Aliens Among Us

Mike Lynn

Client Platform Engineer

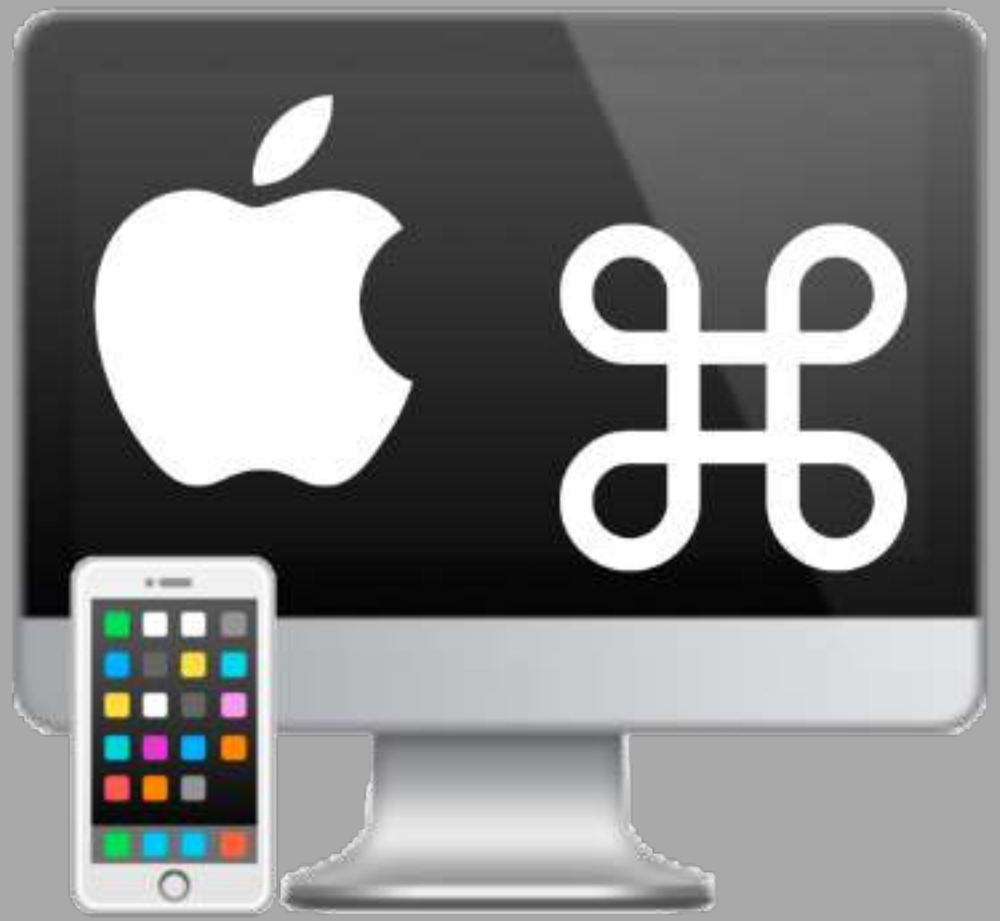
facebook

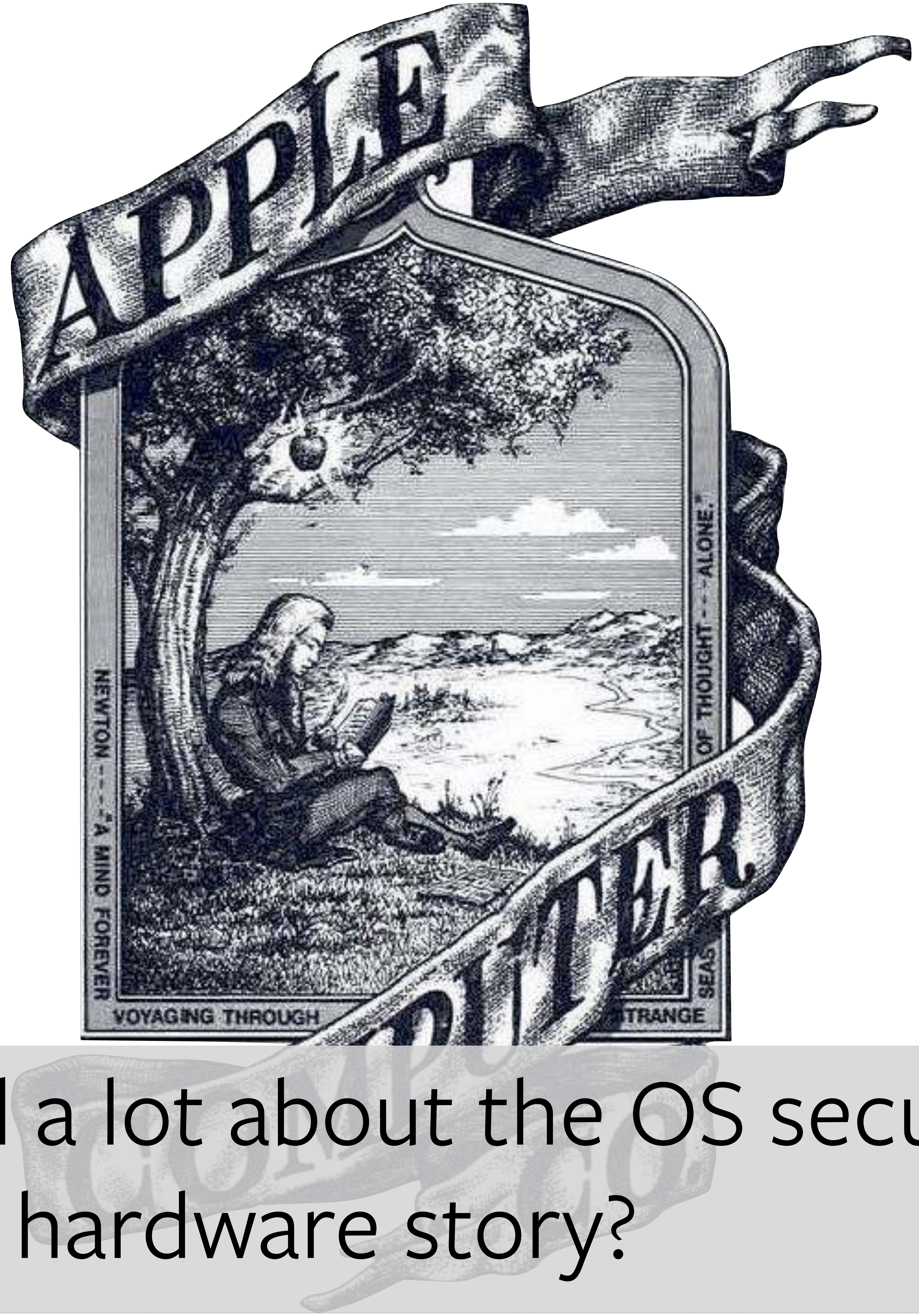
twitter.com / mikeymikey

frogor @
macadmins &
osquery Slack

github.com / pudquick

macadmins.org





So far we've talked a lot about the OS security of Mac.
But what's Apple's hardware story?

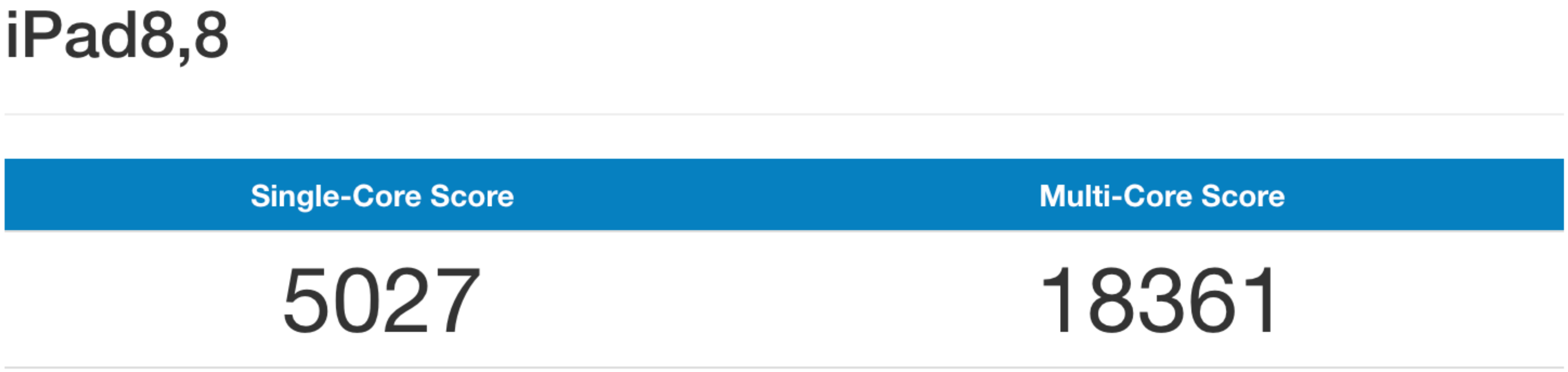
Apple is no stranger to deep hardware change when needed. The first Macs ran on Motorola 68k processors.

To keep up with their speed needs, they later switched to PowerPC processors with emulation providing 68k support.

Then they spent a little time making fun of the competition …

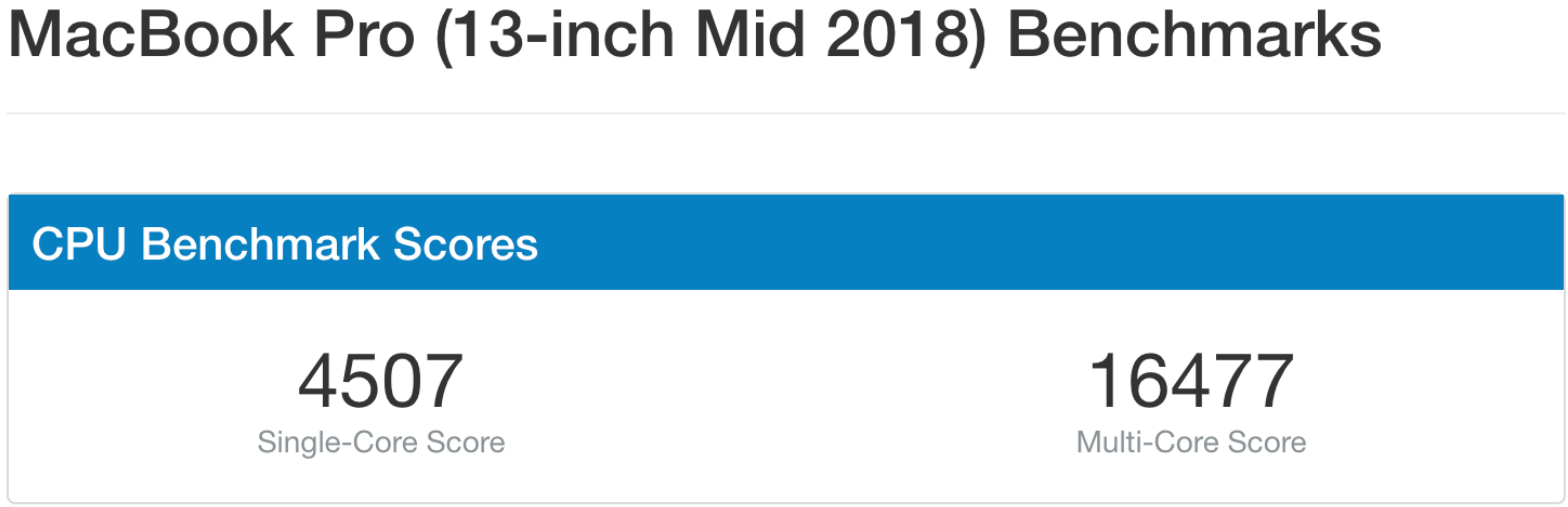
… before switching to praising them, adopting Intel as the new CPU and again relying on emulation during transition.

So what has the mothership been doing lately?



iPad8,8

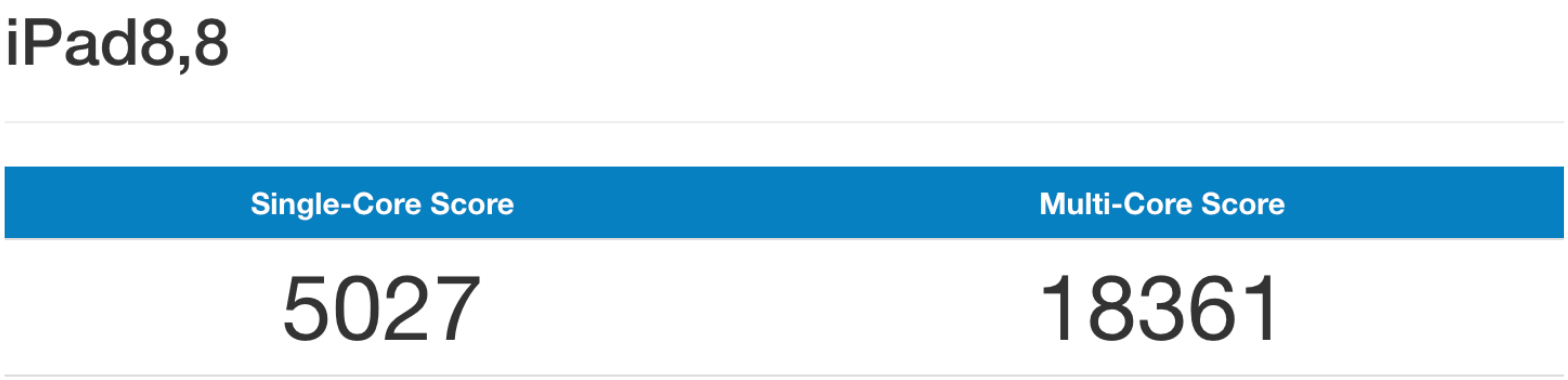
Single-Core Score	Multi-Core Score
5027	18361



MacBook Pro (13-inch Mid 2018) Benchmarks

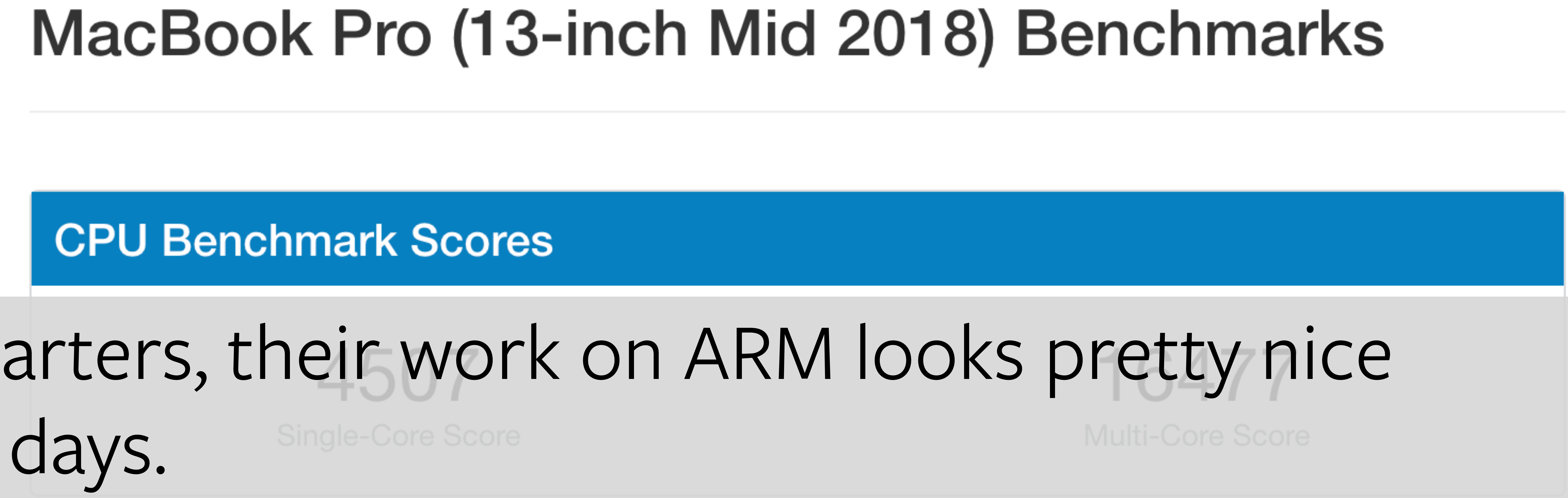
CPU Benchmark Scores

4507	16477
Single-Core Score	Multi-Core Score



iPad8,8

Single-Core Score	Multi-Core Score
5027	18361



MacBook Pro (13-inch Mid 2018) Benchmarks

CPU Benchmark Scores

4507	16477
Single-Core Score	Multi-Core Score

For starters, their work on ARM looks pretty nice these days.

... but since late 2017

… Apple has been introducing alien devices
to the world - Secure Boot Macs with the T2 chip.

iMac Pro

iMac Pro

MacBook Pro

iMac Pro

MacBook Pro

MacBook Air

iMac Pro

MacBook Pro

MacBook Air

Mac Mini

First the iMac Pro and MacBook Pro - and now the MacBook Air and Mac Mini.

iMac Pro

MacBook Pro

MacBook Air

Mac Mini

In fact, more than half their Mac models are now completely unlike anything they’ve shipped before.

Mac Pro

The only ones remaining are - the Mac Pro

Mac Pro (duh)

The only ones remaining are - the Mac Pro (yeah, we know this one is going to change)

Mac Pro (duh)

iMac

The iMac

Mac Pro (duh)

iMac (overdue)

The iMac (due for a refresh)

Mac Pro (duh)

iMac (overdue)

MacBook

The 12″ MacBook

Mac Pro (duh)

iMac (overdue)

MacBook (☠️)

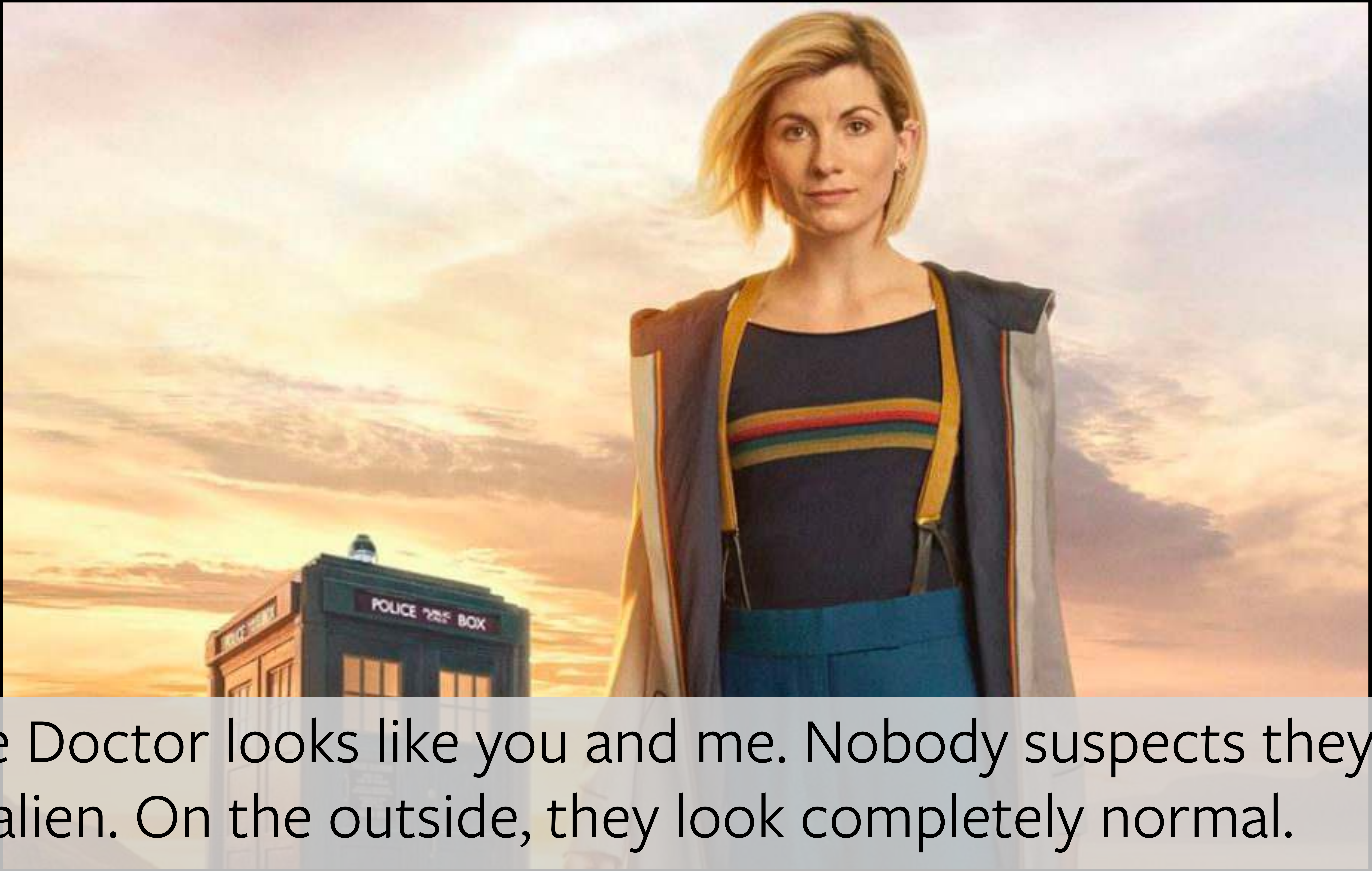
The 12″ MacBook (no idea what Apple will do there)

So how are these devices alien?

SECURE BOOT

No, not like this.

They're alien like this - like The Doctor.



The Doctor looks like you and me. Nobody suspects they're an alien. On the outside, they look completely normal.

”It Just Works”

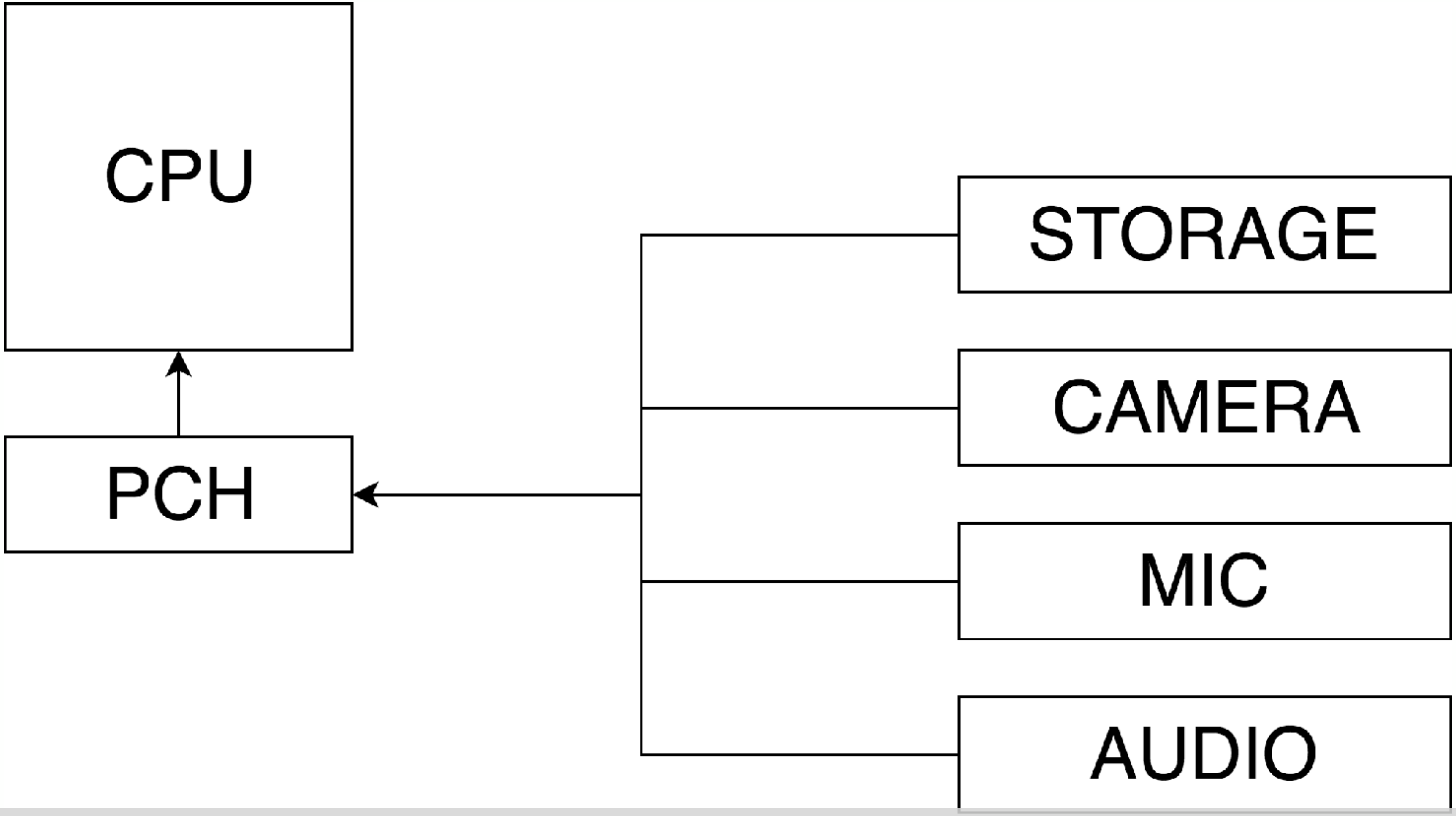
They’re just normal Macs.
They’re just a bit more secure … Not different, surely?

But whereas normal Macs have a single heart under the hood

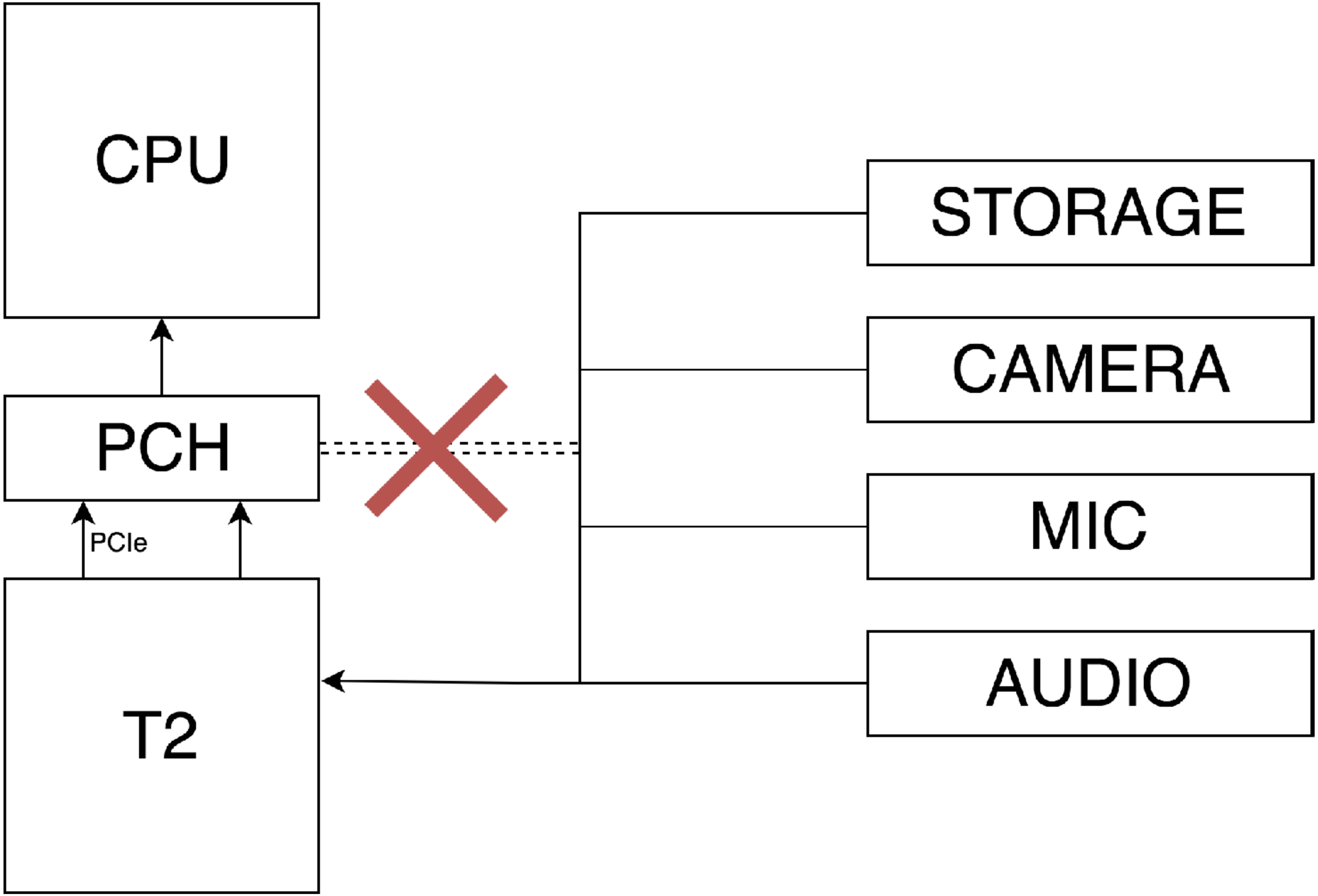
like The Doctor, T2 Secure Boot Macs have two. And the deeper you look, the more differences you see.

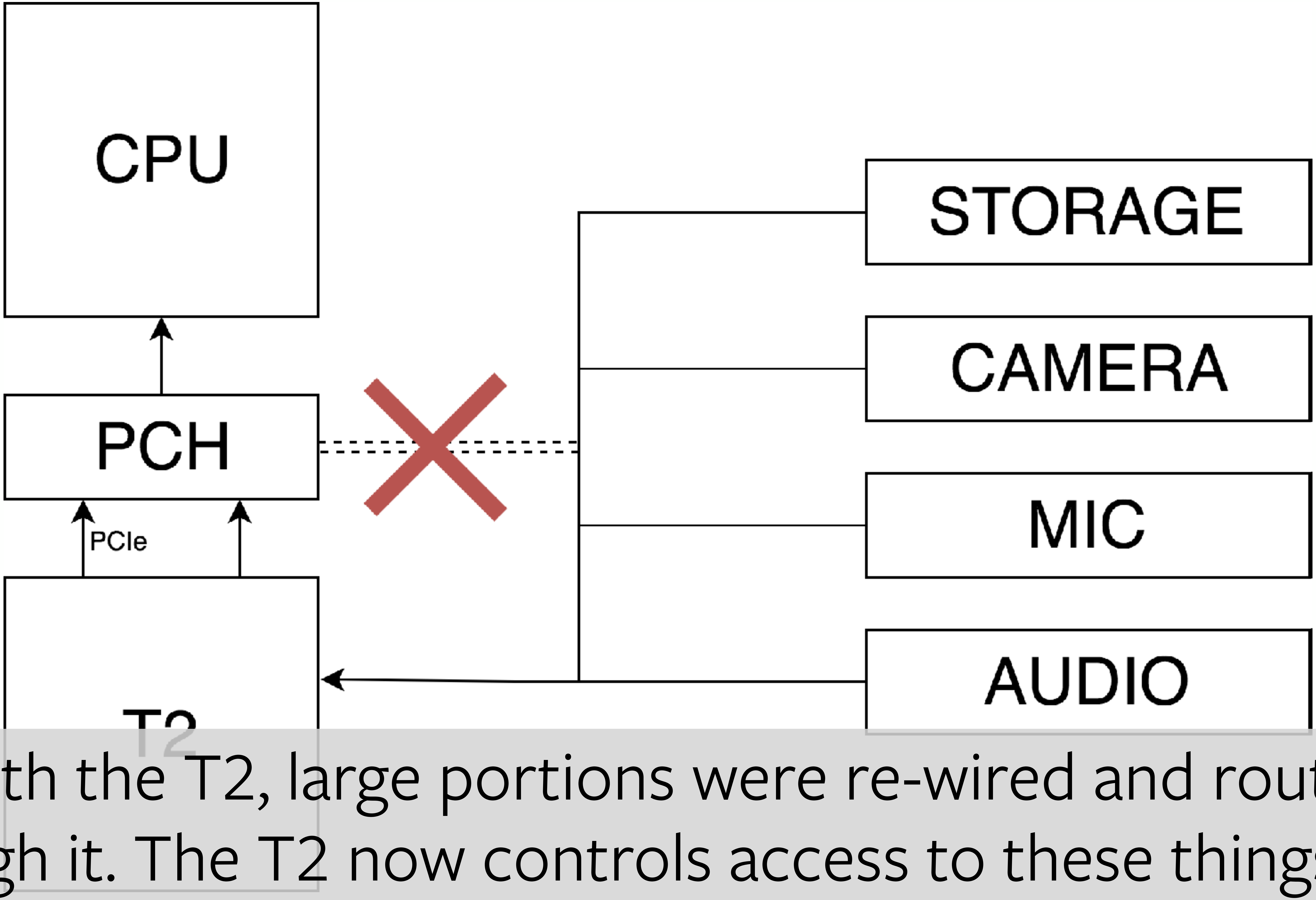
In fact, with how they're wired up - it's less like The Doctor

And more like this little guy. A new brain behind the old.

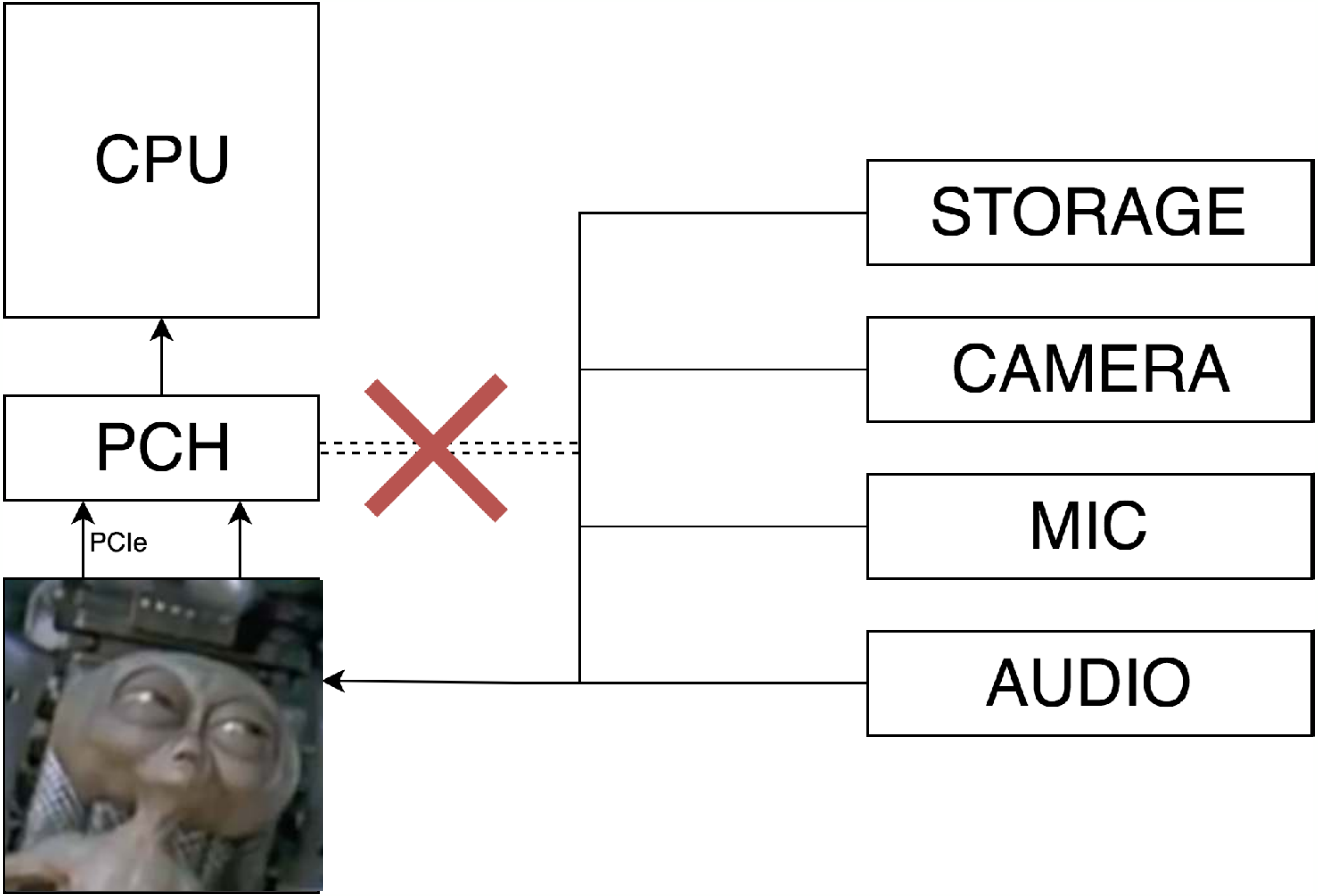


Previous generation Macs were wired more like this, with storage and externals mostly routed directly to the CPU.


CPU
PCH
PCIe
T2
STORAGE
CAMERA
MIC
AUDIO



But with the T2, large portions were re-wired and routed through it. The T2 now controls access to these things.



