LINUX SECURE BOOT WITH TPM AND FDE

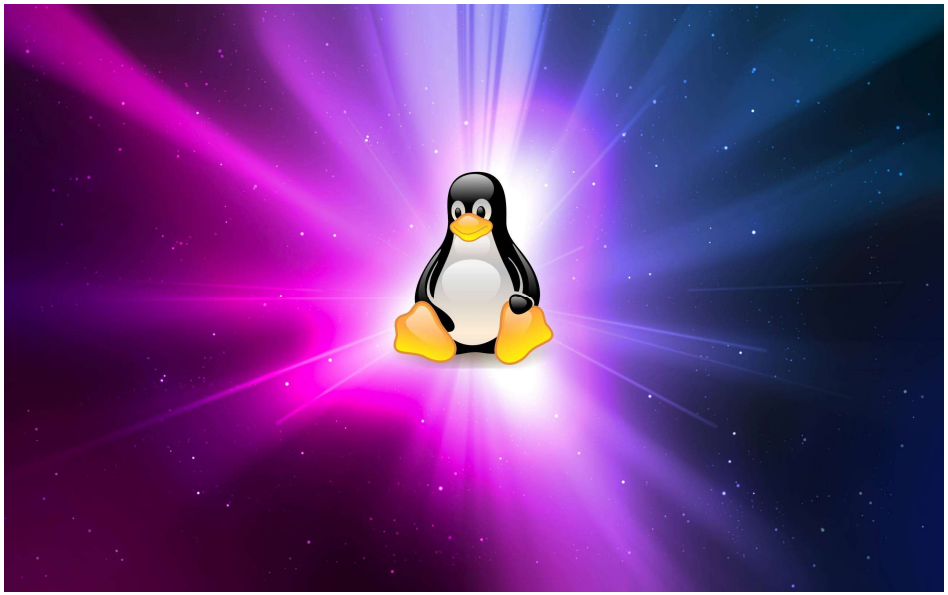
DAVID COLLINS

ABOUT ME

- I work at SEPTA (Philadelphia area public transit)
- My previous experience: Linux sysadmin, Windows sysadmin, network engineer
- My current role: Security engineer
- Email: david@decollins.com
- GitHub: https://github.com/dcollins42

LINUX BOOT SECURITY

- Things I Will Talk About
  - Secure Boot Architecture on x64
  - Secure Boot setup on Kali Linux
  - Trusted Platform Module (TPM) for LUKS key storage

