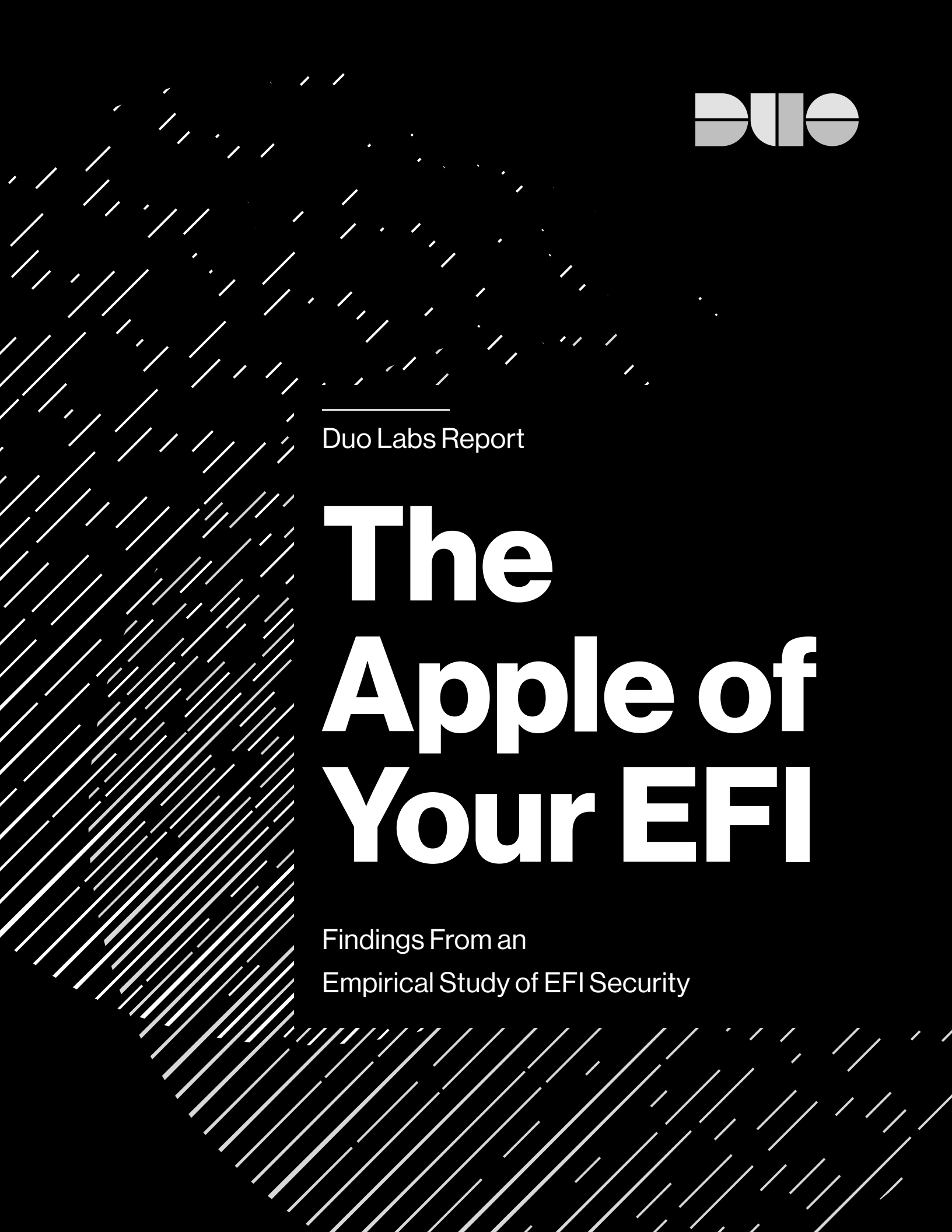


Duo Labs Report

The Apple of Your EFI

Findings From an
Empirical Study of EFI Security

The Apple of Your EFI

Findings From an Empirical Study of EFI Security

AUTHORS
Rich Smith
Pepijn Bruienne

EDITOR
Thu T. Pham

DESIGNER
Chelsea Lewis

VERSION
1.1

Table of Contents

Research Questions & Objectives

In a modern system, the EFI (Extensible Firmware Interface) environment holds particular fascination for security researchers and attackers due to the level of privilege it affords if compromise is successful. EFI is often talked about as operating at privilege level ring -2, which indicates it is operating at a lower level than both the OS (ring 0) and hypervisors (ring -1).

In a nutshell, this means that attacking at the EFI layer gives you control of a system at a level that allows you to circumvent security controls put in place at higher levels, including the security mechanisms of the OS and applications.

In light of recent public releases, there has been increased interest in the security of platform firmware. More specifically, focus has centered on the security of a system's EFI due to the availability of several publicly discussed exploits.

Our Hypotheses: **Firmware Security Support & Visibility**

The state of firmware security as it relates to software security was of particular interest. Our research started with a working hypothesis that there is an asymmetric relationship between the security support vendors provide in the form of patches to their software, as compared to their firmware.

We further hypothesized that both end users and admins make assumptions around the security support provided by vendors to their firmware, while, at the same time, having limited visibility to the reality of the security support actually received.

Finally, we suspected that there would be a divergence between the state of the EFI firmware deployed on systems used in production environments and the expected state of the EFI firmware that should be running, based on the system's hardware and OS versions. The size and characteristics of this discrepancy, was, however, something we were interested in investigating.

Our Research & Analysis: **Why Apple?**

Our research and analysis sought to shine light upon the security of Apple's EFI environment, and to measure the current security state of deployed EFI firmware as compared to the security state of deployed software.

The analysis we conducted focused on Apple's Mac ecosystem. Apple's control over the entire stack across hardware, software, and firmware means its ecosystem provided a more manageable dataset to study.

Equally important is the fact that Apple releases its EFI firmware updates bundled within its OS updates and installs them at the same time, meaning that we could build a definitive map of OS version to EFI version for any given model of Mac hardware. The Wintel PC/OEM ecosystem is far more fragmented and complex in terms of vendor responsibility for the security of the various components, making it more difficult to study and derive conclusive results.

Our Findings: **Expected vs. Actual EFI Versions**

A significant finding of our research was the sizeable deviance between the EFI firmware versions running on production systems and the expected versions based on the OS and hardware versions.

Apple's approach of releasing EFI firmware updates as a component of its overall OS updates allowed us to use the running build version of the OS and the hardware model of the Mac to predict, with certainty, the version of EFI we would expect to be running. We then compared the expected EFI version to the actual EFI version we found to identify discrepancies.

- Our analysis of 73,324 Macs deployed in production environments showed that, on average, 4.2% were running versions of firmware that **did not match** the versions we would expect them to — which could leave them open to publicly disclosed vulnerabilities.

- The level of discrepancy increased significantly above the mean for certain Mac models, with the highest being 43.0% for the iMac 21.5" late 2015 model where 941 out of 2190 real world systems were **running incorrect versions** of EFI firmware.

The size of this discrepancy is somewhat surprising, given that the latest version of EFI firmware should be automatically installed alongside the OS updates. As such, only under extraordinary circumstances should the running EFI version not correspond to EFI version released with the running OS version.

Software Secure but Firmware Vulnerable

Other notable findings detail the discrepancies between the different Mac models that receive software security updates vs. firmware security updates from Apple. It appears that the eligibility for a system to receive firmware updates is dependent on both the model of hardware as well as the version of the OS. This creates multiple possibilities for a system to be seen as ***software secure* but *firmware vulnerable***.

- 16 combinations of Mac hardware and OSs have **never** received any EFI firmware updates over the lifetime of the 10.10 to 10.12.6 versions of OS X/macOS that we analyzed. They do, however, continue to receive security updates from Apple for their OS and bundled software.

- 10 models of Mac received EFI updates in macOS 10.13.0, 10.13.1, and 10.12.6 Security Update 2017-001 did not receive corresponding EFI updates in OS X 10.11 El Capitan Security Update 2017-004.