CPU

PCH

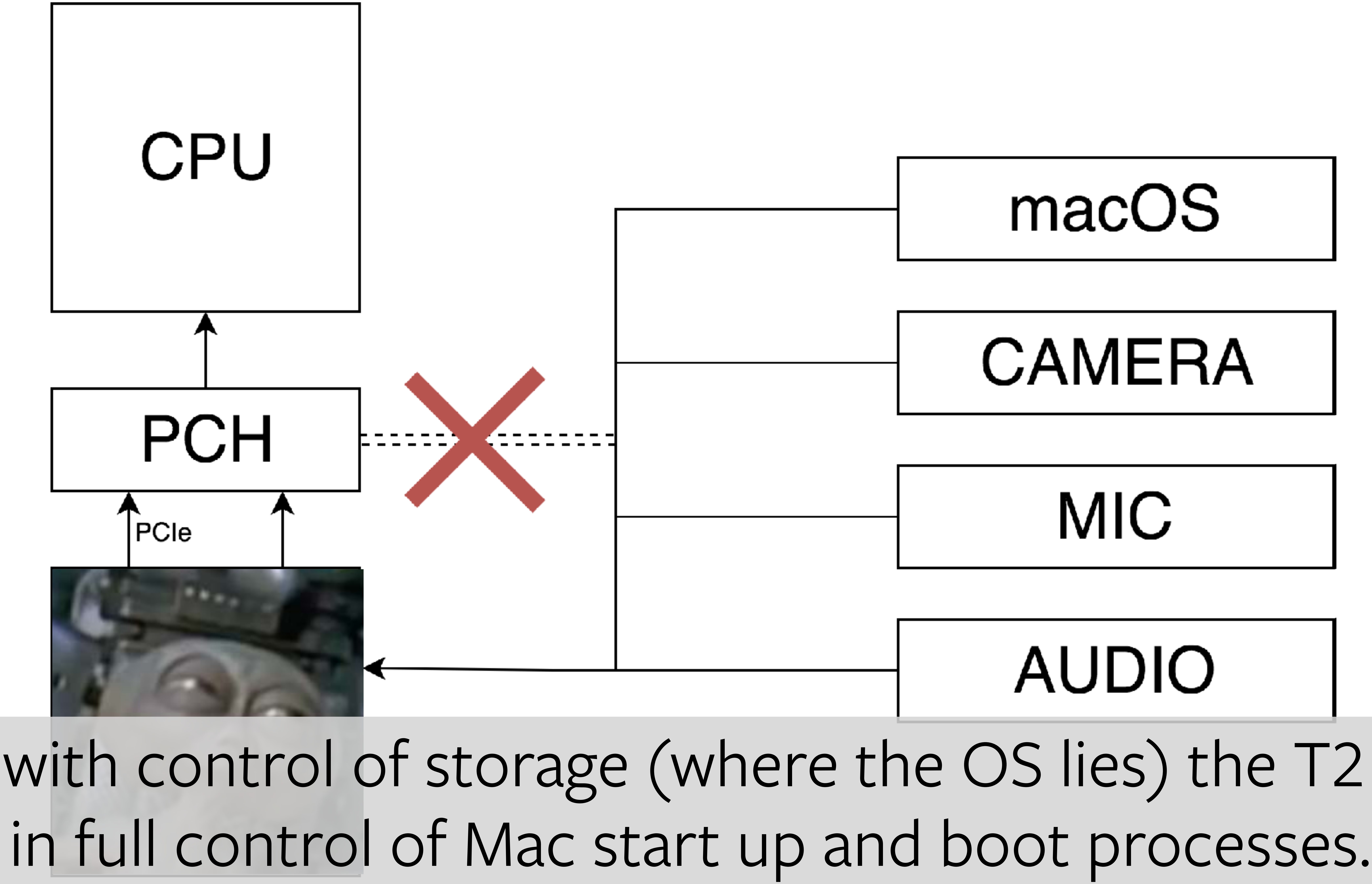
PCIe

STORAGE

CAMERA

MIC

AUDIO

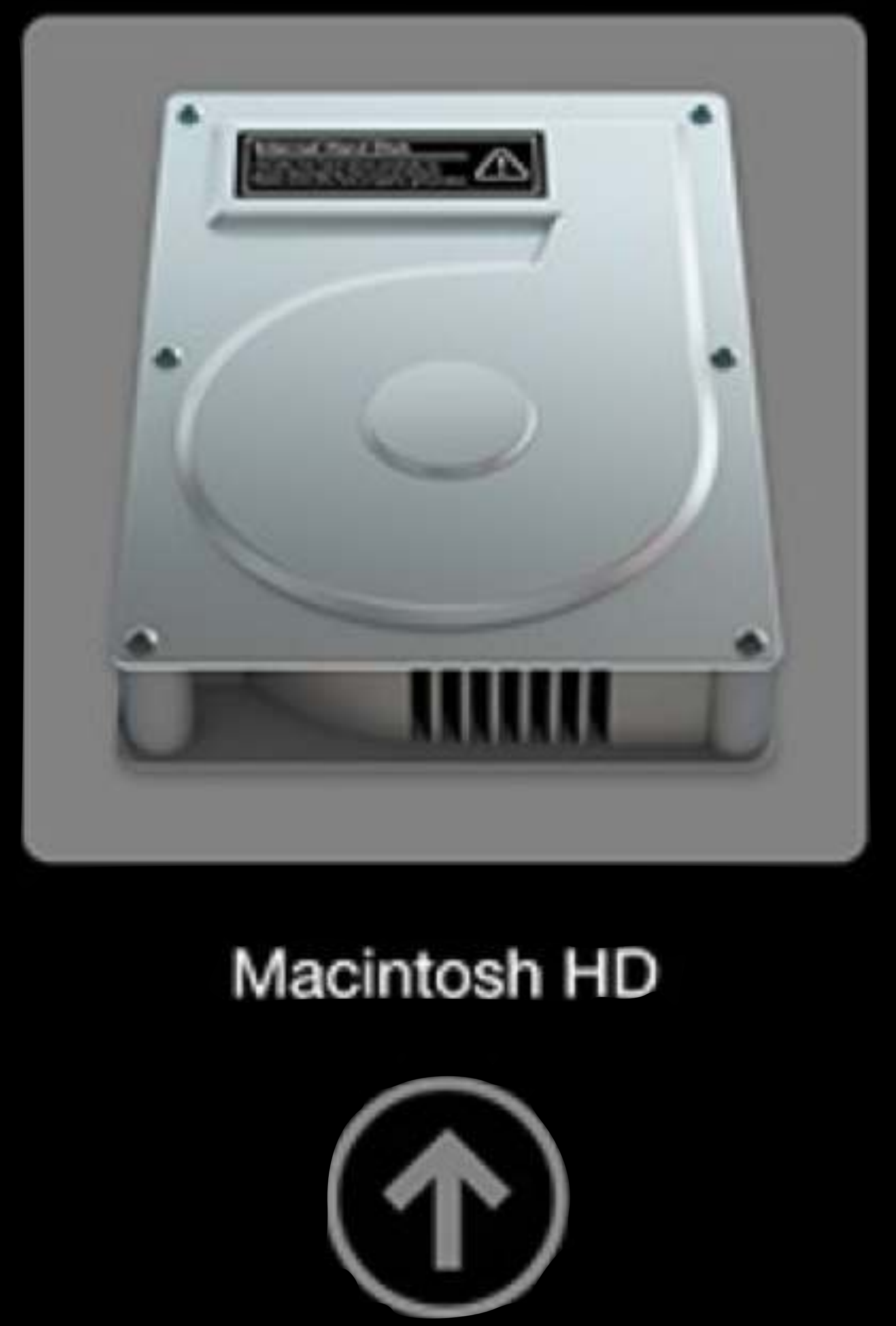


And with control of storage (where the OS lies) the T2 is now in full control of Mac start up and boot processes.

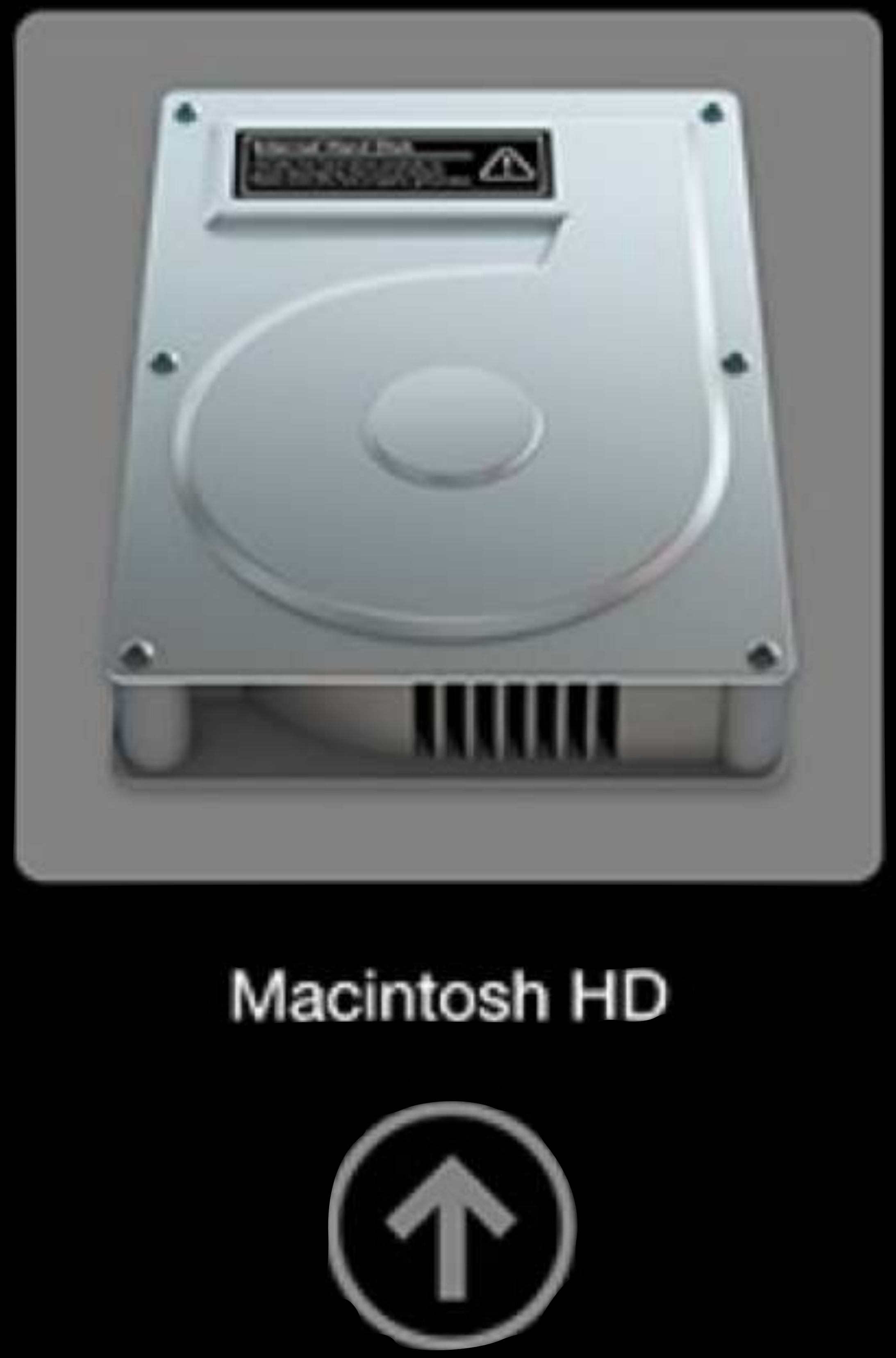
But when the iMac Pro came out in December 2017, surely no one noticed anything different, right?

Apple let a ton of Final Cut Pro X people get sneak previews of the device. They didn′t mention noticing anything.

But the Mac admins of the world did.



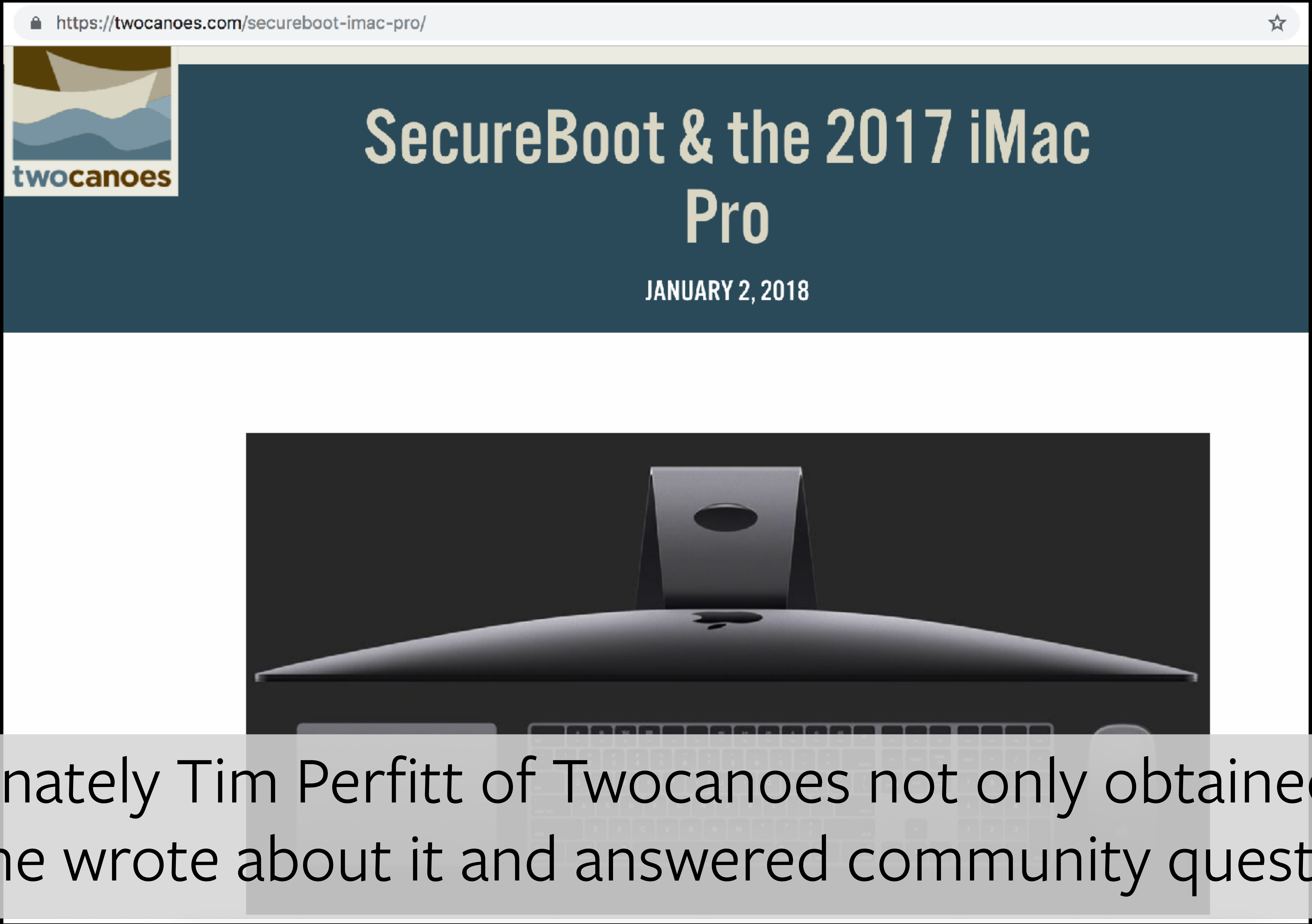
The very first thing noticed was no more NetBoot.
Couldn't even boot off an external drive.



We understood they were "Secure Boot" devices, but this was macOS we were trying to boot. What was going on?

(Un)fortunately, Apple chose their first T2 Mac model carefully. It was chosen to be a small initial rollout.

At $5k+ a piece, the demand would be reduced. But it also made it hard for many admins to get a hold of to study.



https://twocanoes.com/secureboot-imac-pro/

SecureBoot & the 2017 iMac Pro

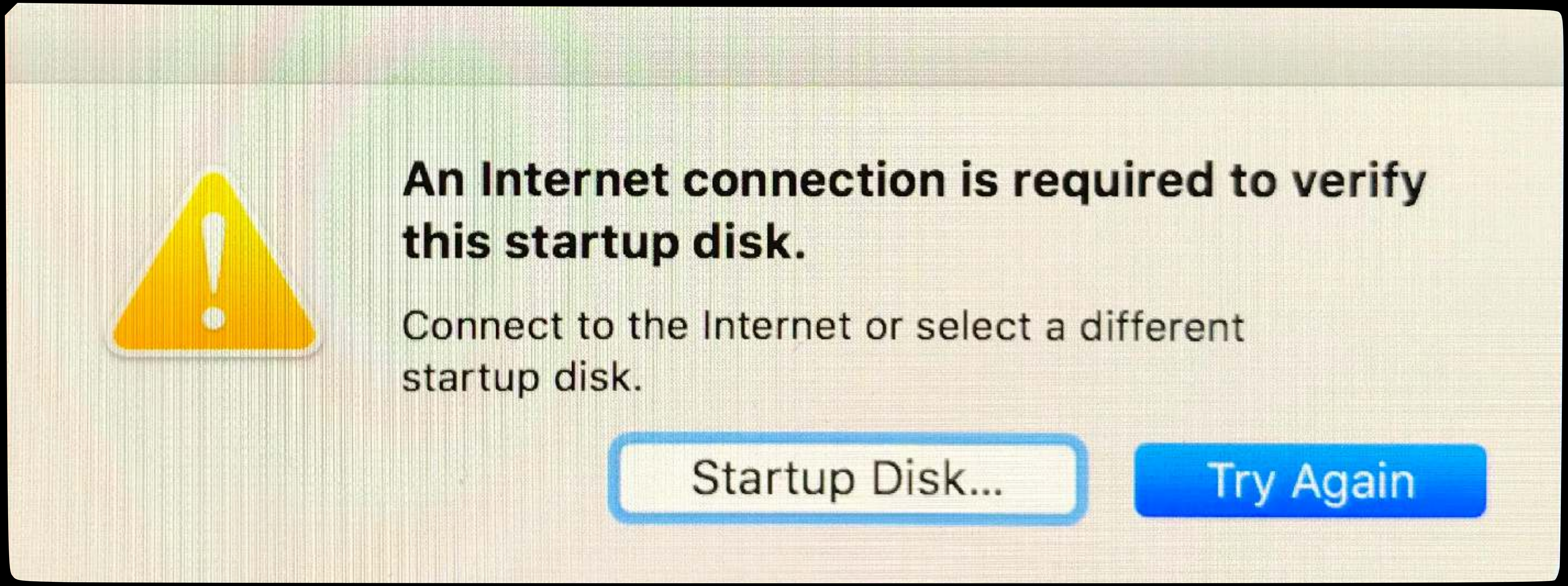
JANUARY 2, 2018

Fortunately Tim Perfitt of Twocanoes not only obtained one, he wrote about it and answered community questions.

SecureBoot & the 2017 iMac Pro

JANUARY 2, 2018

He documented the new security controls and performed network traces and provided many their first look at the T2.



An Internet connection is required to verify this startup disk.

Connect to the Internet or select a different startup disk.

But still other dialogs appeared and were unknown why or how they got triggered. More research was required.

We found an iMac Pro could asr restore an OS image to another iMac Pro. Something else was at play.

When 10.13.4 was released, suddenly it was possible
to do it from other Macs. Did the OS play a component?

Booting Secure

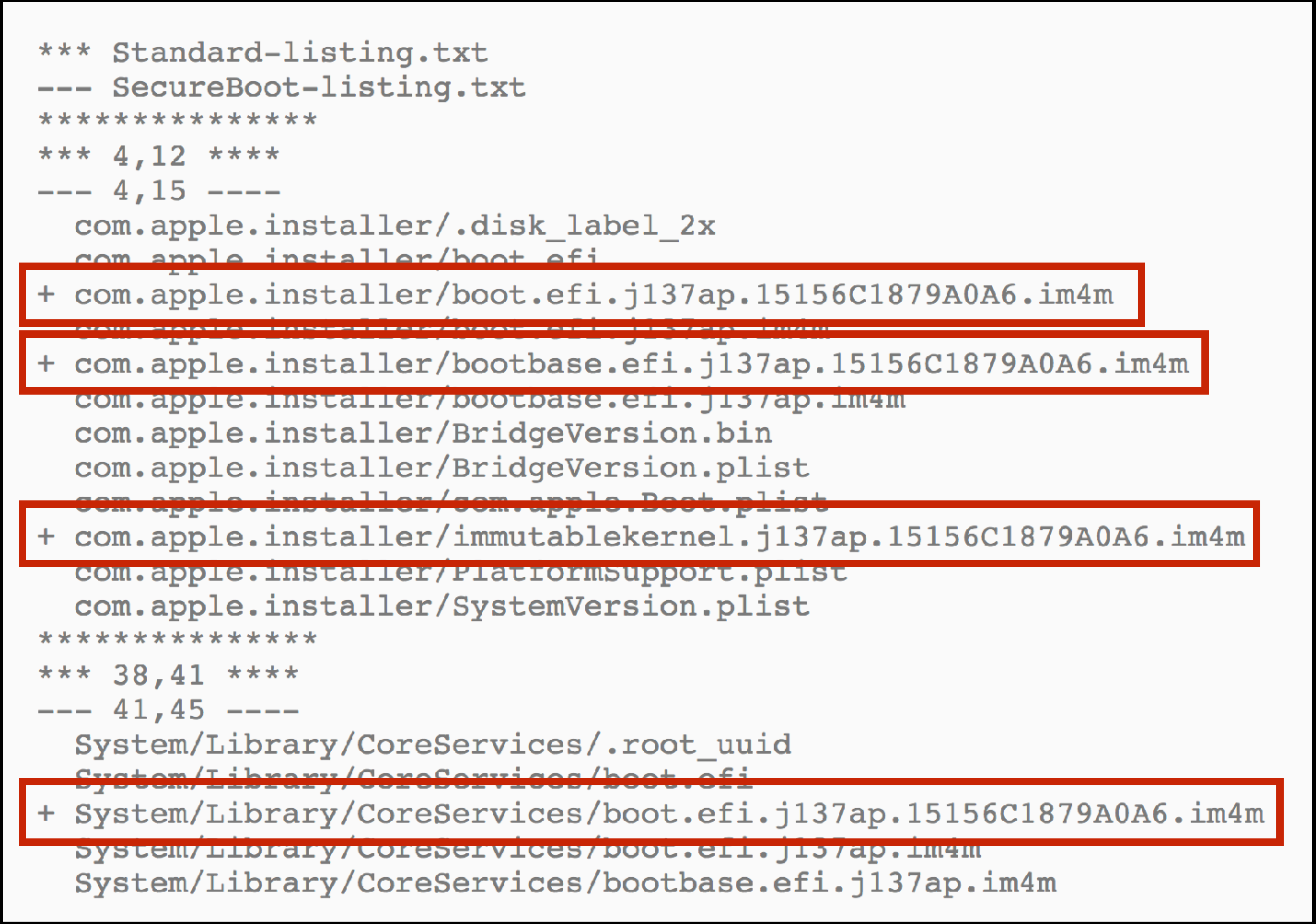
michaellynn.github.io/2018/07/27/booting-secure

At this point there were too many questions and not enough answers. And the MacBook Pro 2018 was arriving.

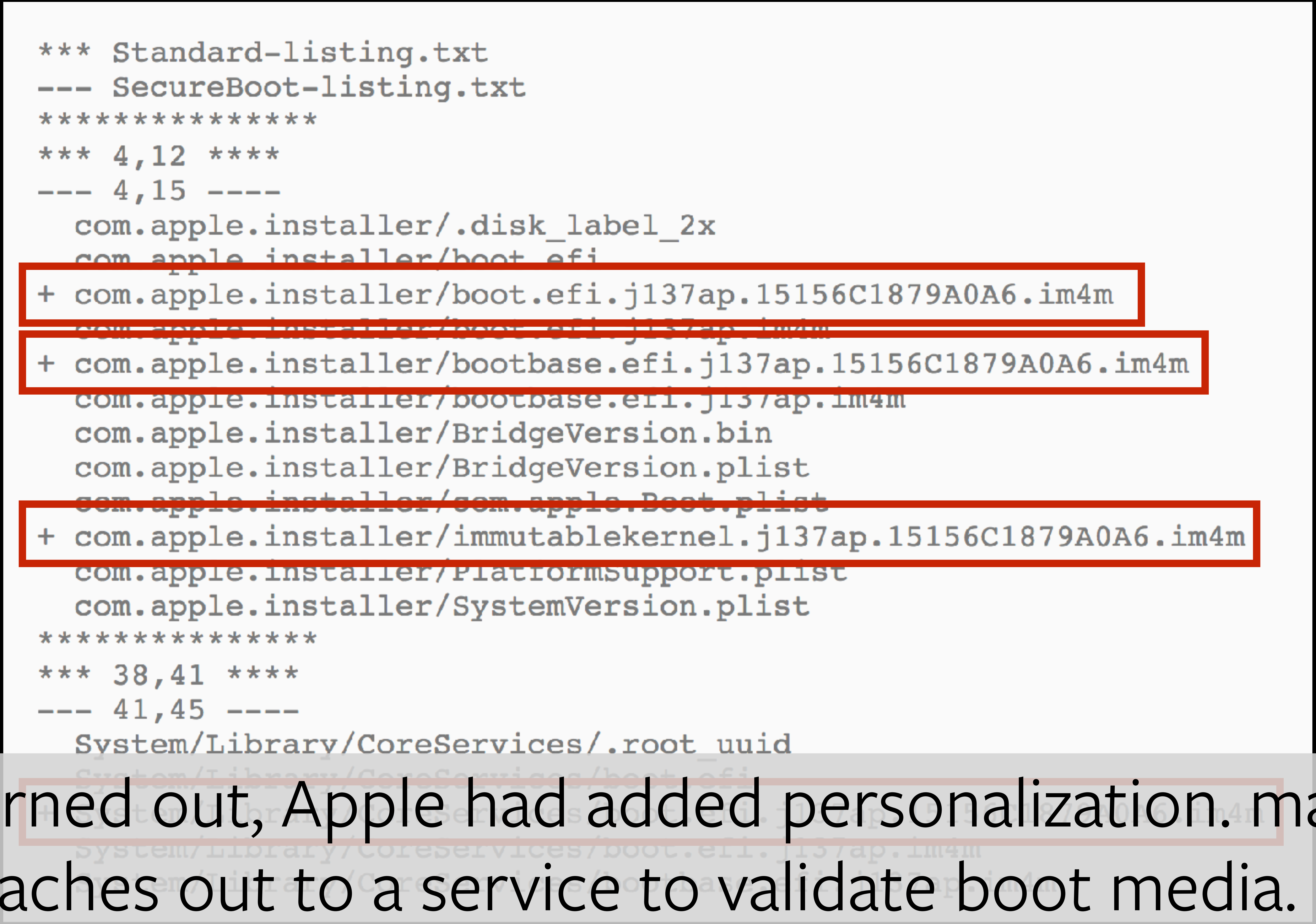
Booting Secure

michaellynn.github.io/2018/07/27/booting-secure

So I dove in and looked at what the OS was doing on these new Secure Boot machines. What exactly was ″Secure″ ?

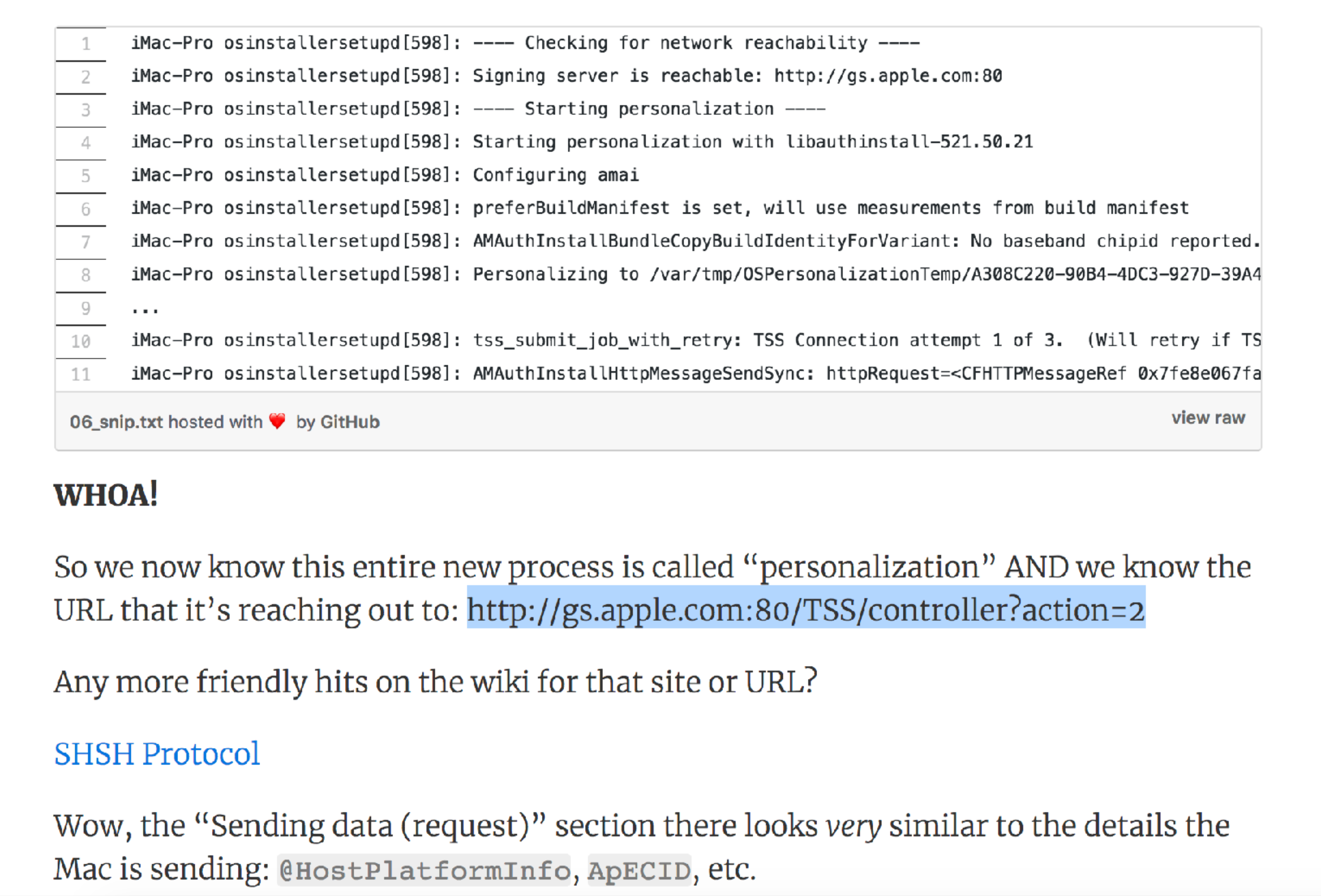


```
*** Standard-listing.txt
--- SecureBoot-listing.txt
***************
*** 4,12 ****
--- 4,15 ----
  com.apple.installer/.disk_label_2x
  com.apple.installer/boot.efi
+ com.apple.installer/boot.efi.j137ap.15156C1879A0A6.im4m
  com.apple.installer/boot.efi.j137ap.im4m
+ com.apple.installer/bootbase.efi.j137ap.15156C1879A0A6.im4m
  com.apple.installer/bootbase.efi.j137ap.im4m
  com.apple.installer/BridgeVersion.bin
  com.apple.installer/BridgeVersion.plist
  com.apple.installer/com.apple.Boot.plist
+ com.apple.installer/immutablekernel.j137ap.15156C1879A0A6.im4m
  com.apple.installer/PlatformSupport.plist
  com.apple.installer/SystemVersion.plist
***************
*** 38,41 ****
--- 41,45 ----
  System/Library/CoreServices/.root_uuid
  System/Library/CoreServices/boot.efi
+ System/Library/CoreServices/boot.efi.j137ap.15156C1879A0A6.im4m
  System/Library/CoreServices/boot.efi.j137ap.im4m
  System/Library/CoreServices/bootbase.efi.j137ap.im4m
```



```
*** Standard-listing.txt
--- SecureBoot-listing.txt
***************
*** 4,12 ****
--- 4,15 ----
  com.apple.installer/.disk_label_2x
  com.apple.installer/boot.efi
+ com.apple.installer/boot.efi.j137ap.15156C1879A0A6.im4m
  com.apple.installer/boot.efi.j137ap.im4m
+ com.apple.installer/bootbase.efi.j137ap.15156C1879A0A6.im4m
  com.apple.installer/bootbase.efi.j137ap.im4m
  com.apple.installer/BridgeVersion.bin
  com.apple.installer/BridgeVersion.plist
  com.apple.installer/com.apple.Boot.plist
+ com.apple.installer/immutablekernel.j137ap.15156C1879A0A6.im4m
  com.apple.installer/PlatformSupport.plist
  com.apple.installer/SystemVersion.plist
***************
*** 38,41 ****
--- 41,45 ----
  System/Library/CoreServices/.root_uuid
  System/Library/CoreServices/boot.efi
+ System/Library/CoreServices/boot.efi.j137ap.15156C1879A0A6.im4m
  System/Library/CoreServices/boot.efi.j137ap.im4m
  System/Library/CoreServices/bootbase.efi.j137ap.im4m
```

As it turned out, Apple had added personalization. macOS now reaches out to a service to validate boot media.



```
iMac-Pro osinstallersetupd[598]: ---- Checking for network reachability ----
iMac-Pro osinstallersetupd[598]: Signing server is reachable: http://gs.apple.com:80
iMac-Pro osinstallersetupd[598]: ---- Starting personalization ----
iMac-Pro osinstallersetupd[598]: Starting personalization with libauthinstall-521.50.21
iMac-Pro osinstallersetupd[598]: Configuring amai
iMac-Pro osinstallersetupd[598]: preferBuildManifest is set, will use measurements from build manifest
iMac-Pro osinstallersetupd[598]: AMAuthInstallBundleCopyBuildIdentityForVariant: No baseband chipid reported.
iMac-Pro osinstallersetupd[598]: Personalizing to /var/tmp/OSPersonalizationTemp/A308C220-90B4-4DC3-927D-39A4
...
iMac-Pro osinstallersetupd[598]: tss_submit_job_with_retry: TSS Connection attempt 1 of 3.  (Will retry if TS
iMac-Pro osinstallersetupd[598]: AMAuthInstallHttpMessageSendSync: httpRequest=<CFHTTPMessageRef 0x7fe8e067fa
```

06_snip.txt hosted with ❤ by GitHub   view raw

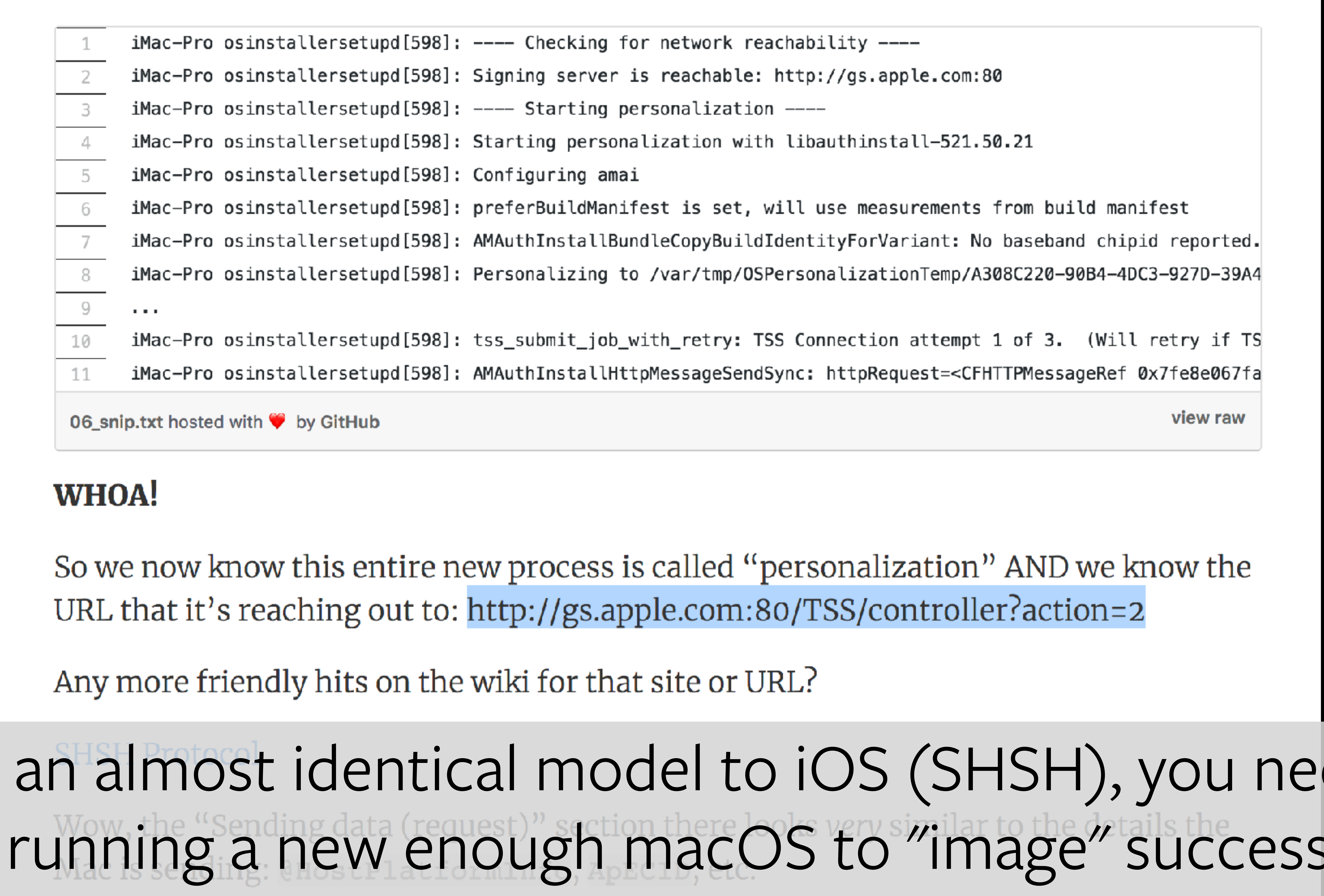
WHOA!