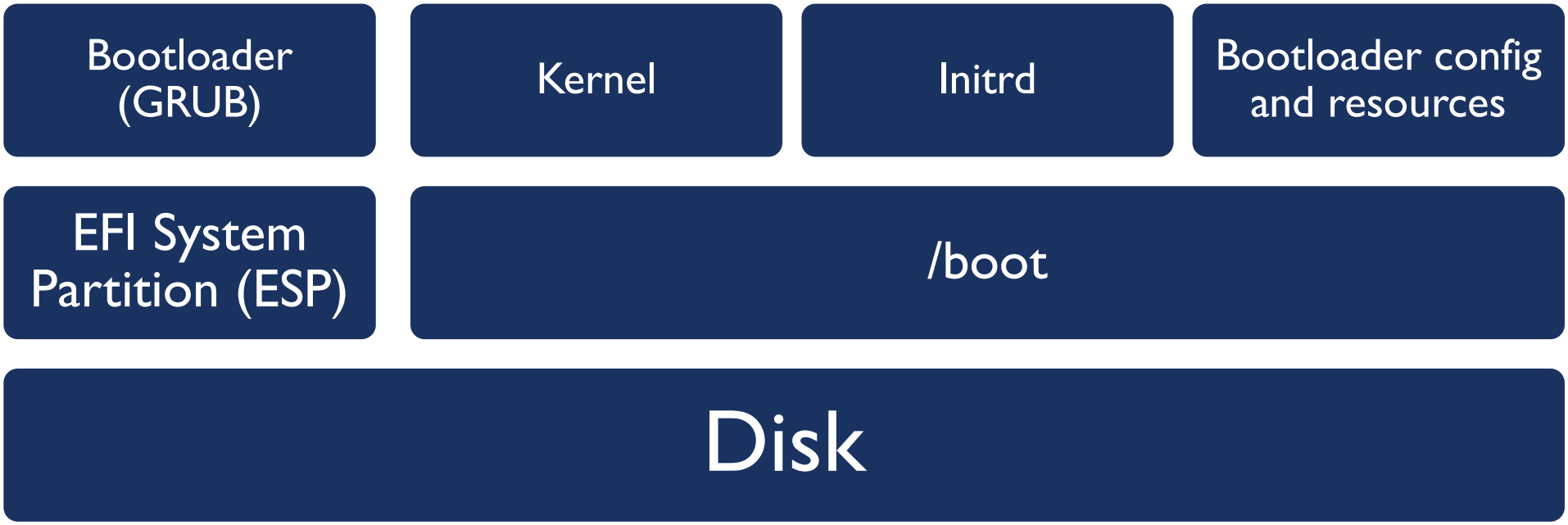


SAFETY TIPS

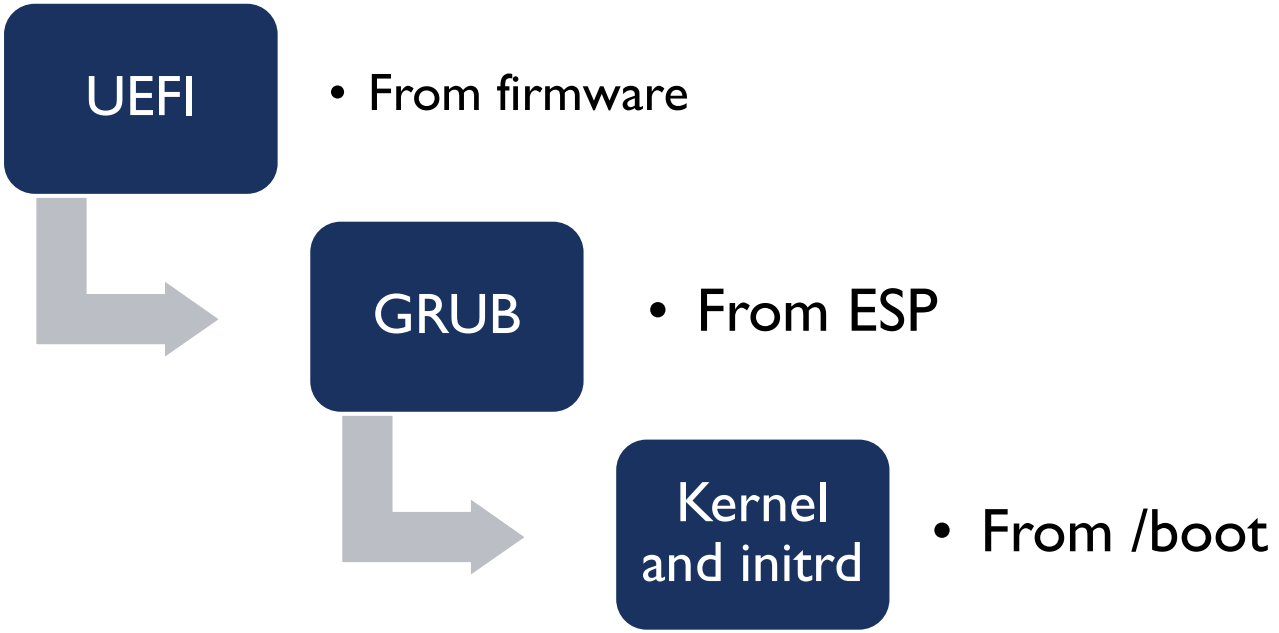
- You can brick your computer or lock yourself out while setting up Secure Boot.
- If you dual-boot with Windows, have a copy of your BitLocker recovery password.
  - `manage-bde -protectors -get C:`
- Secure Boot Platform Keys (PK) allow full control of your computer's boot process. Losing them can brick your computer.

