations on x86 offer an option to boot with “Secure Boot” disabled
- UEFI defines something called “Setup Mode” that allows you to set the PK and KEK
  - This is essentially the state before it leaves the factory
  - Would be a nice way for end users to customize which keys they trust
  - But we don't see many firmwares supporting Setup Mode

Summary

- Secure Boot effectively prevents execution of malware before the Operating System is loaded
  - An attacker with root privilege can still install a trojaned libc, bash, etc
  - Obviously, not a panacea
- Certainly painful from an Open Source perspective

What Does It Mean For Linux?

Is This For Real?

- Windows 8 hardware certification mandates support for Secure Boot
  - All desktop hardware now comes with UEFI Secure Boot enabled by default
- In the server market, we're expecting a slower rate of adoption
  - Virtually no operating system deployed in today's data centers knows how to deal with Secure Boot - yet
  - Some new server hardware already comes with UEFI, but has Secure Boot switched off by default

OS Vendors Have To Deal With It

- Machines with Secure Boot enabled will not boot a Linux kernel unless signed with a “trusted key”
- Linux is Doomed! Film at Eleven!
  - ... not quite yet :-)
- Still, it's an interesting mix of challenges
  - Technical
  - Legal
  - Political

Kernel Community

- If you're a kernel developer, you need to be able to build and run your own kernels
  - Which Secure Boot was designed to prevent
  - One solution is to disable secure boot
- Microsoft UEFI certification is no option for end users
- Needs a technical solution

GPLv3 and Secure Boot

- Our boot loader of choice (grub2) is covered by GPLv3. grub1 and elilo are “GPLv2 or later”
- Preloads + GPLv3: we must allow customers to install an alternative boot loader
  - Legal issues around the “Tivo Clause” in GPLv3
- The Windows Logo Programme Amendment for UEFI boot loader signing says you cannot sign GPL code

Political Issues

- “Evil Empire” rhetorics and conspiracy theories
- We are walking a tightrope in terms of community acceptance
  - Secure Boot is perceived as taking away the end-user's freedom to do with his Hardware whatever he wants
  - FSF calls it “Restricted Boot” rather than “Secure Boot” for this reason
- We cannot announce a solution that leaves the community out in the cold.

Embracing Secure Boot

Minimum Technical Prerequisites

- Boot loader must be signed with a key trusted by the firmware
- Kernel must be signed
- Boot loader must verify the kernel is signed with a trusted key
  - *This* key could be in the firmware's db, or embedded in the boot loader

Getting the Foot In the Door

- As an operating system vendor, we have two ways of getting our boot loader accepted by Secure Boot enabled firmwares

Option A: Be Microsoft

- Work with all IHVs on the planet to get our public key into the firmware database
  - That's a serious scalability issue

Option B: Work With Microsoft

- Use the Microsoft Windows Logo Programme to certify your UEFI loader
  - Microsoft will sign the code with their key
- What this implies:
  - We continue to boot on all Windows certified HW out there
  - You have to sign some serious legal stuff
  - They're asking *lots* of detailed questions on the code we submit

Secure Boot: SUSE® Solution

The MOK: Machine Owner Key

- Return key management back to the user and sysadmin
  - Unconditionally, regardless of what options are missing in the firmware
  - Uniformly, with the same interface and same key formats on all systems
  - Proof of trust by physical presence
  - Implemented by Matthew Garrett, Peter Jones (RH, shim), and Gary Lin (SUSE, MOK)

