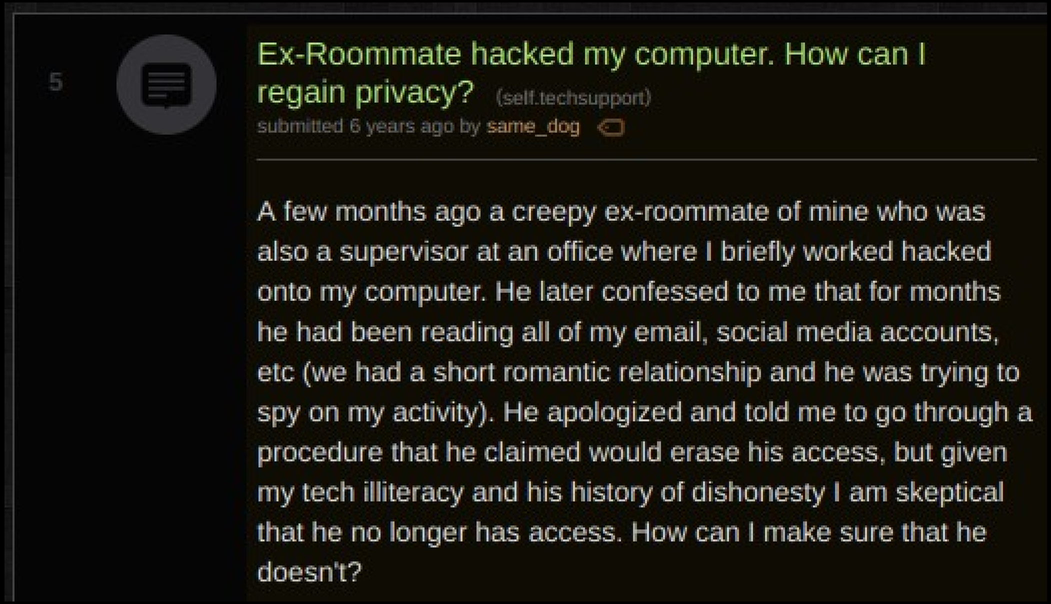
Secure Boot

How it works, how to do it wrong, and how to do it right

Daniel Parks – ISSS advanced talk 17 Sept. 2021

Secure boot might be for you



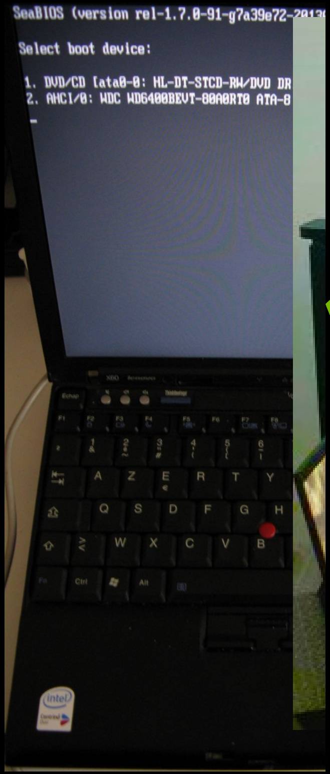
5

Ex-Roommate hacked my computer. How can I regain privacy? (self.techsupport)

submitted 6 years ago by same_dog

A few months ago a creepy ex-roommate of mine who was also a supervisor at an office where I briefly worked hacked onto my computer. He later confessed to me that for months he had been reading all of my email, social media accounts, etc (we had a short romantic relationship and he was trying to spy on my activity). He apologized and told me to go through a procedure that he claimed would erase his access, but given my tech illiteracy and his history of dishonesty I am skeptical that he no longer has access. How can I make sure that he doesn't?