KALI (AND DEBIAN) DEFAULT BOOT SETUP

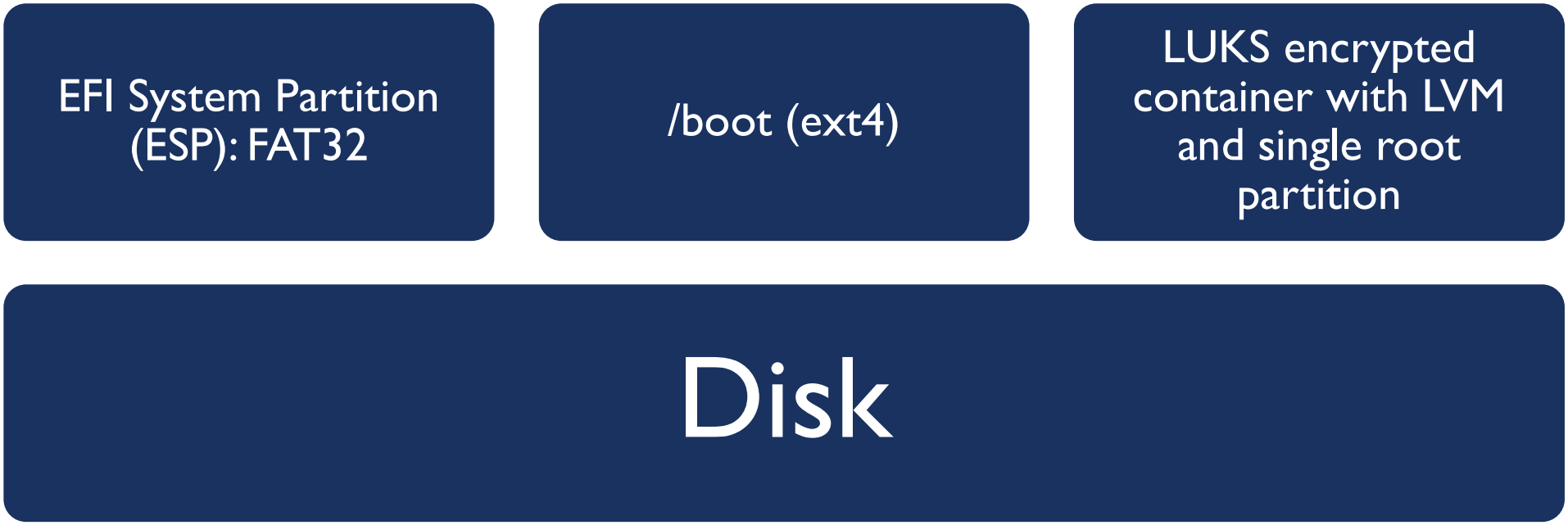
KALI (AND DEBIAN) DEFAULT BOOT SETUP



BOOT PROCESS

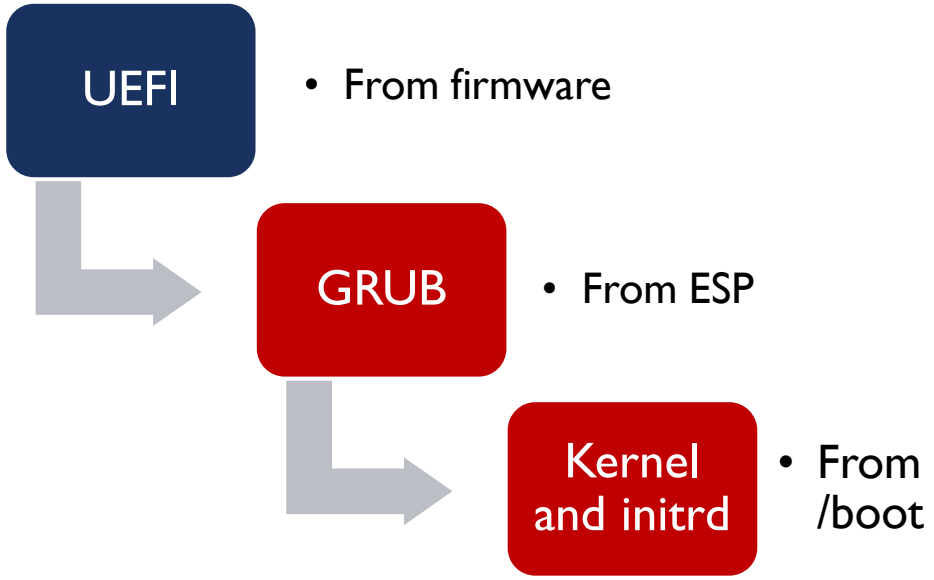


EXAMPLE SYSTEM BOOT SETUP



ATTACKS AGAINST DEFAULT BOOT SETUP

- ESP is always unencrypted
  - Tamper with bootloader.
- /boot is unencrypted
  - Tamper with kernel and initrd.
  - Tampered with bootloader configuration files.



SECURE BOOT ARCHITECTURE X64

Platform Key (PK)

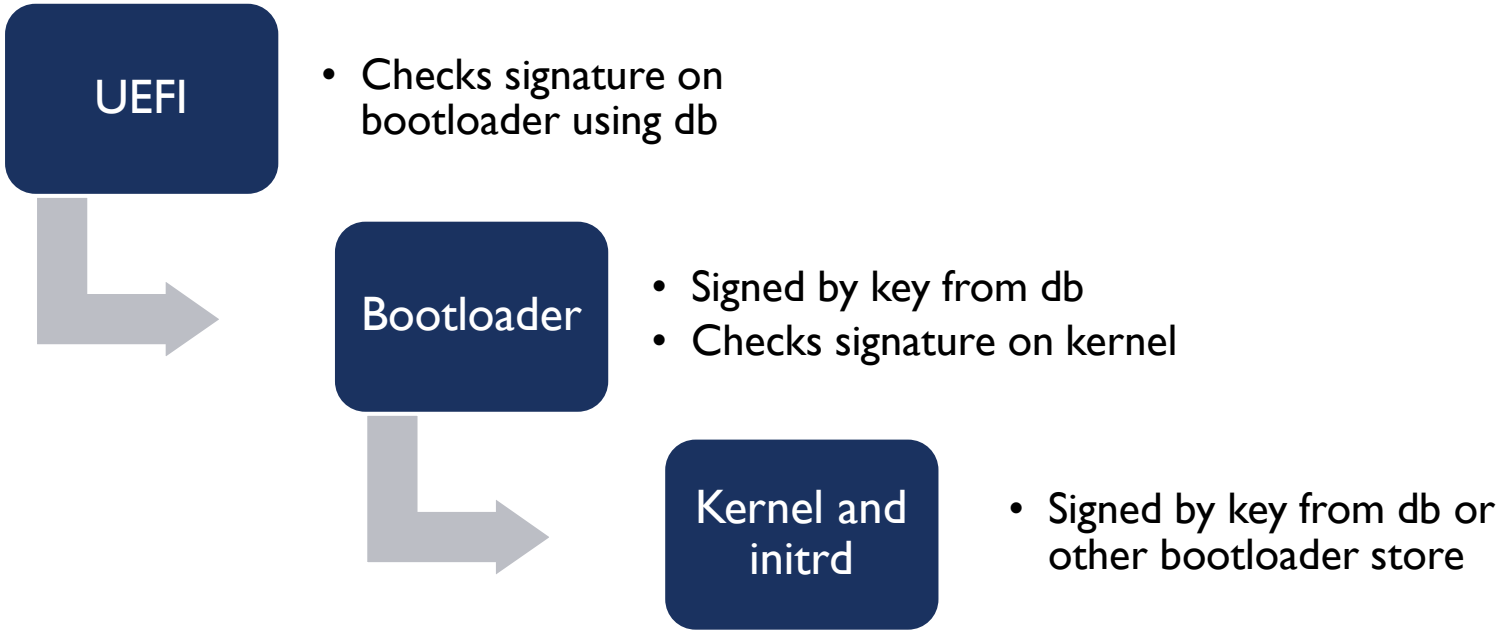
Key Exchange Keys (KEK)

db (signing keys)

dbx (blocked keys)