So we now know this entire new process is called "personalization" AND we know the URL that it's reaching out to: http://gs.apple.com:80/TSS/controller?action=2

Any more friendly hits on the wiki for that site or URL?

SHSH Protocol

Wow, the "Sending data (request)" section there looks *very* similar to the details the Mac is sending: `@HostPlatformInfo`, `ApECID`, etc.



```
iMac-Pro osinstallersetupd[598]: ---- Checking for network reachability ----
iMac-Pro osinstallersetupd[598]: Signing server is reachable: http://gs.apple.com:80
iMac-Pro osinstallersetupd[598]: ---- Starting personalization ----
iMac-Pro osinstallersetupd[598]: Starting personalization with libauthinstall-521.50.21
iMac-Pro osinstallersetupd[598]: Configuring amai
iMac-Pro osinstallersetupd[598]: preferBuildManifest is set, will use measurements from build manifest
iMac-Pro osinstallersetupd[598]: AMAuthInstallBundleCopyBuildIdentityForVariant: No baseband chipid reported.
iMac-Pro osinstallersetupd[598]: Personalizing to /var/tmp/OSPersonalizationTemp/A308C220-90B4-4DC3-927D-39A4
...
iMac-Pro osinstallersetupd[598]: tss_submit_job_with_retry: TSS Connection attempt 1 of 3.  (Will retry if TS
iMac-Pro osinstallersetupd[598]: AMAuthInstallHttpMessageSendSync: httpRequest=<CFHTTPMessageRef 0x7fe8e067fa
```

06_snip.txt hosted with ❤ by GitHub view raw

WHOA!

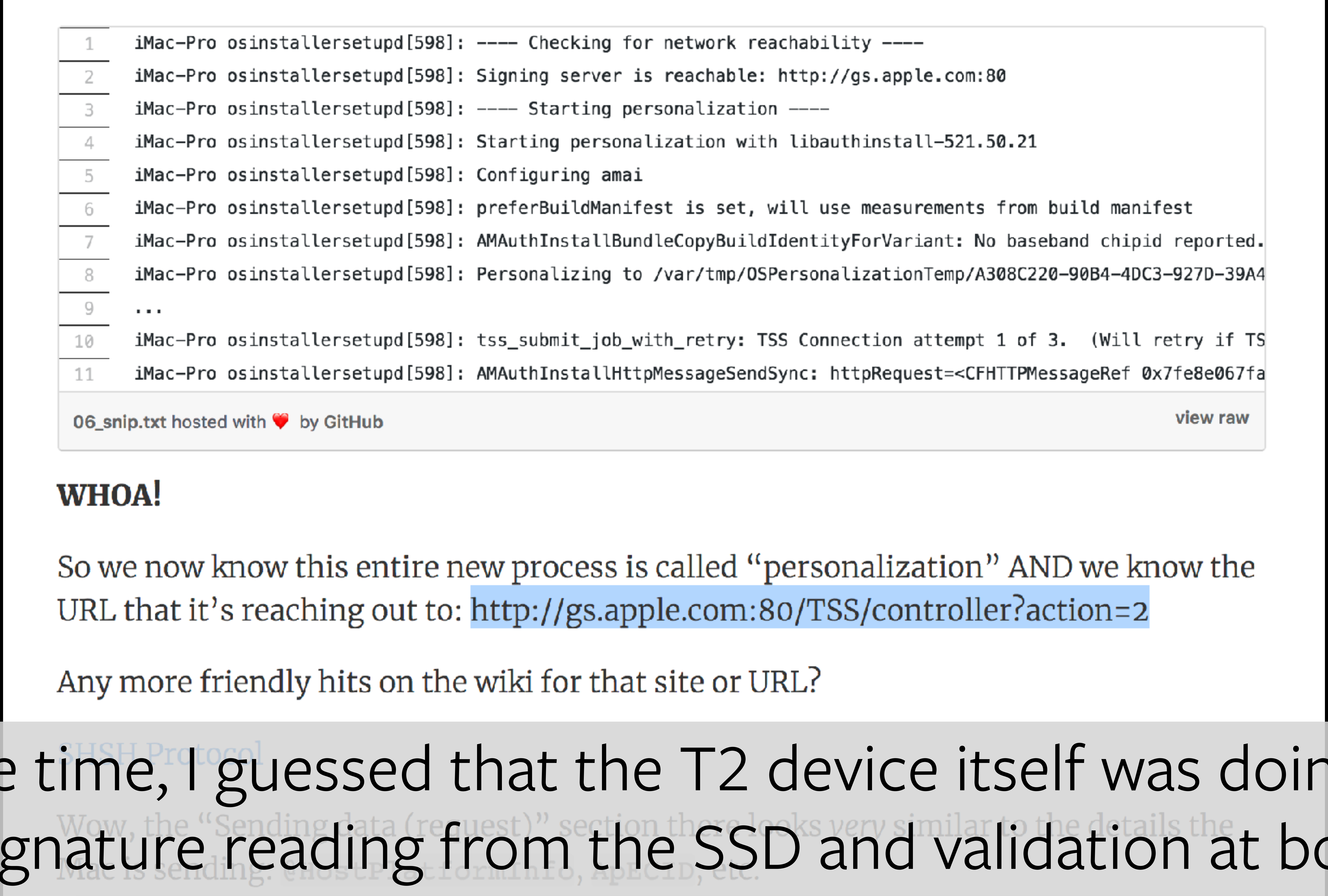
So we now know this entire new process is called “personalization” AND we know the URL that it’s reaching out to: http://gs.apple.com:80/TSS/controller?action=2

Any more friendly hits on the wiki for that site or URL?

SHSH Protocol

Wow, the “Sending data (request)” section there looks *very* similar to the details the Mac is sending: `@HostPlatformInfo`, ApECID, etc.

Using an almost identical model to iOS (SHSH), you needed to be running a new enough macOS to "image" successfully.



```
iMac-Pro osinstallersetupd[598]: ---- Checking for network reachability ----
iMac-Pro osinstallersetupd[598]: Signing server is reachable: http://gs.apple.com:80
iMac-Pro osinstallersetupd[598]: ---- Starting personalization ----
iMac-Pro osinstallersetupd[598]: Starting personalization with libauthinstall-521.50.21
iMac-Pro osinstallersetupd[598]: Configuring amai
iMac-Pro osinstallersetupd[598]: preferBuildManifest is set, will use measurements from build manifest
iMac-Pro osinstallersetupd[598]: AMAuthInstallBundleCopyBuildIdentityForVariant: No baseband chipid reported.
iMac-Pro osinstallersetupd[598]: Personalizing to /var/tmp/OSPersonalizationTemp/A308C220-90B4-4DC3-927D-39A4
...
iMac-Pro osinstallersetupd[598]: tss_submit_job_with_retry: TSS Connection attempt 1 of 3.  (Will retry if TS
iMac-Pro osinstallersetupd[598]: AMAuthInstallHttpMessageSendSync: httpRequest=<CFHTTPMessageRef 0x7fe8e067fa
```

06_snip.txt hosted with ❤ by GitHub view raw

WHOA!

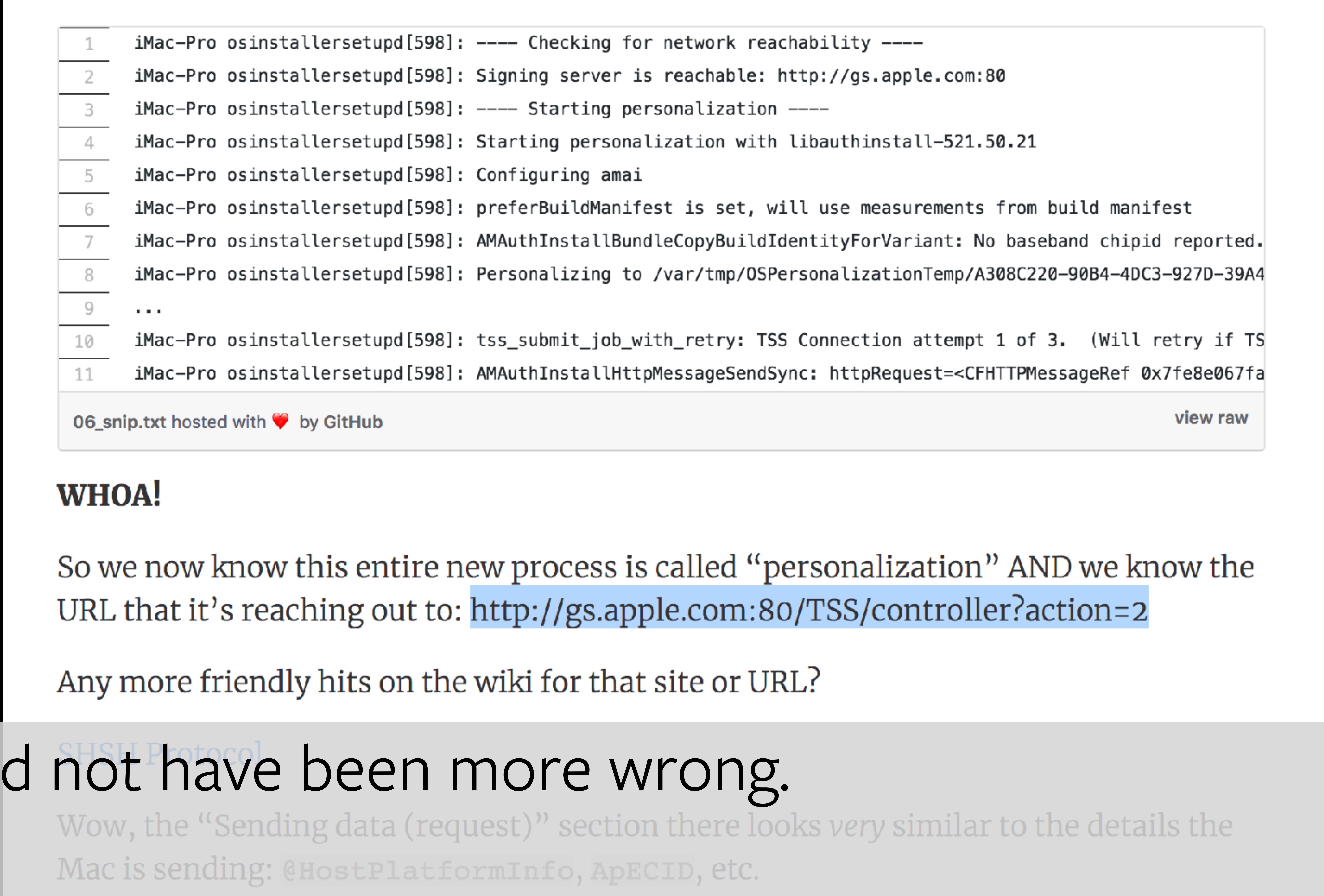
So we now know this entire new process is called "personalization" AND we know the URL that it's reaching out to: http://gs.apple.com:80/TSS/controller?action=2

Any more friendly hits on the wiki for that site or URL?

SHSH Protocol

Wow, the "Sending data (request)" section there looks *very* similar to the details the Mac is sending: @HostPlatformInfo, ApECID, etc.

At the time, I guessed that the T2 device itself was doing the signature reading from the SSD and validation at boot.



```
iMac-Pro osinstallersetupd[598]: ---- Checking for network reachability ----
iMac-Pro osinstallersetupd[598]: Signing server is reachable: http://gs.apple.com:80
iMac-Pro osinstallersetupd[598]: ---- Starting personalization ----
iMac-Pro osinstallersetupd[598]: Starting personalization with libauthinstall-521.50.21
iMac-Pro osinstallersetupd[598]: Configuring amai
iMac-Pro osinstallersetupd[598]: preferBuildManifest is set, will use measurements from build manifest
iMac-Pro osinstallersetupd[598]: AMAuthInstallBundleCopyBuildIdentityForVariant: No baseband chipid reported.
iMac-Pro osinstallersetupd[598]: Personalizing to /var/tmp/OSPersonalizationTemp/A308C220-90B4-4DC3-927D-39A4
...
iMac-Pro osinstallersetupd[598]: tss_submit_job_with_retry: TSS Connection attempt 1 of 3.  (Will retry if TS
iMac-Pro osinstallersetupd[598]: AMAuthInstallHttpMessageSendSync: httpRequest=<CFHTTPMessageRef 0x7fe8e067fa
```

06_snip.txt hosted with ❤ by GitHub

view raw

WHOA!

So we now know this entire new process is called "personalization" AND we know the URL that it's reaching out to: http://gs.apple.com:80/TSS/controller?action=2

Any more friendly hits on the wiki for that site or URL?

SHSH Protocol

Wow, the "Sending data (request)" section there looks *very* similar to the details the Mac is sending: `@HostPlatformInfo`, `ApECID`, etc.

I could not have been more wrong.

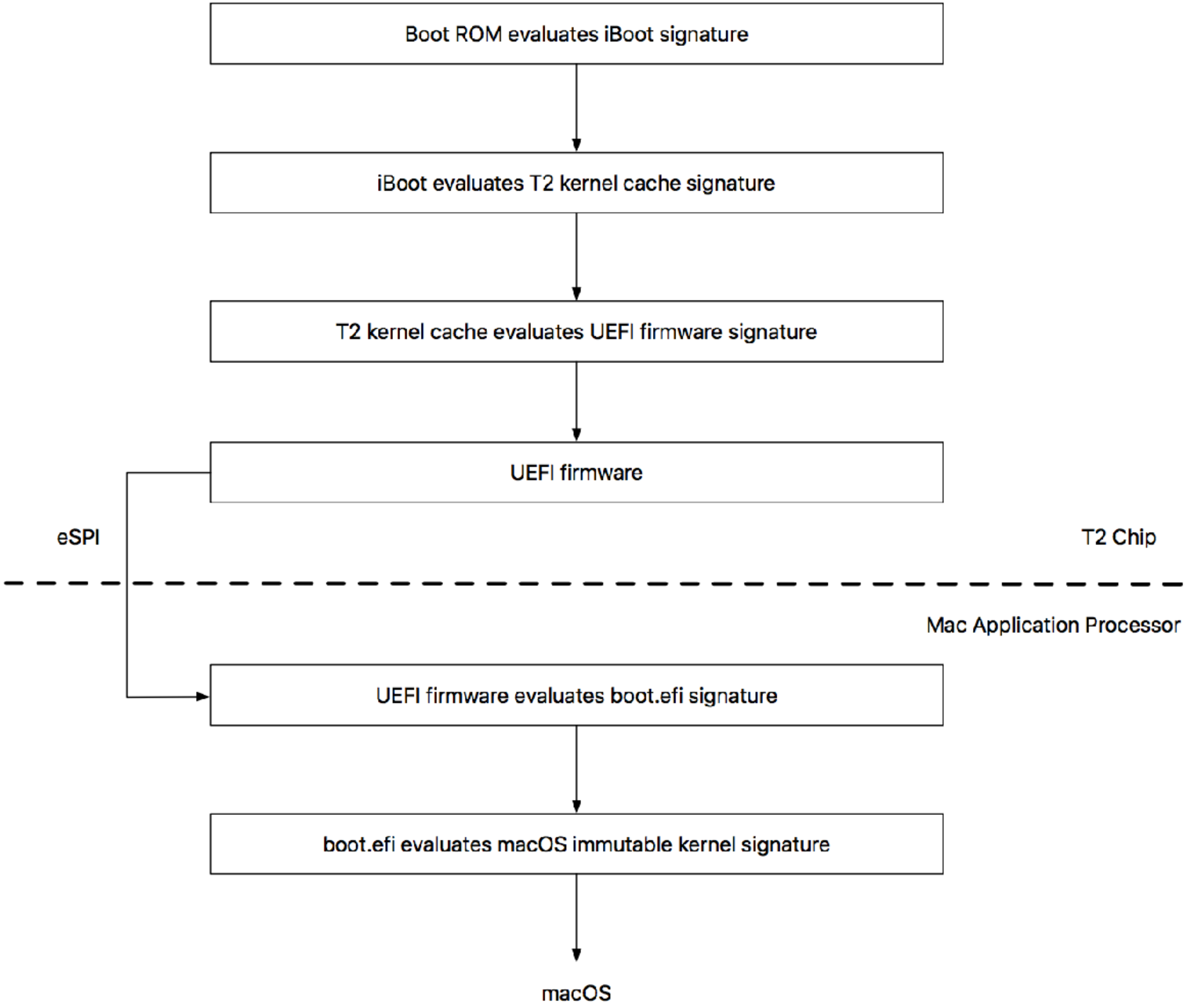
www.apple.com/mac/docs/Apple_T2_Security_Chip_Overview.pdf

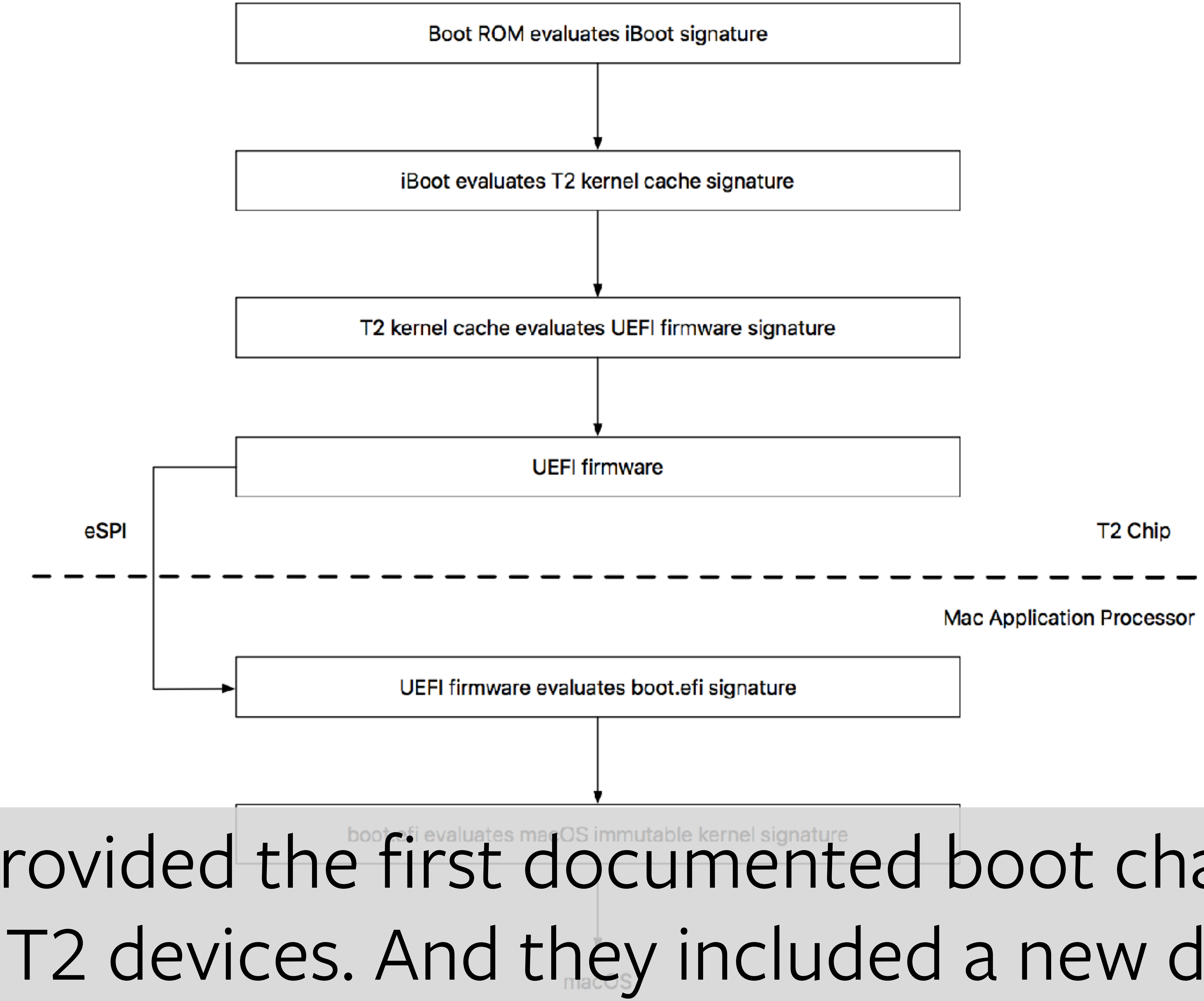
Apple T2 Security Chip

Security Overview

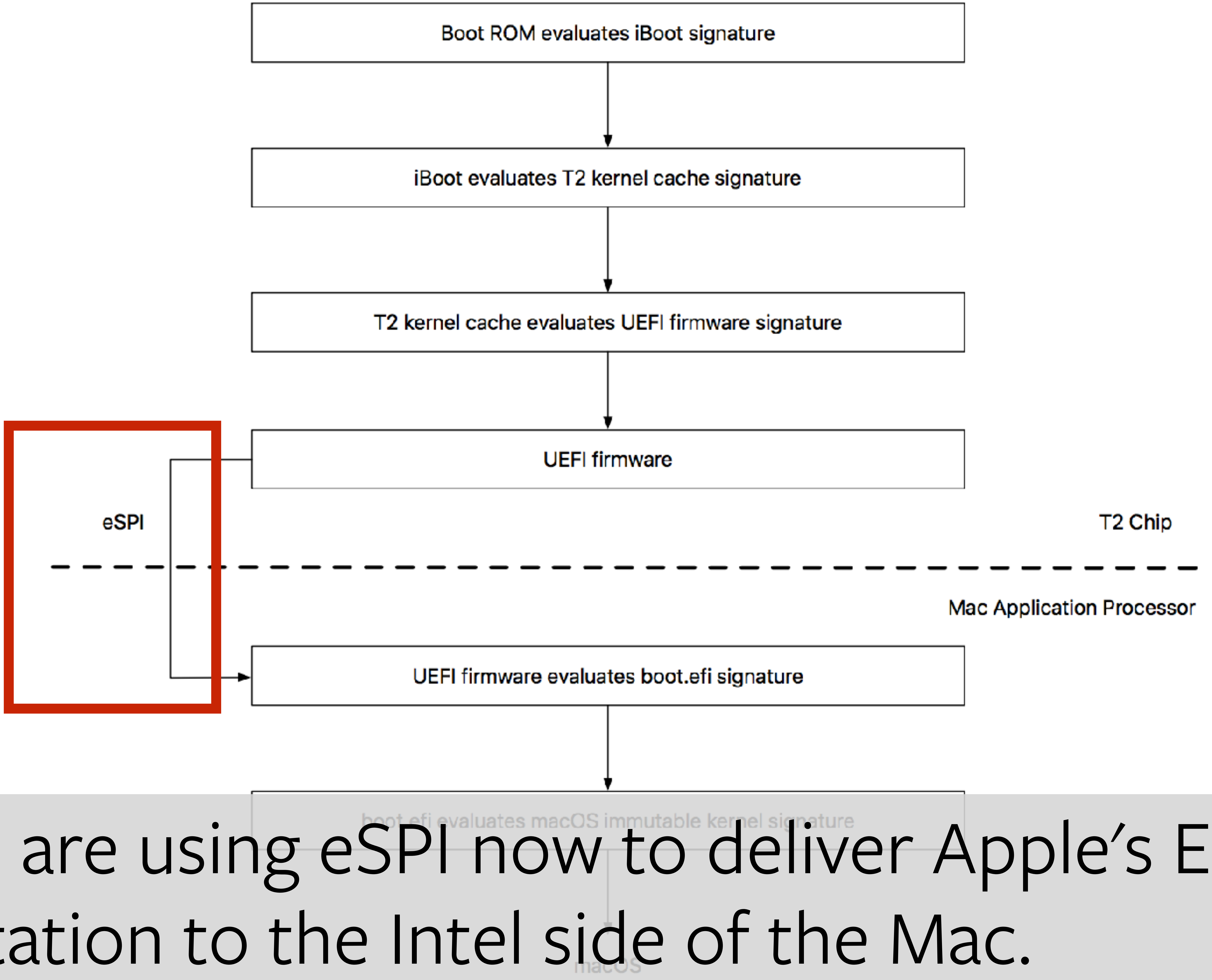
October 2018

Apple just recently released this overview of the T2.
Everyone here should read this.


Boot ROM evaluates iBoot signature
iBoot evaluates T2 kernel cache signature
T2 kernel cache evaluates UEFI firmware signature
UEFI firmware
eSPI
T2 Chip
Mac Application Processor
UEFI firmware evaluates boot.efi signature
boot.efi evaluates macOS immutable kernel signature
macOS


Boot ROM evaluates iBoot signature
iBoot evaluates T2 kernel cache signature
T2 kernel cache evaluates UEFI firmware signature
UEFI firmware
eSPI
T2 Chip
Mac Application Processor
UEFI firmware evaluates boot.efi signature
boot.efi evaluates macOS immutable kernel signature
macOS

In it, they provided the first documented boot chain diagram of T2 devices. And they included a new detail.



Boot ROM evaluates iBoot signature

iBoot evaluates T2 kernel cache signature

T2 kernel cache evaluates UEFI firmware signature

UEFI firmware

eSPI

T2 Chip

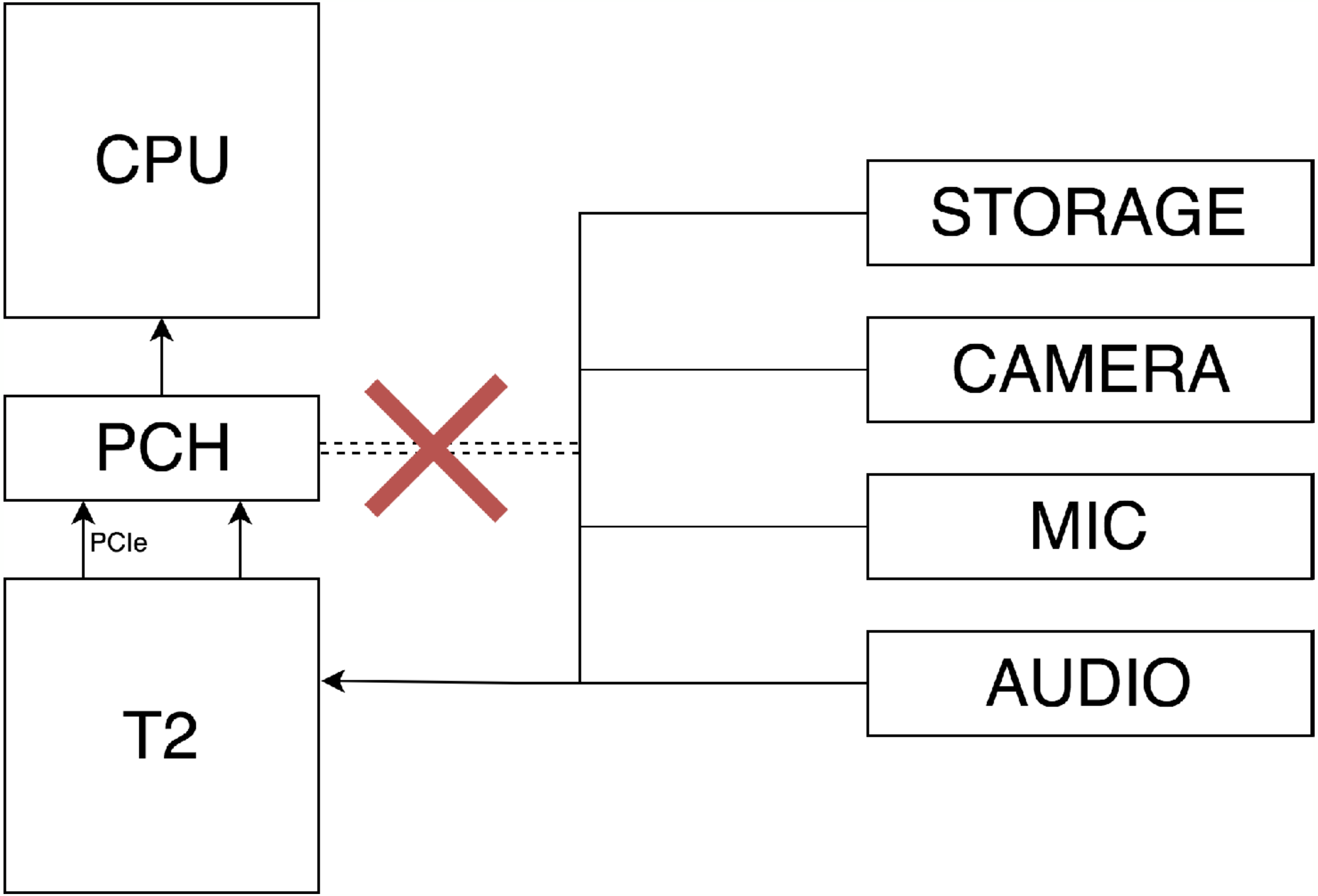
Mac Application Processor

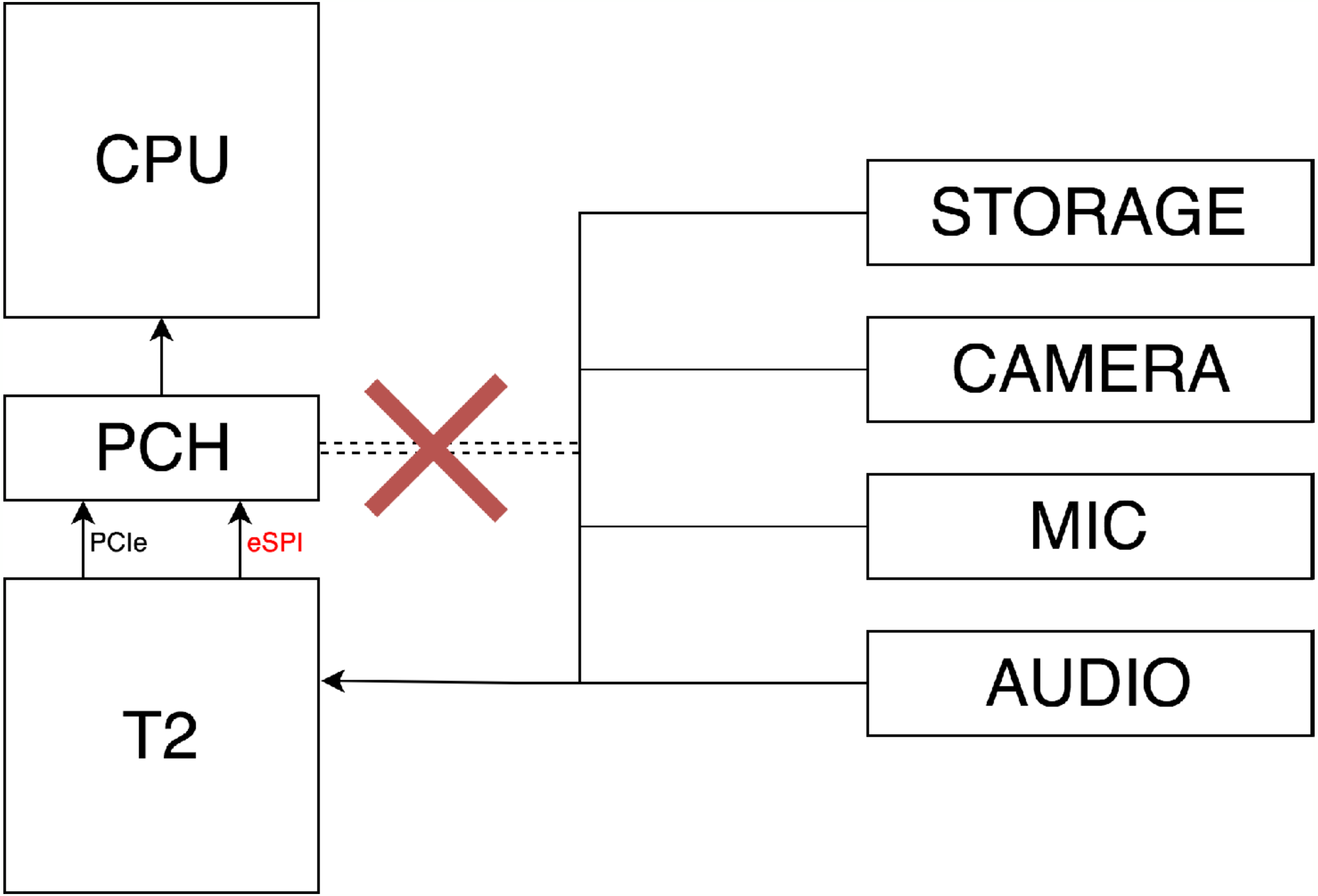
UEFI firmware evaluates boot.efi signature

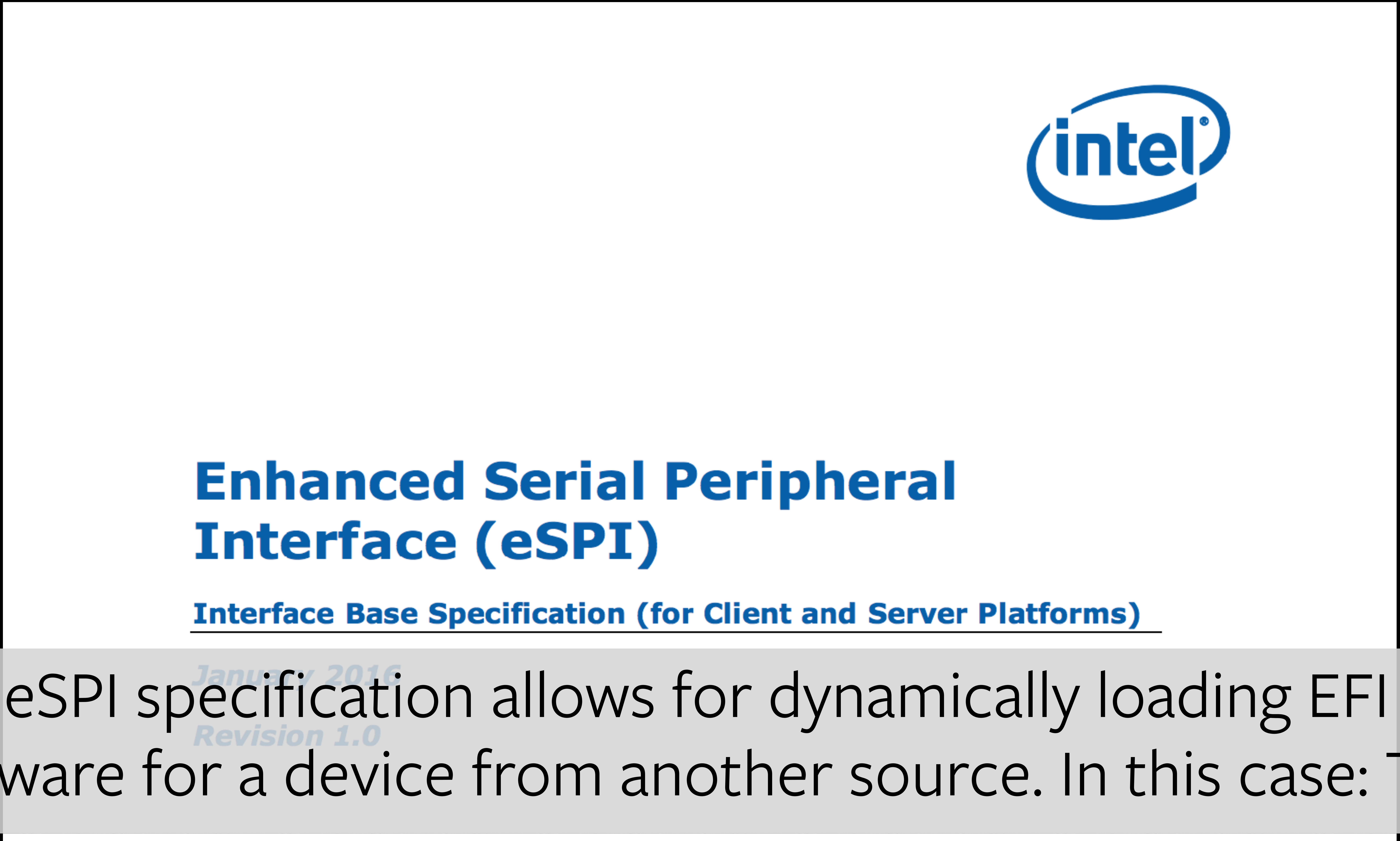
boot.efi evaluates macOS immutable kernel signature

macOS

T2 devices are using eSPI now to deliver Apple’s EFI implementation to the Intel side of the Mac.