The MOK and The Shim

shim

BSD licensed preloader

- UEFI executable, db key signed
- loads grub2

MOKList

Additional key database

- stored in a UEFI BootService-only variable

kManager

Manipulates the MOK database

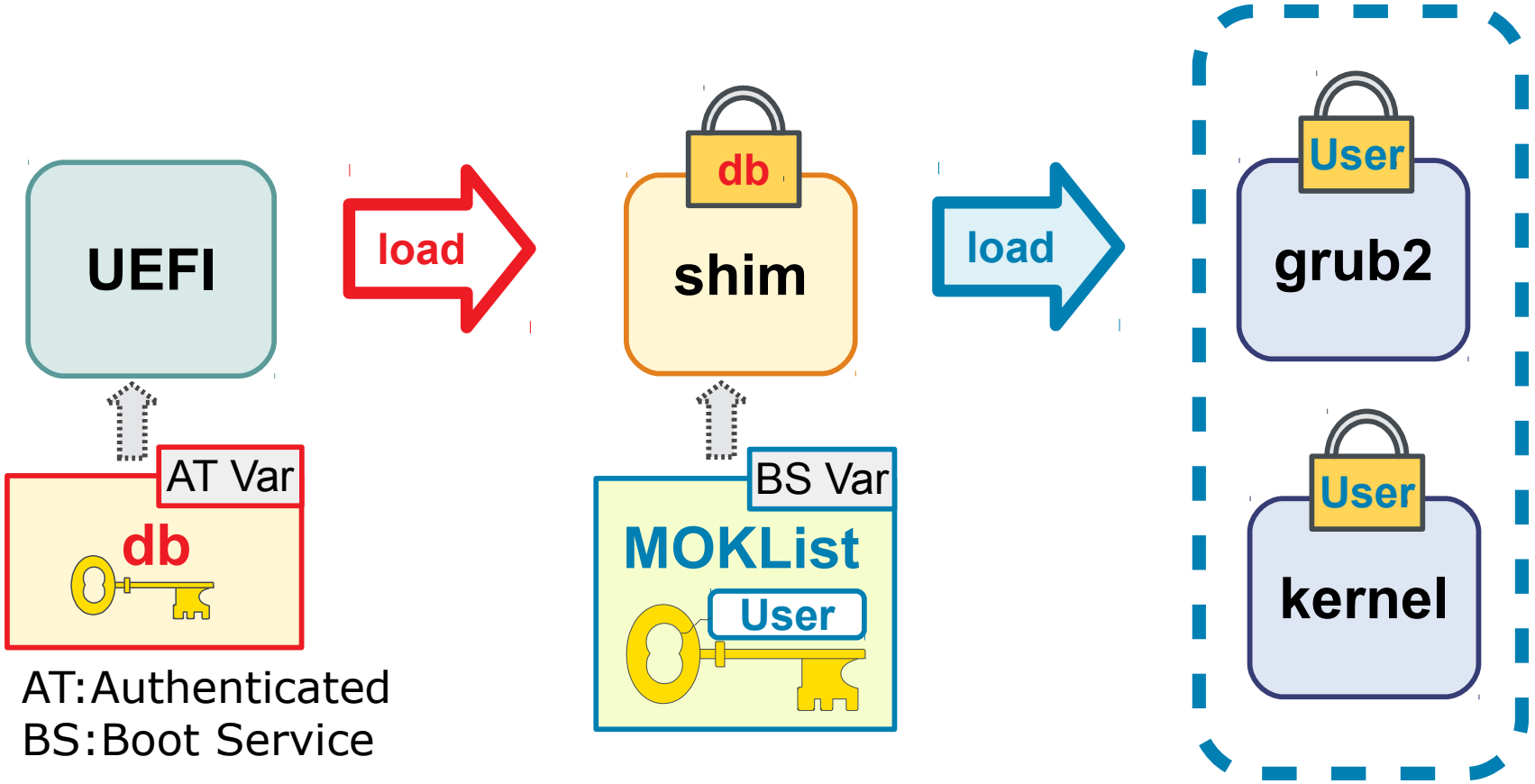
- loaded by shim on demand
- requires user physical presence