SECURE BOOT ARCHITECTURE X64

SECURE BOOT PROCESS



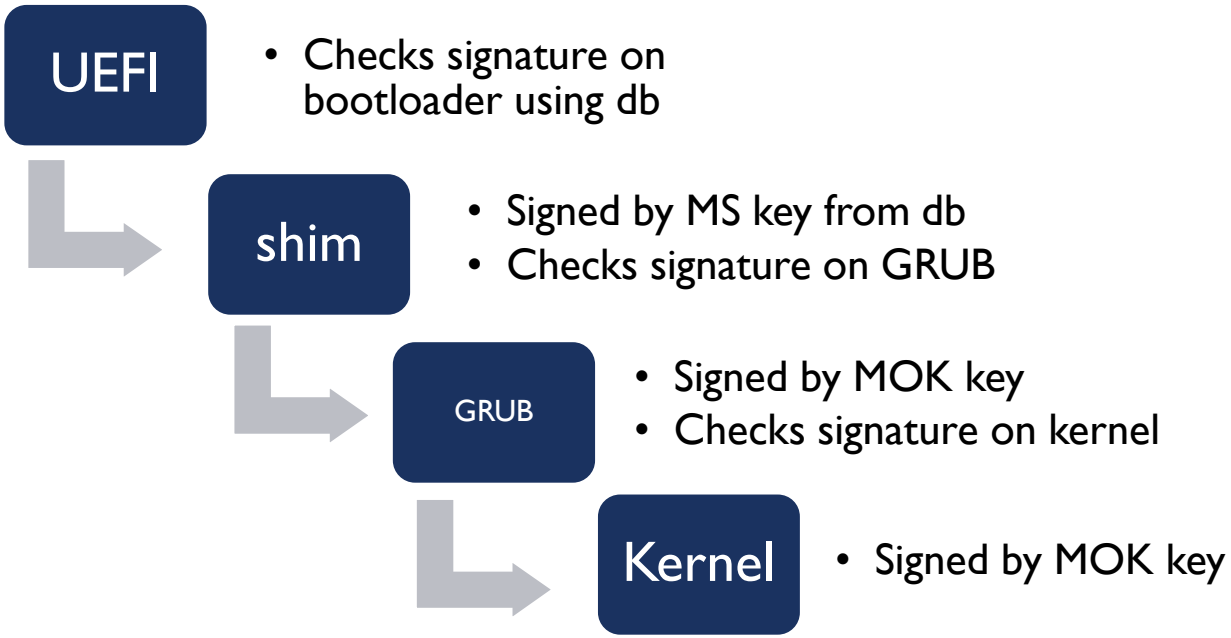
SECURE BOOT OPTIONS

- Use default keys from Microsoft
  - Pros
    - No need to change keys enrolled in UEFI
    - Compatible with dual-booting Windows
  - Cons
    - Susceptible to supply chain attacks
- Use your own generated keys
  - Pros
    - Not susceptible to supply chain attacks
    - Complete control over which bootloaders are authorized
  - Cons
    - Responsibility to properly manage keys

SECURE BOOT OPTIONS

- Secure Boot with MS keys
  - Uses shim bootloader EFI binary that is signed with Microsoft keys
  - shim introduces a new key store called Machine Owner Keys (MOK)
  - shim accepts bootloaders signed with MOKs
  - shim defaults to using GRUB
- Secure boot with custom keys
  - Generate new PK and KEK
  - Enroll PK and KEK in UEFI
  - Generate custom signing key and enroll into db
  - Directly sign GRUB (or other bootloader) with custom signing key

SECURE BOOT PROCESS WITH MS KEYS



GRUB AND KERNEL WITH SECURE BOOT

- GRUB restrictions
  - All GRUB modules must be contained in GRUB EFI image.
  - All files loaded from disk must be signed by GRUB GPG key.
  - Requires Secure Boot Advanced Targeting (SBAT).
- Kernel lockdown mode
  - All kernel modules must be signed.
  - Hibernation is disabled.

SECURE BOOT CONFIGURATION WITH MS KEYS

- Install shim-signed and other prerequisite packages
  - shim-signed
  - sbsigntool
  - build-essential
  - dkms
  - linux-headers for current kernel
  - efivar

```
┌──(thedave㉿zomgtest)-[~]
└─$ sudo apt install shim-signed sbsigntool build-essential dkms linux-headers-$(uname -r) efivar
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
shim-signed is already the newest version (1.40+15.7-1).
sbsigntool is already the newest version (0.9.4-3.1).
build-essential is already the newest version (12.10).
dkms is already the newest version (3.0.11-3).
linux-headers-6.5.0-kali2-amd64 is already the newest version (6.5.3-1kali2).
efivar is already the newest version (37-6).
```