- Put in terms of our real world dataset of 70,000+ systems, there were 3,400 (4.6%) systems identified that are still considered supported by Apple and continue to receive software security updates, but have not received EFI firmware updates.

As a result, those systems continue to be exposed to a number of publicly disclosed EFI vulnerabilities, despite being patched against known public security vulnerabilities in the OS and bundled applications.

Compounding this situation is the lack of easily available information about which systems are currently receiving software and/or firmware security updates from Apple, as we are unable to find any official documentation pertaining to this anywhere. For users and administrators of Apple systems, this means it can be difficult-to-impossible to accurately build a risk profile for their Apple infrastructure since they cannot know which systems will be left exposed to EFI firmware vulnerabilities, or even which versions of firmware contain which known vulnerabilities.

Our Dataset

Our data details the firmware security support provided by Apple based on an analysis of the security updates they have released since 2015-01-27 and covers versions 10.10.0 to 10.13.1. As part of this work, we will be making this data openly available to help admins and end users make more informed risk decisions about the EFI firmware security of their Apple fleets.

This dataset also augments the gathered data with additional context, such as which vulnerabilities specific EFI firmware versions are vulnerable to. We also intend to release RESTful APIs around the data alongside client apps that can easily make use of them to allow admins and users gain more visibility into the state of their fleets' EFI firmware and make remediation recommendations.

Finally, it is important to note that we do not believe that the issues highlighted by our research are limited to just Apple, and are in fact, indicative of industry-wide problems regarding the lower levels of security support and visibility given by vendors to a system's firmware security as compared to a system's software security. In fact, given the more fragmented and heterogeneous environment of the Wintel space, we would expect the situation there to be potentially worse.

Summary of Findings

The high-level summary of the observations and findings are:

On average, 4.2% of real-world Macs used in the production environments analyzed are running an EFI firmware version that's different from what they should be running, based on the hardware model, the OS version, and the EFI version released with that OS version.

The percentages of incorrect EFI versions varies greatly depending on the particular Mac model:

- The late 2015 21.5" iMac has the highest occurrence of incorrect EFI firmware with 43% of systems running incorrect versions.
- This is followed by the 3 variants of the late 2016 13" MacBook Pro with rates of deviance between 35% and 25%.
- In fifth and sixth places came two variants of the early 2011 MacBook Pro showing a deviance from expected EFI firmware versions of 15% and 12%.

Variance from the expected EFI firmware versions is also markedly different across versions of the OS:

- macOS 10.12 (Sierra) had significantly higher average rate of deviance at 10%
- This is followed by OS X 10.11 (El Capitan) with 3.4% and OS X 10.10 (Yosemite) with 2.1%

The data we gathered suggests that there are possible underlying issues with the mechanisms and procedures used by system administrators to update their fleets' OSs that cause systems to fail to update their firmware while successfully updating their OS. It is also not out of the question that Apple's EFI updating mechanisms themselves have an issue that cause EFI updates to fail to install, though we have found no evidence indicating this during our analysis.

An analysis of Apple's update packages indicates that there are 16 Mac models that have not received EFI firmware updates since Apple changed how it deploys the updates in any of analysed OS and security updates despite being able to run one of the three most recent OS versions that still receive software security updates from Apple. This means these systems can be *software secure* but *firmware vulnerable*.

Our analysis of the Apple OS updates that cite EFI security issues as part of the patch present interesting data showing significant numbers of Mac models that do not actually have a patch available to fix the known EFI firmware issues, despite continuing to receive OS and software security updates:

- **Thunderstrike 1 (CVE-2014-4498)**
  47 models capable of running 10.12, 10.11, 10.10 did not have an EFI firmware patch addressing this vulnerability released

- **Thunderstrike 2 (CVE-2015-3692, CVE-2015-3693)**
  31 models capable of running 10.12, 10.11, 10.10 did not have an EFI firmware patch addressing this vulnerability released

- **CVE-2015-7035**
  25 models capable of running 10.12, 10.11, 10.10 did not have an EFI firmware patch addressing this vulnerability released

- **CVE-2016-7585**
  22 models capable of running 10.12, 10.11, 10.10 did not have an EFI firmware patch addressing this vulnerability released; iMac version 16,2 was an anomaly in that it did not receive an EFI update in Security Update 2017.001 for 10.11.x, but did receive an update in OS update 10.12.4 - all comparable iMac models received EFI firmware updates in updates for both 10.11 and 10.12

Further analysis of Apple's updates also highlighted what seems to be the erroneous inclusion of 43 versions of EFI binaries in the 2017-001 security updates for 10.10 and 10.11 that were older than the versions of EFI binaries that were released in the previous updates 2016-003 (10.11) and 2016-007 (10.10). This would indicate a regression or a release QA failing where incorrect versions of EFI firmware were shipped in OS security updates.

The identification of 18 Mac models with only one, two, or three low-numbered versions of EFI firmware in the production dataset and no new EFI binaries in any of the analysed OS updates. This strongly suggests that these models of Mac have never seen a field update of their EFI firmware and continue to have the version they left the factory with.

A variety of other more one-off anomalies and discrepancies within the large corpus of Apple EFI-related data we have gathered, all of which raise questions about the level of QA being afforded to these EFI firmware updates as compared to software security updates.

The research also points to the lack of information and on-system support for EFI firmware security given to Mac admins and users, despite the increasing number of publicly available information and exploits for EFI attacks.

Without exception, the Apple system admins who provided us the version data for their fleets were very surprised by the EFI firmware discrepancies we discovered across their systems as they have received no notification of those discrepancies from the fleet management systems they use.

0.0

What is (U)EFI & Why Does Its Security Matter?

UEFI stands for *Unified Extensible Firmware Interface* specification[1] and has become the industry's standard replacement for the legacy BIOS platform that had been used, revised and held together with much string and sticking tape since it was introduced by IBM in 1975. Both BIOS and UEFI bridge a system's hardware, firmware and OS together to enable it to go from power-on to booting the operating system. The development of the UEFI standard helped address a number of technical limitations of the legacy BIOS introduced so many years ago.[2]

It also helped the industry move away from the largely proprietary and incompatible implementations that the legacy BIOS had devolved into. In many regards, the adoption of UEFI enabled modern system paradigms developed elsewhere in systems engineering over the preceding 40 years to be applied to the pre-boot environments of PC systems, supporting more ubiquitous and modular approaches to be taken.

Such standardization has numerous advantages to OEMs, as well as to hardware and software vendors through the lowering of development and support costs - however, it is also worth recognizing that malicious actors looking to attack systems also benefit from lower costs of development and deployment that a standards-based system presents. The previously highly fragmented and system-specific environment of the legacy BIOS has now been flattened out into a standards-based pre-boot environment that is common across different platforms and architectures. What had previously required very targeted attacks has now become one where the same vulnerabilities and capabilities can be used across large populations of devices.

Aside from the standardization of systems, attacking UEFI implementations is attractive to security researchers due to the level of privilege and stealth it affords an adversary who is successful in their exploitation. The UEFI environment provides a somewhat unique vantage point in terms of system attacks that is often referred to as Ring -2 to indicate how they operate at a level that is below both the OS (Ring 0) and the hypervisor/VMM (Ring -1) and, as such, can undermine the security controls in both.

Successful attack of a system's UEFI implementation provides an attacker with powerful capabilities in terms of stealth, persistence, and direct access to hardware, all in an OS and VMM independent manner. While a detailed discussion of these capabilities is beyond the scope of the background section of this paper, a number of references are given for the interested reader to find out more.[3,4,5]

The point we want to ensure all readers have in mind as we discuss the research in this paper is that for a modern computer system to be considered secure, it is not enough to just focus on the security and up-to-dateness of the OS and software. One also needs to consider the security of the pre-boot environment, which, in a majority of cases, means UEFI. The unfortunate truth is that insecure UEFI can undermine all of the security put in place at the layers above it in the VMM, OS, and applications. Despite the critical nature of the security of UEFI, it is an area that can often be overlooked by both vendors, admins and end users, and it is for these reasons we took the time to evaluate the security support and awareness of a popular UEFI environment.