CPU
PCH
PCIe
T2
STORAGE
CAMERA
MIC
AUDIO


CPU
PCH
PCIe
eSPI
T2
STORAGE
CAMERA
MIC
AUDIO



intel

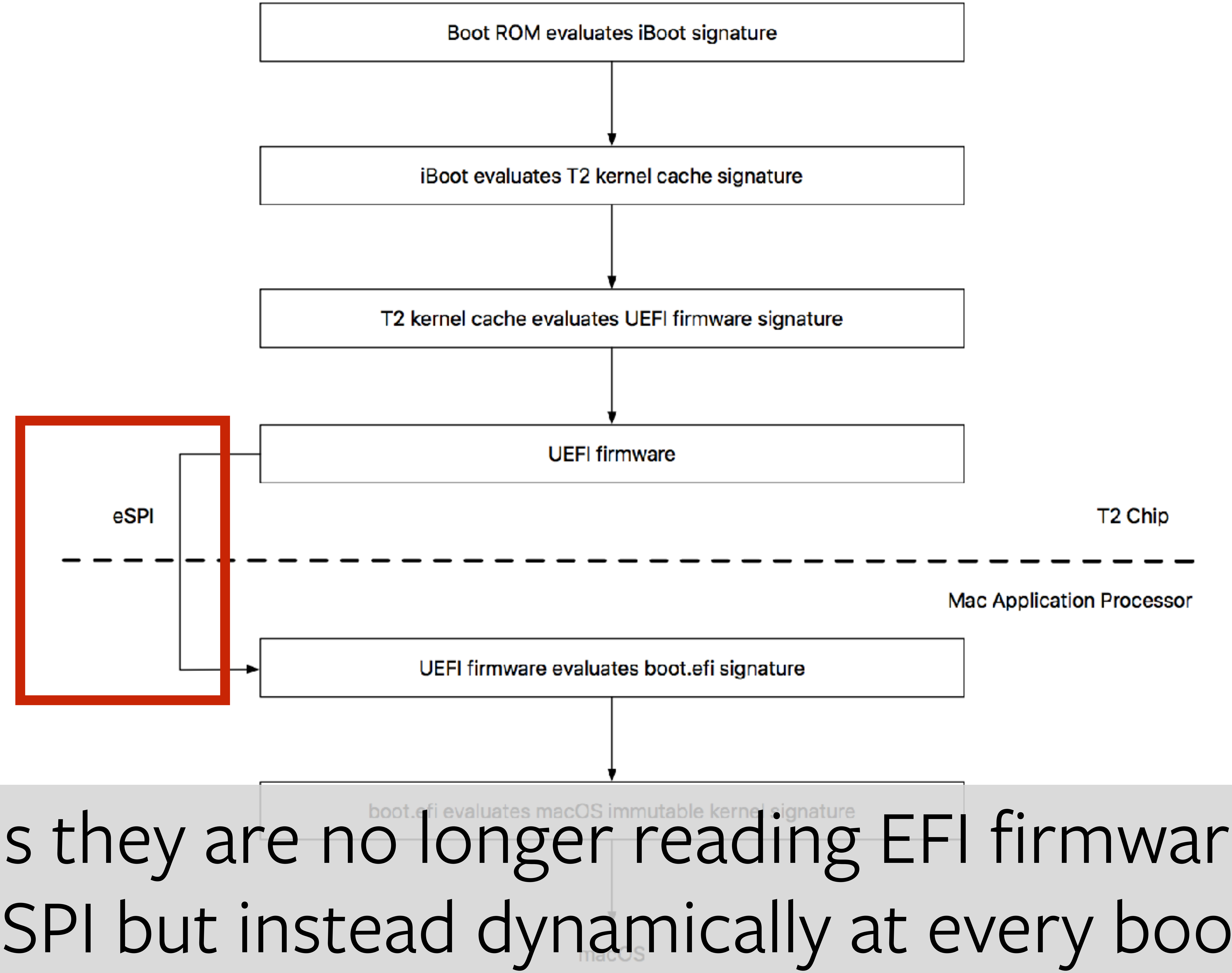
Enhanced Serial Peripheral Interface (eSPI)

Interface Base Specification (for Client and Server Platforms)

January 2016

Revision 1.0

The eSPI specification allows for dynamically loading EFI firmware for a device from another source. In this case: T2



Boot ROM evaluates iBoot signature

iBoot evaluates T2 kernel cache signature

T2 kernel cache evaluates UEFI firmware signature

UEFI firmware

eSPI

T2 Chip

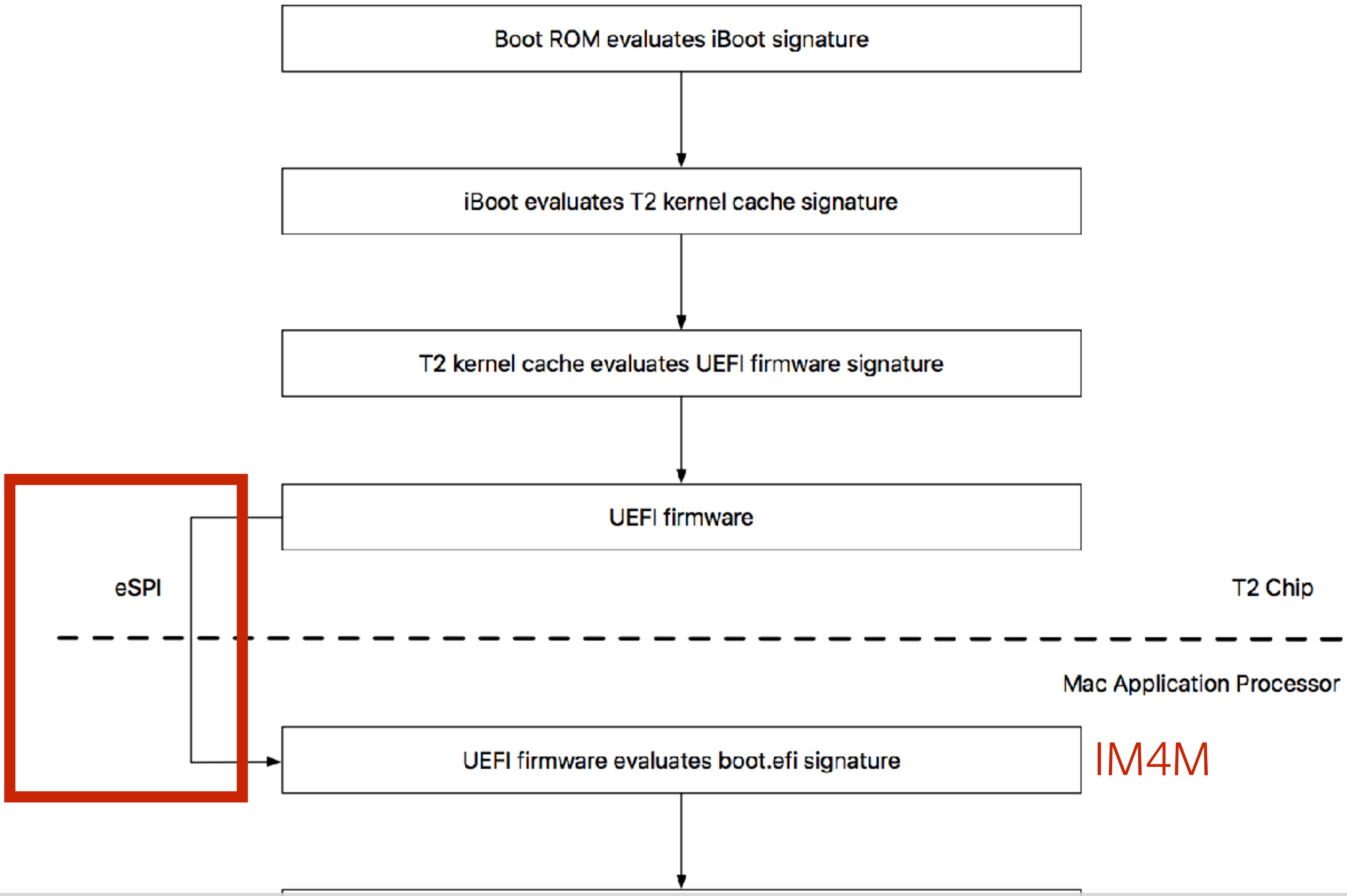
Mac Application Processor

UEFI firmware evaluates boot.efi signature

boot.efi evaluates macOS immutable kernel signature

macOS

This means they are no longer reading EFI firmware from a physical SPI but instead dynamically at every boot.



Boot ROM evaluates iBoot signature

iBoot evaluates T2 kernel cache signature

T2 kernel cache evaluates UEFI firmware signature

UEFI firmware

eSPI

T2 Chip

Mac Application Processor

UEFI firmware evaluates boot.efi signature

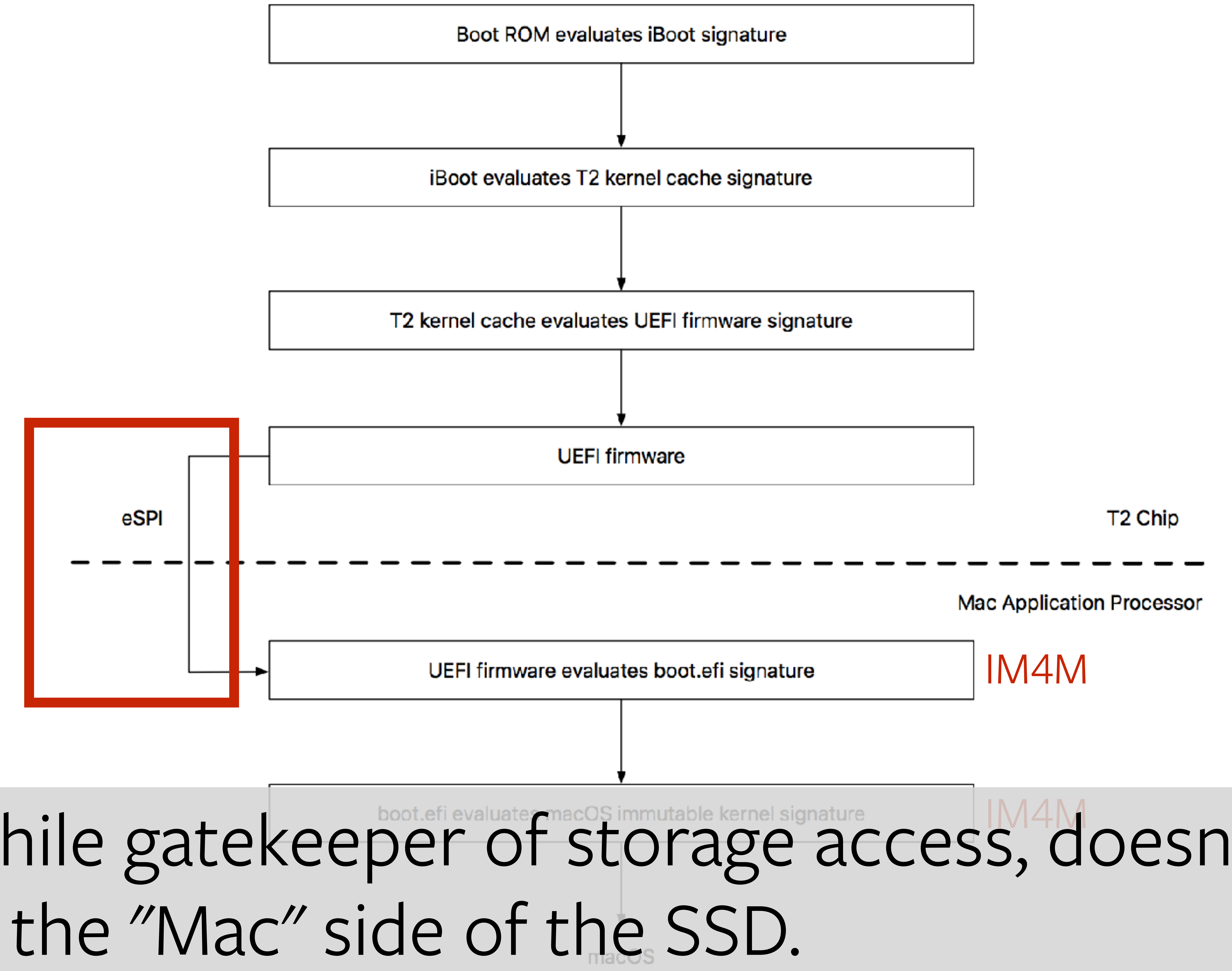
IM4M

boot.efi evaluates macOS immutable kernel signature

IM4M

macOS

The T2 validates the firmware of the machine and send it to be loaded. It’s the firmware that performs sig checks.



Boot ROM evaluates iBoot signature

iBoot evaluates T2 kernel cache signature

T2 kernel cache evaluates UEFI firmware signature

UEFI firmware

eSPI

T2 Chip

Mac Application Processor

UEFI firmware evaluates boot.efi signature

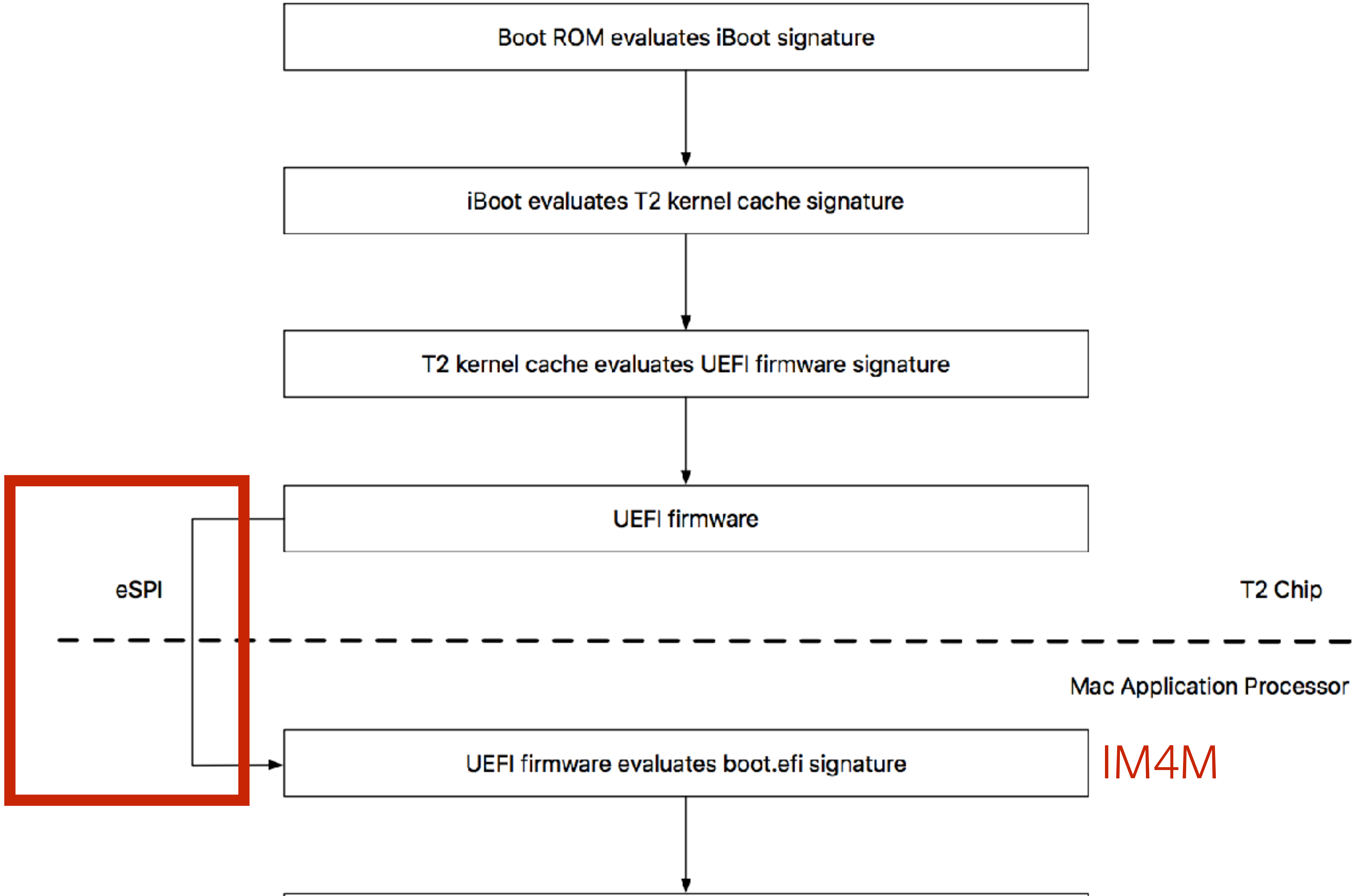
IM4M

boot.efi evaluates macOS immutable kernel signature

IM4M

macOS

The T2, while gatekeeper of storage access, doesn’t ever truly read the ”Mac” side of the SSD.



Boot ROM evaluates iBoot signature

iBoot evaluates T2 kernel cache signature

T2 kernel cache evaluates UEFI firmware signature

UEFI firmware

eSPI

T2 Chip

Mac Application Processor

UEFI firmware evaluates boot.efi signature

IM4M

boot.efi evaluates macOS immutable kernel signature

IM4M

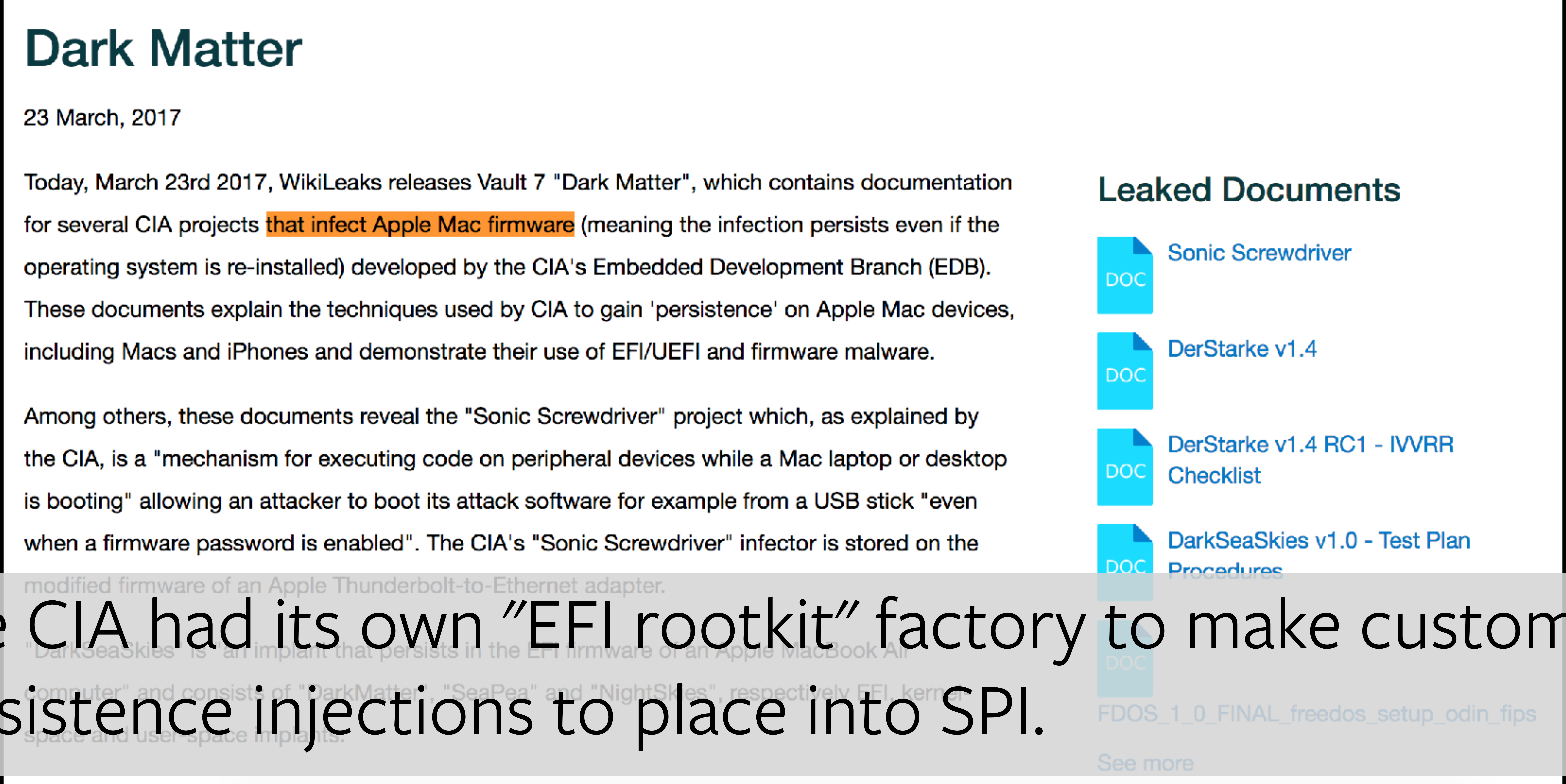
macOS

Its job is to validate the root of the boot chain. Once EFI loads, a T2 Mac is in many ways similar to prior models.

Vault 7 Leaks

Do you remember the Vault 7 leaks?

Vault 7 Leaks



Dark Matter

23 March, 2017

Today, March 23rd 2017, WikiLeaks releases Vault 7 "Dark Matter", which contains documentation for several CIA projects that infect Apple Mac firmware (meaning the infection persists even if the operating system is re-installed) developed by the CIA's Embedded Development Branch (EDB). These documents explain the techniques used by CIA to gain 'persistence' on Apple Mac devices, including Macs and iPhones and demonstrate their use of EFI/UEFI and firmware malware.

Among others, these documents reveal the "Sonic Screwdriver" project which, as explained by the CIA, is a "mechanism for executing code on peripheral devices while a Mac laptop or desktop is booting" allowing an attacker to boot its attack software for example from a USB stick "even when a firmware password is enabled". The CIA's "Sonic Screwdriver" infector is stored on the modified firmware of an Apple Thunderbolt-to-Ethernet adapter.

"DarkSeaSkies" is "an implant that persists in the EFI firmware of an Apple MacBook Air computer" and consists of "DarkMatter", "SeaPea" and "NightSkies", respectively EFI, kernel-space and user-space implants.

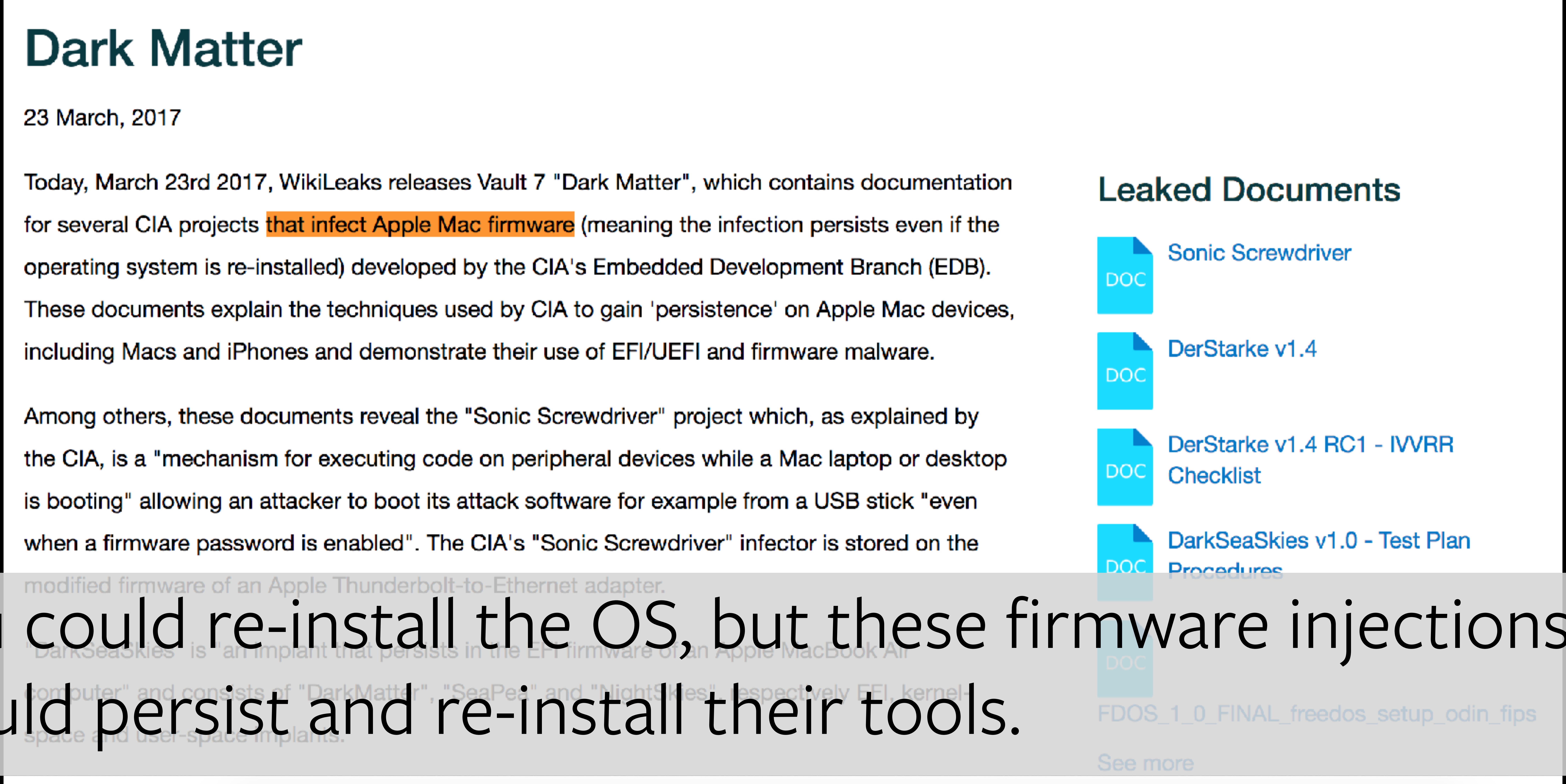
Leaked Documents

- DOC Sonic Screwdriver
- DOC DerStarke v1.4
- DOC DerStarke v1.4 RC1 - IVVRR Checklist
- DOC DarkSeaSkies v1.0 - Test Plan Procedures
- DOC FDOS_1_0_FINAL_freedos_setup_odin_fips

See more

The CIA had its own "EFI rootkit" factory to make custom persistence injections to place into SPI.

Vault 7 Leaks



Dark Matter

23 March, 2017

Today, March 23rd 2017, WikiLeaks releases Vault 7 "Dark Matter", which contains documentation for several CIA projects that infect Apple Mac firmware (meaning the infection persists even if the operating system is re-installed) developed by the CIA's Embedded Development Branch (EDB). These documents explain the techniques used by CIA to gain 'persistence' on Apple Mac devices, including Macs and iPhones and demonstrate their use of EFI/UEFI and firmware malware.

Among others, these documents reveal the "Sonic Screwdriver" project which, as explained by the CIA, is a "mechanism for executing code on peripheral devices while a Mac laptop or desktop is booting" allowing an attacker to boot its attack software for example from a USB stick "even when a firmware password is enabled". The CIA's "Sonic Screwdriver" infector is stored on the modified firmware of an Apple Thunderbolt-to-Ethernet adapter.

"DarkSeaSkies" is "an implant that persists in the EFI firmware of an Apple MacBook Air computer" and consists of "DarkMatter", "SeaPea" and "NightSkies", respectively EFI, kernel-space and user-space implants.

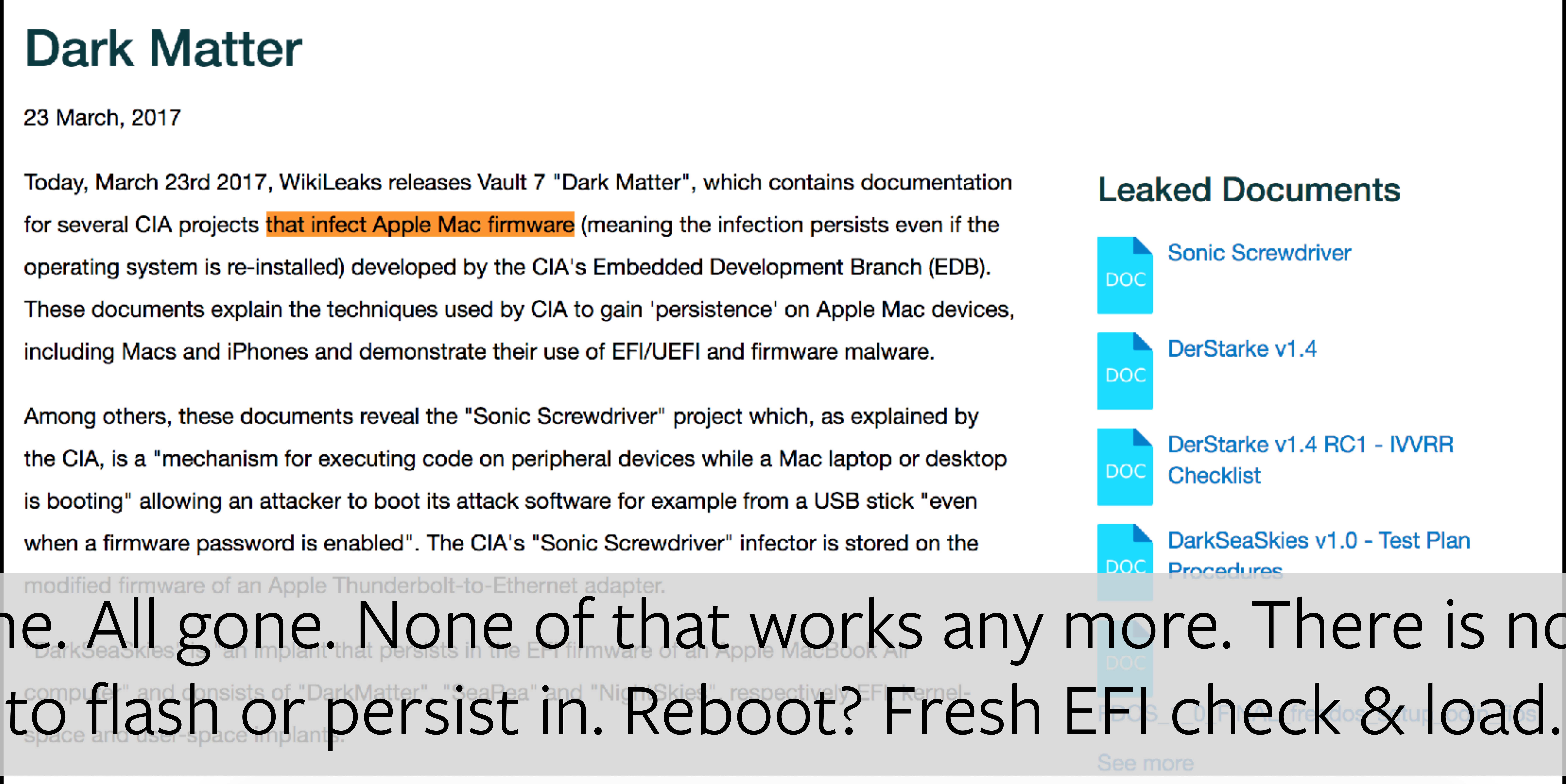
Leaked Documents

- DOC Sonic Screwdriver
- DOC DerStarke v1.4
- DOC DerStarke v1.4 RC1 - IVVRR Checklist
- DOC DarkSeaSkies v1.0 - Test Plan Procedures
- DOC FDOS_1_0_FINAL_freedos_setup_odin_fips

See more

You could re-install the OS, but these firmware injections would persist and re-install their tools.

Vault 7 Leaks



Dark Matter

23 March, 2017

Today, March 23rd 2017, WikiLeaks releases Vault 7 "Dark Matter", which contains documentation for several CIA projects that infect Apple Mac firmware (meaning the infection persists even if the operating system is re-installed) developed by the CIA's Embedded Development Branch (EDB). These documents explain the techniques used by CIA to gain 'persistence' on Apple Mac devices, including Macs and iPhones and demonstrate their use of EFI/UEFI and firmware malware.

Among others, these documents reveal the "Sonic Screwdriver" project which, as explained by the CIA, is a "mechanism for executing code on peripheral devices while a Mac laptop or desktop is booting" allowing an attacker to boot its attack software for example from a USB stick "even when a firmware password is enabled". The CIA's "Sonic Screwdriver" infector is stored on the modified firmware of an Apple Thunderbolt-to-Ethernet adapter.

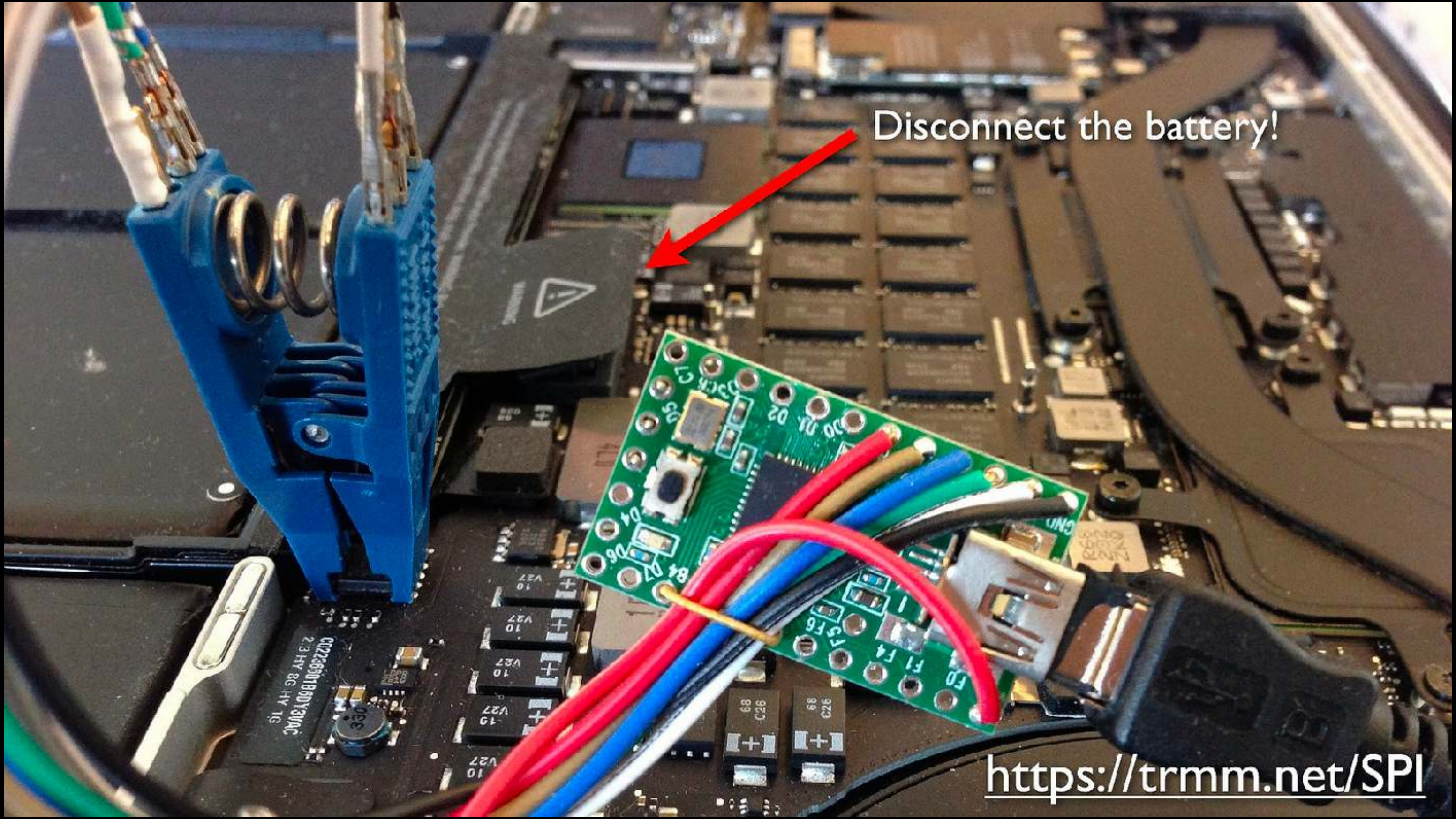
"DarkSeaSkies" is "an implant that persists in the EFI firmware of an Apple MacBook Air computer" and consists of "DarkMatter", "SeaPea" and "NightSkies", respectively EFI, kernel-space and user-space implants.

Leaked Documents

- DOC Sonic Screwdriver
- DOC DerStarke v1.4
- DOC DerStarke v1.4 RC1 - IVVRR Checklist
- DOC DarkSeaSkies v1.0 - Test Plan Procedures
- DOC [illegible]

See more

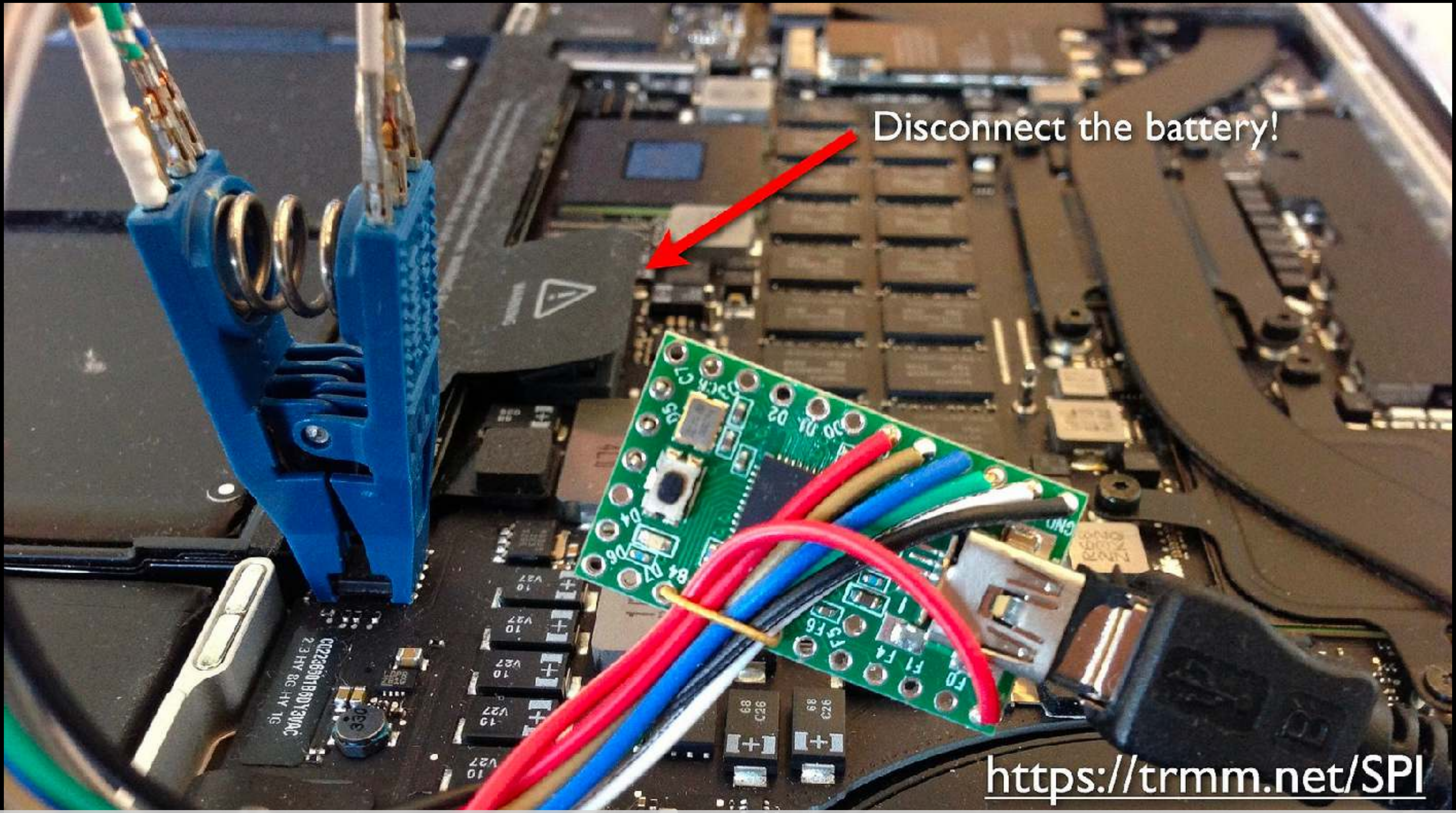
Gone. All gone. None of that works any more. There is no SPI to flash or persist in. Reboot? Fresh EFI check & load.