SECURE BOOT CONFIGURATION WITH MS KEYS

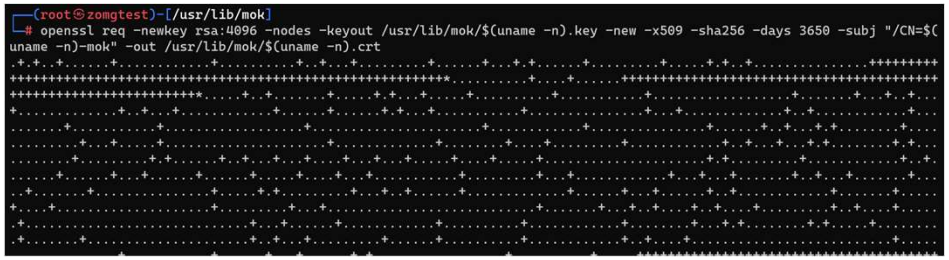
- Install shim to ESP
  - `cd /usr/lib/shim`
  - `cp shimx64.efi.signed /boot/efi/EFI/kali/BOOTx64.EFI`
  - `cp mmx64.efi.signed /boot/efi/EFI/kali/mmx64.efi`
  - `efibootmgr --unicode --disk /dev/nvme0n1 --part 1 --create --label kali-signed --loader /EFI/kali/BOOTx64.efi`

```
┌──(thedave㉿zomgtest)-[/usr/lib/shim]
└─$ sudo cp shimx64.efi.signed /boot/efi/EFI/kali/BOOTx64.EFI

┌──(thedave㉿zomgtest)-[/usr/lib/shim]
└─$ sudo cp mmx64.efi.signed /boot/efi/EFI/kali/mmx64.efi

┌──(thedave㉿zomgtest)-[/usr/lib/shim]
└─$ sudo efibootmgr --unicode --disk /dev/nvme0n1 --part 1 --create --label kali-signed --loader /EFI/kali/BOOTx64.efi
```

```
┌──(thedave㉿zomgtest)-[~]
└─$ ls -alh /usr/lib/shim
total 3.7M
drwxr-xr-x   2 root root 4.0K Oct 16 21:02 .
drwxr-xr-x 122 root root 4.0K Oct 16 21:10 ..
-rw-r--r--   1 root root  108 Jan 30  2023 BOOTX64.CSV
-rw-r--r--   1 root root  84K Jan 30  2023 fbx64.efi
-rw-r--r--   1 root root  86K Jan 30  2023 fbx64.efi.signed
-rw-r--r--   1 root root 829K Jan 30  2023 mmx64.efi
-rw-r--r--   1 root root 830K Jan 30  2023 mmx64.efi.signed
-rw-r--r--   1 root root 918K Jan 30  2023 shimx64.efi
-rw-r--r--   1 root root 927K Aug  4 07:21 shimx64.efi.signed
```

SECURE BOOT CONFIGURATION WITH MS KEYS

- Create and enroll MOK keys
  - `openssl req -newkey rsa:4096 -nodes -keyout /usr/lib/mok/$(uname -n).key -new -x509 -sha256 -days 3650 -subj "/CN=$(uname -n)-mok" -out /usr/lib/mok/$(uname -n).crt`
  - `openssl x509 -outform DER -in /usr/lib/mok/$(uname -n).crt -out /usr/lib/mok/$(uname -n).cer`
  - `mokutil --import /usr/lib/mok/$(uname -n).cer`



SECURE BOOT CONFIGURATION WITH MS KEYS

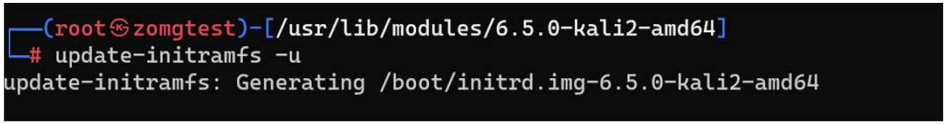
- Very enrolled MOK key
  - `mokutil --list-enrolled`

```
┌──(thedave㉿zomgtest)-[~]
└─$ sudo mokutil --list-enrolled
[sudo] password for thedave:
[key 1]
SHA1 Fingerprint: 53:61:0c:f8:1f:bd:7e:0c:eb:67:91:3c:9e:f3:e7:94:a9:63:3e:cb
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            ed:54:a1:d5:af:87:48:94:8d:9f:89:32:ee:9c:7c:34
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN=Debian Secure Boot CA
        Validity
            Not Before: Aug 16 18:09:18 2016 GMT
            Not After : Aug  9 18:09:18 2046 GMT
        Subject: CN=Debian Secure Boot CA
        Subject Public Key Info:
```

```
[key 2]
SHA1 Fingerprint: b4:2f:2f:44:43:aa:08:27:7f:36:9c:33:d5:0d:dc:ce:ed:cf:fc:70
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            5a:74:40:23:e9:b3:d7:0a:1b:3a:b2:e6:6d:d6:5a:7b:df:64:07:30
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN=zomgtest-mok
        Validity
            Not Before: Oct 17 01:02:49 2023 GMT
            Not After : Oct 14 01:02:49 2033 GMT
        Subject: CN=zomgtest-mok
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
                Modulus:
                    00:eb:92:f9:25:2a:2a:d6:23:7f:3d:0f:78:f2:96:
                    55:e1:2e:07:49:e9:04:17:08:f0:15:15:06:bf:f0:
                    3a:c5:13:65:49:da:a4:dc:4f:ed:5c:67:39:2e:99:
```

SECURE BOOT CONFIGURATION WITH MS KEYS

- Sign kernel with MOK key
  - `sbsign --key /usr/lib/mok/$(uname -n).key --cert /usr/lib/mok/$(uname -n).crt --output /boot/vmlinuz-6.3.0-kali1-amd64 /boot/vmlinuz-6.3.0-kali1-amd64`
- Sign kernel modules with MOK key
  - `cd /usr/lib/modules/$(uname -r)`
  - `sudo find . -name *.ko -exec /usr/lib/modules/$(uname -r)/source/scripts/sign-file sha256 /usr/lib/mok/$(uname -n).key /usr/lib/mok/$(uname -n).cer {} \;`
- Rebuild initramfs
  - `sudo update-initramfs -u`