Note:
While in the above discussion the term 'UEFI' is used to refer to the pre-boot environment, throughout the rest of the paper the abbreviated term 'EFI' is used. The reasons for this are not only to aid readability, but also in recognition that Apple refers to its pre-boot environment as 'EFI' in all of its documentation. Likewise, many contemporary papers released on similar security topics also use 'EFI' in favor of the 'UEFI' acronym and it feels appropriate to follow their lead in this regard.

1.0

A Brief History of Apple EFI & Related Security Research

The importance of ensuring that up-to-date EFI firmware is installed on Macs is well-known among security experts, however, outside of the security scene, this knowledge may be less common. As such, before we jump into the research itself, we thought it useful to give a little history of the security research done on Apple's EFI-enabled platforms, discussing a number of high-profile EFI vulnerabilities and their accompanying exploits that have been publicly disclosed. If you are already familiar with both the security history and mechanisms involved with Apple's EFI, then jump to **Section 6** to see our new findings.

Using Apple's EFI Implementation & Gaining Persistence

In 2012, Loukas K, better known as "snare," presented EFI-related research at the 12th annual Black Hat USA conference in Las Vegas.[6] In his research, he outlined the process of finding vulnerabilities in Apple's EFI implementation and presented a number of ways to use these vulnerabilities to gain persistence.

The two most promising, and therefore most concerning, methods of gaining persistence identified by snare included:

- Using PCIe option ROMs on external peripherals to inject and run malicious code in the EFI environment.
- Re-flashing the EFI firmware from userland with a modified version that enables malicious code to survive complete OS reinstalls. This only applied to older Mac models that did not use signed firmware code.

The Birth of ThunderStrike to Run Malicious Code

The work done by snare was expanded upon in 2014 by Trammell Hudson of Two Sigma who presented at the 31st edition of the Chaos Computer Club Conference (or 31c3).[7,8] Hudson took both the PCIe option ROM and firmware flash research done by snare and was able to create a PoC (proof of concept) combining hardware and software, using an Apple Thunderbolt to Ethernet adapter, which he named ThunderStrike.

The adapter contained modified firmware code that, when connected to a vulnerable Mac, performed an EFI flash rewrite with modified firmware that allowed the attacker to run arbitrary code at the Ring -2 level while at the same time "fixing" Apple's vulnerable code and effectively closing the door behind itself, preventing further exploitation.

This facilitated simple "**evil maid**" or physical drive-by attacks where an attacker with a modified adapter would be able to plug it into a target system, perform a forced reboot, wait, unplug the adapter and walk away knowing that the target Mac had been successfully compromised with permanently modified EFI firmware. Apple first addressed the PCIe option ROM vulnerability in the OS X 10.10.2 and Security Update 2015-001 for OS X 10.9.5[9] as CVE-2014-4498.[10]

ThunderStrike 2: Less Hardware, More Viral

Following up his earlier work, Trammell Hudson then partnered with Xeno Kovah and Corey Kallenberg of LegbaCore at DEF CON 23 in 2015 to present ThunderStrike 2.[11] This improved version of the original ThunderStrike PoC combined a number of previously disclosed vulnerabilities to enable attacking the EFI firmware without the need for a physical adapter to be used, while, at the same time, adding the ability to further infect any attached Thunderbolt devices in order to virally spread the exploit to other targets.

The main vulnerabilities used in ThunderStrike 2 were patched by Apple in the OS X 10.10.4 update and Security Update 2015-005 on 06/30/2015, as CVE-2015-3692[12] and CVE-2015-3693.[13]

A New Exploit Toolkit Emerges: Sonic Screwdriver

Related to these two major PoCs, it was revealed in March 2017 that as part of the alleged CIA "Vault 7" leaks released by Wikileaks, a very similar exploit toolkit had existed even prior to Hudson's and snare's work, dubbed "Sonic Screwdriver."[14]

This exploit toolkit has been shown to have used the same vulnerabilities that have been publicly discussed in order to implement CIA-created payloads such as the one contained in the same leaked documents named "DerStarke."[15] These tools were developed in November 2012 shortly after the publication of snare's work and thought to be directly based on it.

Recent DMA Attacks

Security researcher Ulf Frisk published his findings related to using the PCIe bus to perform DMA (Direct Memory Access) attacks on against data on disk or in-memory using an external device,[16, 17] once again leveraging unpatched vulnerabilities in Apple's EFI to gain access to sensitive data such as Filevault 2 (FDE) decryption passphrases.

1.1

Changes for macOS 10.13 High Sierra

With the release of macOS 10.13 High Sierra, Apple shipped the largest-ever number of EFI updates in a single OS update. The reason behind the large number of changed EFI firmware payloads is because Apple changed the default filesystem in macOS 10.13 from HFS+ (Mac OS Extended) to APFS, as was announced during WWDC 2017.

In order for the early boot environment to be able to mount an APFS volume, an "ApfsJumpStart" DXE driver was added to Apple's EFI environment. Apple's High Sierra system requirements also meant that a number of older Mac models that had not received any EFI updates bundled with an OS or security update now received their first EFI firmware update. The newly-added models are:

- iMac (Late 2009), aka iMac10,1
- MacBook (Late 2009), aka MacBook6,1
- Mac Pro (Mid-2010), aka MacPro5,1

Possibly because of the issues with EFI update reliability outlined elsewhere in this paper, Apple started to release versions of APFS-capable EFI firmware for some models with the last two macOS 10.12 Sierra point updates, 10.12.5 and 10.12.6. By "pre-seeding" the EFI updates prior to the High Sierra release, Apple's intentions may have been to reduce the chance of failed EFI upgrades causing issues for its customers.

In contrast to this, we noted a difference in supported models for macOS 10.11 El Capitan. There are 10 Mac models that did not receive EFI updates in the 10.11 Security Update 2017-003, whereas they did receive them in macOS 10.12.5, 10.12.6 and macOS 10.13. This can be seen as indicating both good and bad news. The good being that it appears that Apple is keeping the EFI updates distributed with macOS 10.12 security updates and macOS 10.13 OS updates in lockstep, which significantly improves the divergent state of previous OS and Security update cycles.

The bad news is, OS X 10.11 looks like it will continue to have discrepancies in the versions of EFI being released for it compared to newer versions of the OS and certain Mac models that perpetuate the possibility of systems being *software secure but firmware vulnerable*.

The ten models of Mac that saw EFI updates in the 10.13 and 10.12 OS branches but didn't see EFI updates in 10.11.6 Security Update 2017-004 are:

- IM181
- IM183
- MB101
- MBA31
- MBP131
- MBP132
- MBP133
- MBP141
- MBP142
- MBP143

From a version nomenclature standpoint, Apple appears to have adopted a slightly different strategy with the EFI firmware updates released since 10.13. Apple now appears to follow a versioning strategy where they are increasing the four-digit version string and mostly leaving the build (**BXX**) string at **B00**; this is in contrast to the previous approach of the four-digit version remaining constant and the build number being incremented. We still noticed a number of endianness problems with the build string despite this, as EFI payloads for the following models show:

- IM101_00CF_00B
- IM111_0037_00B
- IM112_005B_00B
- IM121_004D_00B
- MB61_00CB_00B
- MB71_003D_00B
- MBA31_0067_00B
- MBP61_005A_00B
- MBP71_003D_00B
- MBP81_004D_00B
- MM41_0045_00B

As more updates are released in the 10.13 branch and the security updates of 10.12 and 10.11, we will have more data points to see what the new normal is in terms of EFI version strings. However, early indications are that the build numbers are less important than they were prior to 10.13 and it is the four-digit version that we will see incrementing over the releases.

Support for MacPro (Mid-2010)

As noted, one of the models that received its first EFI update since Apple changed the deployment method is the Mac Pro (Mid-2010) or as we will refer to it from here on, the MacPro5,1. It is unclear whether Apple decided not to ship any EFI updates because this model requires following the manual process as outlined earlier or because of other internal reasons.