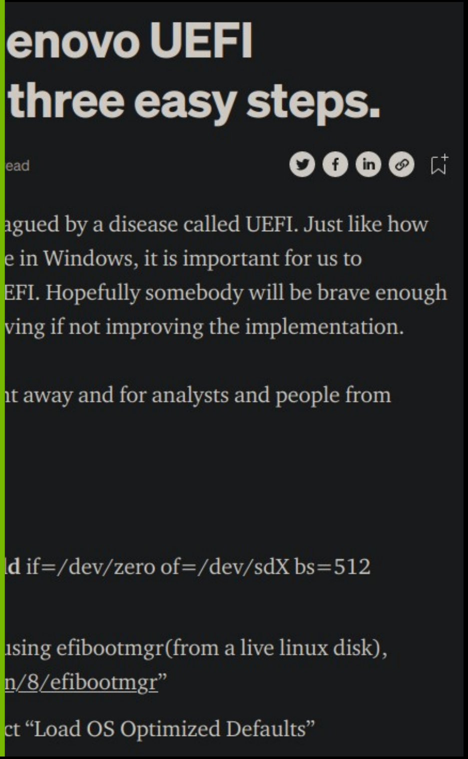
Secure boot might not be for you



```
SeaBIOS (version rel-1.7.0-91-g7a39e72-2013

Select boot device:

1. DVD/CD [ata0-0: HL-DT-STCD-RW/DVD DR
2. AHCI/0: WDC WD6400BEVT-80A0RT0 ATA-8
_
```



enovo UEFI
three easy steps.

ead

agued by a disease called UEFI. Just like how
e in Windows, it is important for us to
EFI. Hopefully somebody will be brave enough
ving if not improving the implementation.

nt away and for analysts and people from

ld if=/dev/zero of=/dev/sdX bs=512

using efibootmgr(from a live linux disk),
n/8/efibootmgr"

ct "Load OS Optimized Defaults"

How secure boot works

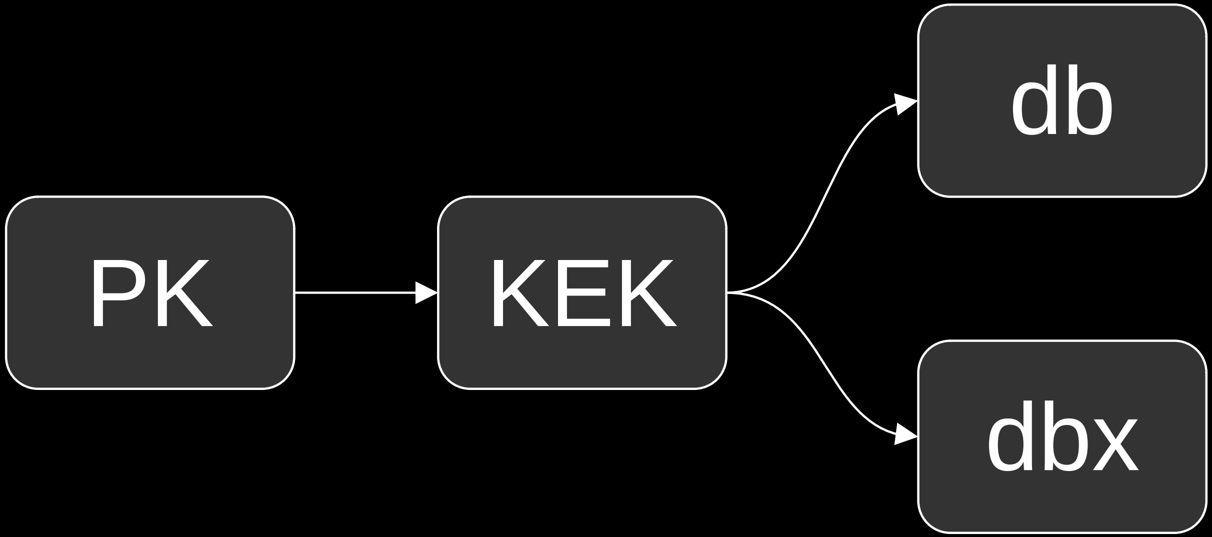
1. Platform verifies UEFI

- Not actually required by UEFI spec
- Implemented by most vendors
  - Intel CSME and AMD PSP
  - Some Dell computers: RoT is EC
  - Some HP computers: RoT is HP ESC