Disconnect the battery!

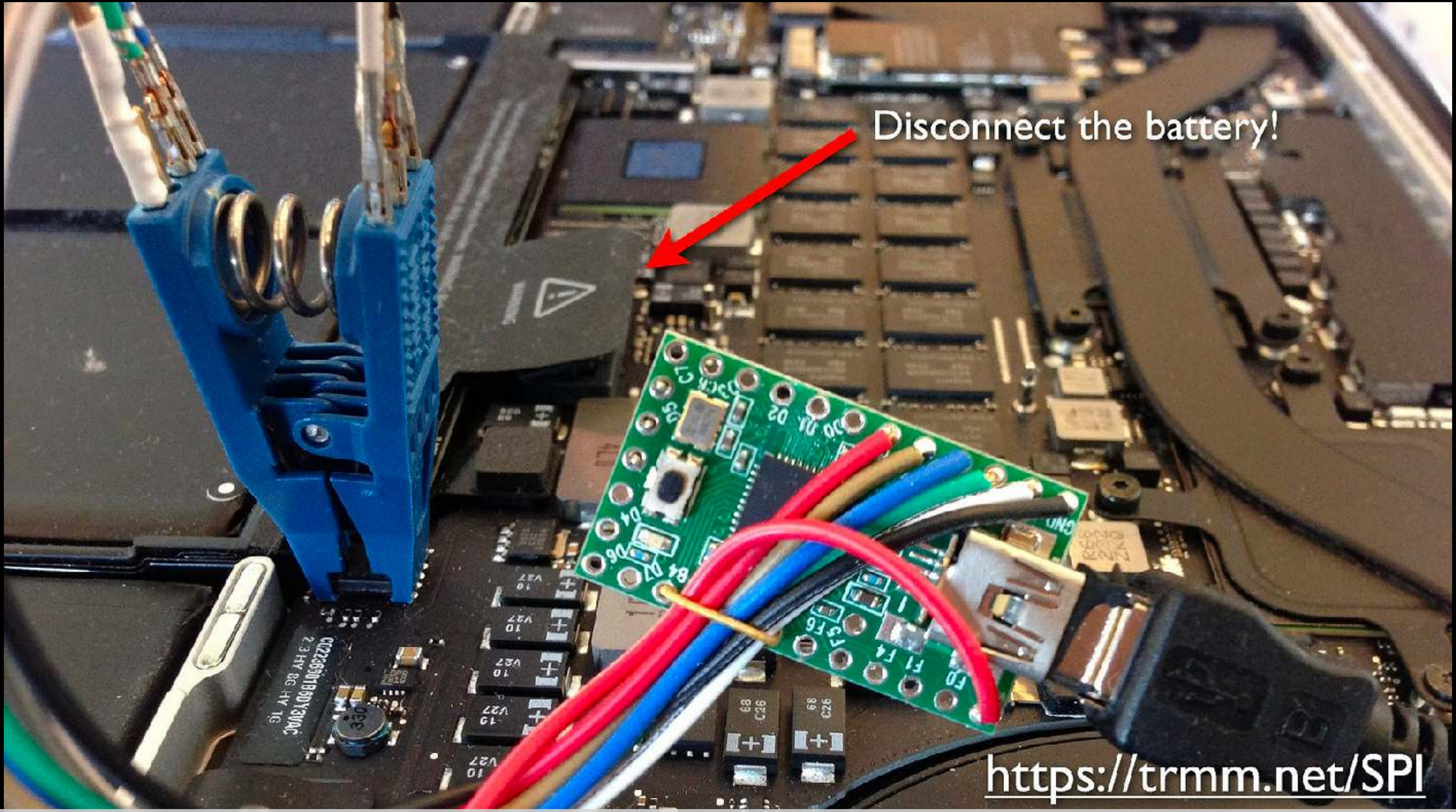
https://trmm.net/SPI

Thunderstrike



Thunderstrike

Thunderstrike - the same idea, only more real as a possible attack vector against companies and individuals.



Disconnect the battery!

https://trmm.net/SPI

Again - gone.

Thunderstrike

No NetBoot

No SPI / firmware flash

Need Apple's personalization

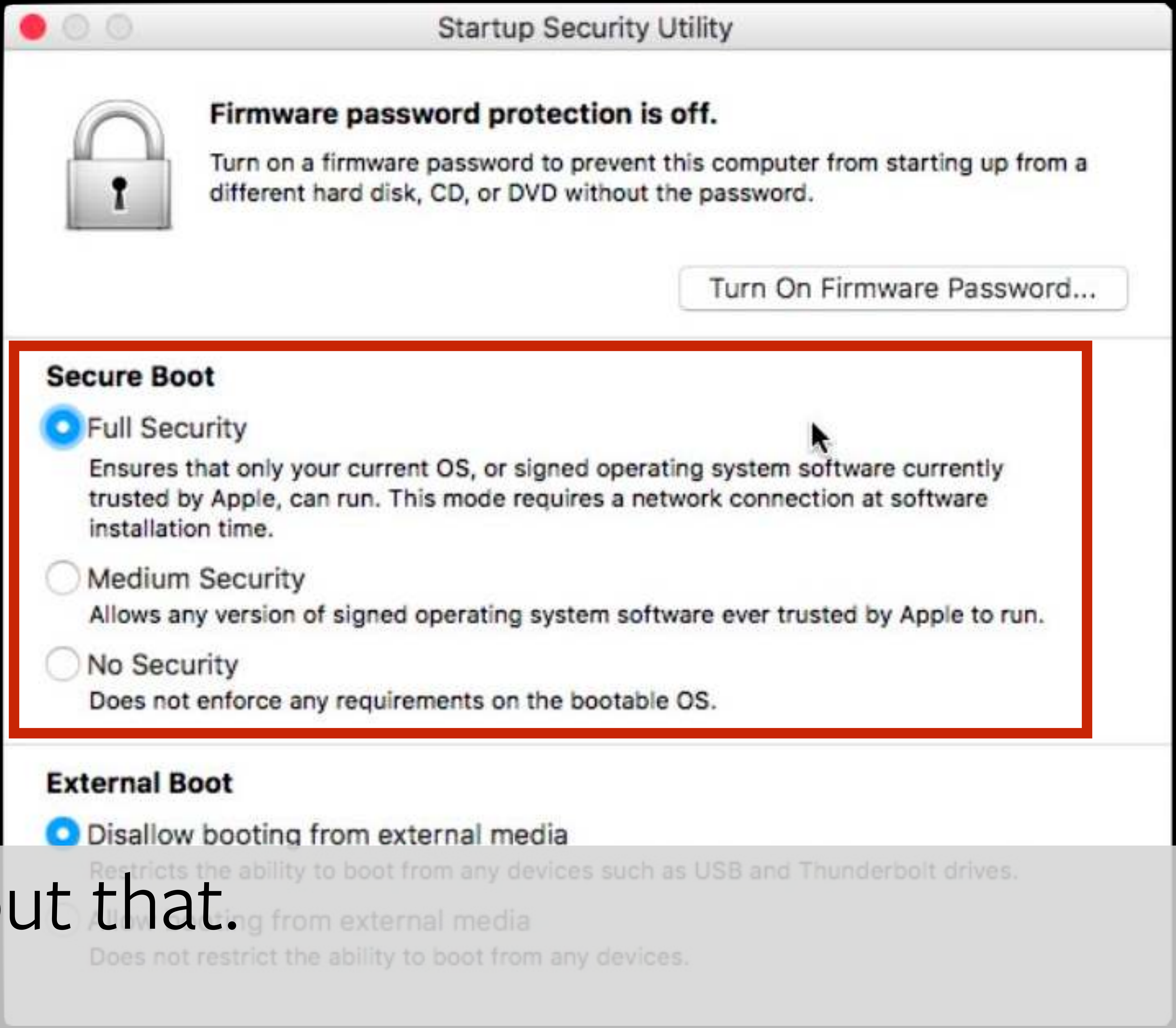
So we've seen some interesting details on differences. But we also mentioned controls and settings for Secure Boot.

No NetBoot

No SPI / firmware flash

Need Apple’s personalization

Surely an administrator could turn these things off if it was impacting their old workflows.



Firmware password protection is off.

Turn on a firmware password to prevent this computer from starting up from a different hard disk, CD, or DVD without the password.

Secure Boot

● Full Security

Ensures that only your current OS, or signed operating system software currently trusted by Apple, can run. This mode requires a network connection at software installation time.

○ Medium Security

Allows any version of signed operating system software ever trusted by Apple to run.

○ No Security

Does not enforce any requirements on the bootable OS.

External Boot

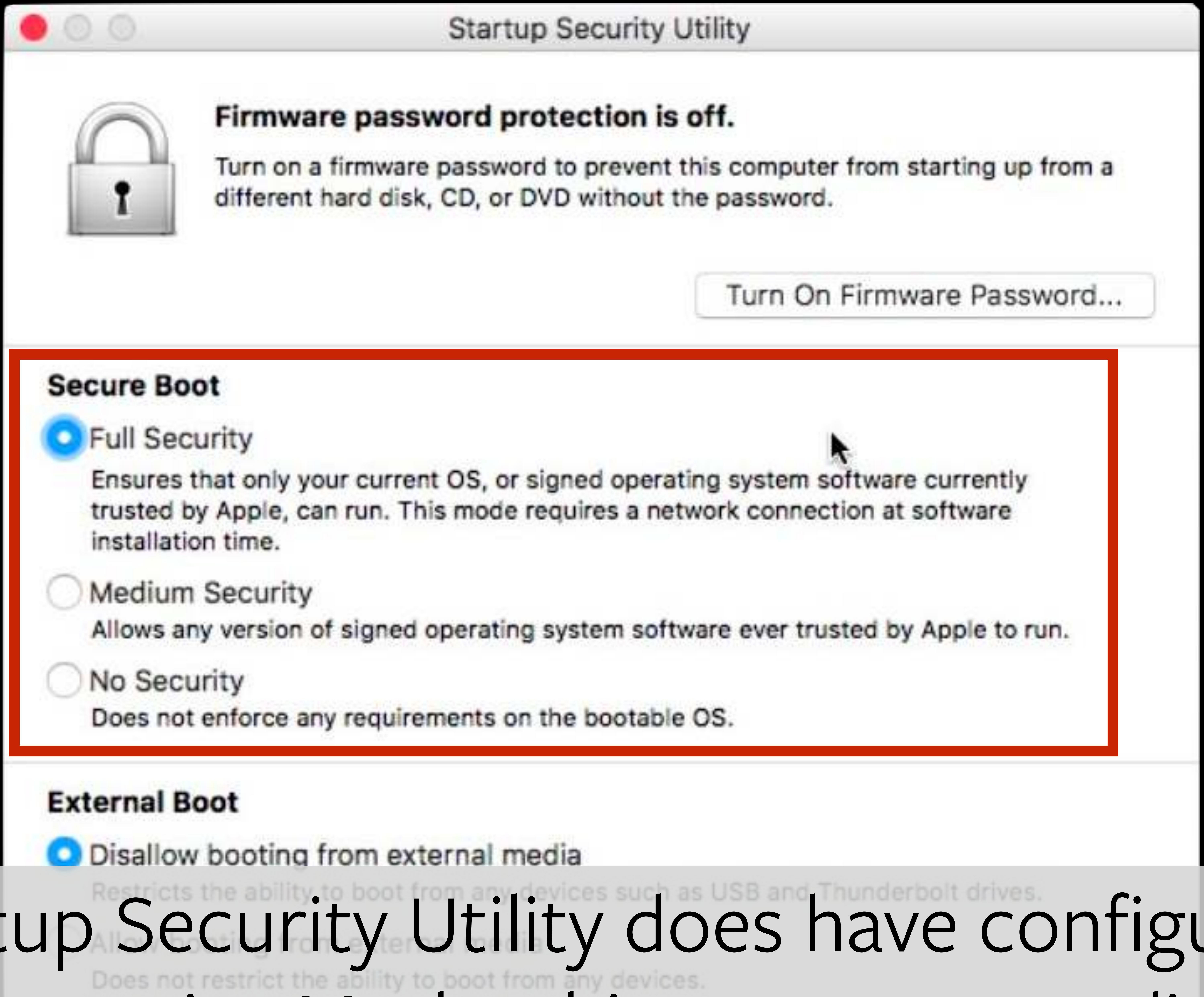
● Disallow booting from external media

Restricts the ability to boot from any devices such as USB and Thunderbolt drives.

○ Allow booting from external media

Does not restrict the ability to boot from any devices.

Let's talk about that.



Startup Security Utility

Firmware password protection is off.

Turn on a firmware password to prevent this computer from starting up from a different hard disk, CD, or DVD without the password.

Turn On Firmware Password...

Secure Boot

● Full Security

Ensures that only your current OS, or signed operating system software currently trusted by Apple, can run. This mode requires a network connection at software installation time.

○ Medium Security

Allows any version of signed operating system software ever trusted by Apple to run.

○ No Security

Does not enforce any requirements on the bootable OS.

External Boot

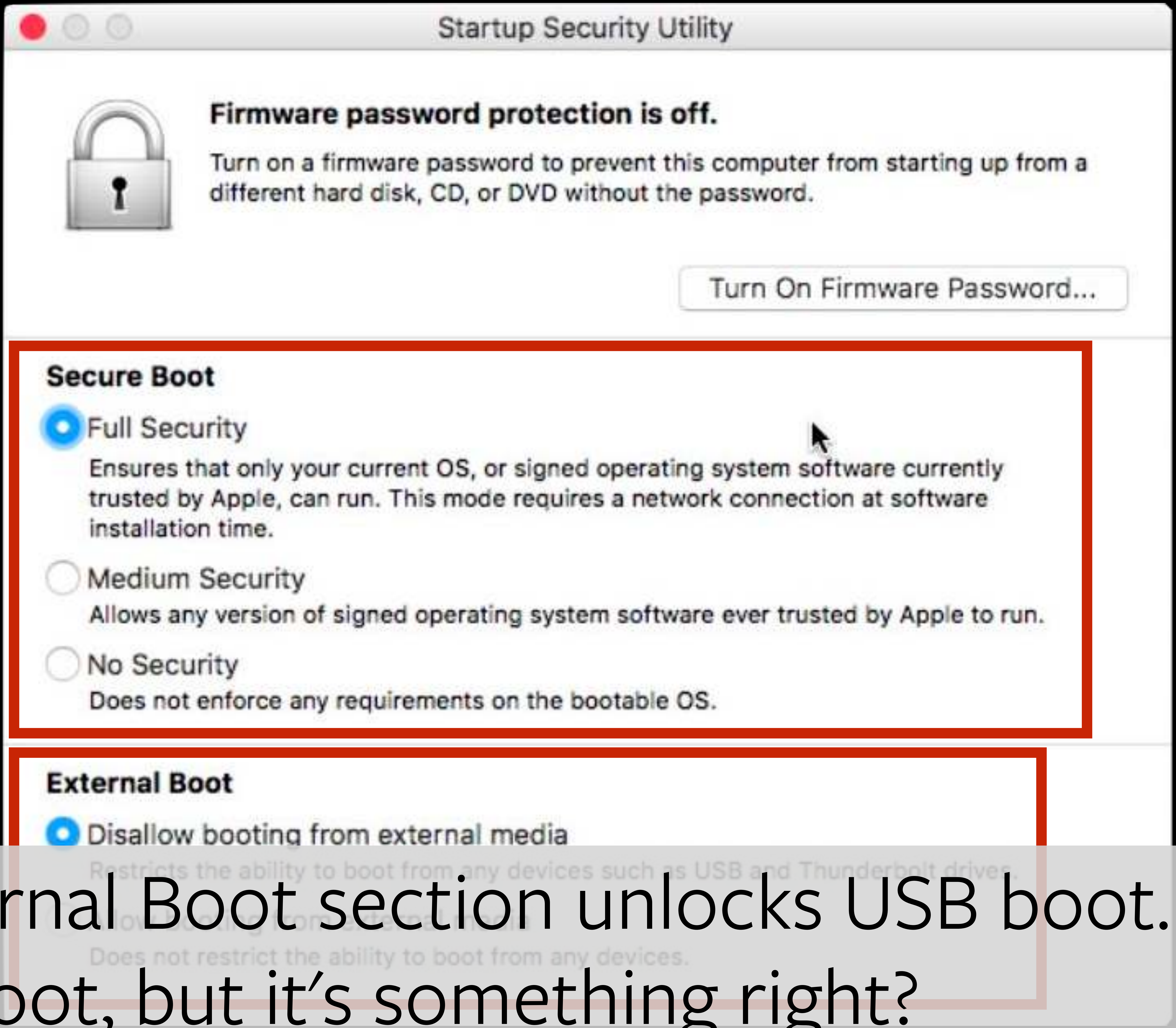
● Disallow booting from external media

Restricts the ability to boot from any devices such as USB and Thunderbolt drives.

○ Allow booting from external media

Does not restrict the ability to boot from any devices.

Yes, the Startup Security Utility does have configuration for disabling security. Maybe this stops personalization.



Startup Security Utility

Firmware password protection is off.

Turn on a firmware password to prevent this computer from starting up from a different hard disk, CD, or DVD without the password.

Secure Boot

● Full Security
Ensures that only your current OS, or signed operating system software currently trusted by Apple, can run. This mode requires a network connection at software installation time.

○ Medium Security
Allows any version of signed operating system software ever trusted by Apple to run.

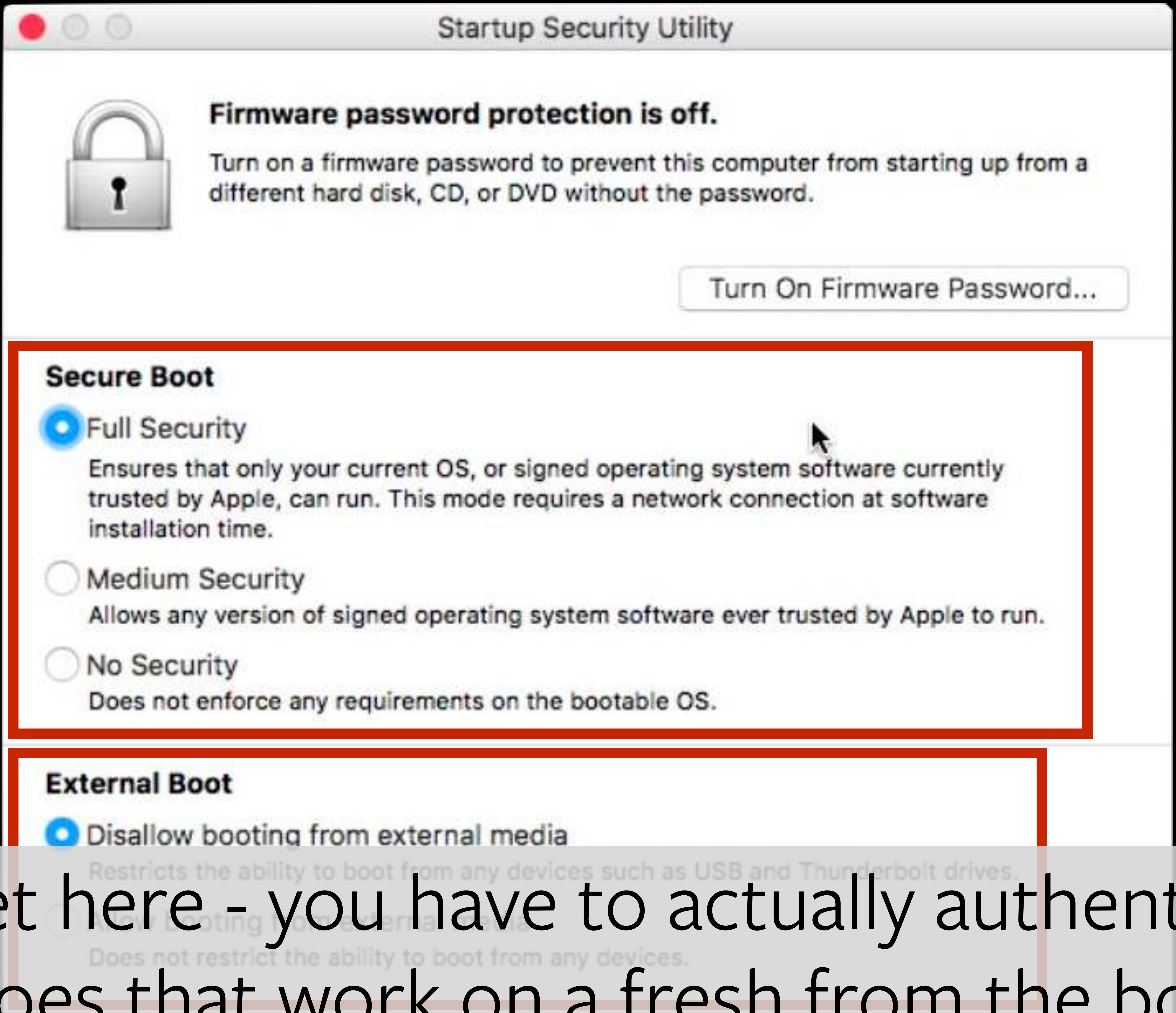
○ No Security
Does not enforce any requirements on the bootable OS.

External Boot

● Disallow booting from external media
Restricts the ability to boot from any devices such as USB and Thunderbolt drives

○ Allow booting from external media
Does not restrict the ability to boot from any devices.

And the External Boot section unlocks USB boot.
It’s not NetBoot, but it’s something right?



Startup Security Utility

Firmware password protection is off.

Turn on a firmware password to prevent this computer from starting up from a different hard disk, CD, or DVD without the password.

Secure Boot

● Full Security

Ensures that only your current OS, or signed operating system software currently trusted by Apple, can run. This mode requires a network connection at software installation time.

○ Medium Security

Allows any version of signed operating system software ever trusted by Apple to run.

○ No Security

Does not enforce any requirements on the bootable OS.

External Boot

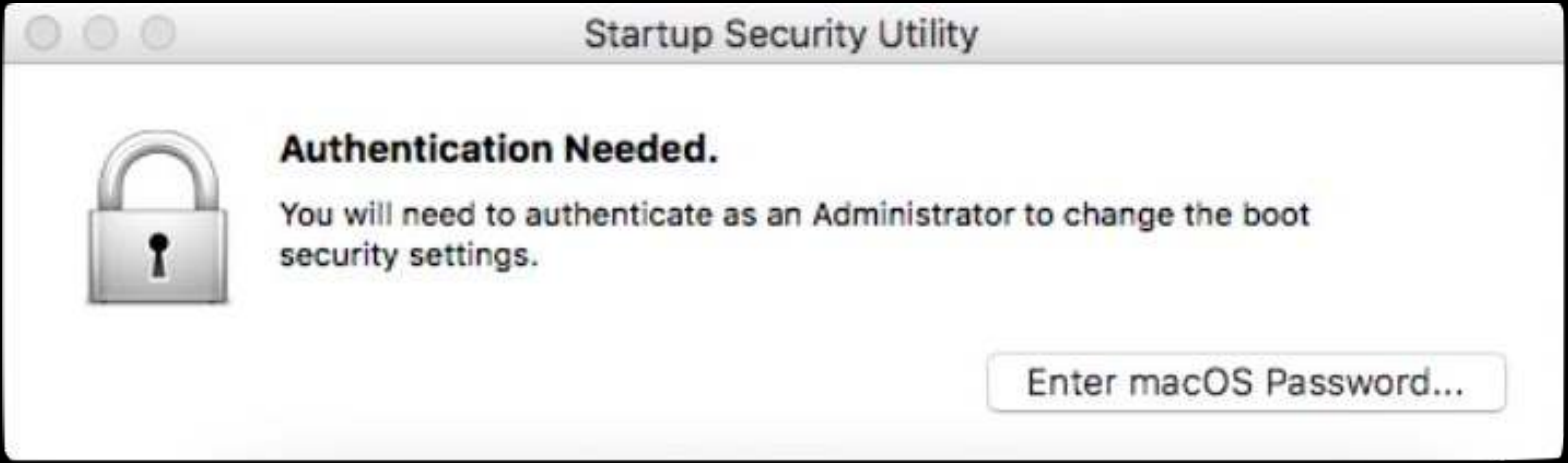
● Disallow booting from external media

Restricts the ability to boot from any devices such as USB and Thunderbolt drives.

○ Allow booting from external media

Does not restrict the ability to boot from any devices.

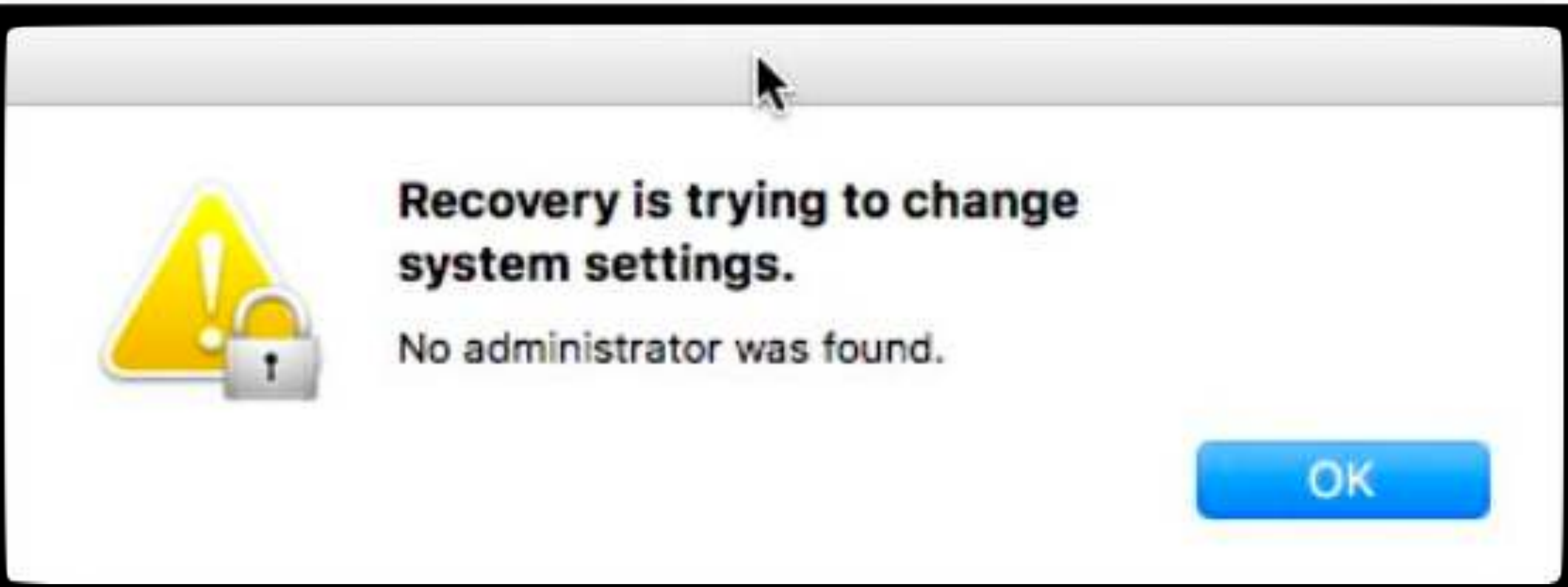
Oh. But to get here - you have to actually authenticate first … how does that work on a fresh from the box Mac?



Startup Security Utility

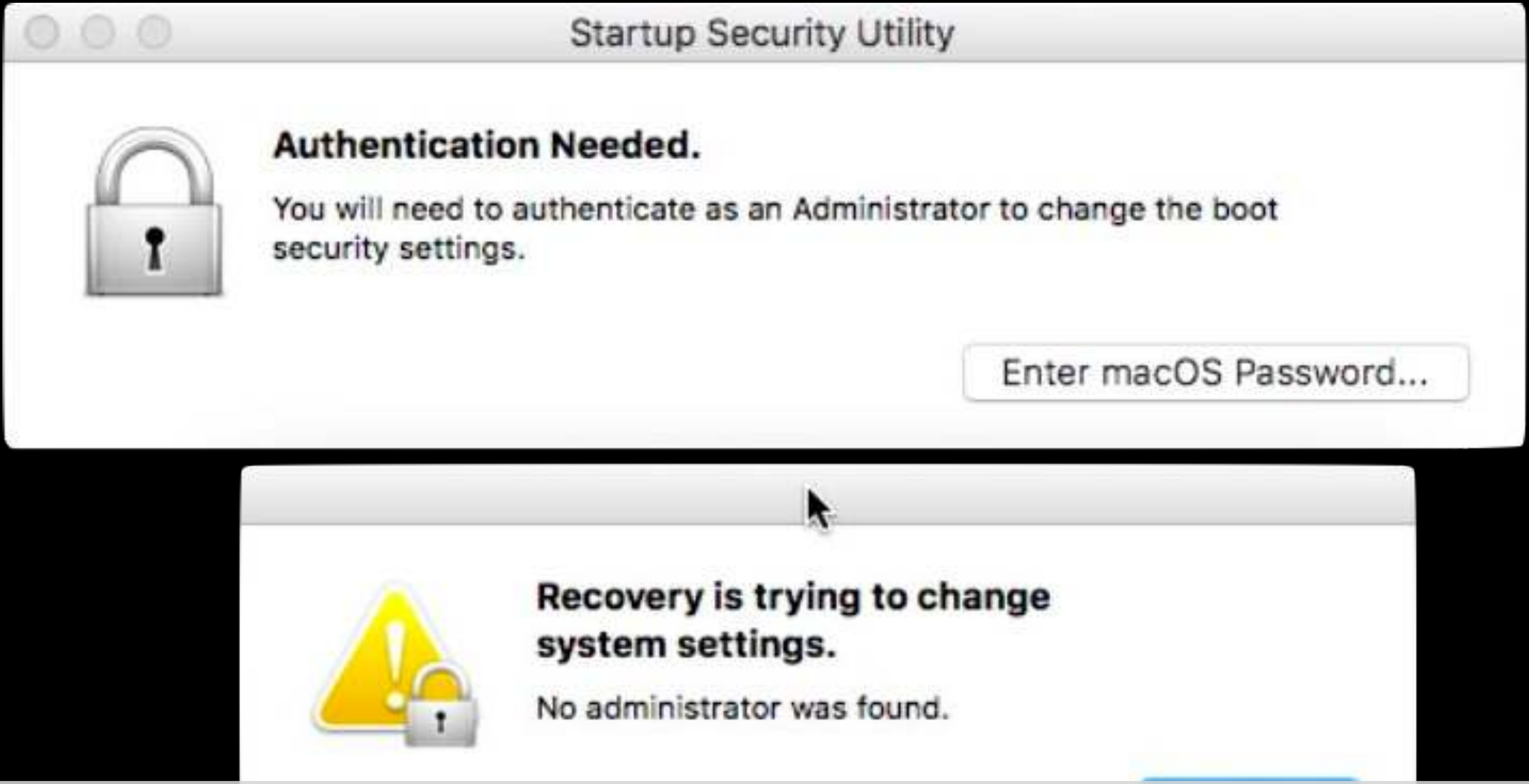
Authentication Needed.

You will need to authenticate as an Administrator to change the boot security settings.



Recovery is trying to change system settings.

No administrator was found.



Startup Security Utility

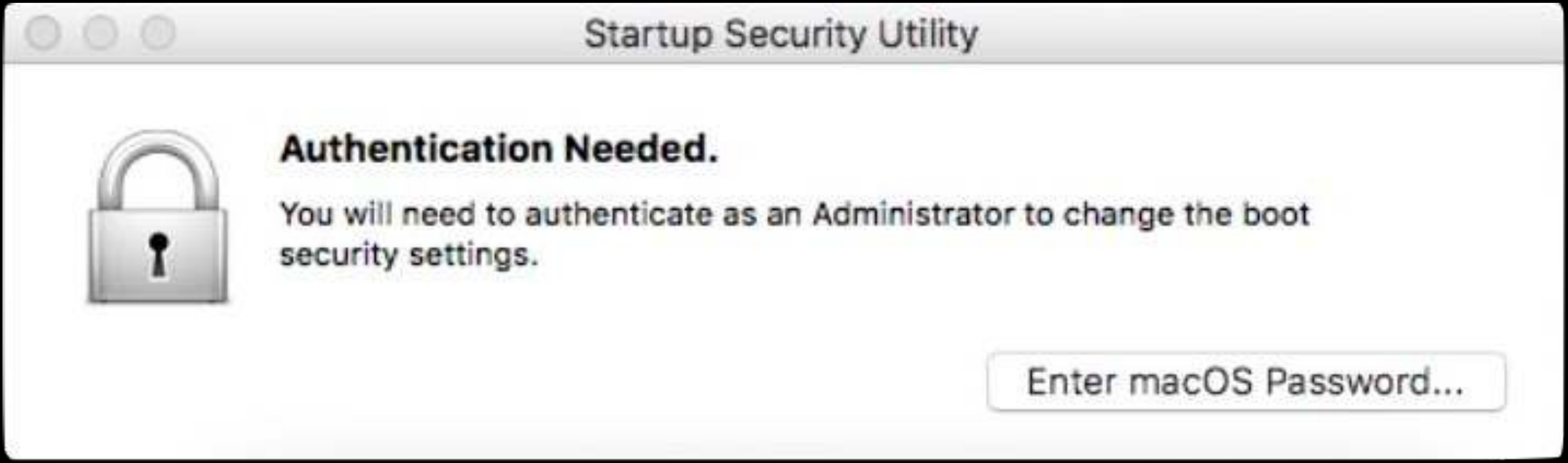
Authentication Needed.

You will need to authenticate as an Administrator to change the boot security settings.

Recovery is trying to change system settings.

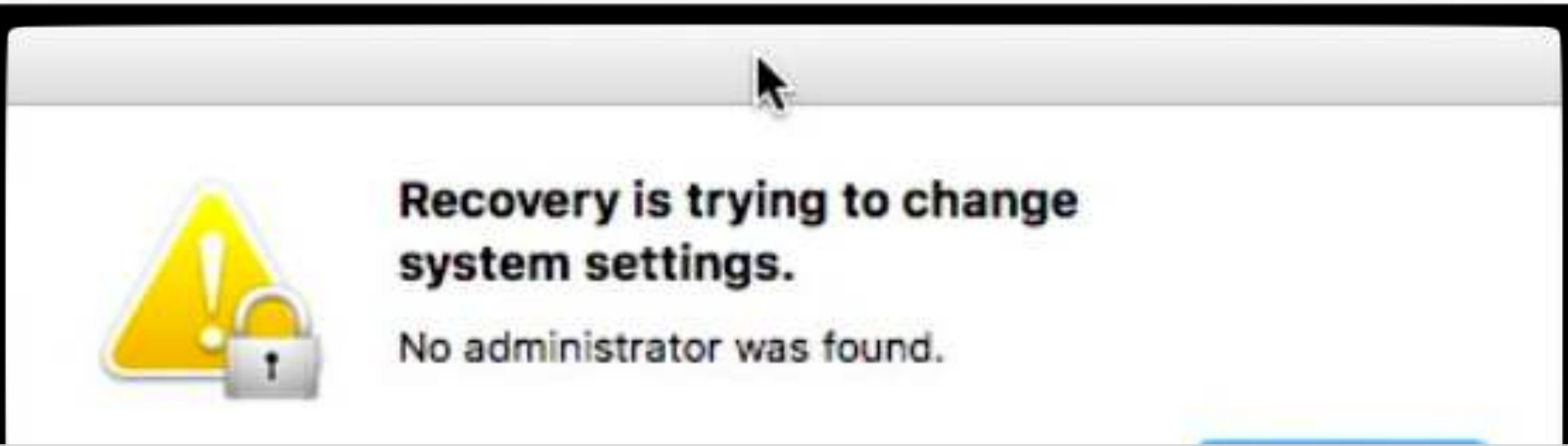
No administrator was found.