Yet because APFS support in EFI is a requirement for macOS 10.13, Apple had no choice but to update the firmware of the MacPro5,1, which is still very popular with professional users in the graphic design and audiovisual fields. The way Apple managed to support this Mac model is interesting and worth discussing.

Apple implemented the required manual update for the MacPro5,1 through a plugin for the macOS InstallAssistant which is the executable that runs after the user downloads and runs “Install macOS High Sierra” from the Mac App Store. During the program flow, InstallAssistant performs the following calls and branches on any MacPro5,1 that requires the EFI update:

- Calls **apfsSupportedByROM** to determine target’s APFS capability, returns false

- Calls **dataForNVRAMKey** and reads “**4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:FirmwareFeatures**”, which returns the current EFI version

- Calls **isLatestEFIForK5B**, which compares the EFI version retrieved from the previous step
  - The name of the procedure references an internal codename for the MacPro5,1, “**K5B**”

- Calls **isEqualToString**, which compares the host EFI version to “**MP51.0082.B00**”, returns true

- Calls **beginLegcayFirmwareUpdate**

- Instantiates the external **LegacyFirmware.bundle** plugin found in **/Applications/Install macOS High Sierra.app/Contents/Plugins**

- Calls **LFFirmwareUpdateWindowController**, which shows a dialog informing the user of the firmware update requirement

- Prompts user to click “Shutdown” in the **InstallAssistant LegacyFirmware** plugin UI

- Waits for **[LFFirmwareUpdateWindowController shutdownClicked]**

- Instantiates and allocates **OSISClient**

- Calls **[OSISClient stageLegacyFirmwareUpdateWithOptions:]**

- Establishes XPC connection to **osishelperd**, which is a privileged helper process that runs alongside **InstallAssistant**

- Calls **[OSISHelper stageLegacyFirmwareUpdateWithReply:]**

- Locates ***.efi**, ***.fd** and ***.crc32** files in **/Applications/Install macOS High Sierra.app/Contents/Resources/Firmware**

- Calls **NSTask** to make a system call:
  - **/usr/sbin/bless -mount / -firmware EfiUpdaterApp2.efi -payload MP51_0083_00B_LOCKED.fd -options -x efi-apple-payload0-data <contents of *.crc32 file> --verbose**

- System shuts down

- As instructed, the user must now hold down the power button and wait for sustained beep, firmware update commences

We performed a brief comparative analysis of the updated MacPro5,1 firmware and determined that outside of the ApfsJumpStart and PchResetRuntime DXE code additions, no other changes were present. This implies that Apple did not include any of the security patches that were released for other Mac models and only included the EFI functionality necessary to boot from an APFS volume.

The Introduction of eficheck

A new EFI-focused tool called **`eficheck`** shipped with macOS 10.13 as well. The **`eficheck`** tool runs as a scheduled system daemon, but can also be invoked manually from the command line. It uses a library of Apple-curated whitelists to compare the active EFI firmware version to that of known-good versions.

The tool's intention is to alert the end user if an unknown EFI version is found and to offer the option to submit the information to Apple. Based on our use of **`eficheck`**, we can say that it only performs integrity checks and it does not alert users to out-of-date versions of running EFI firmware.

Our Commentary

While not exhaustive, the above illustrates that both public and private research into EFI security, and how to exploit any vulnerabilities found, has been actively pursued by a range of organizations for at least the last five years and likely quite a while longer.

Even though the volume of EFI vulnerabilities is nowhere near as high as for software-based vulnerabilities, the threats are real and carry with them some extra complexity given the system layer being attacked is below the OS's kernel (Ring 0), making it inherently hard to detect the presence of any firmware-based compromise and harder to remove.

When taken in combination with the generally low level of knowledge most admins and users have about the role of EFI in their systems as compared to the OS and applications, alongside the lack of visibility to the 'up-to-dateness' of the installed EFI firmware, we quickly get to a point where the majority of Apple consumers need to rely on Apple to take full responsibility for ensuring their EFI firmware is patched against the latest vulnerabilities.

Apple has made obvious efforts in trying to make the updates of EFI firmware something that admins and users don't have to consider separately and we think this is an approach that adds significant security to Mac systems. As of OS X 10.10, Apple has consistently bundled EFI firmware updates with the OS and security updates themselves, instead of as a separate update with its own installer, improving the likelihood that important EFI security patches are actually applied.

To offer some background on this, it is important to know how EFI firmware updates were treated prior to 2015. Before then, EFI firmware updates were distributed separately and they required manual intervention on each Mac in order to boot into a dedicated EFI firmware update mode.[18] This process was end-user-unfriendly and labor-intensive for Mac admins who were in charge of many hundreds or thousands of Macs.

As a result, the EFI firmware updates would often be skipped and, because there had been no major public security vulnerabilities prior to 2015, they were mostly ignored. The laissez-faire attitude was finally shattered when Thunderstrike 1 was published which seemed to force Apple's hand to make changes in order to ensure that EFI firmware updates deploy alongside OS and security updates.

2.0

How Does a Mac Update Its EFI Firmware & How Do You Find Your EFI Version?

Warning! Running some of the commands seen below could result in bricking your Mac. This data is included for background and completeness, but we do not encourage messing with your EFI firmware unless you know how to recover, should an error occur. Consider yourselves duly warned!

While the standard route by which Apple's Mac EFI firmware gets installed alongside an OS update is deliberately invisible to the end user, it is still worth understanding the process by which EFI firmware is updated so we can understand the EFI lifecycle in more detail. This also allows for manual intervention as needed. There doesn't appear to be any official Apple documentation on the technical details of the EFI update process, so what is found in this and the following sections is a combination of what was learned as the research was conducted, as well as what we learned from other sources of information we found in research released by others.

Mac EFI Firmware Installation

Within an Apple OS update, there is a **`FirmwareUpdate.pkg`** bundle to install EFI firmware containing a postinstall action shell script located at **`FirmwareUpdate.pkg/postinstall_actions/update`** which triggers the firmware installation. The content of the shellscript shows the two main actions taken to update the firmware:

```
#!/bin/sh
/usr/libexec/FirmwareUpdateLauncher -p "$PWD/Tools"
/usr/libexec/efiupdater -p "$PWD/Tools/EFIPayloads"
```

Figure 1.
Contents of the **`FirmwareUpdate.pkg/postinstall_action/update`** *shell script*

The script shows two separate updaters being ran: **`/usr/libexec/FirmwareUpdateLauncher`** and **`/usr/libexec/efiupdater`**. **`FirmwareUpdateLauncher`** seems to be a multi updater that is able to update pretty much all the other non-EFI firmware in a modern Mac system such as the SMC, SSD, and USB-C controllers. Such hardware can also have EFI elements to enable the installation process for other firmware (e.g. like **`MultiUpdater.efi`** and **`ThorUtil.efi`**), but these EFI components are out of scope for the discussion of core EFI updates themselves. For the sake of this paper, we will focus on **`efiupdater`**, as that is the binary that is responsible for updating the EFI.

While the **`efiupdater`** binary is not well documented, it's relatively easy to reverse engineer how it works (see also Section 3. EFI updater board ID behavior). The method used in the postinstall script simply calls the binary with a '**`-p`**' argument and the path to a directory containing all of the EFI firmware updates within an OS update. The determination of which update is the correct one to install on a system is made by the **`efiupdater`** binary itself, and is based on the board ID of the Mac running the command and the board ID(s) contained in the firmware binaries themselves. There doesn't appear to be a way to direct the **`efiupdater`** to install a single EFI update - it wants a whole directory to pick from.