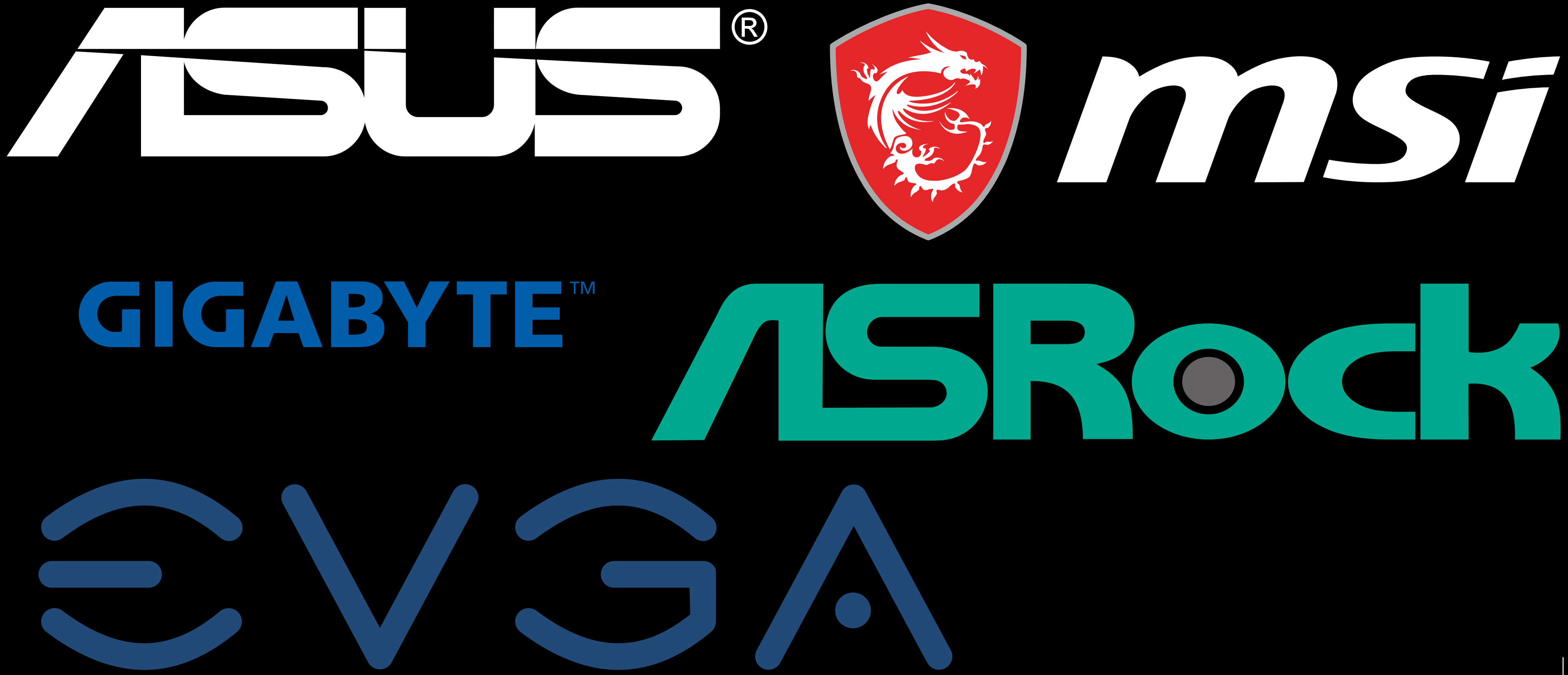
1.5 Aside: How to do it wrong

- CVE-2020-8705
  - Laptop manufactures decided to disable Boot Guard when resuming from S3
  - *Every* manufacturer tested by the researcher did this
  - Why???