It doesn′t. Apple has pinned unlocking these configurations to SecureToken (crypto) macOS admin accounts.



Authentication Needed.

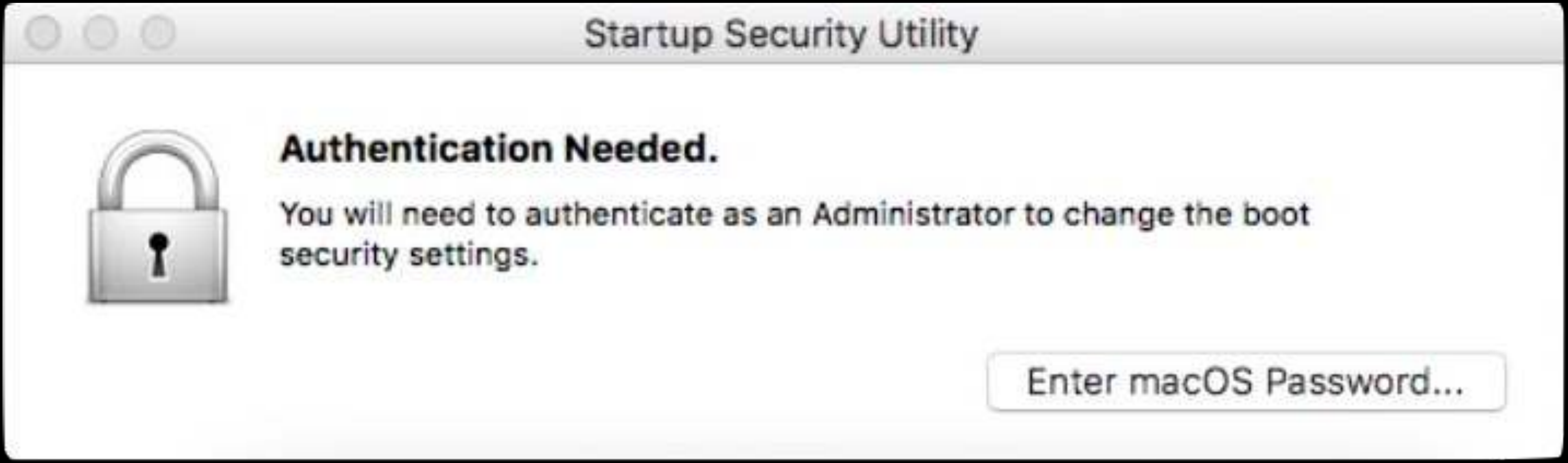
You will need to authenticate as an Administrator to change the boot security settings.



Recovery is trying to change system settings.

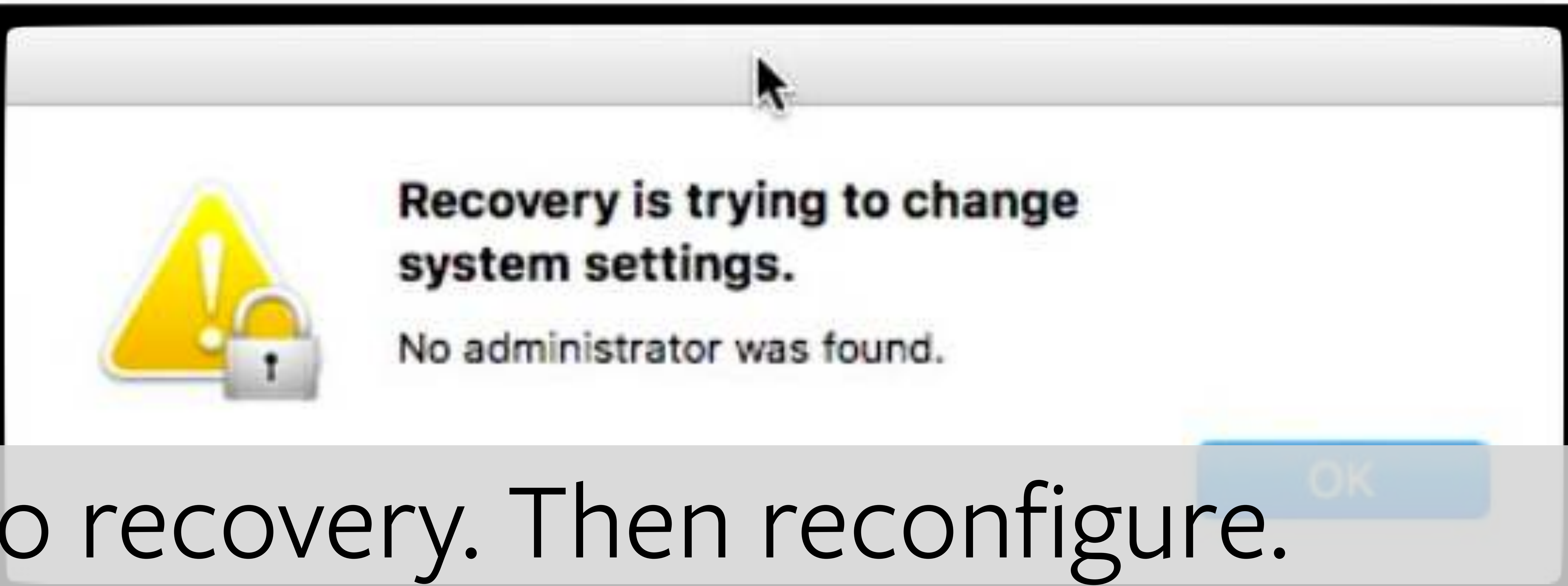
No administrator was found.

This means before you can change these settings, you have to go through macOS setup fully at least once.



Authentication Needed.

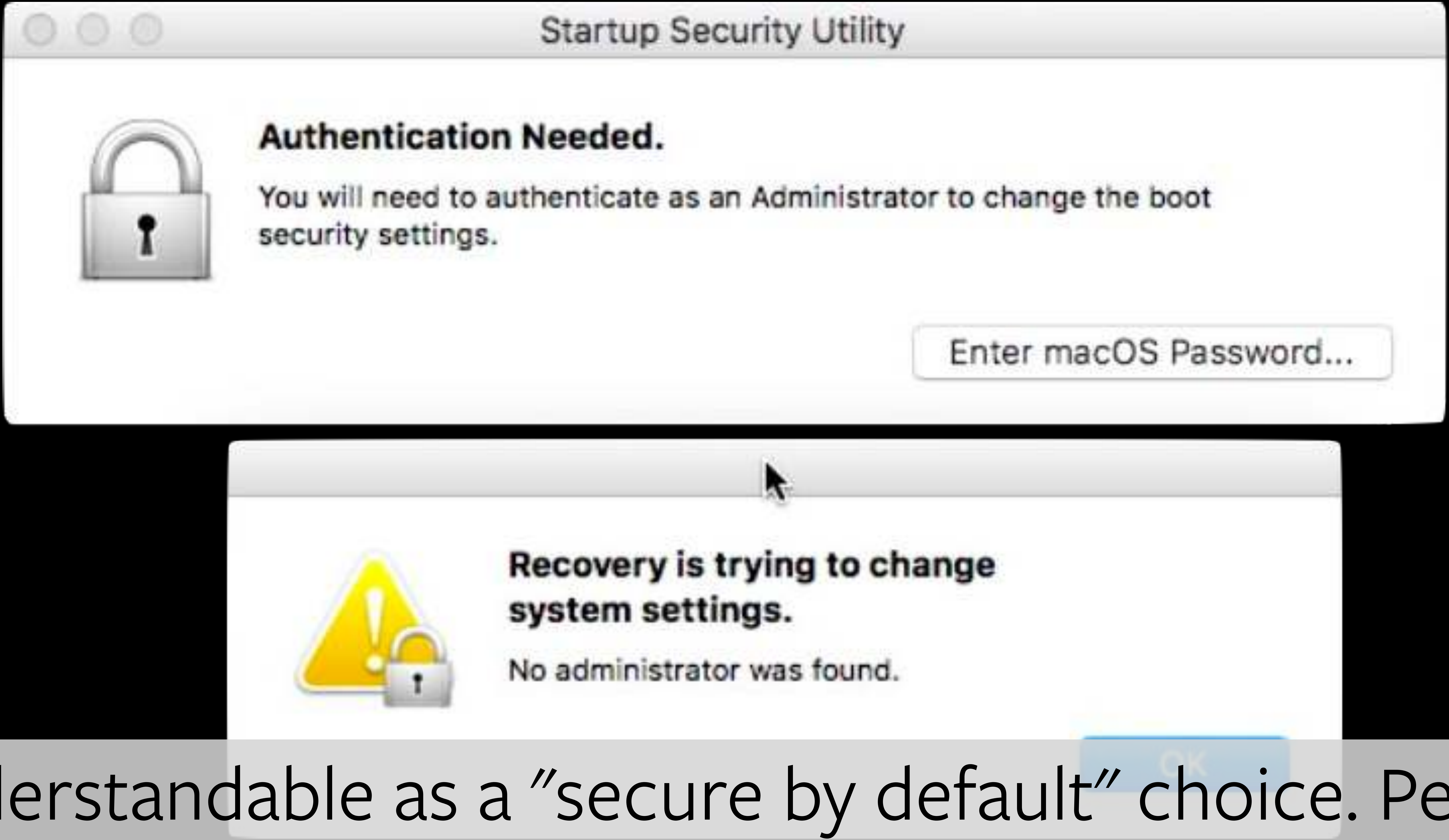
You will need to authenticate as an Administrator to change the boot security settings.



Recovery is trying to change system settings.

No administrator was found.

Then reboot to recovery. Then reconfigure.
Then you can wipe image it (like you intended to all along).



Startup Security Utility

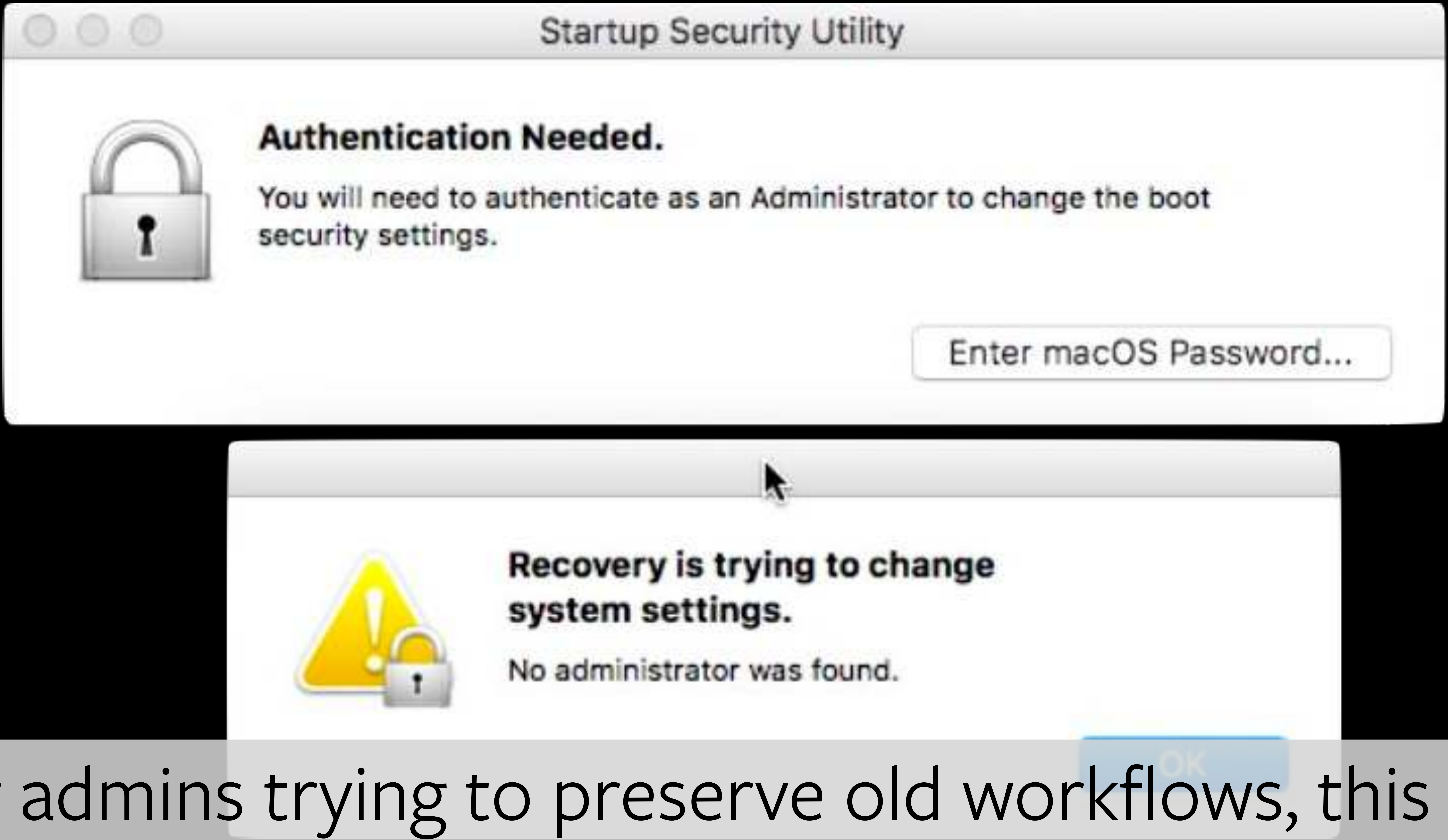
Authentication Needed.

You will need to authenticate as an Administrator to change the boot security settings.

Recovery is trying to change system settings.

No administrator was found.

It’s understandable as a "secure by default" choice. Personal devices benefit from this quite a bit.



Startup Security Utility

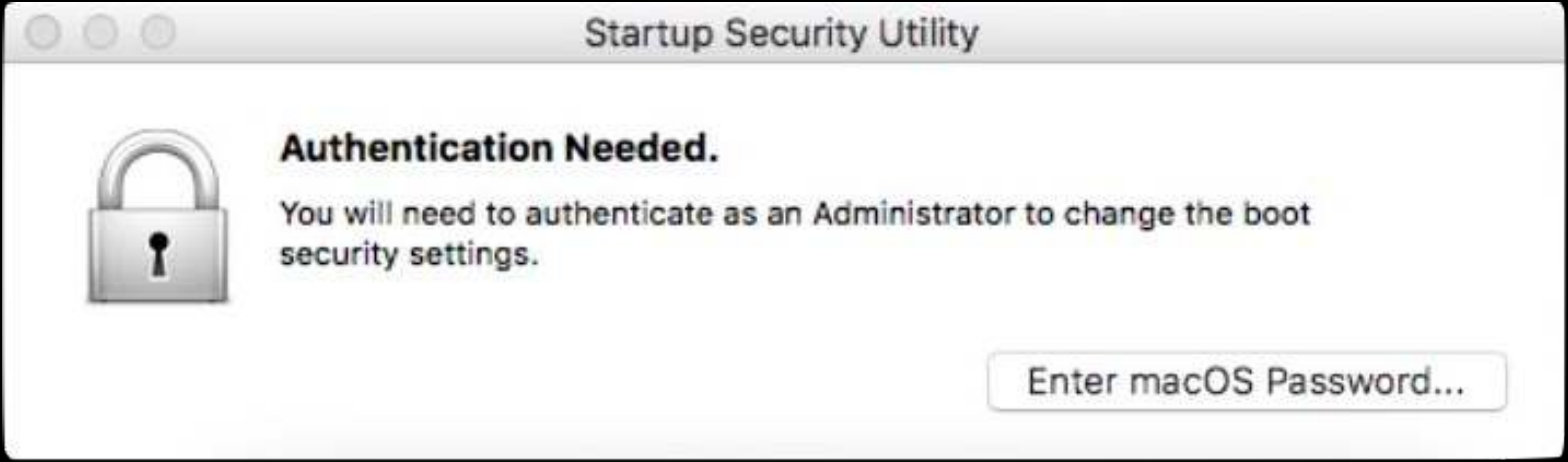
Authentication Needed.

You will need to authenticate as an Administrator to change the boot security settings.

Recovery is trying to change system settings.

No administrator was found.

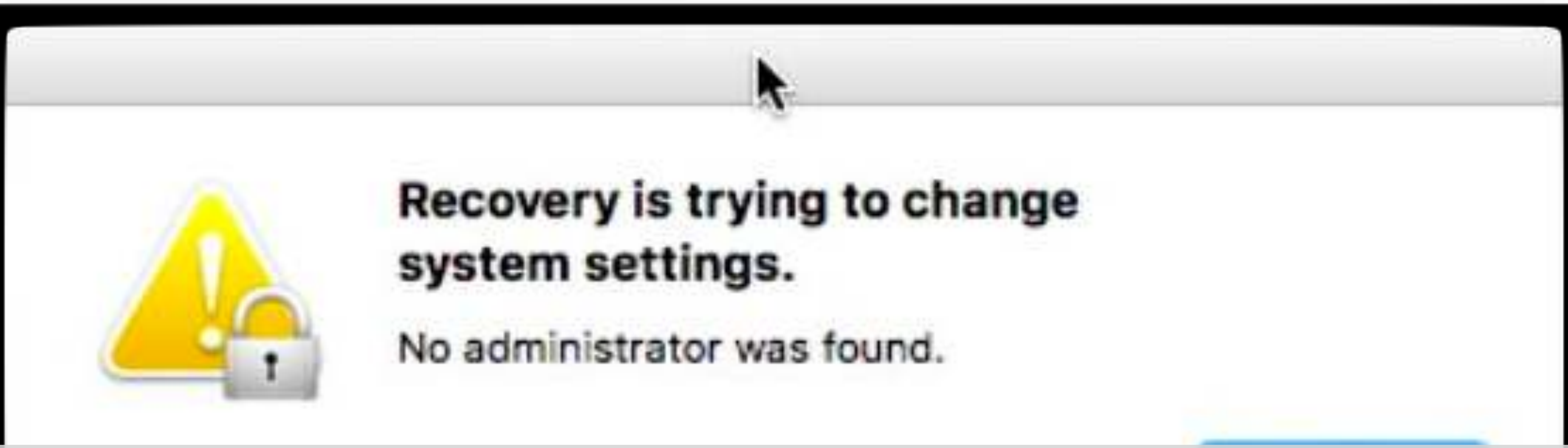
But for admins trying to preserve old workflows, this is just more change and steps on top of steps.



Startup Security Utility

Authentication Needed.

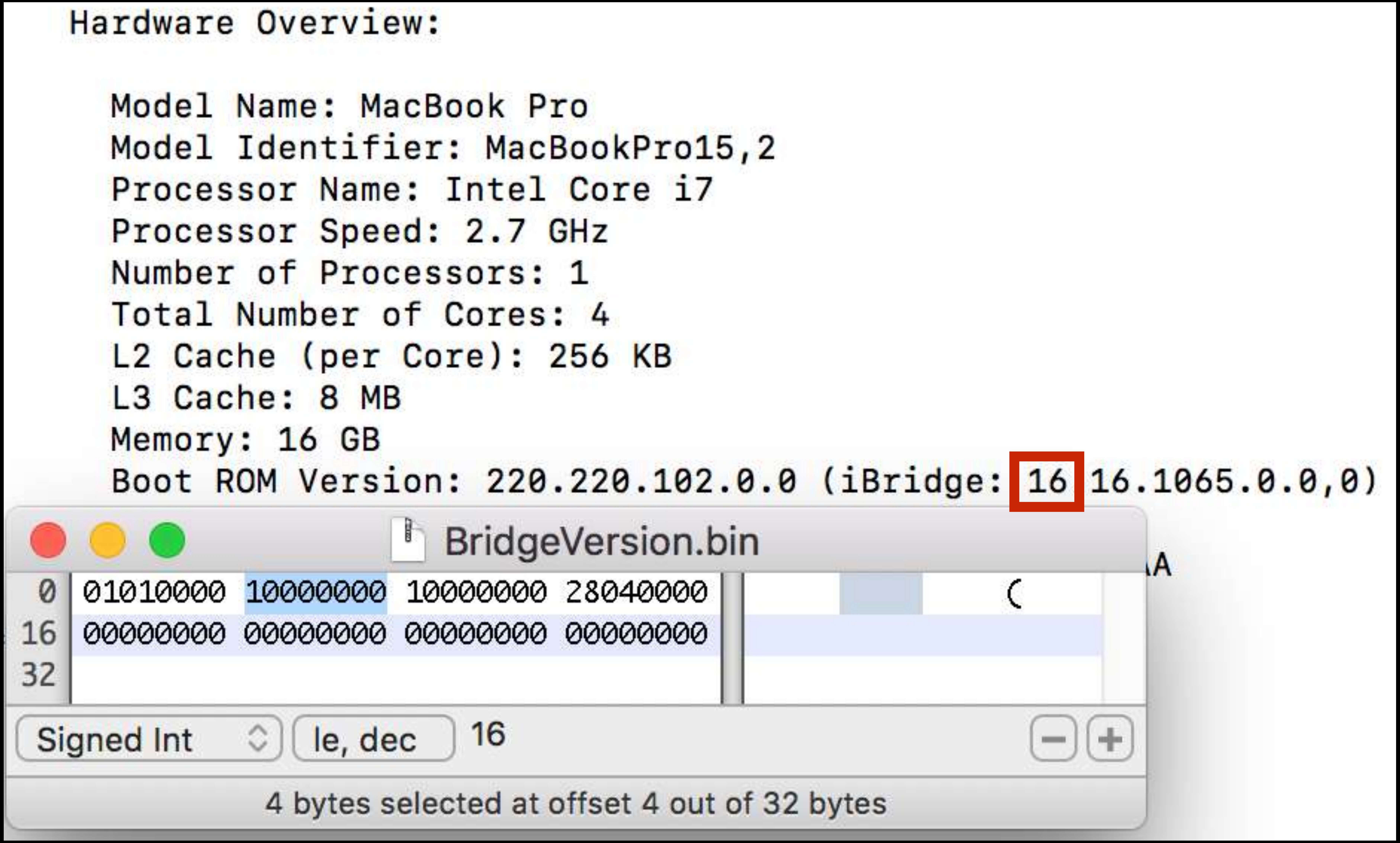
You will need to authenticate as an Administrator to change the boot security settings.



Recovery is trying to change system settings.

No administrator was found.

But hey - once you’re done with those steps, it’s mostly like an old Mac - right?



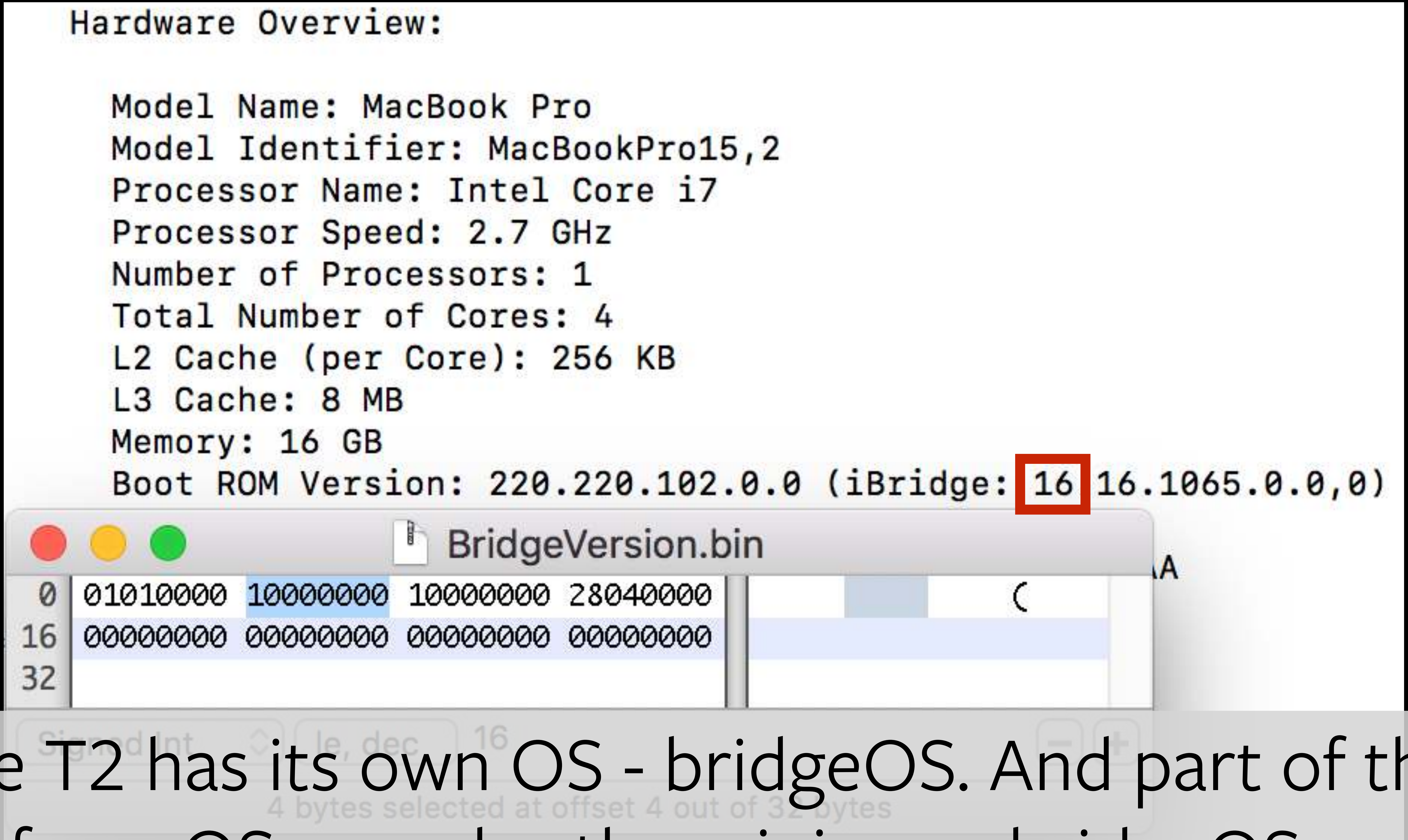
```
Hardware Overview:

  Model Name: MacBook Pro
  Model Identifier: MacBookPro15,2
  Processor Name: Intel Core i7
  Processor Speed: 2.7 GHz
  Number of Processors: 1
  Total Number of Cores: 4
  L2 Cache (per Core): 256 KB
  L3 Cache: 8 MB
  Memory: 16 GB
  Boot ROM Version: 220.220.102.0.0 (iBridge: 16 16.1065.0.0,0)
```

BridgeVersion.bin

```
 0  01010000 10000000 10000000 28040000      (
16  00000000 00000000 00000000 00000000
32
```

Signed Int le, dec 16

4 bytes selected at offset 4 out of 32 bytes



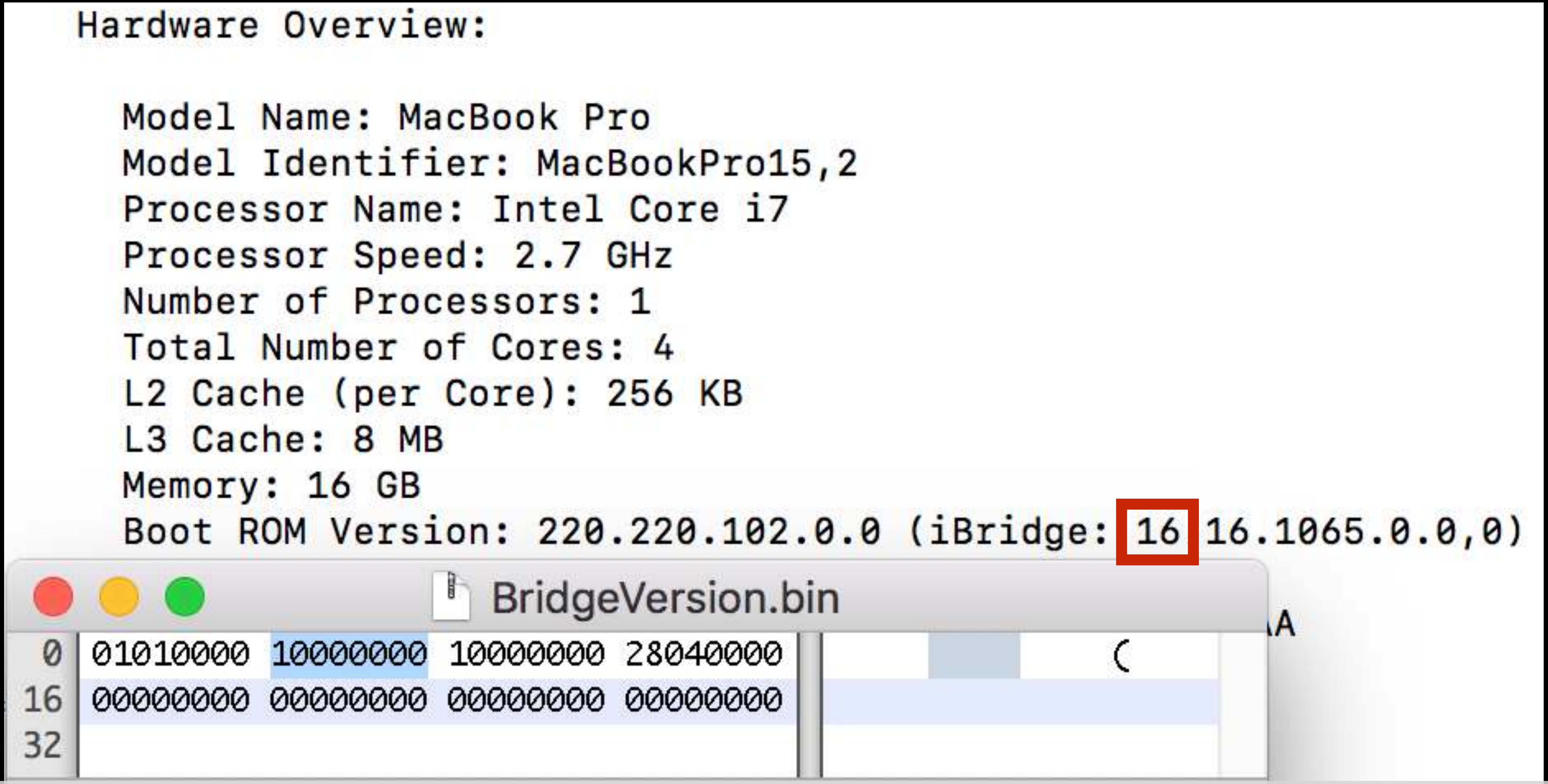
```
Hardware Overview:

  Model Name: MacBook Pro
  Model Identifier: MacBookPro15,2
  Processor Name: Intel Core i7
  Processor Speed: 2.7 GHz
  Number of Processors: 1
  Total Number of Cores: 4
  L2 Cache (per Core): 256 KB
  L3 Cache: 8 MB
  Memory: 16 GB
  Boot ROM Version: 220.220.102.0.0 (iBridge: 16 16.1065.0.0,0)
```

BridgeVersion.bin

```
 0  01010000 10000000 10000000 28040000          (
16  00000000 00000000 00000000 00000000
32
```

4 bytes selected at offset 4 out of 32 bytes

Well the T2 has its own OS - bridgeOS. And part of the install of macOS encodes the minimum bridgeOS version.

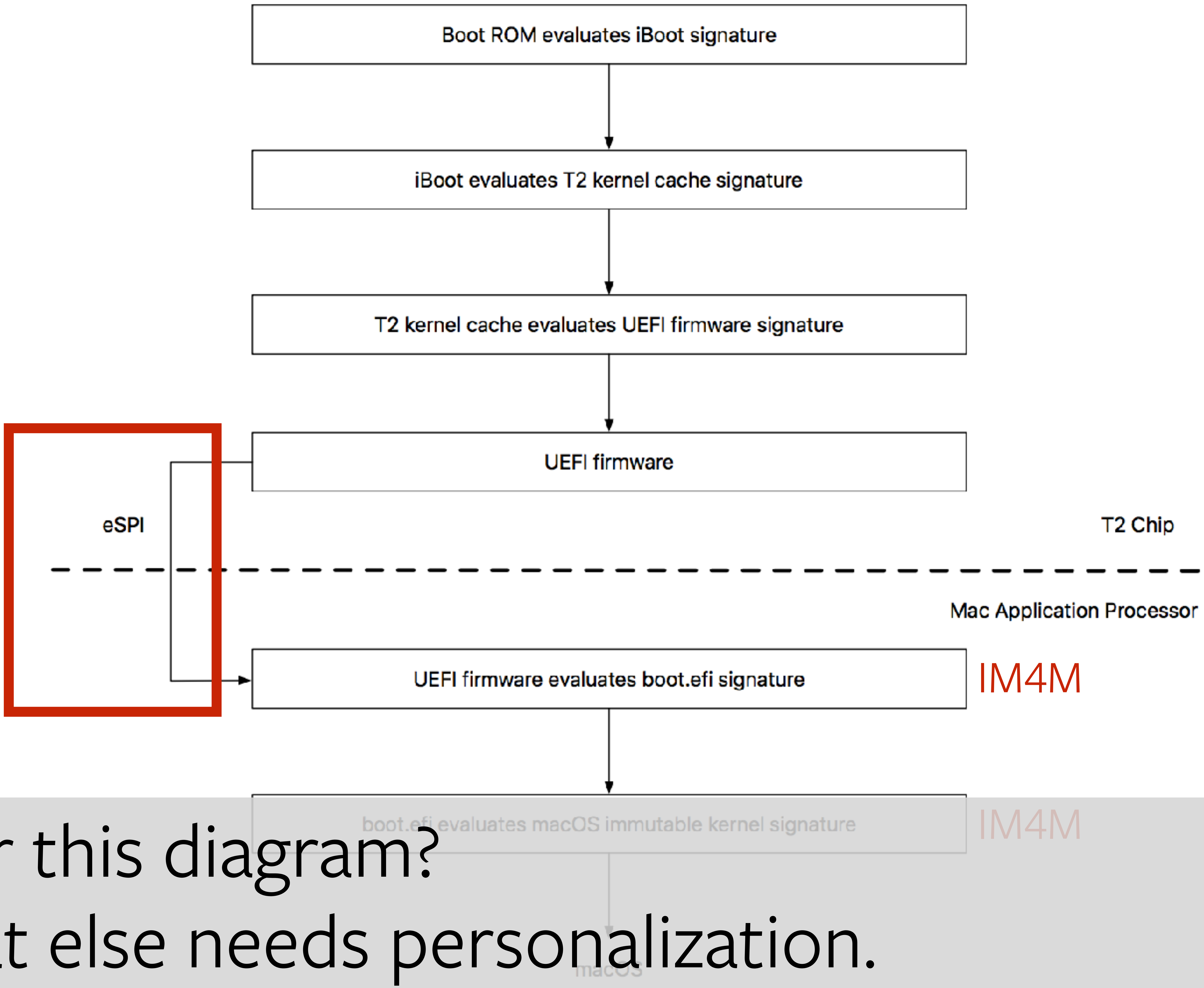


```
Hardware Overview:

  Model Name: MacBook Pro
  Model Identifier: MacBookPro15,2
  Processor Name: Intel Core i7
  Processor Speed: 2.7 GHz
  Number of Processors: 1
  Total Number of Cores: 4
  L2 Cache (per Core): 256 KB
  L3 Cache: 8 MB
  Memory: 16 GB
  Boot ROM Version: 220.220.102.0.0 (iBridge: 16 16.1065.0.0,0)
```

BridgeVersion.bin

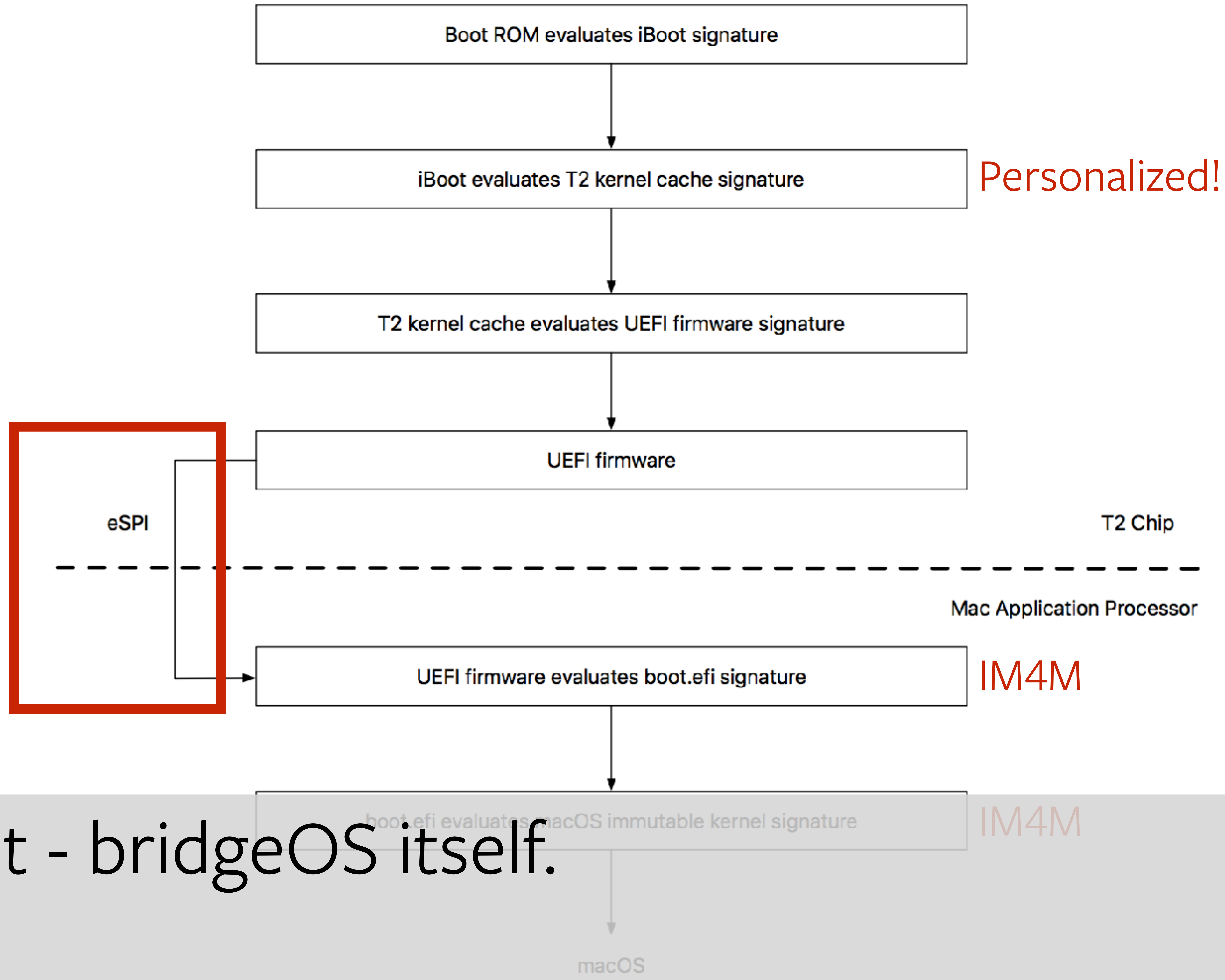
```
 0  01010000 10000000 10000000 28040000            (
16  00000000 00000000 00000000 00000000
32
```

If macOS starts up, the Mac will check and see if it's on the right version of bridgeOS. If it's not - it'll want to update.