The **`efiupdater`** binary needs to be run as root and takes care of all of the steps of copying the EFI update where it needs to go as well as blessing[19] it, it's a one stop and simple way to update the EFI firmware. Example output from running this command is given in Figure 2:

```
bash-3.2# /usr/libexec/efiupdater -p ~/FirmwareUpdatesTest/
Raw EFI Version string: MBP111.88Z.0138.B18.1702171721
EFI currentVersion: [0000000001380018]
EFI updateVersion:  [0000000001380021]
EFI found at IODeviceTree:/efi
Will need to copy 8523776 bytes to EFI system partition
Aggregate boot path is IODeviceTree:/PCI0@0/RP06@1C,5/SSD0@0/PRT0@0/PMP@0/@0:2
GPT detected
Booter partition required at index 3
System partition found
Booter partition found
Preferred system partition found: disk0s1
Returning booter information dictionary:
<CFBasicHash 0x7f99db7002a0 [0x7fffbe787da0]>{type = mutable dict, count = 3,
entries =>
    0 : <CFString 0x1063d8a60 [0x7fffbe787da0]>{contents = "System Partitions"} = (
   disk0s1
)
    1 : <CFString 0x1063d9240 [0x7fffbe787da0]>{contents = "Data Partitions"} = (
   disk0s2
)
    2 : <CFString 0x1063d9260 [0x7fffbe787da0]>{contents = "Auxiliary Partitions"} = (
   disk0s3
)
}

Substituting ESP disk0s1
Mounting at /Volumes/bless.lFhE
Executing "/sbin/mount"
Returned 0
Creating /Volumes/bless.lFhE/EFI/APPLE/FIRMWARE if needed
Deleting previous contents of /Volumes/bless.lFhE/EFI/APPLE/FIRMWARE
Deleting /Volumes/bless.lFhE/EFI/APPLE/FIRMWARE/MBP111_0138_B18_LOCKED.scap (8520304
bytes)
Opened dest at /Volumes/bless.lFhE/EFI/APPLE/FIRMWARE//MBP111_0138_B21_LOCKED.scap for
writing
preallocation not supported on this filesystem for /Volumes/bless.lFhE/EFI/APPLE/
FIRMWARE//MBP111_0138_B21_LOCKED.scap

/Volumes/bless.lFhE/EFI/APPLE/FIRMWARE//MBP111_0138_B21_LOCKED.scap created
successfully
Relative path of /Volumes/bless.lFhE/EFI/APPLE/FIRMWARE//MBP111_0138_B21_LOCKED.scap is
\EFI\APPLE\FIRMWARE\MBP111_0138_B21_LOCKED.scap
IOMedia disk0s1 has UUID A99B4ECD-E003-4057-9F8F-9E27F4CFB546
Executing "/sbin/umount"
Returned 0
Write to RTC: 0
Setting EFI NVRAM:
<CFBasicHash 0x7f99db405420 [0x7fffbe787da0]>{type = mutable dict, count = 1,
entries =>
    2 : <CFString 0x1063d8aa0 [0x7fffbe787da0]>{contents = "efi-apple-recovery"} =
<CFString 0x7f99db700e70 [0x7fffbe787da0]>{contents = "<array><dict><key>IOMatch</
key><dict><key>IOProviderClass</key><string>IOMedia</string><key>IOPropertyMatch</
key><dict><key>UUID</key><string>A99B4ECD-E003-4057-9F8F-9E27F4CFB546</
string></dict></dict><key>BLLastBSDName</key><string>disk0s1</string></
dict><dict><key>IOEFIDevicePathType</key><string>MediaFilePath</string><key>Path</
key><string>\EFI\APPLE\FIRMWARE\MBP111_0138_B21_LOCKED.scap</string></dict></array>"}
}

Background color default set successfully
```

Figure 2.
Output from running the efiupdater application

The **efiupdater** will only attempt to upgrade an EFI firmware that is newer than the firmware that it detects you are currently running. If an older firmware update is detected, it just prints out the version strings and exits with a return code of 1.

```
Raw EFI Version string: MBP111.88Z.0138.B25.1702171721
EFI currentVersion: [0000000001380025]
EFI updateVersion:  [0000000001380021]
```

There is a '**--force-update**' switch to the **efiupdater** that causes versions to be ignored, meaning that older versions of firmware can be set for installation upon the next reboot, while the older firmware can be set up for installation; whether a downgrade actually occurs is down to the checks enforced by the currently running EFI firmware. During our research, we found no situation where Apple's EFI allowed downgrades to occur. Shortened output from using the switch is below and shows that the EFI **currentVersion** has been nulled out to pass the version check:

```
/usr/libexec/efiupdater -p ~/FirmwareUpdatesTest/ --force-update
EFI currentVersion: [0000000000000000]
EFI updateVersion:  [0000000001380021]
EFI found at IODeviceTree:/efi
Will need to copy 8523776 bytes to EFI system partition
...
...
```

Once the **efiupdater** command has run, you can inspect that the EFI file has been copied to the EFI partition and has been blessed by running the following commands:

```
#Mount the EFI partition
bash-3.2# mount -tmsdos /dev/disk0s1 /tmp/efi
#Look at the contents of the FIRMWARE directory
bash-3.2# ls /tmp/efi/EFI/APPLE/FIRMWARE/
MBP111_0138_B21_LOCKED.scap
```

This shows the EFI update has been copied to the EFI partition. The following command shows the protected nvram variable that has been set by the bless command that causes the EFI update file that has been copied to the partition to be flashed during recovery mode boot. Manually trying to set the nvram variable always fails.

```
bash-3.2# nvram -px efi-apple-recovery
<output redacted>
efi-apple-recovery    <array><dict><key>IOMatch</
key><dict><key>IOProviderClass</key><string>IOMedia</
string><key>IOPropertyMatch</key><dict><key>UUID</key><string>C7BE0AAC-
8FB7-4C11-BF3C-B988FD479B24</string></dict></dict><key>BLLastBSDName</
key><string>disk0s1</string></dict><dict><key>IOEFIDevicePathType</
key><string>MediaFilePath</string><key>Path</key><string>\EFI\APPLE\FIRMWARE\
MBP111_0138_B21_LOCKED.scap</string></dict></array>%00
```

As can be seen in the green above, the **efi-apple-recovery** nvram variable has been set to a plist type structure containing the path to the EFI update file located on the partition we previously observed.

In all our tests using the '**--force-update**' switch, there appears to be another level of checking that takes place within Apple's pre-boot EFI environment that prevents rollback to an older version of EFI. While the EFI update process has been set up by the OS, a reflash never takes place in the pre-boot update environment.

Now that we understand how Apple's EFI updater works, we can recreate the main steps of blessing a new EFI update manually with the following command that makes use of the undocumented **-firmware** switch to the **bless** command. It is also important to use the **-recovery** switch as, without that, the **bless** command fails with the error 'Could not set boot properties: 0xe00002bc Error while writing firmware updater for EFI.'

```
bless -mount / -firmware ~/Desktop/MBP111_0138_B21_LOCKED.scap --verbose
--recovery
EFI found at IODeviceTree:/efi
Will need to copy 8523776 bytes to EFI system partition
Aggregate boot path is IODeviceTree:/PCI0@0/RP06@1C,5/SSD0@0/PRT0@0/PMP@0/@0:2
GPT detected
...
...
<output redacted>
```

Much of the output from this command is the same as the output you see when you run the **efiupdater** command as it just calls **bless** directly after doing a variety of checks and validations.

This concludes the whistle-stop overview of how Apple updates EFI firmware and provides you the tools to take control of updating EFI firmware manually, should you desire.

3.0

EFI Updater Board ID Behavior

While most Mac models have a one-to-one relation with the physical EFI firmware upgrades, some models do not have a dedicated named upgrade file associated with them. Examples of such models are the **MacBookPro11,3** and **MacBookPro11,5** as well as a few others (see **Table 2** in Appendix A for a full list).

Throughout our dataset, we observed that the entries for these models appeared to be using the same EFI firmware as that of the previous model in the range, i.e. MacBookPro11,3 received MacBookPro11,2 firmware, while MacBookPro11,5 received MacBookPro11,4 firmware. While this seemed somewhat logical considering the MBP11,3 and MBP11,5 models were very minor revisions to the model line, we wanted to better understand the process that determines what EFI firmware version these "orphaned" models are assigned.