2. UEFI chain of trust

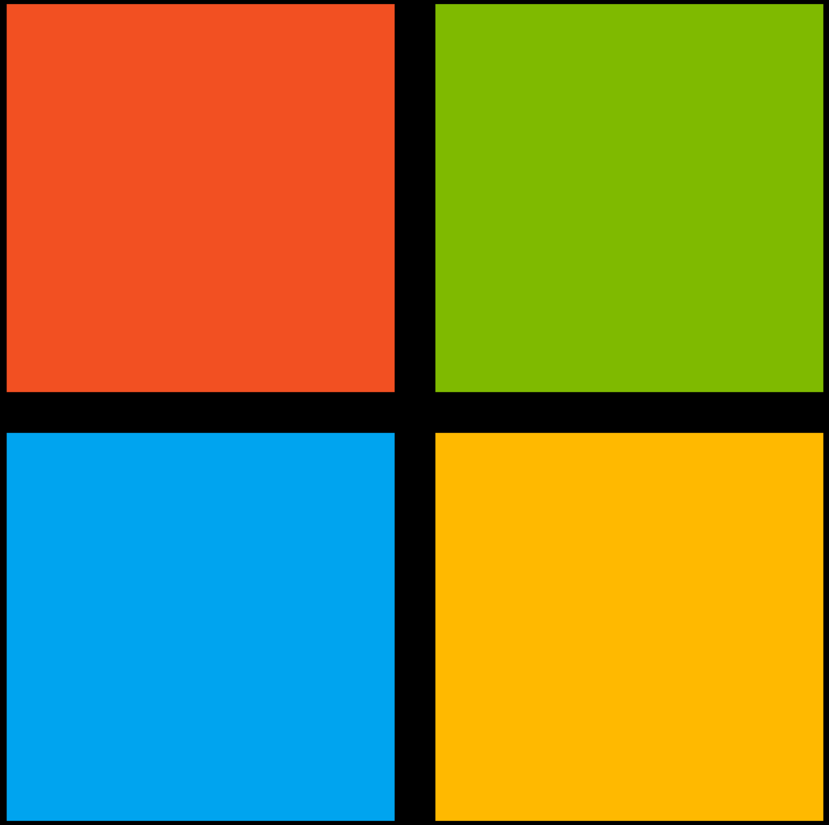


2. Who owns the PK?

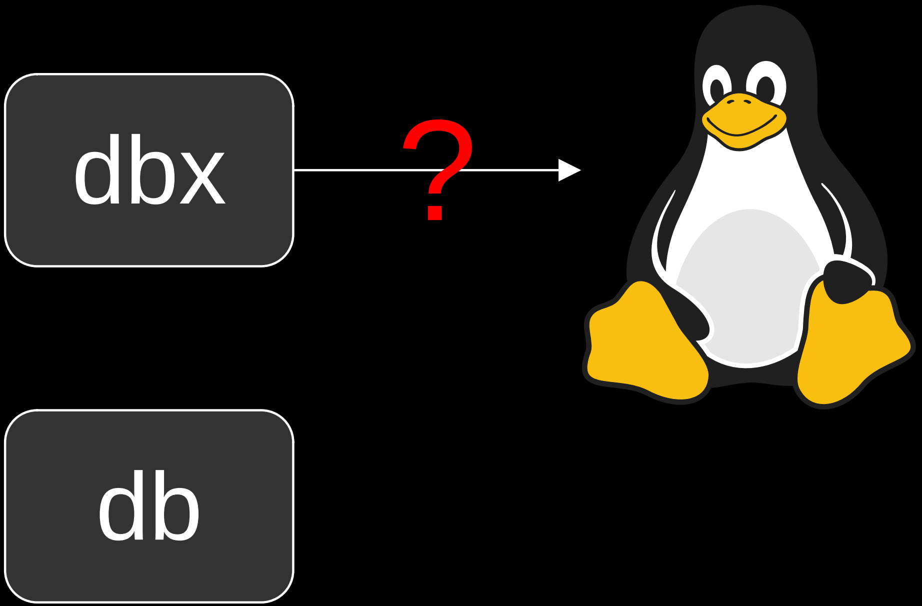


2. Who owns the KEK?

ASUS®
msi
GIGABYTE™
ASRock
EVGA



3. EFI binary verification process

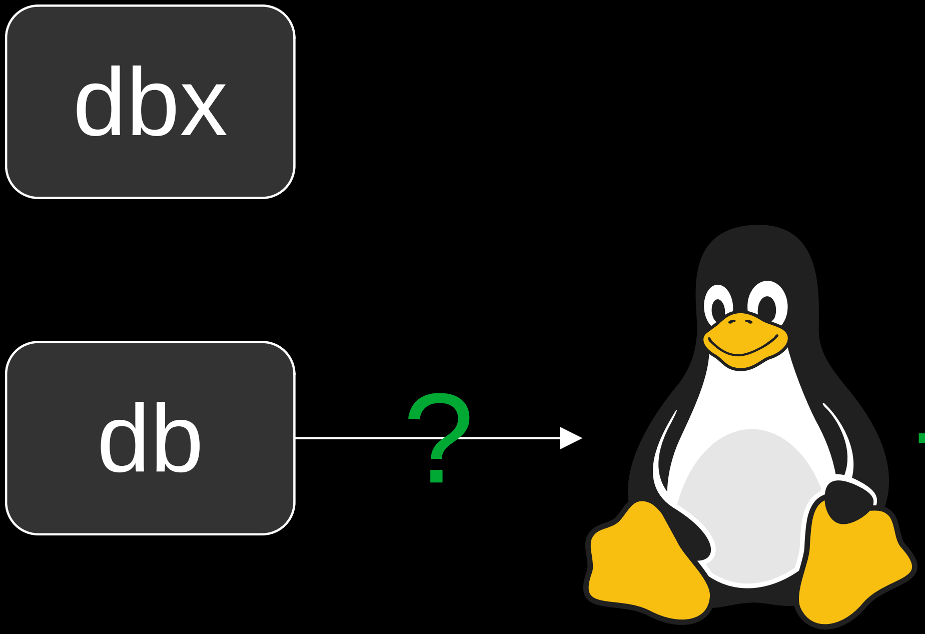


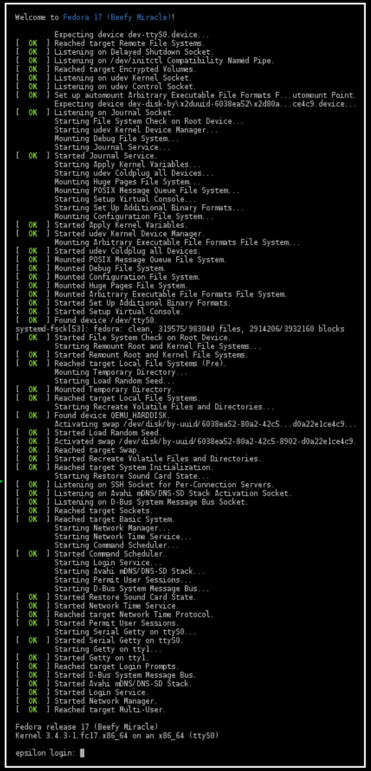
dbx

?

db

3. EFI binary verification process





```
Welcome to Fedora 17 (Beefy Miracle)!

         Expecting device dev-ttyS0.device...
[  OK  ] Reached target Remote File Systems.
[  OK  ] Listening on Delayed Shutdown Socket.
[  OK  ] Listening on /dev/initctl Compatibility Named Pipe.
[  OK  ] Reached target Encrypted Volumes.
[  OK  ] Listening on udev Kernel Socket.
[  OK  ] Listening on udev Control Socket.
[  OK  ] Set up automount Arbitrary Executable File Formats F...utomount Point.
         Expecting device dev-disk-by\x2duuid-6038ea52\x2d80a...ce4c9.device...
[  OK  ] Listening on Journal Socket.
         Starting File System Check on Root Device...
         Starting udev Kernel Device Manager...
         Mounting Debug File System...
         Starting Journal Service...
[  OK  ] Started Journal Service.
         Starting Apply Kernel Variables...
         Starting udev Coldplug all Devices...
         Mounting Huge Pages File System...
         Mounting POSIX Message Queue File System...
         Starting Setup Virtual Console...