mokutil

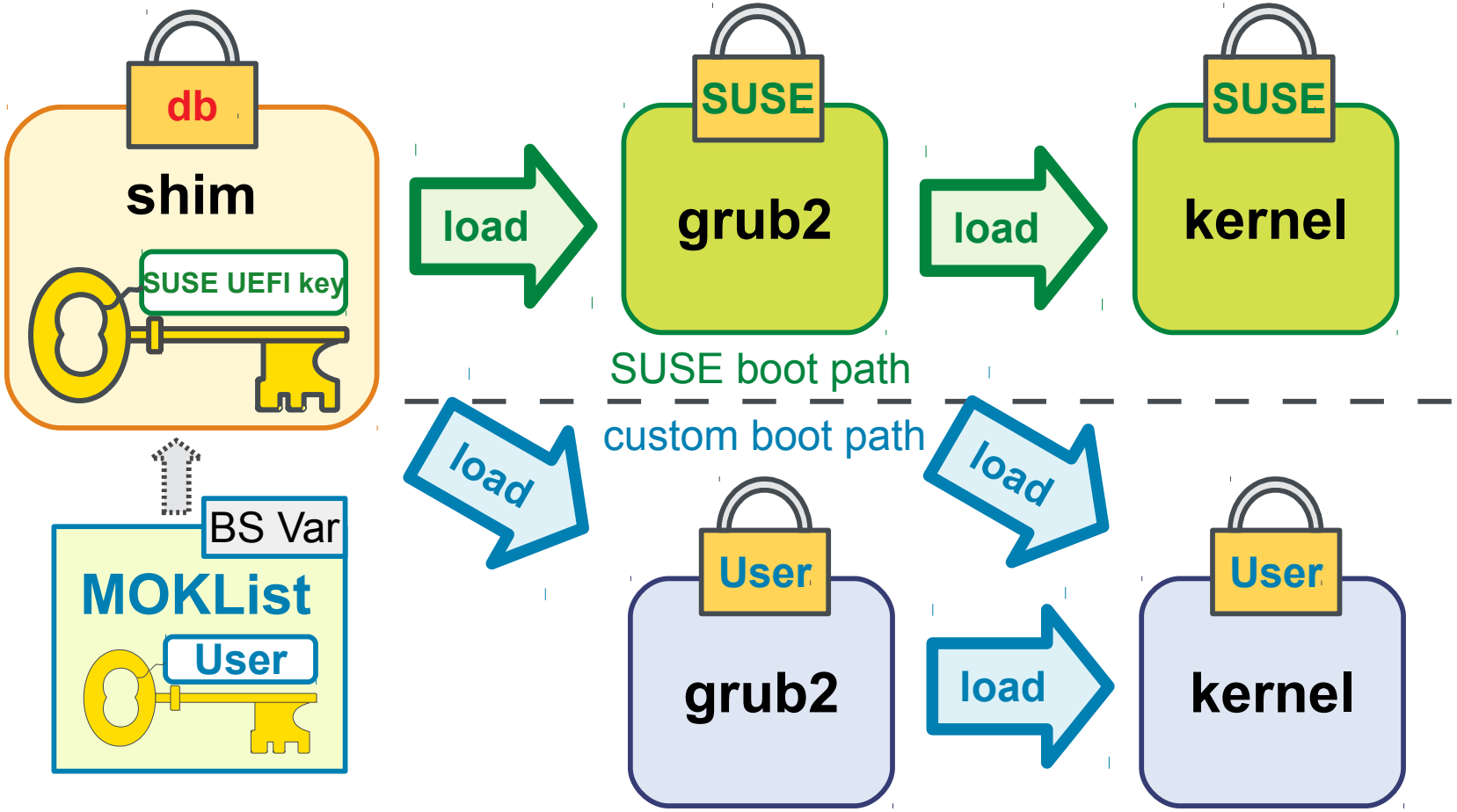
Linux utility

- issues requests to MokManager

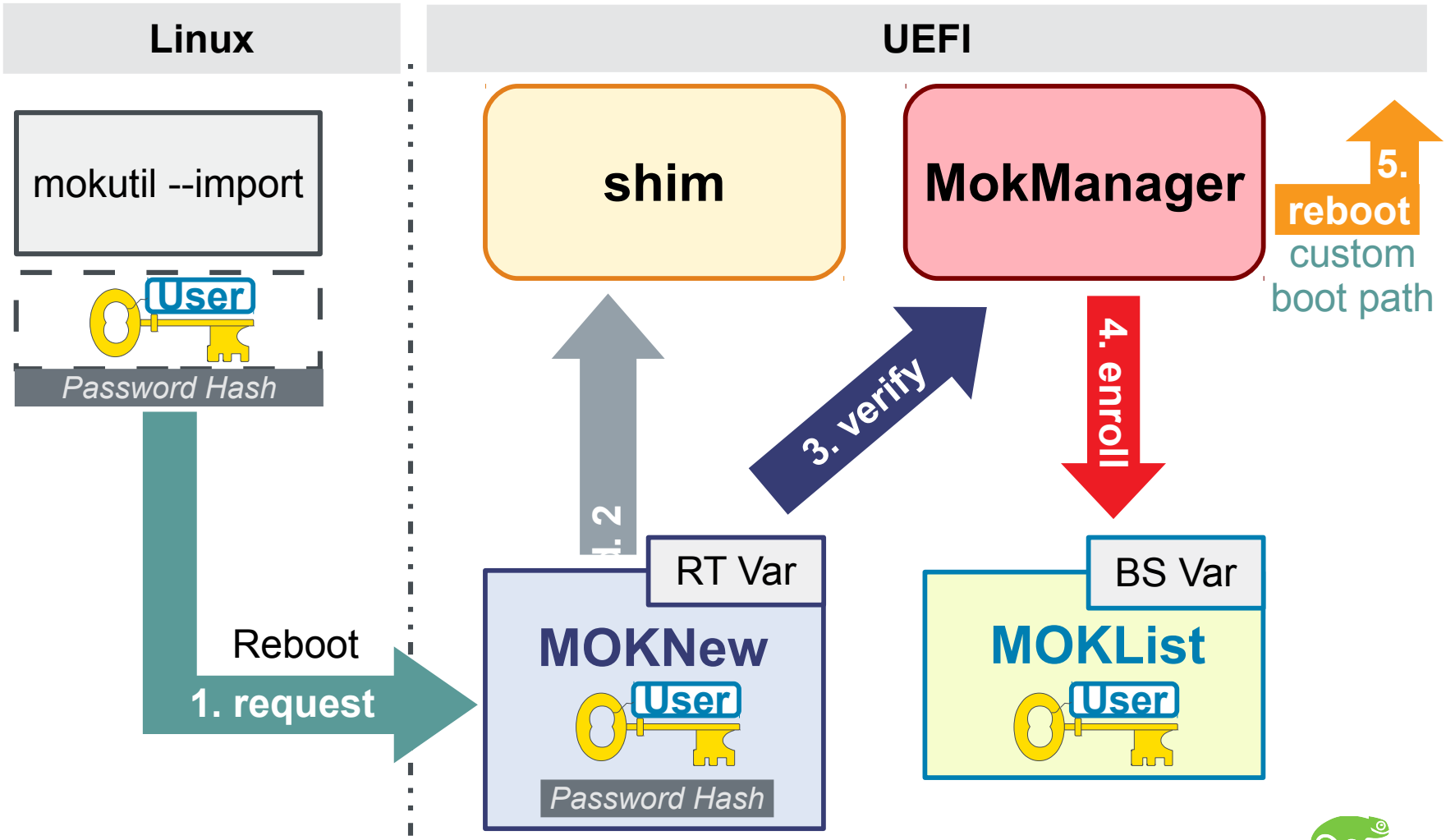
Secure Boot With MOK



Multi-boot With MOK



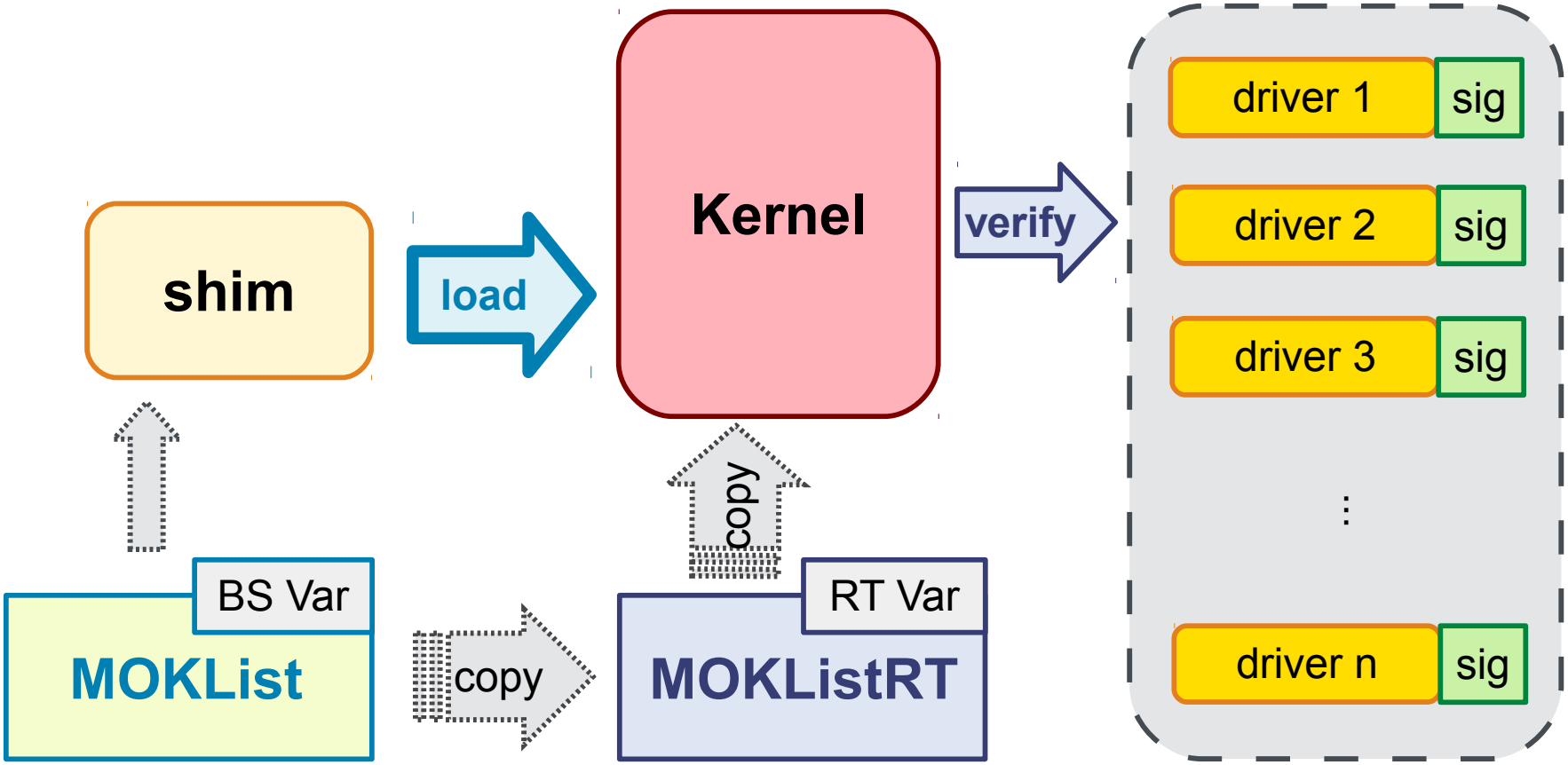
Enrolling a MOK key



What Should Be Signed?

- UEFI mandates only EFI binaries to be signed
  - that protects against bootloader attacks only
  - not much value for a lot of hassle
- The value is in a kernel that can be trusted to be intact
  - hence kernel signing
  - modules need to be signed, too
  - and some functionality disabled: direct /dev/mem writes, iopl()

Module Verification With MOK



MOK and Modules

- MOK allows for easy installation of 3rd party modules
- An addition to the MOKList is requested upon installation of a module package
- The user then approves the addition on a subsequent reboot
  - yes, a reboot is needed to prove intent and physical presence of the machine owner
  - this is somewhat less convenient than using KEKs and AVs, but, and that's most important, works in all cases, even when you can't install your own KEK in the firmware