3.1. Reverse Engineering efiupdater: How Does It Pick an Eligible Firmware File?

As already discussed, the EFI upgrade is initiated by `/usr/libexec/efiupdater` by enumerating a target folder containing EFI firmware updates and selecting an eligible firmware file to perform an update with. If the target folder does not contain an eligible firmware upgrade file because no compatible and/or newer version was found, the tool takes no action. We decided to reverse engineer `efiupdater` in order to understand how it picks eligible firmware. Here's a summary of that work:

When `efiupdater` runs, it checks for the presence of two possible option flags: `-p` (payload) and `--force-update`. When no option flags are provided, the tool simply prints the currently installed EFI firmware version to stdout; both as a simplified string and as a raw version string:

```
Raw EFI Version string: MBP133.88Z.0226.B23.1704201604
       EFI currentVersion: [0000000002260023]
       EFI updateVersion:  [0000000000000000]
```

When run with the **-p** flag, a path containing one or more EFI firmware update files is expected and the tool iteratively inspects them, looking for a compatible update:

```
/usr/libexec/efiupdater -p ~/Documents/EFI/MacBookPro/
Raw EFI Version string: MBP133.88Z.0226.B23.1704201604
       EFI currentVersion: [0000000002260023]
       EFI updateVersion:  [0000000002230000]
```

In the above example, an EFI firmware update file compatible with the model the tool was run on was found but its version (**EFI updateVersion**) was older than what is currently installed (**EFI currentVersion**). This means no update would be possible since **efiupdater** does not allow downgrading. The force flag can be used, and while this will cause the EFI update to be copied to the correct partition and the nvram variables to be set, a reflash does not occur in the pre-boot EFI environment as the currently running EFI firmware does not allow downgrading.

After disassembling **efiupdater**, we found the option flag handling in the **main()** function together with a number of other functions that made it clear how the tool determines model eligibility. As part of the code that handles the force flag, a subroutine creates a **force-efi-update** property in the **"option" IORegistry** realm using **IORegistryEntryCreateCFProperty**, which results in an nvram variable that is read at boot time and initiates the EFI firmware update.

Next, a subroutine uses **IORegistryEntrySearchCFProperty** to search **IORegistry** for two properties: **board-id** and **board-rev**. The **board-id** property is an internal Apple identifier that uniquely labels a Mac model in a way that the more generic model identifier can not. The **board-id** is formatted as **Mac-<8 byte hex value>** and can be retrieved from the command line using this command:

```
       $ ioreg -p "IODeviceTree" -r -n / -d 1  | perl -n -e '/board-
id.*\<\"(.*)\"\>.*/ && print "$1\n"'

       Mac-A5C67F76ED83108C
```

3.2.
Do EFI Firmware Updates Store a List of Board-ID Entries?

To our knowledge, the board-rev property is rarely used outside of Apple, so we concluded that, in most cases, the board-id would be the logical identifier for **`efiupdater`** to use for determining eligibility. The question was then: *do EFI firmware updates store a list of compatible board-id entries that tell `efiupdater` whether a Mac model is compatible or not?*

To answer this, we decided to take a look at the firmware for a model which we knew was found to be installed for more than one model: MacBookPro11,4. We had observed this version of firmware also being installed on the MacBookPro11,5 so our first task was to determine the board-ids for the MacBookPro11,4 and MacBookPro11,5.

Luckily, there are multiple online sources that have complete lists of **`board-id`** to Mac model mappings.[20, 21, 22] We determined that **MBP115** has the board-id **`Mac-06F11F11946D27C5`**, while **MBP114** has **`Mac-06F11FD93F0323C5`**. Our next step was to load an EFI firmware update for the MBP114 in **UEFITool**[23] and perform a hex string search for the hex component of the **`board-id`** for both models.

As expected, we found single entries for both **`06F11F11946D27C5`** and **`06F11FD93F0323C5`** in the GUID **`781F254A-C457-5D13-9275-1BF5D56E0724`** - a raw entry consisting of a 4-byte header (**`0x7C000019`**) followed by one or more 8-byte hex entries representing the hex portion of a typical **`Mac-<hex string> board-id`**, with a maximum of 120 bytes available for board-id storage, or 15 unique entries.

With this knowledge in hand, we returned to the decompiled **`efiupdater`** code to continue our analysis. Following the retrieval of the board-id, we determined that the next step is to iterate through the EFI firmware updates found at the path passed in with **`-p`**.

For each file, the subroutine in charge of finding and opening firmware update files steps into another subroutine that reads the file into memory and jumps to the correct offset within the firmware file (**`0xbc0`** or **`0x1060`**, depending on the relative age of the model) for the **`781F254A-C457-5D13-9275-1BF5D56E0724`** GUID, which we now know contains compatible **`board-id`** entries.

These subroutines continue to iterate over firmware files until the newest firmware version is found. The version of the on-disk firmware is determined by scanning the file for the hex string **`0x2449534f49424924`** which is "**`$IBIOSI$`**" in ASCII representation.

The next 24 bytes following this identifier are then split on the "." character using **`strtok()`** and subsequently scanned for **`"88Z"`** which is the start of the EFI version string. The contents of the version string, formatted as XXXX.YYY where X and Y are both hexadecimal values are converted to unsigned long integers with **`strtoul()`** for numerical comparison.

A complete table of all known EFI models to Board ID mappings is given in **Table 1**, a table showing Mac models that use firmware versions named after other Mac models is given in **Table 2**. Both tables can be found in **Appendix A**.

3.3.
How efiupdater Handles EFI Firmware Updates for Macs Based on Their Board-ID

We now have a complete understanding of how **`efiupdater`** handles EFI firmware updates for Mac models without dedicated EFI updates: the tool retrieves the **`board-id`** for the Mac it is run on, loads one or more firmware updates, parses the contents of a specific GUID in the EFI bundle and attempts to find a match for the **`board-id`** it retrieved. This allows Apple to use one firmware update for multiple compatible models without the need to ship multiple identical files. A simplified flowchart of the above process is illustrated in Figure 3.