Boot ROM evaluates iBoot signature
iBoot evaluates T2 kernel cache signature
T2 kernel cache evaluates UEFI firmware signature
UEFI firmware
eSPI
T2 Chip
Mac Application Processor
UEFI firmware evaluates boot.efi signature
IM4M
boot.efi evaluates macOS immutable kernel signature
IM4M
macOS

Remember this diagram?

Guess what else needs personalization.



Boot ROM evaluates iBoot signature

iBoot evaluates T2 kernel cache signature

Personalized!

T2 kernel cache evaluates UEFI firmware signature

UEFI firmware

eSPI

T2 Chip

Mac Application Processor

UEFI firmware evaluates boot.efi signature

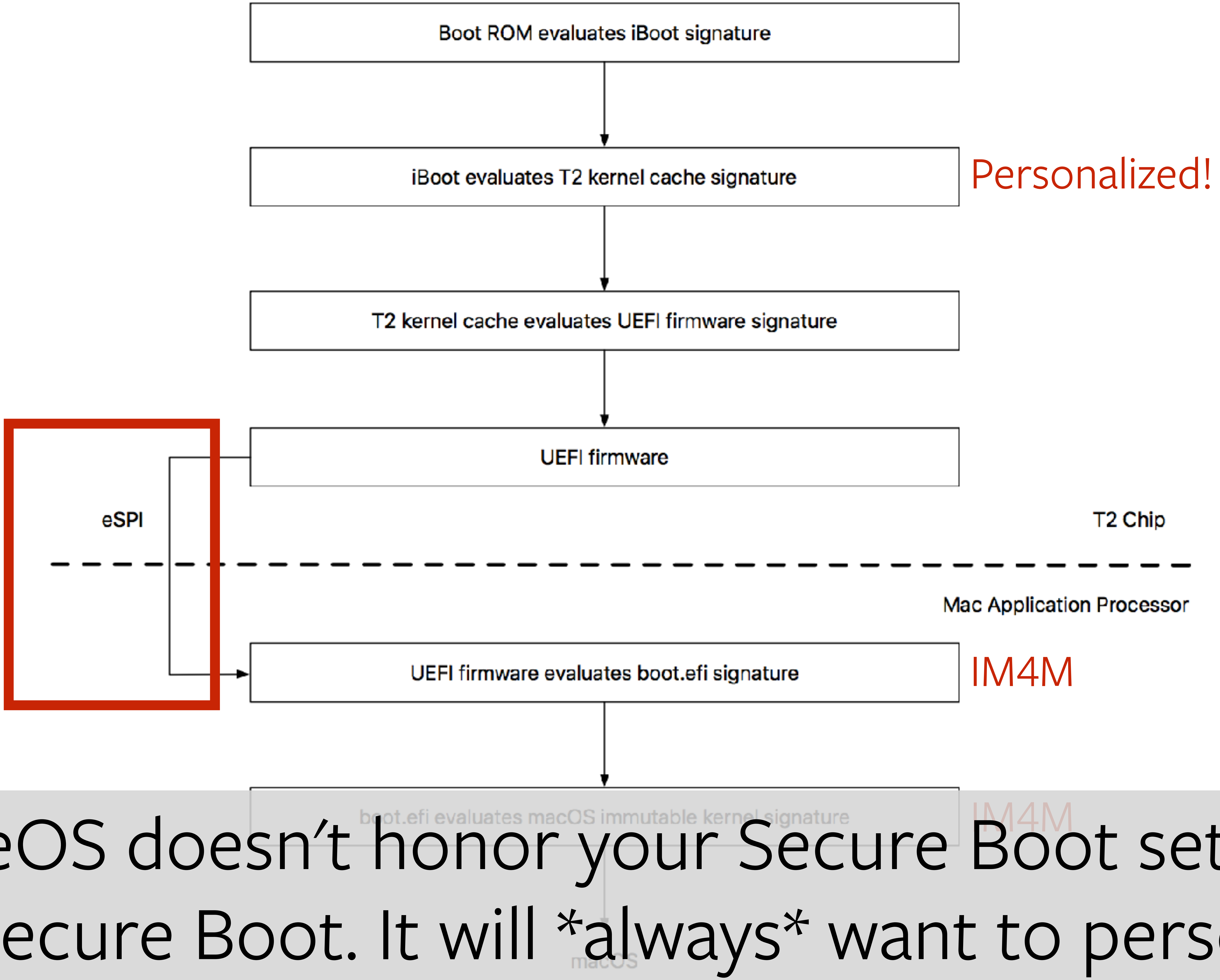
IM4M

boot.efi evaluates macOS immutable kernel signature

IM4M

macOS

That’s right - bridgeOS itself.



Boot ROM evaluates iBoot signature

iBoot evaluates T2 kernel cache signature

Personalized!

T2 kernel cache evaluates UEFI firmware signature

UEFI firmware

eSPI

T2 Chip

Mac Application Processor

UEFI firmware evaluates boot.efi signature

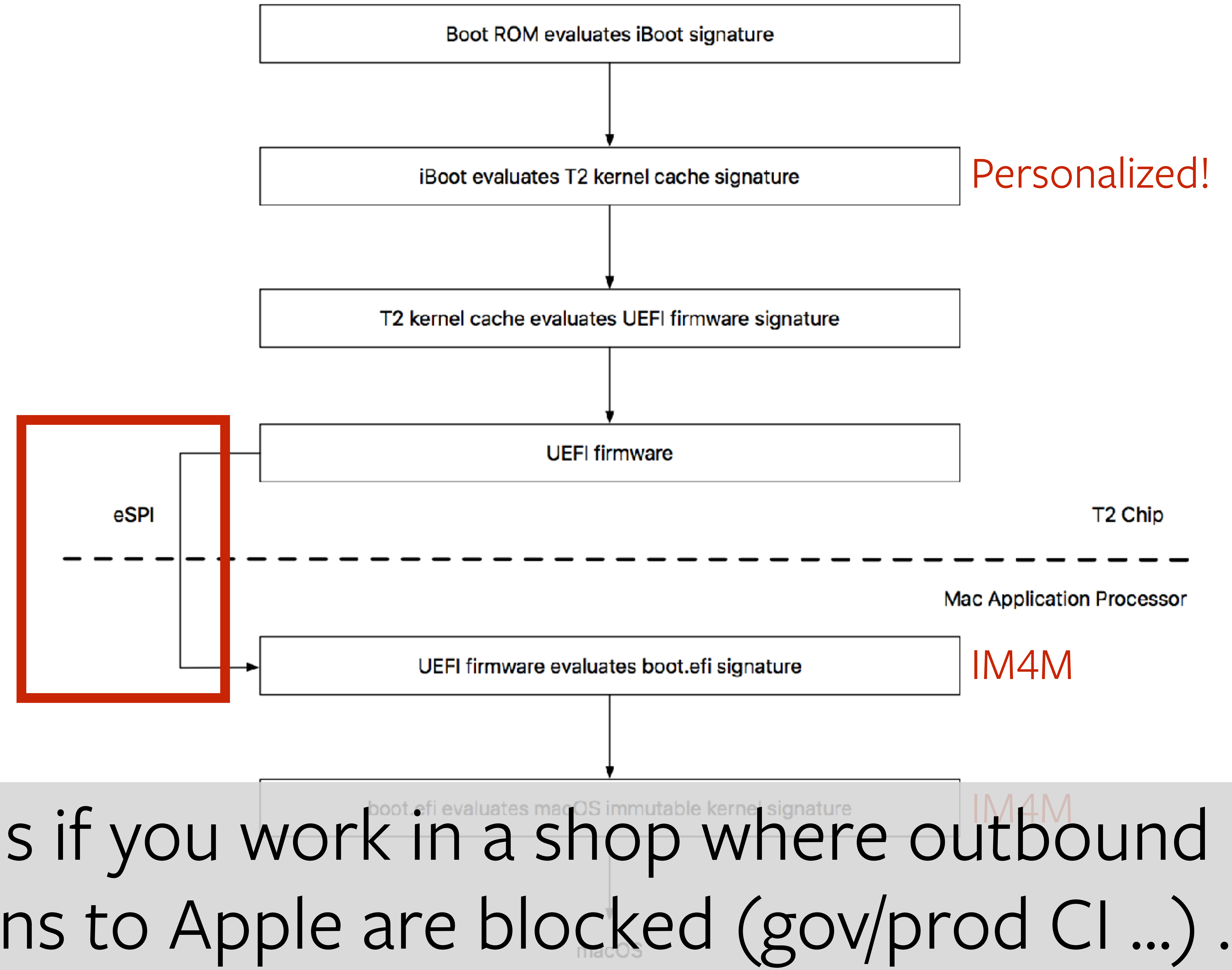
IM4M

boot.efi evaluates macOS immutable kernel signature

IM4M

macOS

And bridgeOS doesn’t honor your Secure Boot settings for disabling Secure Boot. It will *always* want to personalize.



Boot ROM evaluates iBoot signature

iBoot evaluates T2 kernel cache signature

Personalized!

T2 kernel cache evaluates UEFI firmware signature

UEFI firmware

eSPI

T2 Chip

Mac Application Processor

UEFI firmware evaluates boot.efi signature

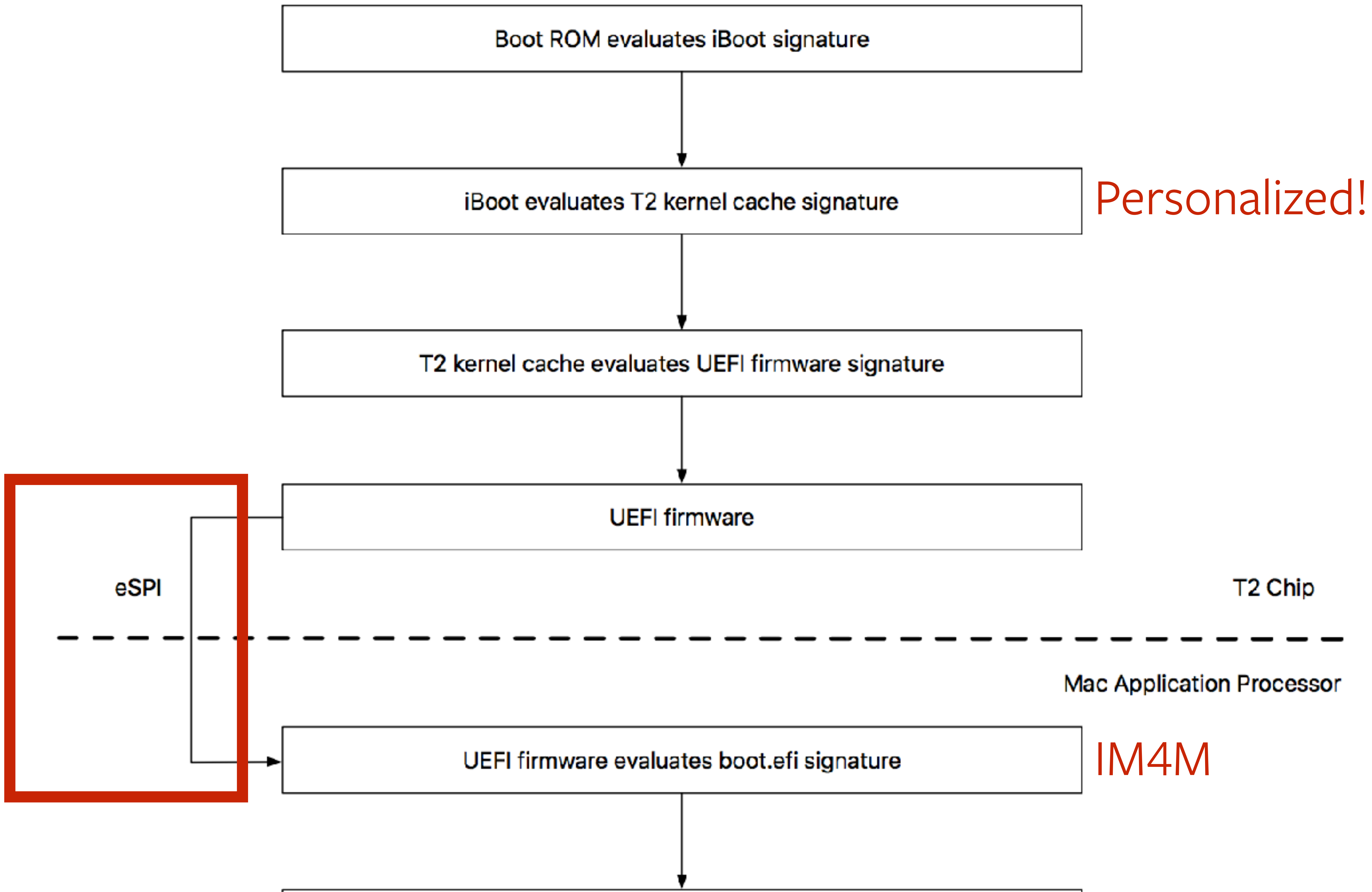
IM4M

boot.efi evaluates macOS immutable kernel signature

IM4M

macOS

This means if you work in a shop where outbound network connections to Apple are blocked (gov/prod CI …) …



Boot ROM evaluates iBoot signature

iBoot evaluates T2 kernel cache signature

Personalized!

T2 kernel cache evaluates UEFI firmware signature

UEFI firmware

UEFI firmware evaluates boot.efi signature

IM4M

boot.efi evaluates macOS immutable kernel signature

IM4M

macOS

… your new Macs will likely never be able to update. What is the better security: Blocking Apple? Or not updating?

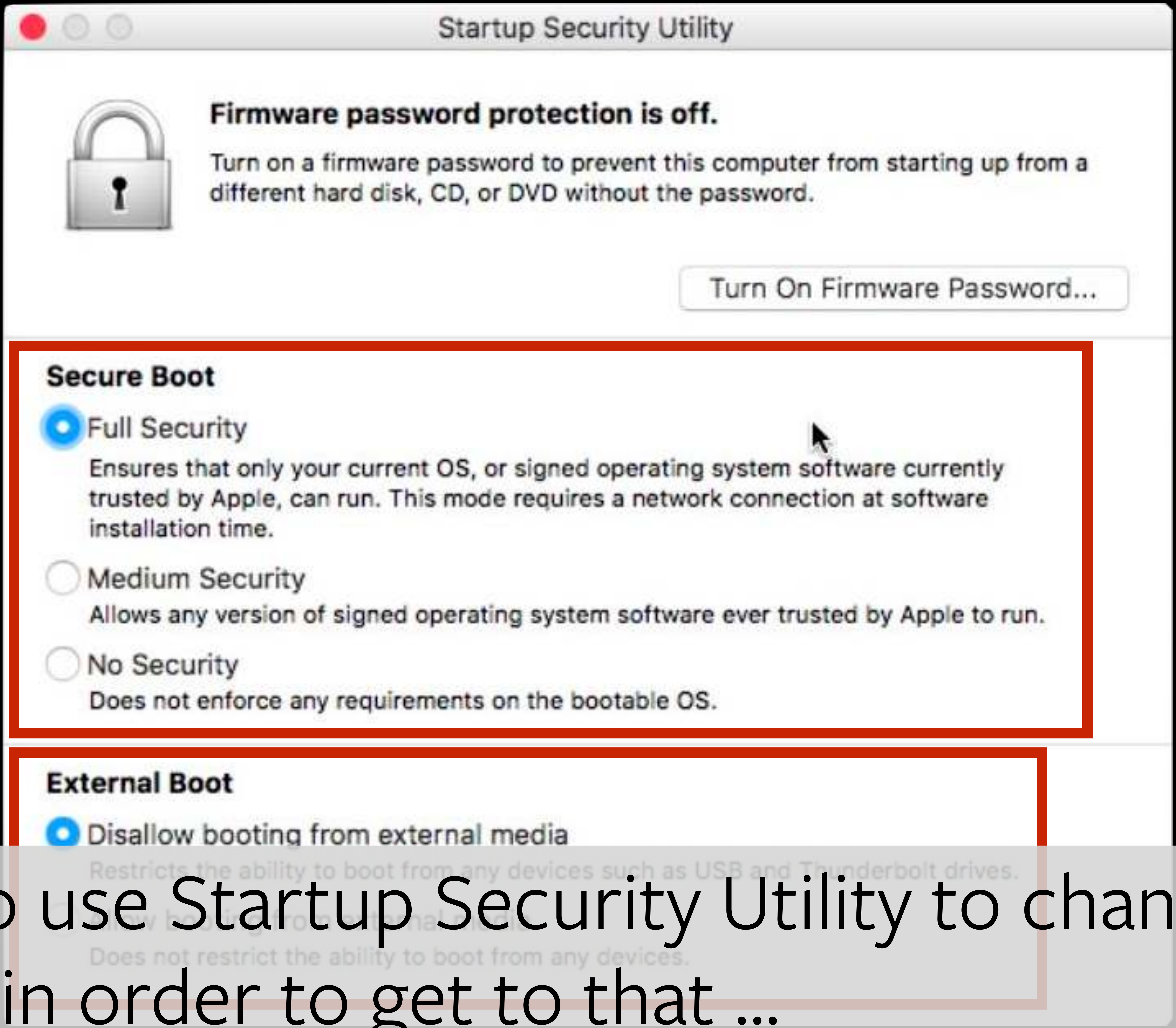
I’m sure as heck glad I don’t have to provision these!

I can hear a lot of you in the audience saying ”woo, I’m in security/forensics - glad I don’t have to deal with this!”

… I have presents for you!

A lot of forensics tool kits out there right now involve boot media to capture a ″clean″ image of the machine′s drive.

If you'll remember - by default, Secure Boot Macs don't support this.



Startup Security Utility

Firmware password protection is off.

Turn on a firmware password to prevent this computer from starting up from a different hard disk, CD, or DVD without the password.

Secure Boot

● Full Security
Ensures that only your current OS, or signed operating system software currently trusted by Apple, can run. This mode requires a network connection at software installation time.

○ Medium Security
Allows any version of signed operating system software ever trusted by Apple to run.

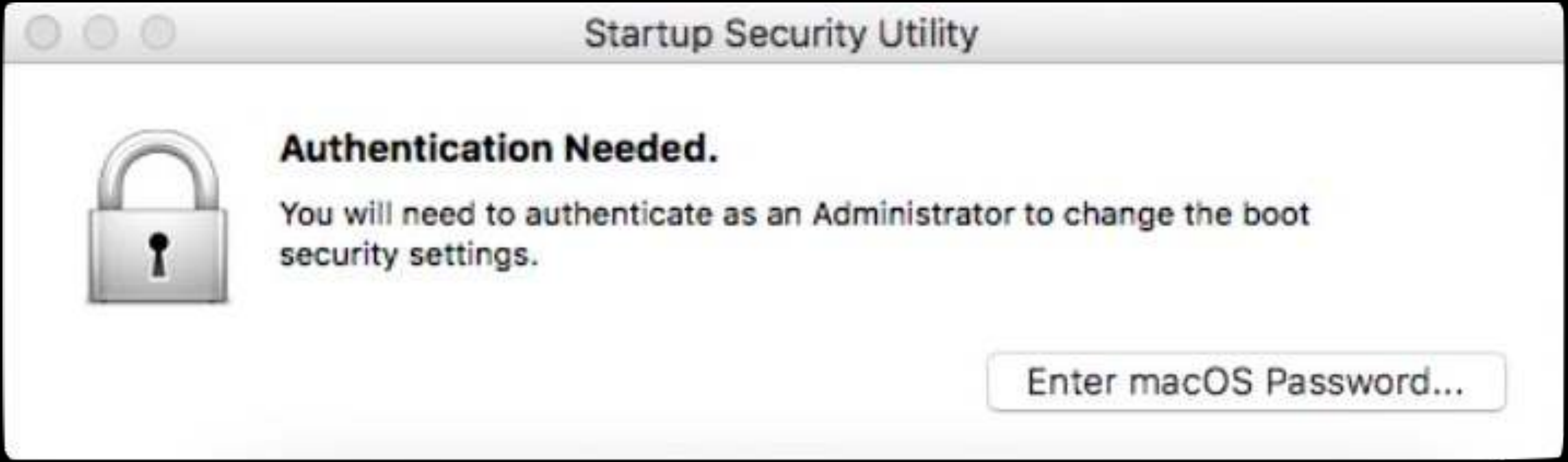
○ No Security
Does not enforce any requirements on the bootable OS.

External Boot

● Disallow booting from external media
Restricts the ability to boot from any devices such as USB and Thunderbolt drives.

○ Allow booting from external media
Does not restrict the ability to boot from any devices.

You'll need to use Startup Security Utility to change those settings. But in order to get to that …

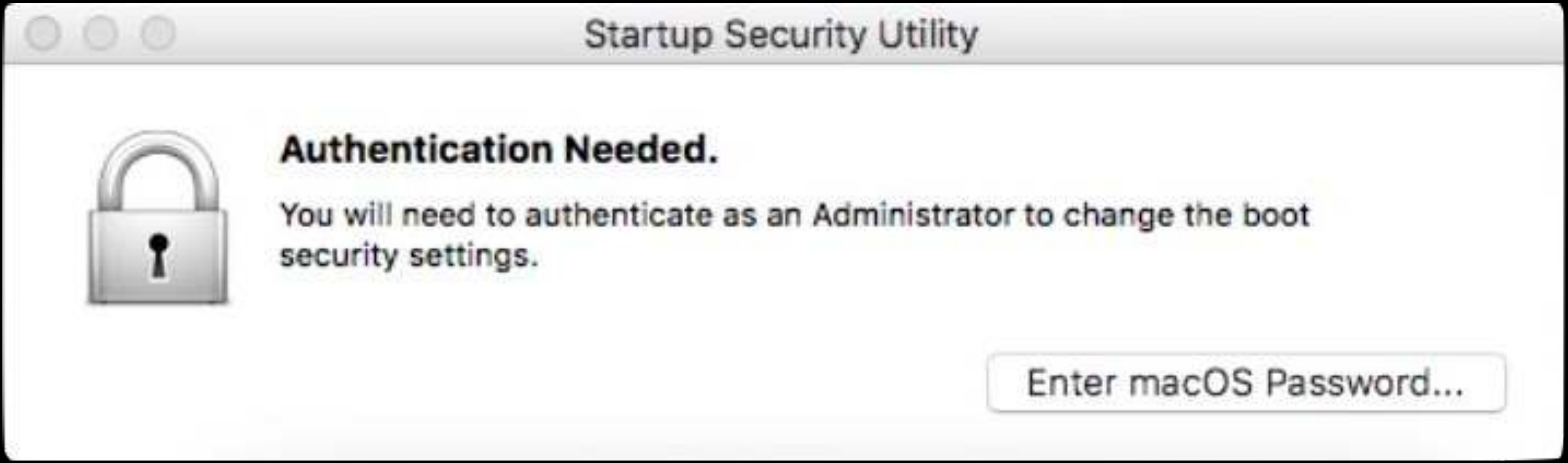


Startup Security Utility

Authentication Needed.

You will need to authenticate as an Administrator to change the boot security settings.

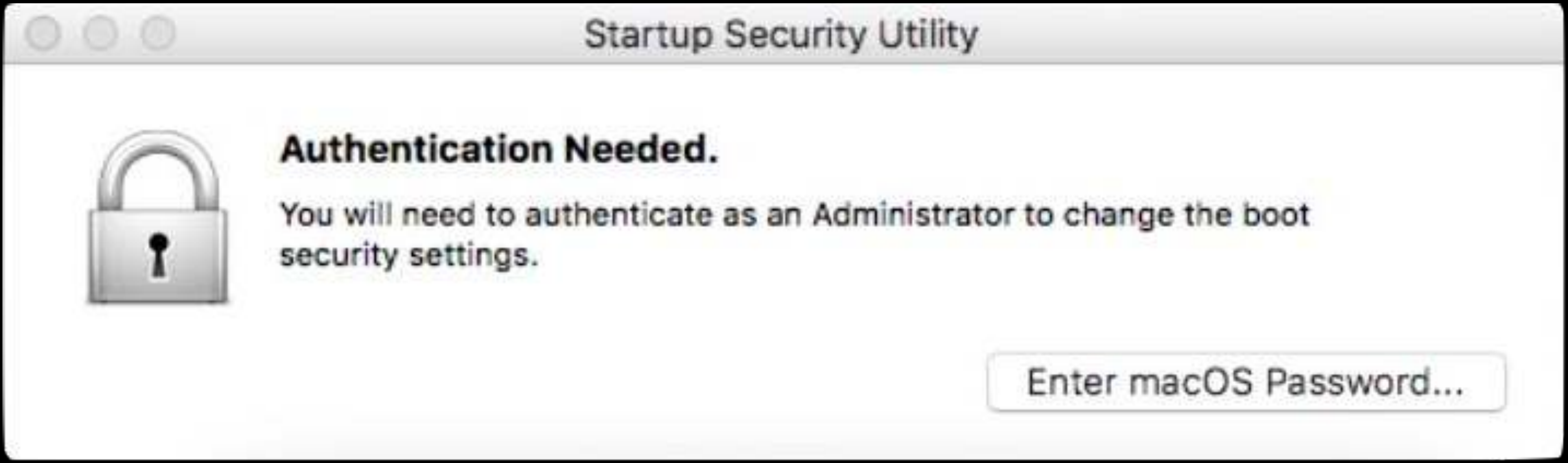
First you'll need to authenticate. And not just any login, it needs to be a SecureToken admin.



Authentication Needed.

You will need to authenticate as an Administrator to change the boot security settings.

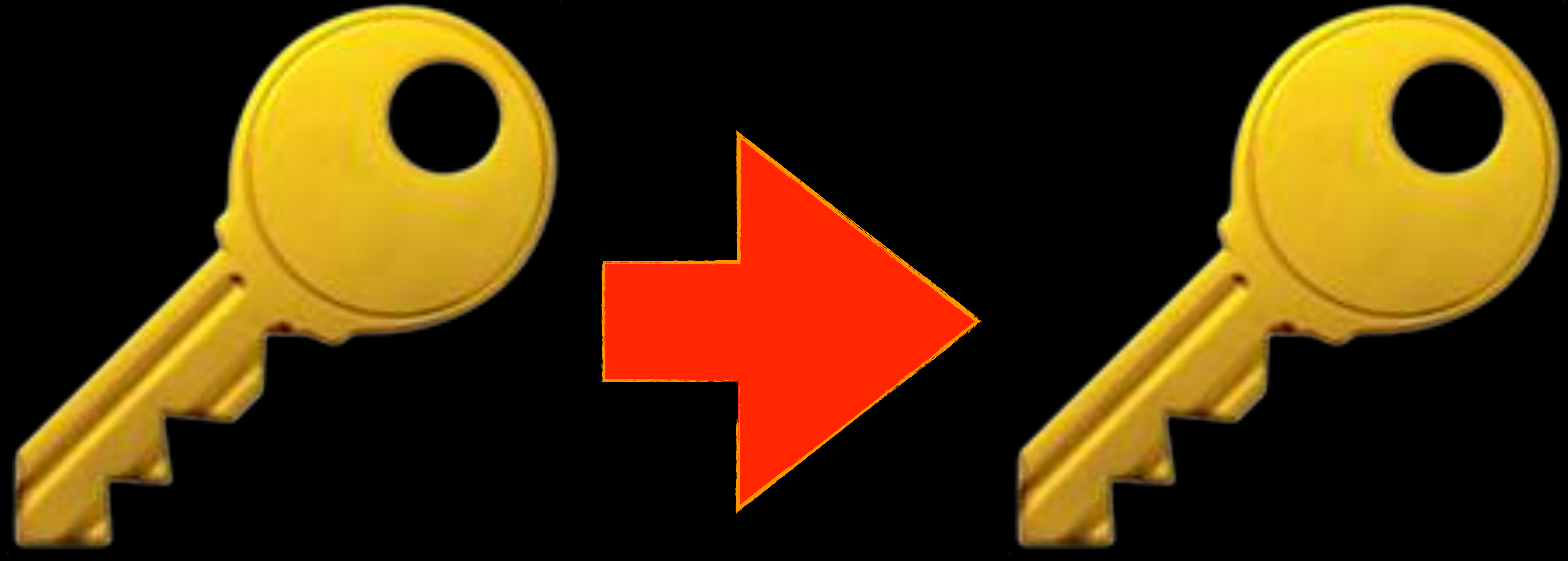
By default on macOS, that's usually the first person to log into a device - usually the main user of the device.



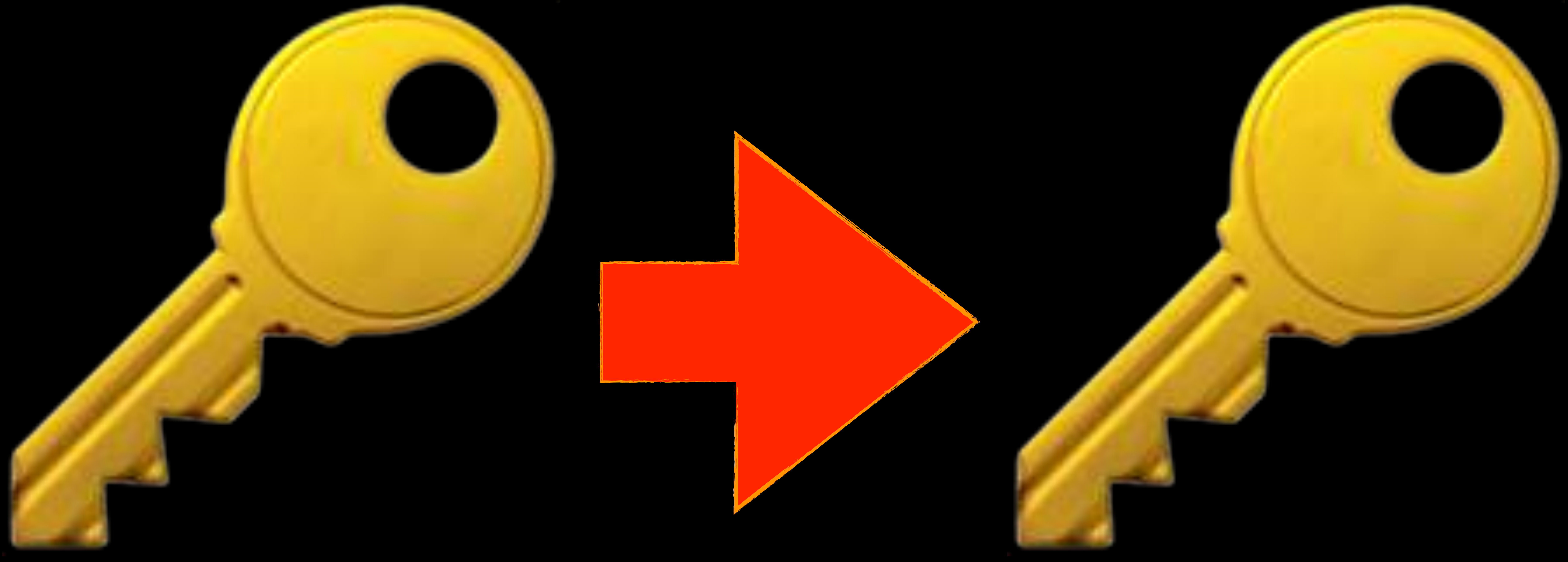
Authentication Needed.

You will need to authenticate as an Administrator to change the boot security settings.

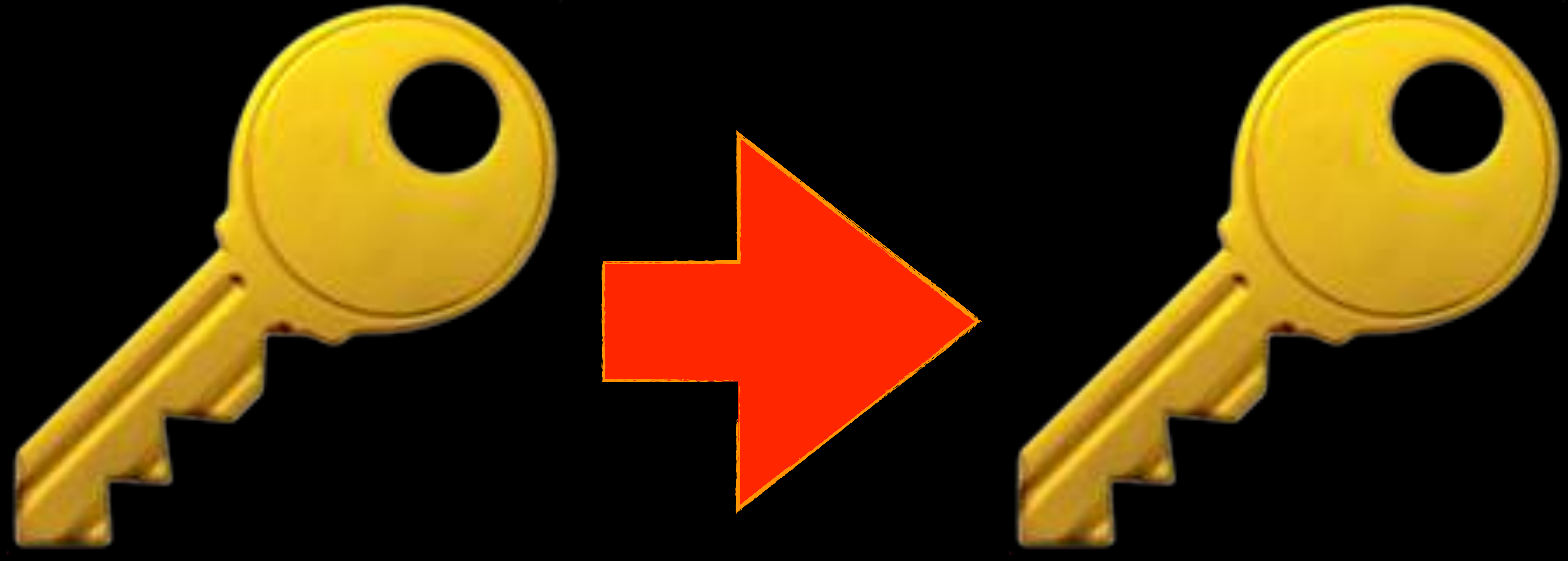
Do you know their password? Will asking them for it be possible for the investigation you're trying to do?



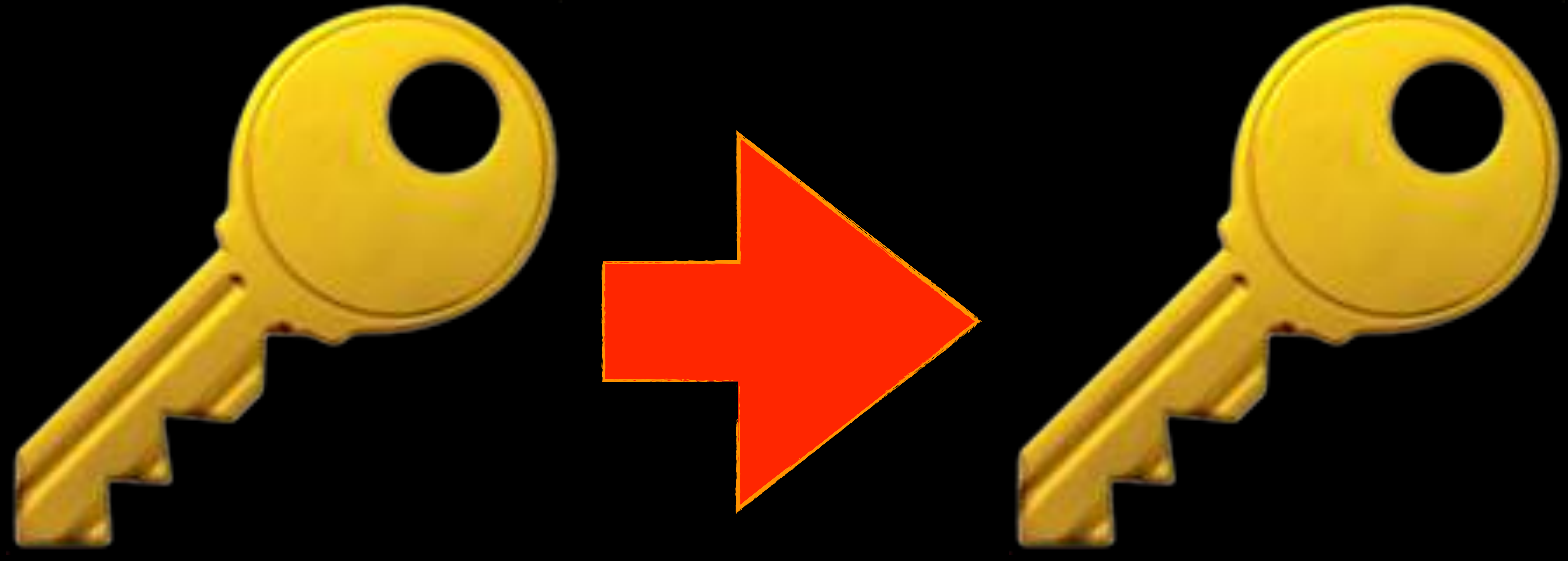
SecureToken users are special in macOS. You can’t just use root to make a new one. You need another one’s password.