         Starting Set Up Additional Binary Formats...
         Mounting Configuration File System...
[  OK  ] Started Apply Kernel Variables.
[  OK  ] Started udev Kernel Device Manager.
         Mounting Arbitrary Executable File Formats File System...
[  OK  ] Started udev Coldplug all Devices.
[  OK  ] Mounted POSIX Message Queue File System.
[  OK  ] Mounted Debug File System.
[  OK  ] Mounted Configuration File System.
[  OK  ] Mounted Huge Pages File System.
[  OK  ] Mounted Arbitrary Executable File Formats File System.
[  OK  ] Started Set Up Additional Binary Formats.
[  OK  ] Started Setup Virtual Console.
[  OK  ] Found device /dev/ttyS0.
systemd-fsck[53]: fedora: clean, 319575/983040 files, 2914206/3932160 blocks
[  OK  ] Started File System Check on Root Device.
         Starting Remount Root and Kernel File Systems...
[  OK  ] Started Remount Root and Kernel File Systems.
[  OK  ] Reached target Local File Systems (Pre).
         Mounting Temporary Directory...
         Starting Load Random Seed...
[  OK  ] Mounted Temporary Directory.
[  OK  ] Reached target Local File Systems.
         Starting Recreate Volatile Files and Directories...
[  OK  ] Found device QEMU_HARDDISK.
         Activating swap /dev/disk/by-uuid/6038ea52-80a2-42c5...d0a22e1ce4c9...
[  OK  ] Started Load Random Seed.
[  OK  ] Activated swap /dev/disk/by-uuid/6038ea52-80a2-42c5-8902-d0a22e1ce4c9.