SECURE BOOT CONFIGURATION WITH MS KEYS

- Create SBAT file
  - `echo "sbat,1,SBAT Version,sbat,1,https://github.com/rhboot/shim/blob/main/SBAT.md\ngrub,3,Free Software Foundation,grub,2.06,https://www.gnu.org/software/grub/\ngrub.ubuntu,1,Ubuntu,grub2,2.06-2ubuntu14.1,https://www.ubuntu.com/" > /usr/share/grub/sbat.csv`

```
┌──(root㉿zomgtest)-[/boot]
└─# echo "sbat,1,SBAT Version,sbat,1,https://github.com/rhboot/shim/blob/main/SBAT.md\ngrub,3,Free Software Foundation,grub,2.06,https://www.gnu.org/software/grub/\ngrub.ubuntu,1,Ubuntu,grub2,2.06-2ubuntu14.1,https://www.ubuntu.com/" > /usr/share/grub/sbat.csv
```

SECURE BOOT CONFIGURATION WITH MS KEYS

- Create GRUB GPG key
  - `sudo gpg --gen-key`
  - `sudo gpg --export FD1CF31A6A31C1429D1D2B00D931CCE4607EB176 | sudo tee /usr/lib/mok/$(uname -n).asc > /dev/null`

SECURE BOOT CONFIGURATION WITH MS KEYS

- Sign GRUB modules and other files
  - `cd /usr/lib/grub/x86_64-efi`
  - `sudo find . -name "*.mod" -exec gpg --detach-sign {} \;`
  - `sudo find . -name "*.lst" -exec gpg --detach-sign {} \;`
  - `sudo find . -name "*.img" -exec gpg --detach-sign {} \;`
  - `cd /boot/grub`
  - `sudo find . -type f -exec gpg --detach-sign {} \;`
  - `cd /boot`
  - `find . -name "vmlinuz*" -exec gpg --detach-sign {} \;`
  - `find . -name "initrd*" -exec gpg --detach-sign {} \;`

```
┌──(root㉿zomgtest)-[/usr/lib/grub/x86_64-efi]
└─# sudo find . -name "*.mod" -exec gpg --detach-sign {} \;
```

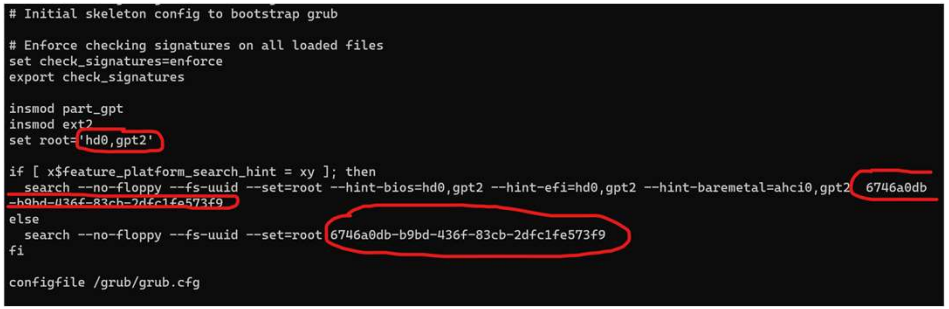
```
┌──(root㉿zomgtest)-[/usr/lib/grub/x86_64-efi]
└─# cd /boot/grub

┌──(root㉿zomgtest)-[/boot/grub]
└─# sudo find . -type f -exec gpg --detach-sign {} \;
```

```
┌──(root㉿zomgtest)-[/usr/lib/grub/x86_64-efi]
└─# cd /boot/grub

┌──(root㉿zomgtest)-[/boot/grub]
└─# sudo find . -type f -exec gpg --detach-sign {} \;
```

SECURE BOOT CONFIGURATION WITH MS KEYS

- Create skeleton GRUB config

```
# Initial skeleton config to bootstrap grub

# Enforce checking signatures on all loaded files
set check_signatures=enforce
export check_signatures

insmod part_gpt

insmod ext2

set root='hd0,gpt2’

if [ x$feature_platform_search_hint = xy ]; then

  search --no-floppy --fs-uuid --set=root --hint-bios=hd0,gpt2 --hint-efi=hd0,gpt2 --hint-baremetal=ahci0,gpt2  6d7c5735-63c3-4a53-888e-4461904e6239

else

  search --no-floppy --fs-uuid --set=root 6d7c5735-63c3-4a53-888e-4461904e6239fi

configfile /grub/grub.cfg
```



SECURE BOOT CONFIGURATION WITH MS KEYS

- Build GRUB standalone image
  - `GRUB_MODULES="acpi all_video boot btrfs cat chain configfile echo efifwsetup efinet ext2 fat font gettext gfxmenu gfxterm gfxterm_background gzio halt help hfsplus iso9660 jpeg keystatus loadenv loopback linux ls lsefi lsefimmap lsefisystab lssal memdisk minicmd normal ntfs part_apple part_msdos part_gpt password_pbkdf2 png probe reboot regexp search search_fs_uuid search_fs_file search_label sleep smbios squash4 test true video xfs zfs zfscrypt zfsinfo cpuid linuxefi play tpm cryptodisk gcry_arcfour gcry_blowfish gcry_camellia gcry_cast5 gcry_crc gcry_des gcry_dsa gcry_idea gcry_md4 gcry_md5 gcry_rfc2268 gcry_rijndael gcry_rmd160 gcry_rsa gcry_seed gcry_serpent gcry_sha1 gcry_sha256 gcry_sha512 gcry_tiger gcry_twofish gcry_whirlpool luks lvm efi_uga efi_gop crypto disk diskfilter pcidump setpci lspci"`
  - `GRUB_DIR="/usr/lib/grub/x86_64-efi"`
  - `GRUB_PUB_KEY="/usr/lib/mok/$(uname -n).asc"`
  - `GRUB_SBAT="/usr/share/grub/sbat.csv"`
  - `GRUB_OUTPUT="/root/grubx64.efi"`
  - `GRUB_INIT_CONFIG="/boot/grub/grub-initial.cfg"`
  - `grub-mkstandalone --directory "$GRUB_DIR" --format x86_64-efi --modules "$GRUB_MODULES" --pubkey "$GRUB_PUB_KEY" --sbat "$GRUB_SBAT" --output "$GRUB_OUTPUT" "boot/grub/grub.cfg=$GRUB_INIT_CONFIG"`