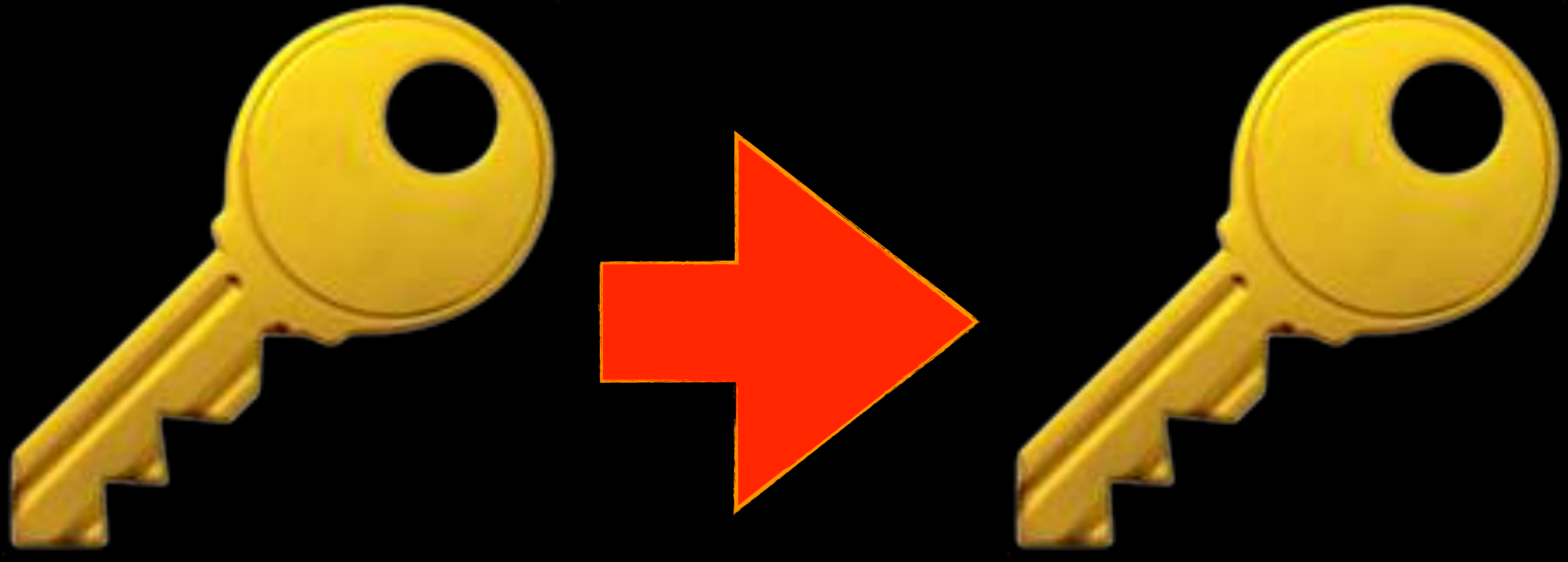
Right now your admins may be blissfully unaware of this.
Their workflows may result in a single SecureToken user.



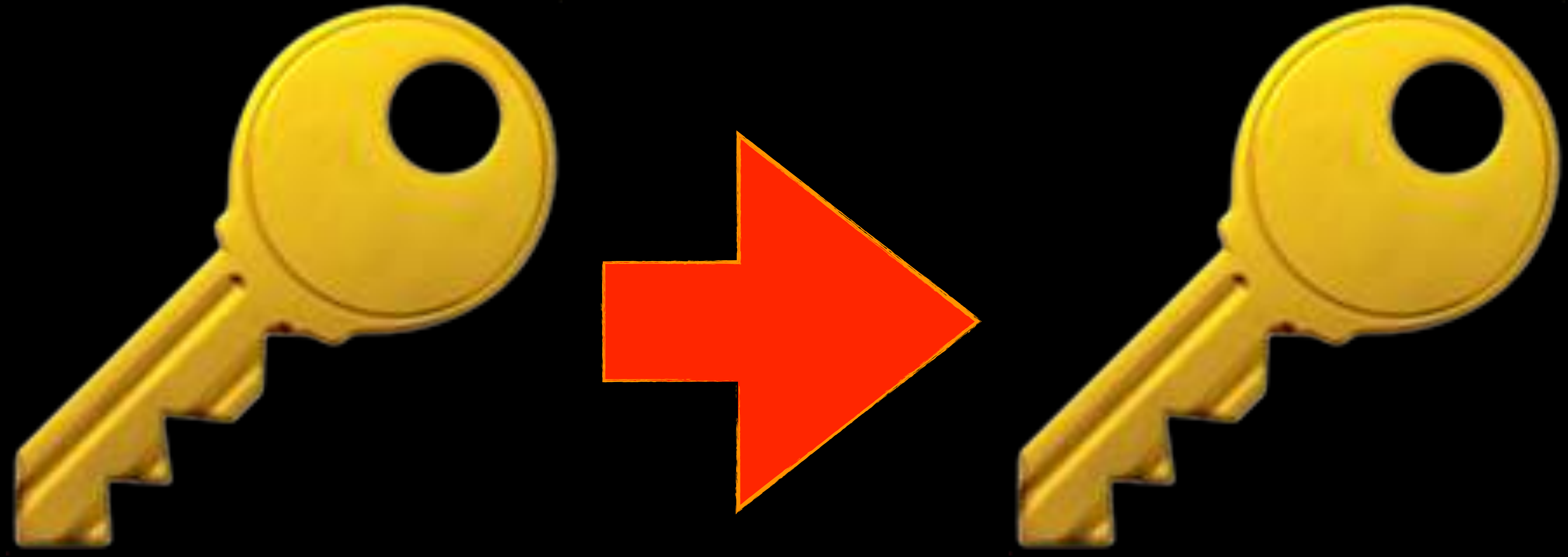
Apple doesn’t have extensive (any?) enterprise workflows & tooling around automatic creation of SecureToken admins.



If your environment needs your forensic toolkits to work,
you may need to find out what the process is right now.



Most enterprise management is moving to MDM. Maybe ask your rep/contact about MDM workflows for this?



… But it’s all ok - target disk mode still works, right?

Duo Security, Inc. [US] | https://duo.com/blog/apple-imac-pro-and-secure-storage

Duo Labs / Featured Article / May 2, 2018

Apple iMac Pro and Secure Storage

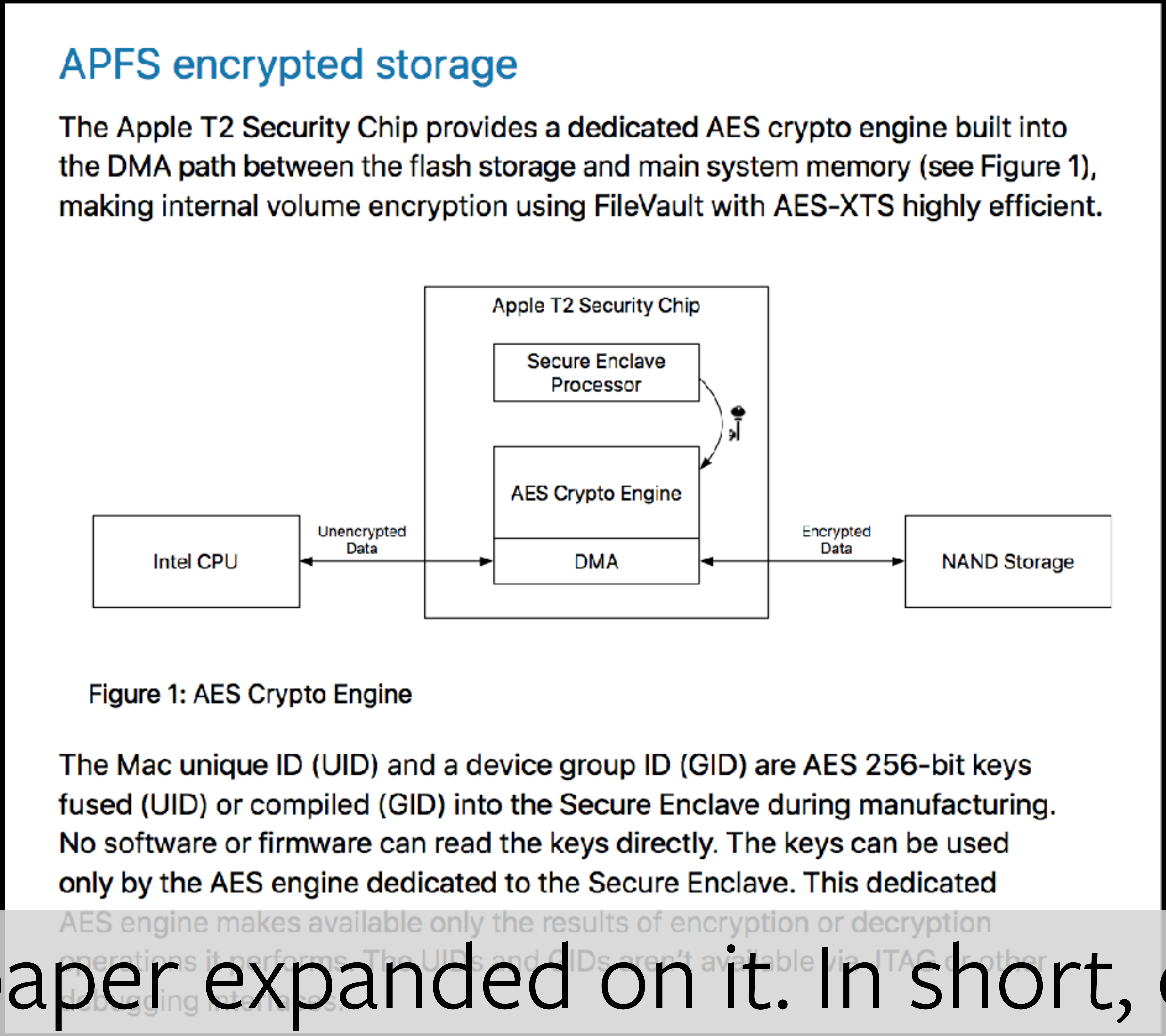
by Pepijn Bruienne

Introduction

With the introduction of the iMac Pro by Apple in late December 2017, a number of brand-new features premiered on the macOS platform. While Apple had already introduced a dedicated security coprocessor with the inclusion of the T1 processor in the late 2016 MacBook Pro with Touch Bar, it served only a few specific tasks like driving the Touch Bar display and touch screen, reading fingerprints from the Touch ID sensor and storing fingerprint data in the T1's Secure Enclave Processor or SEP.

In keeping with its limited system architecture role, the T1 Application Processor or AP was based on a 32-bit ARMv7 processor similar to the S1 AP used in the Apple Watch. Its dedicated watchOS-based OS is named BridgeOS, while the hardware identifies as iBridge.

Let's talk about that. Duo Security released a great initial research paper on T2 storage and how it's different.



APFS encrypted storage

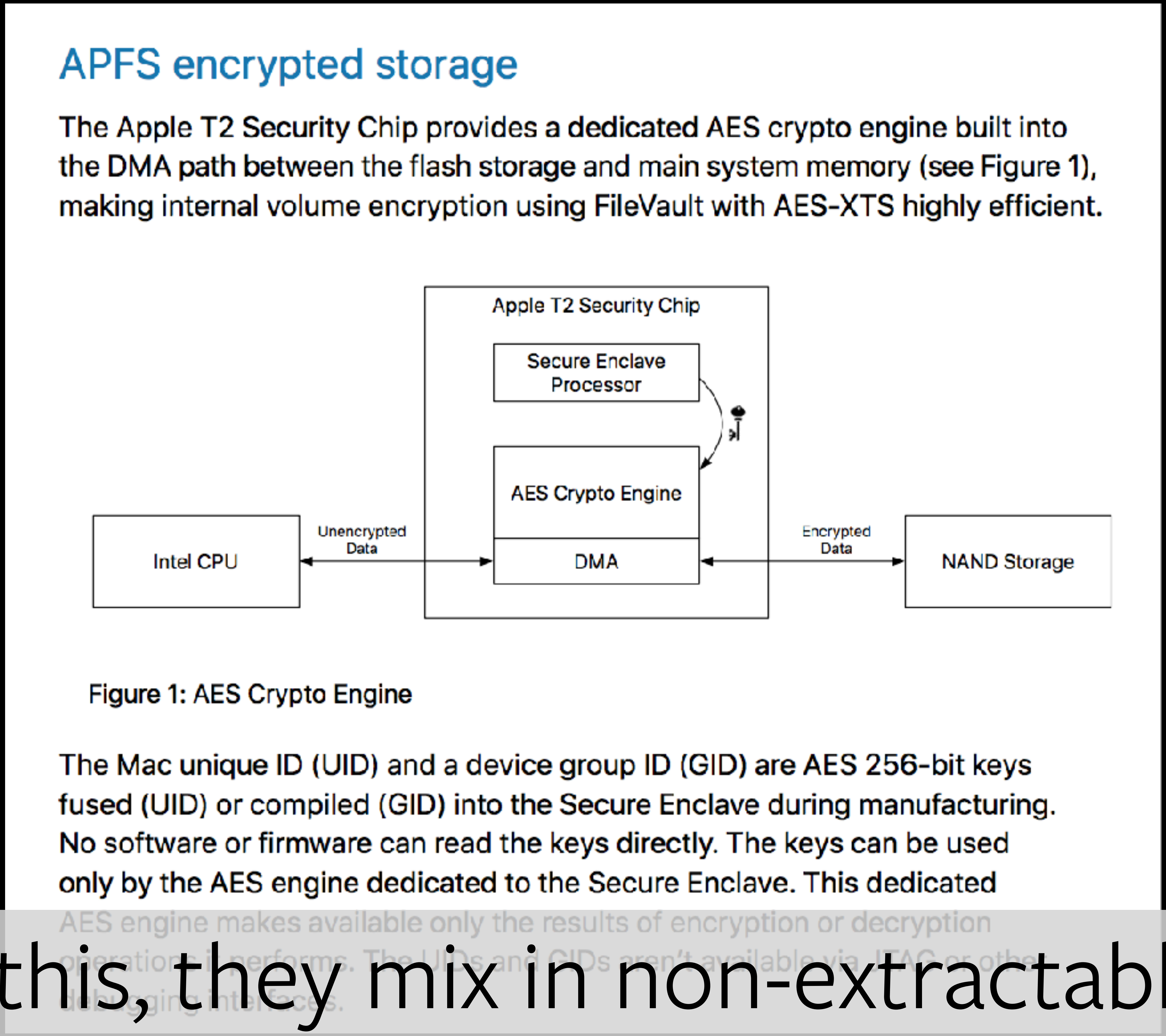
The Apple T2 Security Chip provides a dedicated AES crypto engine built into the DMA path between the flash storage and main system memory (see Figure 1), making internal volume encryption using FileVault with AES-XTS highly efficient.

Apple T2 Security Chip
Secure Enclave Processor
AES Crypto Engine
DMA
Intel CPU
Unencrypted Data
Encrypted Data
NAND Storage

Figure 1: AES Crypto Engine

The Mac unique ID (UID) and a device group ID (GID) are AES 256-bit keys fused (UID) or compiled (GID) into the Secure Enclave during manufacturing. No software or firmware can read the keys directly. The keys can be used only by the AES engine dedicated to the Secure Enclave. This dedicated AES engine makes available only the results of encryption or decryption operations it performs. The UIDs and GIDs aren't available via JTAG or other debugging interfaces.

Apple's white paper expanded on it. In short, encrypted volumes on T2 devices do it at a lower disk level.



APFS encrypted storage

The Apple T2 Security Chip provides a dedicated AES crypto engine built into the DMA path between the flash storage and main system memory (see Figure 1), making internal volume encryption using FileVault with AES-XTS highly efficient.

Apple T2 Security Chip
Secure Enclave Processor
AES Crypto Engine
Intel CPU
Unencrypted Data
DMA
Encrypted Data
NAND Storage

Figure 1: AES Crypto Engine

The Mac unique ID (UID) and a device group ID (GID) are AES 256-bit keys fused (UID) or compiled (GID) into the Secure Enclave during manufacturing. No software or firmware can read the keys directly. The keys can be used only by the AES engine dedicated to the Secure Enclave. This dedicated AES engine makes available only the results of encryption or decryption operations it performs. The UIDs and GIDs aren't available via JTAG or other debugging interfaces.

When they do this, they mix in non-extractable hardware secrets to the volume unlock key.

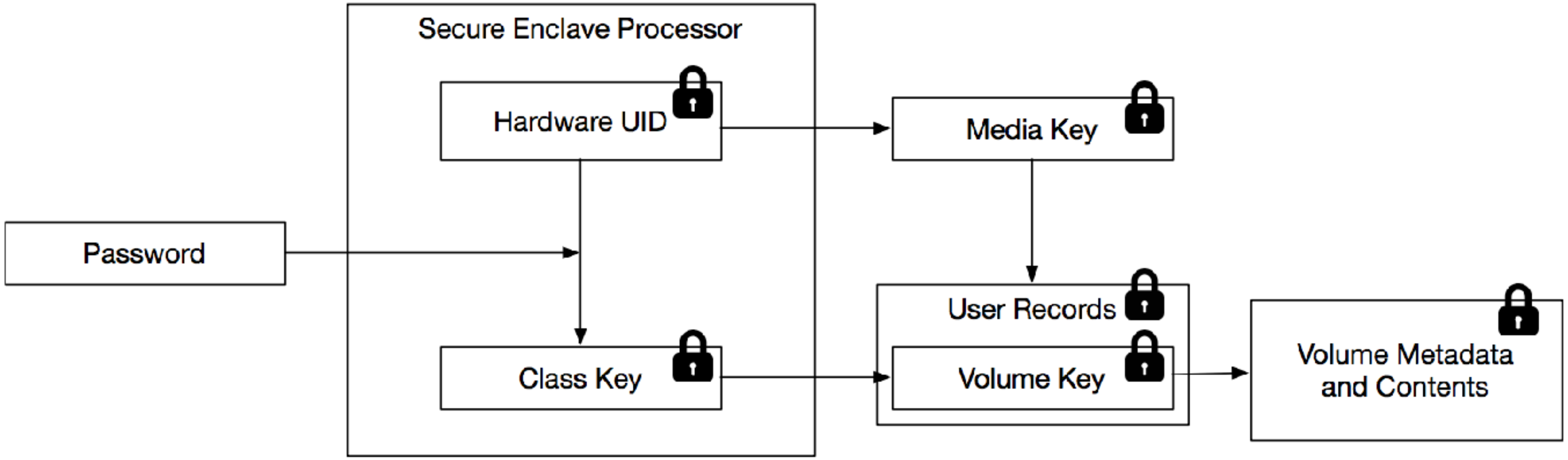


Figure 2: FileVault key hierarchy

This means if you don't have recovery keys for a FileVault disk, even if you grab every byte, you're missing the SEP.

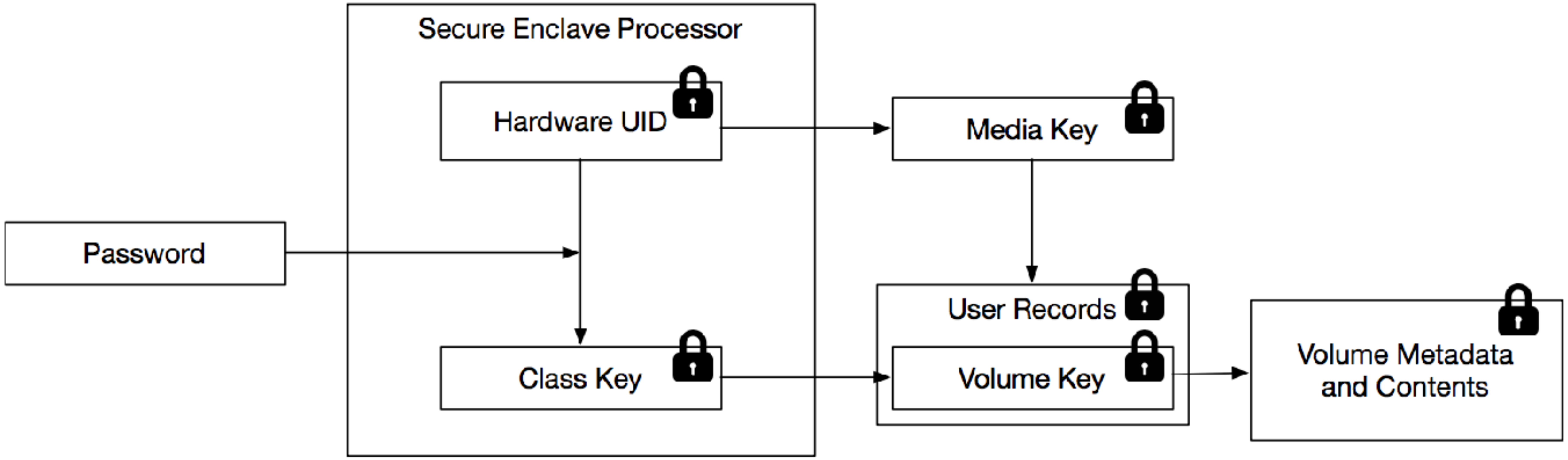
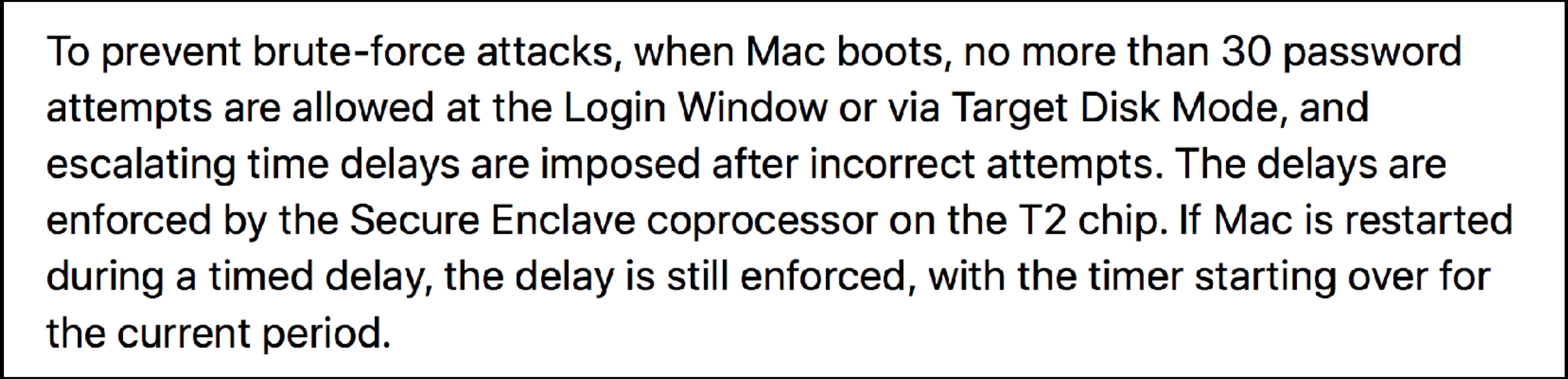


Figure 2: FileVault key hierarchy

And without that, you can't do offline dictionary attacks. You'll have to resort to brute forcing raw keys.

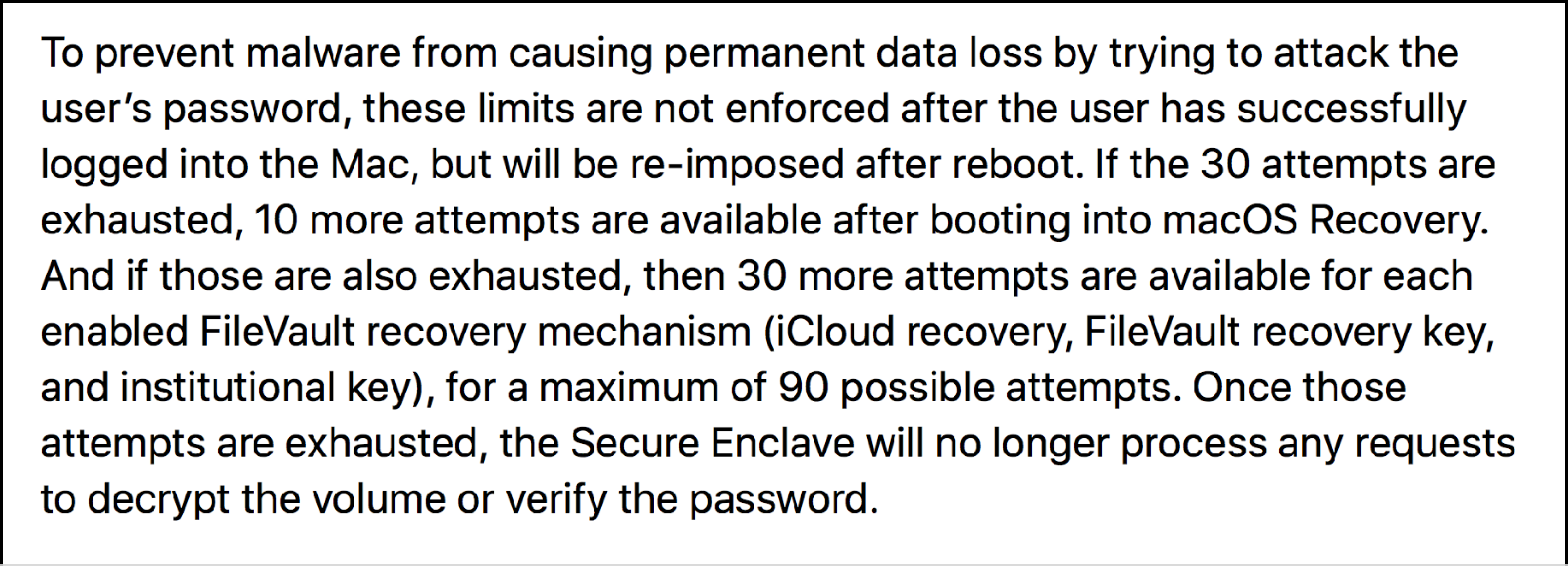
Good luck with that.
I’ll wait here while the solar system comes to an end first.

″Oh″, you say, ″I’ll just brute over target disk mode″



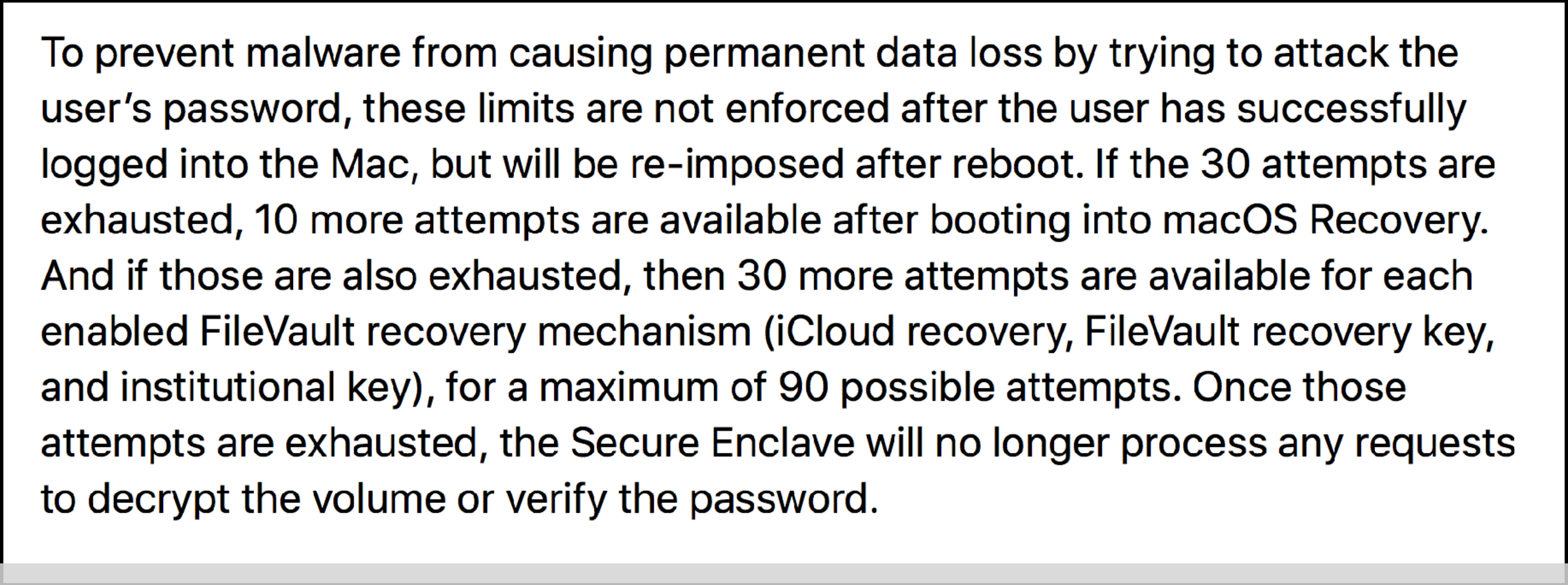
To prevent brute-force attacks, when Mac boots, no more than 30 password attempts are allowed at the Login Window or via Target Disk Mode, and escalating time delays are imposed after incorrect attempts. The delays are enforced by the Secure Enclave coprocessor on the T2 chip. If Mac is restarted during a timed delay, the delay is still enforced, with the timer starting over for the current period.

Here's some fun new details in that same paper. Like iOS devices, there's now enforced delays for wrong guesses.



To prevent malware from causing permanent data loss by trying to attack the user’s password, these limits are not enforced after the user has successfully logged into the Mac, but will be re-imposed after reboot. If the 30 attempts are exhausted, 10 more attempts are available after booting into macOS Recovery. And if those are also exhausted, then 30 more attempts are available for each enabled FileVault recovery mechanism (iCloud recovery, FileVault recovery key, and institutional key), for a maximum of 90 possible attempts. Once those attempts are exhausted, the Secure Enclave will no longer process any requests to decrypt the volume or verify the password.

… annnnd 90 tries ever. TOTAL. Period. 30 per key type. After that, the SEP will ignore additional tries.



To prevent malware from causing permanent data loss by trying to attack the user’s password, these limits are not enforced after the user has successfully logged into the Mac, but will be re-imposed after reboot. If the 30 attempts are exhausted, 10 more attempts are available after booting into macOS Recovery. And if those are also exhausted, then 30 more attempts are available for each enabled FileVault recovery mechanism (iCloud recovery, FileVault recovery key, and institutional key), for a maximum of 90 possible attempts. Once those attempts are exhausted, the Secure Enclave will no longer process any requests to decrypt the volume or verify the password.

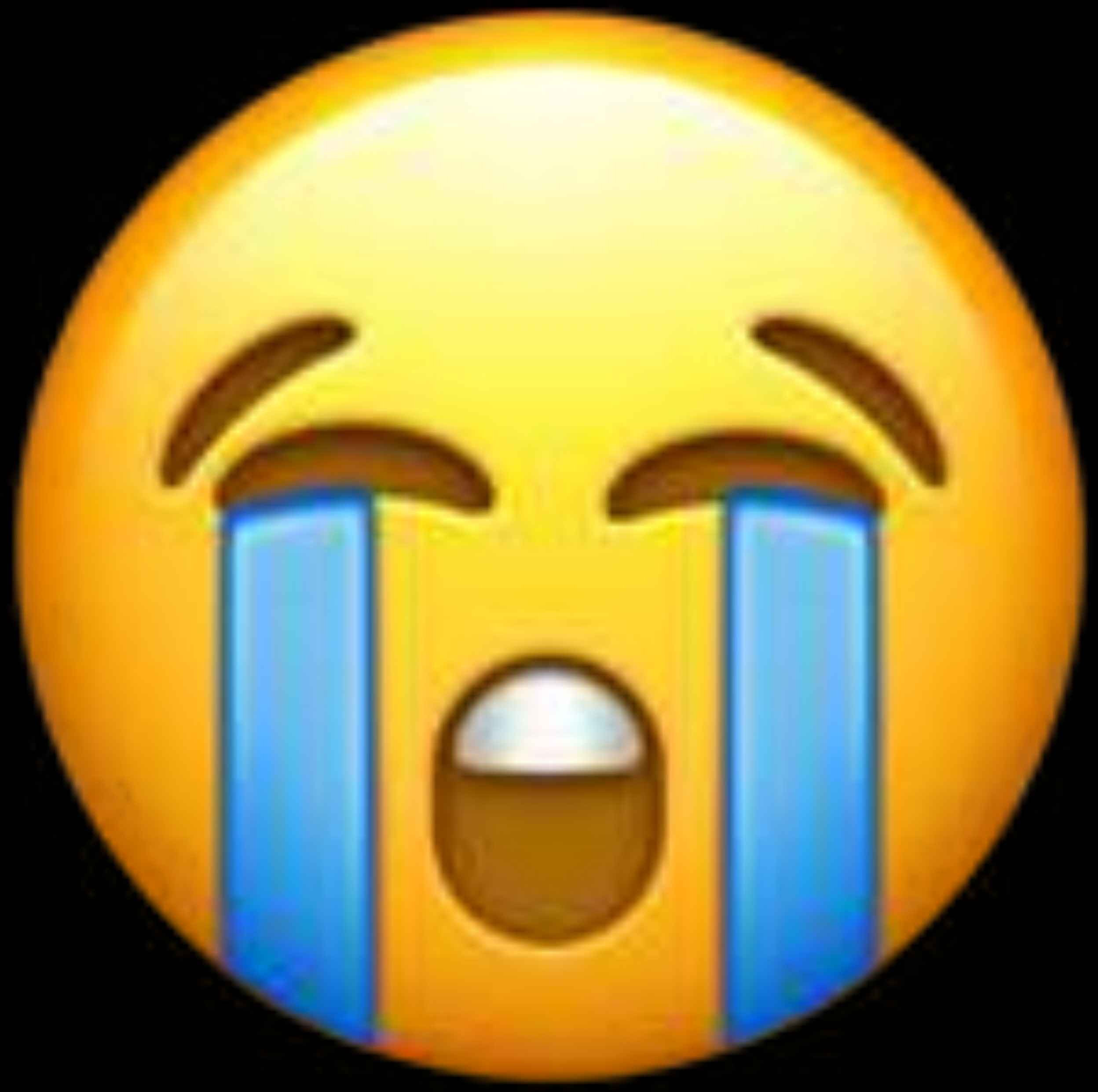
I hope your dictionary is awesome.

Apple views data privacy as extremely important. They are going to do everything they can to increase protections.

If you don’t have proper management of access (IRK, PRK, SecureToken, etc.), you may get shut out of these Macs.

And because of how they're designed, even if you have legal reason to access the data - Apple won't be able to help you.

This can be complicated by bugs and rapid OS change - and Apple needs your help so they can get this right.



So don’t cry about things changing.

1. Talk To Your Admins!

Talk to your admins! They may already be deep in this. They may not have seen it yet. Get informed and work together!

1. **Talk To Your Admins!**

2. **Get a Secure Boot Mac**

If you don’t have one yet - GET ONE. NOW.

1. **Talk To Your Admins!**
2. **Get a Secure Boot Mac**
3. **PRK key escrow / IRK**

Make sure your recovery key escrow is working well.
If you don't have a user's password, you may not get in.

1. Talk To Your Admins!
2. Get a Secure Boot Mac
3. PRK key escrow / IRK
4. Firmware password

And while I’m at it - SecureToken may not be the end of how T2 access is controlled.

1. Talk To Your Admins!
2. Get a Secure Boot Mac
3. PRK key escrow / IRK
4. Firmware password

Firmware passwords offer an additional level of device security and I can see them tying into Secure Boot someday

1. Talk To Your Admins!
2. Get a Secure Boot Mac
3. PRK key escrow / IRK
4. Firmware password

If you’re not managing your firmware passwords on your devices - you should think about looking into it.

1. Talk To Your Admins!
2. Get a Secure Boot Mac
3. PRK key escrow / IRK
4. Firmware password
5. Test and file radars!

1. **Talk To Your Admins!**
2. **Get a Secure Boot Mac**
3. **PRK key escrow / IRK**
4. **Firmware password**
5. **Test and file radars!**

And test everything! These are aliens! (but the good kind)
Not everything you do is guaranteed to work.

1. **Talk To Your Admins!**
2. **Get a Secure Boot Mac**
3. **PRK key escrow / IRK**
4. **Firmware password**
5. **Test and file radars!**

And if you find a bug? Report it! Help Apple make this transition process as smooth as possible.

But most of all …

thank you!

thank you!
(questions?)