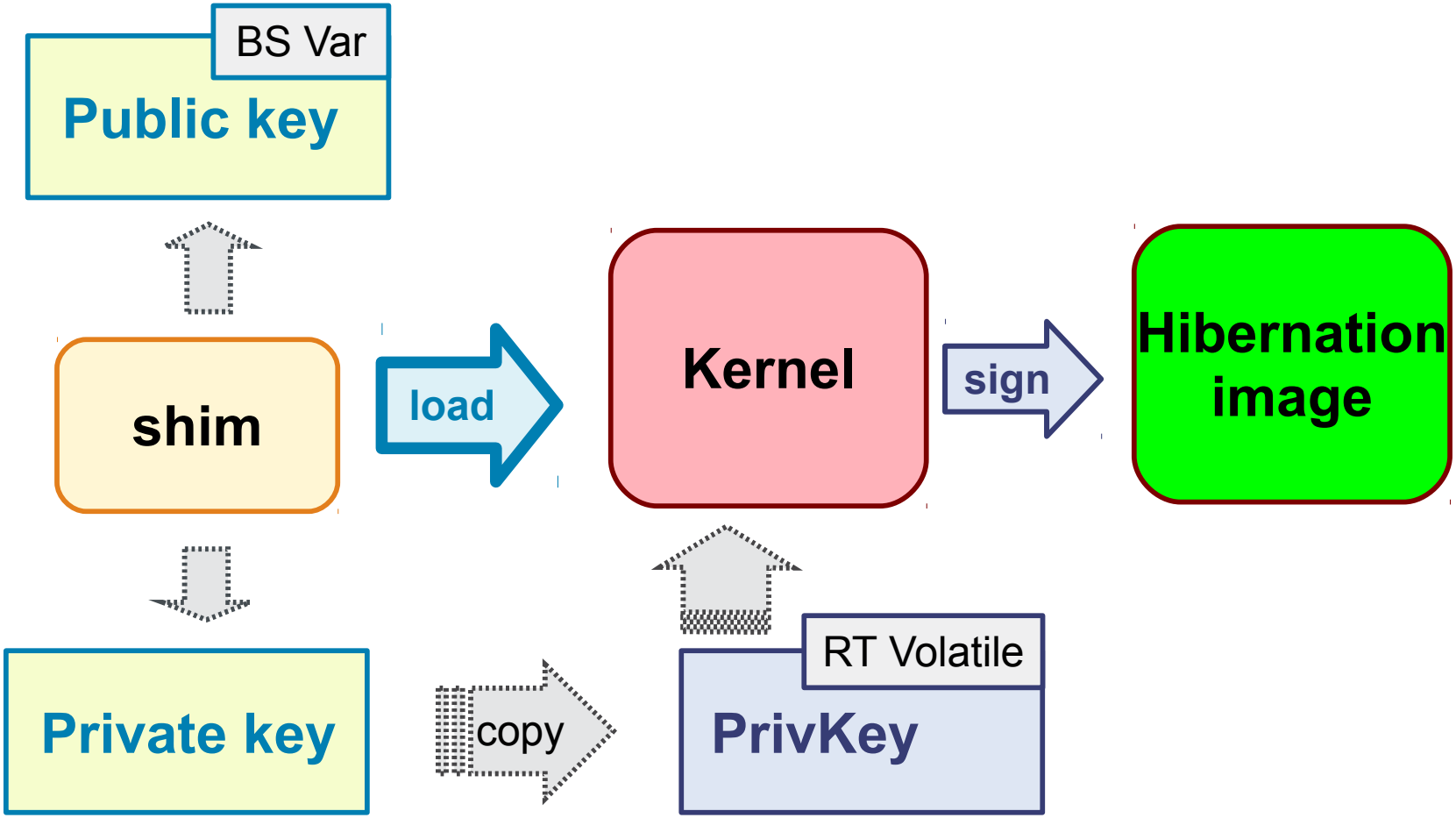
Restrictions in Secure Boot Mode

- No direct access to IO port, must use kernel interface
  - KMS drivers are required for graphics card
- No direct access to memory
  - No /dev/mem, no /dev/rmem
- Not possible to load unsigned 3rd party modules