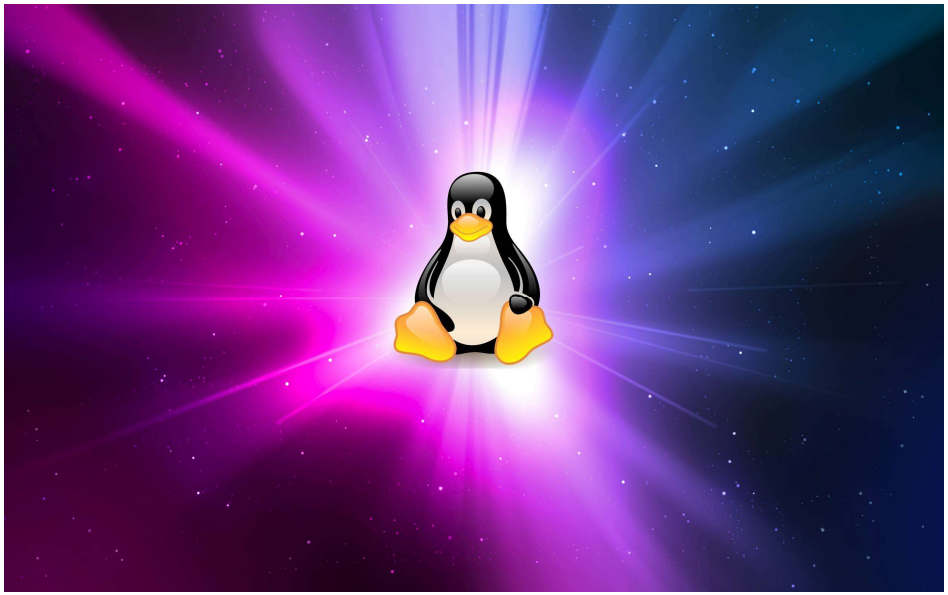
```
┌──(root㉿zomgtest)-[/boot]
└─# GRUB_MODULES="acpi all_video boot btrfs cat chain configfile echo efifwsetup efinet ext2 fat font gettext gfxmenu gf
xterm gfxterm_background gzio halt help hfsplus iso9660 jpeg keystatus loadenv loopback linux ls lsefi lsefimmap lsefisy
stab lssal memdisk minicmd normal ntfs part_apple part_msdos part_gpt password_pbkdf2 png probe reboot regexp search sea
rch_fs_uuid search_fs_file search_label sleep smbios squash4 test true video xfs zfs zfscrypt zfsinfo cpuid linuxefi pla
y tpm cryptodisk gcry_arcfour gcry_blowfish gcry_camellia gcry_cast5 gcry_crc gcry_des gcry_dsa gcry_idea gcry_md4 gcry_
md5 gcry_rfc2268 gcry_rijndael gcry_rmd160 gcry_rsa gcry_seed gcry_serpent gcry_sha1 gcry_sha256 gcry_sha512 gcry_tiger
gcry_twofish gcry_whirlpool luks lvm efi_uga efi_gop crypto disk diskfilter pcidump setpci lspci"

┌──(root㉿zomgtest)-[/boot]
└─# GRUB_DIR="/usr/lib/grub/x86_64-efi"

┌──(root㉿zomgtest)-[/boot]
└─# GRUB_PUB_KEY="/usr/lib/mok/$(uname -n).asc"

┌──(root㉿zomgtest)-[/boot]
└─# GRUB_SBAT="/usr/share/grub/sbat.csv"

┌──(root㉿zomgtest)-[/boot]
└─# GRUB_OUTPUT="/root/grubx64.efi"

┌──(root㉿zomgtest)-[/boot]
└─# GRUB_INIT_CONFIG="/boot/grub/grub-initial.cfg"

┌──(root㉿zomgtest)-[/boot]
└─# grub-mkstandalone --directory "$GRUB_DIR" --format x86_64-efi --modules "$GRUB_MODULES" --pubkey "$GRUB_PUB_KEY" --s
bat "$GRUB_SBAT" --output "$GRUB_OUTPUT" "boot/grub/grub.cfg=$GRUB_INIT_CONFIG"
```

SECURE BOOT CONFIGURATION WITH MS KEYS

- Copy GRUB standalone image to ESP
  - `cp /root/grubx64.efi /boot/efi/EFI/kali`
- Sign GRUB standalone image
  - `sbsign --key /usr/lib/mok/$(uname -n).key --cert /usr/lib/mok/$(uname -n).crt --output /boot/efi/EFI/kali/grubx64.efi /boot/efi/EFI/kali/grubx64.efi`