[  OK  ] Reached target Swap.
[  OK  ] Started Recreate Volatile Files and Directories.
[  OK  ] Reached target System Initialization.
         Starting Restore Sound Card State...
[  OK  ] Listening on SSH Socket for Per-Connection Servers.
[  OK  ] Listening on Avahi mDNS/DNS-SD Stack Activation Socket.
[  OK  ] Listening on D-Bus System Message Bus Socket.
[  OK  ] Reached target Sockets.
[  OK  ] Reached target Basic System.
         Starting Network Manager...
         Starting Network Time Service...
         Starting Command Scheduler...
[  OK  ] Started Command Scheduler.
         Starting Login Service...
         Starting Avahi mDNS/DNS-SD Stack...
         Starting Permit User Sessions...
         Starting D-Bus System Message Bus...
[  OK  ] Started Restore Sound Card State.
[  OK  ] Started Network Time Service.
[  OK  ] Reached target Network Time Protocol.
[  OK  ] Started Permit User Sessions.
         Starting Serial Getty on ttyS0...
[  OK  ] Started Serial Getty on ttyS0.
         Starting Getty on tty1...
[  OK  ] Started Getty on tty1.
[  OK  ] Reached target Login Prompts.
[  OK  ] Started D-Bus System Message Bus.
[  OK  ] Started Avahi mDNS/DNS-SD Stack.
[  OK  ] Started Login Service.
[  OK  ] Started Network Manager.
[  OK  ] Reached target Multi-User.

Fedora release 17 (Beefy Miracle)
Kernel 3.4.3-1.fc17.x86_64 on an x86_64 (ttyS0)

epsilon login: █
```