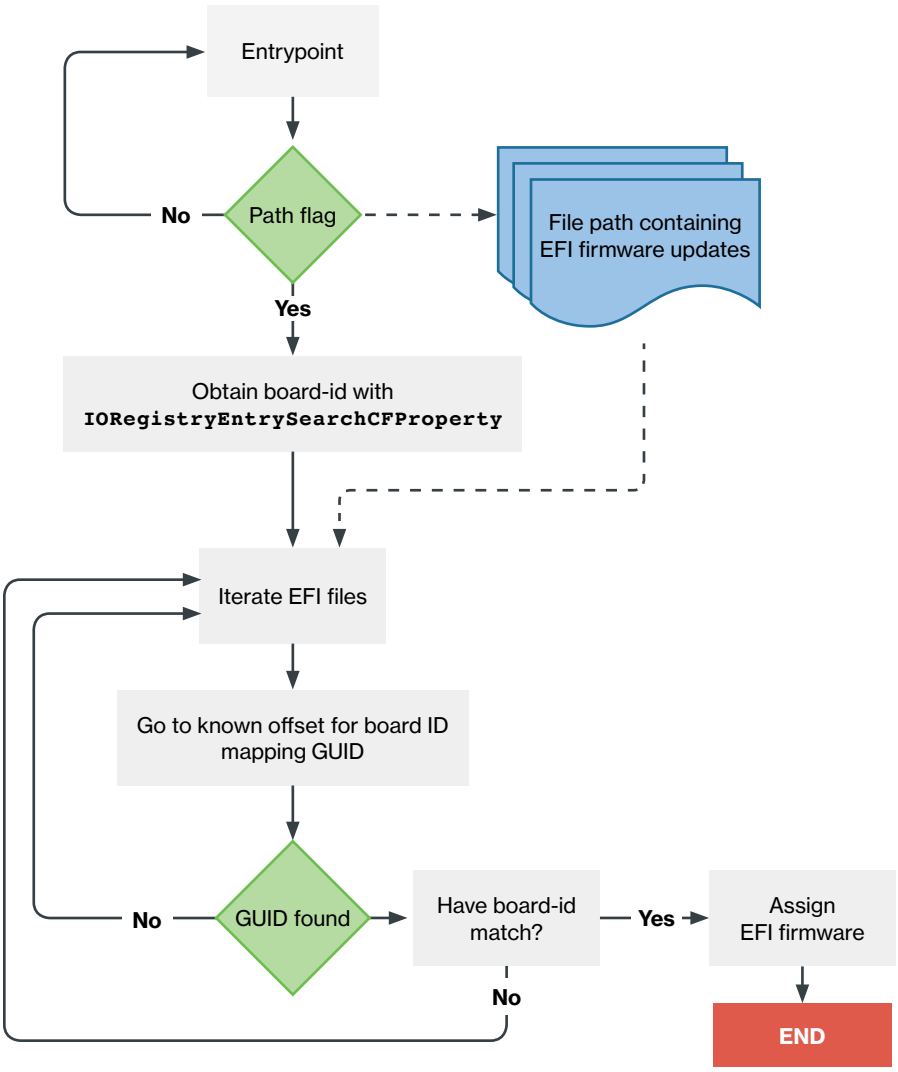


Figure 3.
Flow diagram of the process **`efiupdater`** *follows to locate the correct EFI update file*

Results

4.0

Initial Research Questions

Like all research, we had a series of questions we wanted to answer.
The questions we started this project with were:

- If you are fully patched and up to date with regards to the macOS/OS X software, are you also fully patched in terms of your EFI firmware?
- Is there a difference between the firmware security patches that are available between the different major OS versions (in this case, 10.10, 10.11 and 10.12) even though EFI is independent of the running OS?
- Is there a difference between the firmware security patches that are available between different models of Mac hardware?
- Are there any discrepancies between the expected version of firmware that a system is running and the actual version of firmware a system is running when we look over large numbers of real-world systems that would be indicative of more widespread EFI update issues?
- More generally, is there information that could be shared about non-obvious areas of EFI updates and their security that could help Apple admins and users make their systems more secure?

These questions came from the more general hypothesis that there was an unequal relationship between the security support provided for software as opposed to firmware. Additionally, we hypothesized that the exact situation with firmware security patches and which systems received them was not well-understood, nor something that had been investigated publicly.

It was with these points in mind that we began the project.

5.0

Research Methodology

Our research consisted of two main activities:

- The first focused on the analysis of the Mac software update packages themselves, the firmware updates they contained, and which systems the firmware updates would apply to.

- The second was a large-scale data collection and analysis of 73,383 Mac systems and the versions of software, hardware and firmware they were running.

The analysis of the firmware and software updates released by Apple covered all Mac updates released during the period between OS X 10.10 (released October 16, 2014) and macOS 10.13.1 (released October 31 2017). Of the 73,383 real world systems from which data was collected, 54,744 of them were running OS versions that were 10.10.x, 10.11.x or 10.12.x. At the time of data collection from these systems, macOS 10.13 had not been released. The full list of updates analyzed and the firmware they contained is given in Table 3 in **Appendix A**.

Firmware updates from Apple come bundled within the OS updates and are held in a package file called **`FirmwareUpdate.pkg`**. This can be extracted from the larger OS update bundle manually using standard system tools or through the use of an application like **`Pacifist`**.[24] Since Apple update packages are really just xar archives, the following commands will allow you to extract the **`FirmwareUpdate.pkg`** package from the overall OS update package:

```
xar -xf /tmp/mac_os_upd_10-12-2.pkg FirmwareUpdate.pkg -C /tmp/extracted_
firmware
```

The **`FirmwareUpdate.pkg`** package is, in turn, a directory that contains four files, one of which is a gzip compressed tar archive named Scripts. It can have its contents extracted using the following:

```
tar -zxf  /tmp/extracted_firmware/FirmwareUpdate.pkg -C Scripts
```

A second method of extracting the payload from the package is to use the native **`pkgutil`** tool and its **`expand`** verb:

```
pkgutil --expand /tmp/mac_os_upd_10-12-2.pkg /tmp/extracted_firmware
cd /tmp/extracted_firmware/FirmwareUpdate.pkg/Scripts/Tools/EFIPayloads
```

Either method will produce the directory structure of Scripts that contains a Tools subdirectory that, in turn, contains the EFIPayloads directory.

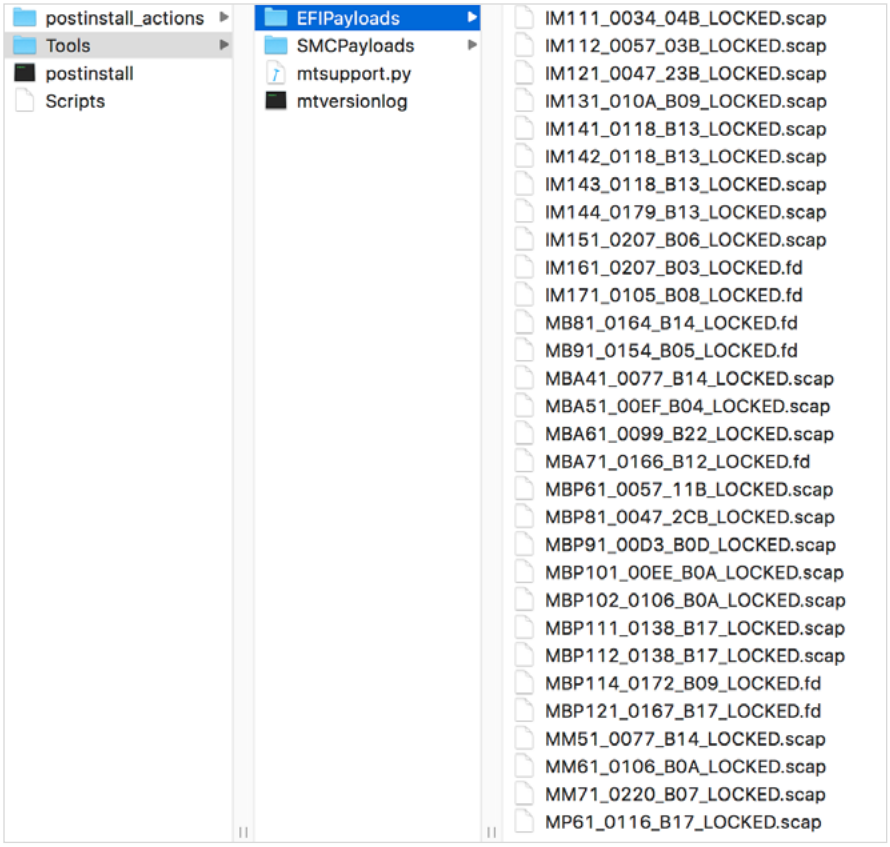


Figure 4.
Contents of an **`EFIPayloads`** *directory in an* **`FirmwareUPdate.pkg`** *bundle*

It is in the **`EFIPayloads`** directory where the actual EFI update binaries are stored. The naming of the files in that directory indicate the Mac hardware model in the section before the first underscore, which is followed by the version of the EFI firmware file itself.

The only deviation from the recipe of analysis outlined above was for two EFI specific updates released by Apple called EFI Update Mac 2015 and EFI Update Mac 2015-002 that contained a differently named package called **`EFIUpdate.pkg`**, rather than **`FirmwareUpdate.pkg`**. The structure and contents of that package was, however, consistent with what is described above aside from the name of the package.

Now that we know how to get access to the EFI binaries in an update, we can build up our dataset correlating EFI binary versions observed in an Apple update against the model of Mac hardware the EFI binary was for; this is repeated for every OS update. This produced datasets that were similar to those in Table A.