SECURE BOOT CONFIGURATION WITH MS KEYS

- **THAT'S IT!**
- **REBOOT!**



SECURE BOOT CONFIGURATION WITH CUSTOM KEYS

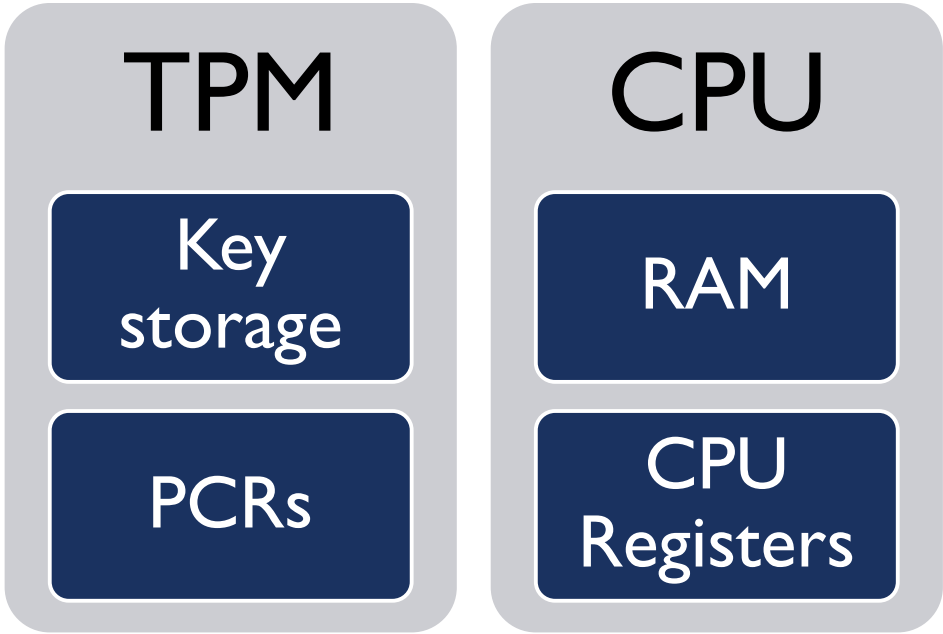
- Generate keys
  - PK
  - KEK
  - db signing
- Enroll keys in UEFI
  - Varies by UEFI manufacturer
- Other steps are the same as when using MS keys
  - No shim
  - Sign GRUB and kernel

SECURE BOOT CONFIGURATION WITH CUSTOM KEYS

- Other options when using custom keys
  - Ditch GRUB and use another bootloader
    - Systemd-boot
    - rEFInd
  - Create monolithic image with Linux kernel and initrd in single EFI binary

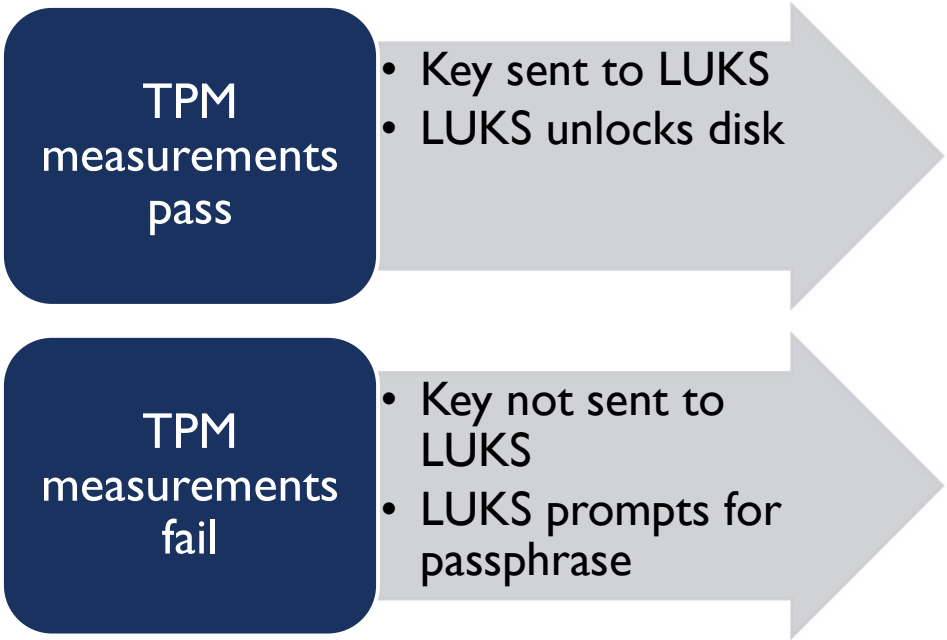
TPM AND LUKS

- TPM can store keys for decrypting LUKS partition
- TPM measures boot environment
  - Measurements stored in Platform Configuration Registers (PCR)
  - Key released only when PCRs match values used when key was sealed
  - PCR0: Core System Firmware executable code
  - PCR2: extended or pluggable executable code
  - PCR4: hash value of bootloader
  - PCR7: Secure Boot state



TPM AND LUKS

- Sealing LUKS key with TPM PCR state stores the key in TPM encrypted with the value of PCR registers.
- If PCR values change, the TPM will not decrypt the LUKS key
- If anything in the boot chain changes, you have to use your LUKS password and then reseal the LUKS key.



REFERENCES

- References
  - https://wiki.archlinux.org/title/Unified_Extensible_Firmware_Interface/Secure_Boot
  - https://wiki.debian.org/SecureBoot
  - https://ruderich.org/simon/notes/secure-boot-with-grub-and-signed-linux-and-initrd
  - https://blastrock.github.io/fde-tpm-sb.html
  - https://www.rodsbooks.com/efi-bootloaders/secureboot.html
  - https://pawitp.medium.com/full-disk-encryption-on-arch-linux-backed-by-tpm-2-0-c0892cab9704

QUESTIONS?

- Questions?