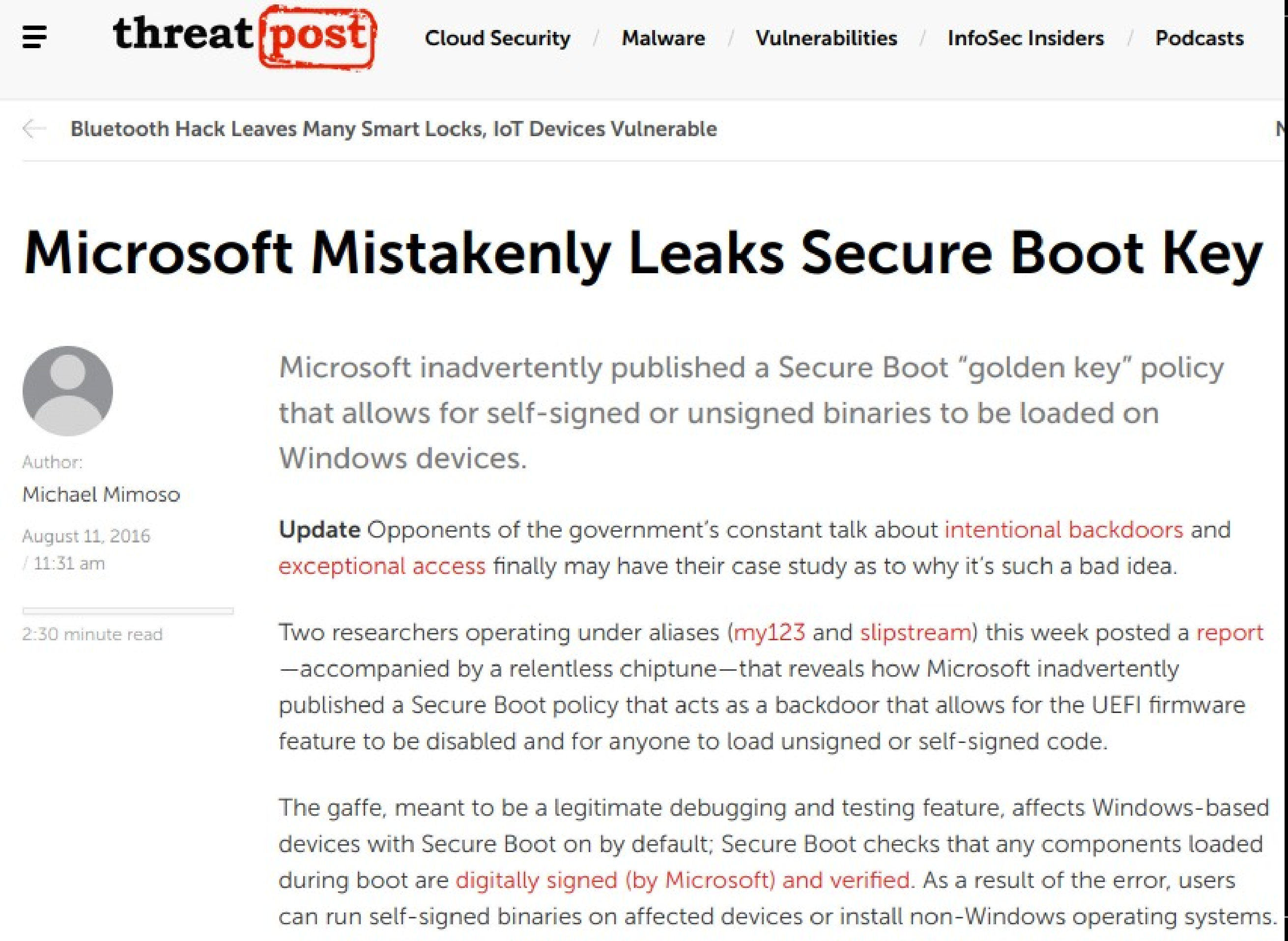
4. Microsoft stuff

- Laptops ship with Microsoft’s KEK
- db and dbx distributed through Windows Update
- Other OSes can get their bootloaders approved but it’s a pain
- If it gets approved, Microsoft gives back a copy signed with their db key

4.5 Aside: What happens if someone does it wrong

- CVE-2020-10713: buffer overflow in GRUB
- All vulnerable versions have to have their hashes added to the dbx

4.75 Aside: How to really do it wrong



Microsoft Mistakenly Leaks Secure Boot Key

Author:
Michael Mimoso

August 11, 2016
/ 11:31 am

2:30 minute read

Microsoft inadvertently published a Secure Boot "golden key" policy that allows for self-signed or unsigned binaries to be loaded on Windows devices.

Update Opponents of the government's constant talk about intentional backdoors and exceptional access finally may have their case study as to why it's such a bad idea.

Two researchers operating under aliases (my123 and slipstream) this week posted a report—accompanied by a relentless chiptune—that reveals how Microsoft inadvertently published a Secure Boot policy that acts as a backdoor that allows for the UEFI firmware feature to be disabled and for anyone to load unsigned or self-signed code.

The gaffe, meant to be a legitimate debugging and testing feature, affects Windows-based devices with Secure Boot on by default; Secure Boot checks that any components loaded during boot are digitally signed (by Microsoft) and verified. As a result of the error, users can run self-signed binaries on affected devices or install non-Windows operating systems.

5. UEFI Secure Boot Modes

- Setup – no PK
- User – can’t change PK

How secure boot works *on Linux*

Option 1: PreLoader

- Signed with Microsoft keys
- Verifies bootloader + kernel with hashes
- Requires physical access to enroll hashes

Option 2: shim

- Signed with Microsoft keys
- Made for GRUB
- Verifies bootloader + kernel with hashes / sig
- Requires physical access to enroll hashes / signatures
- Signing key is MOK

Option 3: Enroll your own keys

- This is the fun one

⚠

Why would you want to do that?

- Microsoft's keys don't do much to protect you from people with physical access
- Using your own keys could make it really annoying to do anything
- Especially if you set a BIOS password

1. Generate your own PK & KEK

```
$ uuidgen --random > GUID.txt
$ openssl req -newkey rsa:4096 -nodes -keyout PK.key -new -x509 -sha256 -days
3650 -subj "/CN=my Platform Key/" -out PK.crt
$ openssl x509 -outform DER -in PK.crt -out PK.cer
$ cert-to-efi-sig-list -g "$(< GUID.txt)" PK.crt PK.esl
$ sign-efi-sig-list -g "$(< GUID.txt)" -k PK.key -c PK.crt PK PK.esl PK.auth
$ openssl req -newkey rsa:4096 -nodes -keyout KEK.key -new -x509 -sha256 -days
3650 -subj "/CN=my Key Exchange Key/" -out KEK.crt
$ openssl x509 -outform DER -in KEK.crt -out KEK.cer
$ cert-to-efi-sig-list -g "$(< GUID.txt)" KEK.crt KEK.esl
$ sign-efi-sig-list -g "$(< GUID.txt)" -k PK.key -c PK.crt KEK KEK.esl KEK.auth
```