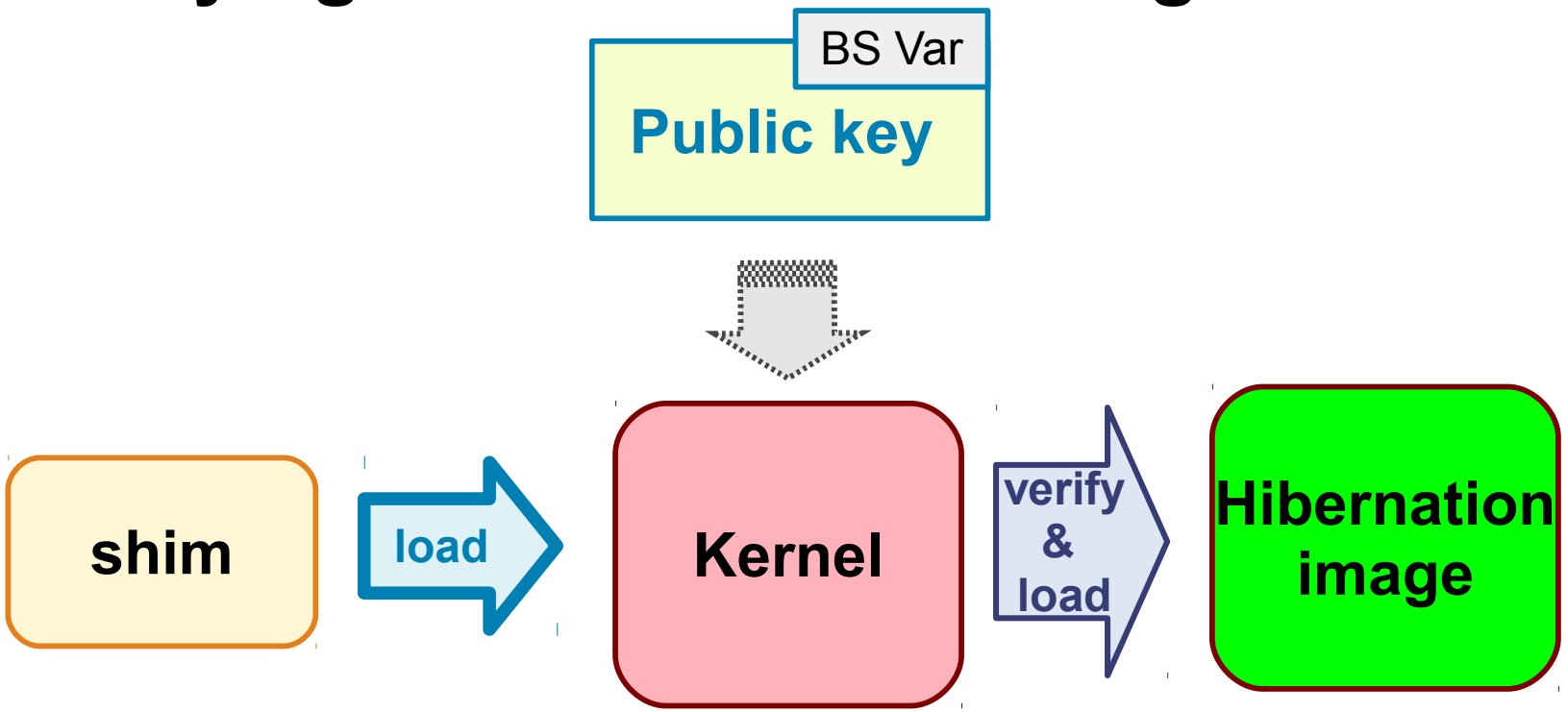
Further Limitations

- Hibernation
  - aka S4 or Suspend-to-disk
  - stores RAM image on disk
  - image can be manipulated, or even created by malware
  - it's a hole in the security model
  - solution: HMAC Signatures
  - implementation by Joey Lee (SUSE) in progress
- kexec and kdump
  - allow for executing an entirely new OS
  - must do signature verification
  - in-kernel implementation by Vivek Goyal (Red Hat) in progress

Signing the Hibernation Image



Verifying the Hibernation Image



Secure Boot in SLE11 SP3/openSUSE

Install SP3/openSUSE on a Secure Boot System

- DVD image should boot nicely with Secure Boot enabled
- If DVD image isn't recognized properly (some firmwares are “funky”), run isohybrid –uefi on ISO image and dump ISO image to an USB stick (not possible for MINI ISO)
- Secure Boot support will be automatically enabled by installer.
- If you are installing on UEFI platform but Secure Boot isn't enabled, you can enable support in Bootloader section, Bootloader options / Support Secure Boot.
- Your boot manager will have a entry named

  “SUSE Linux Enterprise 11 SP3/openSUSE 12.3”

Secure Boot Implementation Details on SP3

- Grub2 and shim are hidden behind an “elilo” wrapper
- In /etc/elilo.conf, “secure-boot = false|auto|true” is used to determine if grub2/shim should be installed on EFI partition or just elilo.
- Careful with “secure-boot = auto”, if you enable then disable Secure Boot in firmware. Currently, “auto” is set when Secure Boot is detected at install time. Will switch to “yes” to prevent breakage
- SUSE key is available in /etc/uefi/certs/ (need to be converted to X509 format to be enrolled, see http://en.opensuse.org/openSUSE:UEFI_Secure_boot_using

Enrolling MOK

- Needed to allow 3rd party module (or kernel) to be accepted by shim / Secure Boot
- Use “mokutil --import <key.der>” then reboot
- You'll need to type a password for MOK List (or use –root-pw to use root password. Installer will do that automatically if 3rd party module are enrolled at install time)
- Reboot and accept changes in shim

Summary

- With Secure Boot support, SLE 11 SP3 will allow you to secure systems even more strict (if required), trade off will be for dump scenarios

- MOK handling allows flexibility for testing, upgrading and 3rd party support

- SUSE integration for lower effort handling

Corporate Headquarters
Maxfeldstrasse 5
90409 Nuremberg
Germany

+49 911 740 53 0 (Worldwide)
www.suse.com

Join us on:
www.opensuse.org