2. Generate your own db

```
$ openssl req -newkey rsa:4096 -nodes -keyout db.key -new -x509 -sha256 -days 3650 -subj "/CN=my Signature Database key/" -out db.crt

$ openssl x509 -outform DER -in db.crt -out db.cer

$ cert-to-efi-sig-list -g "$(< GUID.txt)" db.crt db.esl

$ sign-efi-sig-list -g "$(< GUID.txt)" -k KEK.key -c KEK.crt db db.esl db.auth
```

2.5 Aside: dbx

- dbx is optional!
- You own the keys – don’t sign anything bad
- If you do: create dbx or rotate keys

3. Sign a binary (GRUB)

```
# sbsign --key db.key --cert db.crt
--output /boot/vmlinuz-linux
/boot/vmlinuz-linux
```

```
# sbsign --key db.key --cert db.crt
--output esp/EFI/BOOT/BOOTx64.EFI
esp/EFI/BOOT/BOOTx64.EFI
```

4. Put firmware in setup mode

- Boot into BIOS interface
- Vendors call it whatever they want
- Delete the platform key??
- Switch to setup mode??
- Reset secure boot keys??

5. Enroll your keys

```
# mkdir -p /etc/secureboot/keys/{db,dbx,KEK,PK}
# cp db.auth /etc/secureboot/keys/db/
# cp KEK.auth /etc/secureboot/keys/KEK/
# cp PK.auth /etc/secureboot/keys/PK/
# sbkeysync --verbose
# sbkeysync --verbose --pk
```

Platform automatically switches to User Mode

This is a pain

- sbctl by Morten Linderud aka Foxboron

```
$ sbctl create-keys
⟹ Creating secure boot keys …
  → Using UUID d6e9af79-c6b5-4b43-b893-dbb7e6570142 …
⟹ Signing [ … ]/keys/PK/PK.der.esl with [ … ]/keys/PK/PK.key …
⟹ Signing [ … ]/keys/KEK/KEK.der.esl with [ … ]/keys/PK/PK.key …
⟹ Signing [ … ]/keys/db/db.der.esl with [ … ]/keys/KEK/KEK.key …

$ sbctl enroll-keys
⟹ Syncing /usr/share/secureboot/keys to EFI variables …
⟹ Synced keys!

$ sbctl sign /boot/EFI/BOOT/BOOTX64.EFI
  → Signing /boot/EFI/BOOT/BOOTX64.EFI …
```

Issues

- Unencrypted hard drive + sbctl keys = bad
- Grub = bad (sorta)
- Bootloaders are slow

General recommendations

- Validate *all* boot config
- Don’t use Microsoft keys
- Set a bios password
- Use EFISTUB when available
- Use disk encryption in combination with secure boot
  - Preferred: require password at boot
  - Use sleep mode so that you don’t have to decrypt the hard drive every time

Aside: sleep

- Make sure you’re using deep sleep

-
  ```
  $ cat /sys/power/mem_sleep
  s2idle [deep]
  ```
- `[s2idle] deep → mem_sleep_default=deep`
- Might have to enable a bios setting
- Might have to hack ACPI tables

Aside: sleep, part 2

- Sleep can be approximated with hibernation + FDE + TPM
- Not really recommended

Aside: Android & CrOS

- Chrome OS uses depthcharge
- Android is just sorta custom
- Some phones let you enroll custom verified boot keys

Demo: physical access

CC Attributions

- Coreboot - https://commons.wikimedia.org/wiki/File:Coreboot%2BseaBIOS%2Bon-x60.JPG
- Gaming PC - https://www.deviantart.com/marcusburns1977/art/My-PC-Built-758744133
- Tux - https://commons.wikimedia.org/w/index.php?curid=48629023

Further Reading

Download slides to click links.

- Secure Boot Chain in UEFI
- Establishing the Root of Trust
- Unified Extensible Firmware Interface/Secure Boot
- Managing EFI Boot Loaders for Linux: Controlling Secure Boot
- HP Sure Start Whitepaper