OS Update	MBP91	MBP121	IM142
com.apple.pkg.update.security.2017-001Yosemite.14F2315	00D3 B0C	0167 B14	0118 B12
com.apple.pkg.update.os.SecUpd2017-002El Capitan.15G1510	00D3 B15	0167 B24	0118 B20
com.apple.pkg.update.os.MacBookProUpdate.16F2104	00D3 B15	0167 B24	0118 B44

Table A.
An example of the OS update, Mac hardware model and EFI firmware lookup tables that were constructed from analysis of the update packages.

In order to make the analysis of these large sets of Apple firmware updates easier and quicker, a small script was written to automate the task of extracting and tabulating the firmware's characteristics. As part of this research, the tooling developed is being released for end users and admins to use to get a better understanding of their systems' EFI state.[25]

The second area of analysis was across 73,383 Mac systems deployed in production roles across a range of organizations. Administrators at these organizations were kind enough to help in this research by supplying anonymized data regarding the software and firmware versions of their fleets. This data allowed us to work with a representative dataset of real world production Mac systems that we could query and compare against the idealized dataset derived from the OS updates themselves.

The data was gathered by running a distributed query of endpoints using one of two methods:

- A prepared **`osquery`**[26] statement that gathers Model ID, OS version, OS build and EFI boot rom version

```
select hardware_model, os_version.version as os_version, build as os_build,
p.version as rom_version from os_version, platform_info as p JOIN system_
info;
```

A one-liner shell script that uses system tools to gather the same information:

```
echo "$(ioreg -l | awk '/product-name/ { split($0, line, "\""); printf("%s\n",
line[4]); }')\t$(sw_vers -productVersion)\t$(sw_vers -buildVersion)\t$(ioreg -d
3 -p IODeviceTree -n rom | awk -F\" '/version/{print $4}')"
```

Sample output from methods looks as following:

```
+----------+---------+----------+--------------------------+
| hw_model | os_vers | os_build | rom_vers                 |
+----------+---------+----------+--------------------------+
| iMac12,1 | 10.12.5 | 16F73    | IMP121.NNZ.NNNN.BNN.NNNN |
+----------+---------+----------+--------------------------+
```

Once collected, these were put into a single large dataset that comprised the real world data observed about the hardware, software and firmware on a range of systems deployed in production roles across a range of organizations.

With both of these datasets in hand, we were in a position to be able to cross-correlate the observed real world data against the idealized data derived from Apple's updates. Since Apple automates the installation of firmware updates within the larger OS updates, we were able to strongly correlate the expected versions of firmware that should be present on a system given the model of the Mac hardware and the build version of the OS. It is the intersection and analysis of these two fairly simple datasets that allowed us to answer many of our initial questions.

6.0

Analyzing the Data, What Was Found?

Now that we have created our two main datasets and have a way to correlate them, we could dig into the actual analysis. What follows is a discussion of the main items of interest that we discovered during our analysis alongside additional context of what the security impacts of the findings may be.

6.1. Comparing the Running to the Expected Firmware Version, Based on the Current OS Patch Level

As described in the **Research Methodology** section, a large-scale data analysis was conducted across 73,324 Mac systems. Of those systems, there were 65,853 Mac systems that were running an OS build in the 10.10 (Yosemite), 10.11 (El Capitan) or 10.12 (Sierra) families for which we had the corresponding OS and EFI update data from the analysis of update packages themselves. The remaining 7,471 systems were running versions of OS X older than 10.10.

For these 65,853 systems, we then compared the version of the EFI that was installed against the version of EFI that we would expect to be installed based upon the running OS version and the EFI binaries we observed being bundled in that update.

At the top level, this analysis showed that over 4.2% (or 2,282) of the sampled Apple Mac systems were ***not*** running the version of EFI firmware that was released with the OS version they were running. This figure was unexpectedly high given that Apple Mac's firmware updates come bundled with their OS update and should be installed automatically, rather than relying on a separate EFI specific update having to be applied.

Given a version of the OS a Mac system is running, there should be a high degree of certainty as to which firmware version it is running. As part of the installation process that updates the OS to a given version, it should also automatically update the firmware to the latest version that comes bundled with it.

Digging deeper into the data showed a number of other interesting trends. A breakdown of the most interesting observations are given below:

When segmented on the basis of different Mac models, there were certain newer models of hardware that had **significantly** higher rates of unexpected versions of EFI running. The top five models of Mac showing above-average deviations had the following percentages of unexpected EFI firmware:

- The iMac16,2 (iMac 21.5", discontinued 6/5/2017) topped the list with 43% having unexpected EFI firmware versions.
- All members of the MacBookPro13 series (late 2016 MacBook Pro) were found to have between 35% and 25% of reported endpoints running an unexpected version of EFI firmware that was older than the version that shipped with their current OS build version.
- In fifth place was the Macbook Pro 8,2 (early 2011 models) with 14.9% that had older-than-expected EFI firmware.
- These significant deviations from the average across all models of 4.2% of systems running an older-than-expected version of EFI firmware raises significant questions as to why some models are more prone than others to have older EFI firmware. Particularly interesting is the clustering of the three models of the MacbookPro13 series.

When segmented on the basis of OS version, there was also a significant clustering of discrepancies:

- Macs running builds of macOS 10.12 Sierra appeared to have the highest overall incidence of out-of-date EFI firmware with an average of 9.5% running a version of firmware that was older than the versions that shipped with that OS's build.
- Macs running OS X 10.11 El Capitan were second with an average of 3.4% running older-than-expected EFI versions.
- Systems running OS X 10.10 Yosemite had the lowest percentage with 2.1% of EFI firmware versions that were older than expected.
- This gave a combined percentage across OS X 10.10, OS X 10.11 and macOS 10.12 as 4.2% running an older-than-expected version of EFI firmware.
- Overall, our analysis across different OS builds suggests that Mac systems with more recent versions of macOS are up to 4.5 times as likely to have an EFI version that was older than expected in relation to the installed OS build version: 2.1% for those running OS X Yosemite vs. 9.5% for macOS Sierra.

A more granular breakdown of the systems running unexpected firmware versions is given below in Table 3 in **Appendix A**.

Factors Contributing to the Failure to Update EFI

While we are able to observe that older-than-expected and vulnerable versions of firmware are running on large populations of deployed Macs, we are not able to determine the exact root cause that is behind these figures. In reality, we expect that there are likely a variety of different factors at play that contribute to the results we observed.

The sheer number of affected systems alongside the manner in which they cluster depending on OS and hardware version gives us confidence that the anomalies are not purely a result of user error on the part of system owners and it is, in fact, reflective of some kind of failure in the way EFI firmware updates are installed.

Not every method of updating OS X/macOS is equivalent and some methods are seemingly not able to update the EFI firmware. One notable example of this is if the update is being performed via an update source that is using Target Disk Mode.

In such a scenario, the OS will update but the EFI will remain at the previous version. This may explain some of the EFI version discrepancies if systems had a new major OS build installed via the target disk installation method, however it would not account for the discrepancies arising from OS updates installed after the initial base OS installation.

It would also be reasonable to assume that there may be other conditions where installation methods are not able to update the EFI, although the conditions that could preclude the update of the EFI are not seemingly documented anywhere and so would be unknown to most admins and users.

A more concerning potential contributing factor is that, if the EFI update process fails for some reason, the user is not notified, and continues to not be notified, about the fact that their version of EFI is old and, in some cases, potentially vulnerable. This creates a situation where end users and admins believe they are running the most secure and up-to-date system components they can and are, in fact, in a position of potential blind vulnerability to attacks they thought they were secured against.

Compounding this issue further is that without manually carving up an OS update package and knowing the undocumented commands you have to run to update an EFI firmware image, there is no official way to update the EFI image without a full reinstall of the OS update. An anecdotal sampling of the experienced admins at the organizations who shared their system version data with us for this study showed that, without exception, they were all very surprised at the discrepancies in their fleet's EFI versions and that they have received no notifications of those discrepancies from the management systems they use.

Regardless of the root cause, the fact that this is the real-world state of EFI firmware security for Mac deployments strongly indicates there is some form of widespread failure at play, further highlighting the lack of visibility, notification, and control that users and admins have over the security of their system's firmware.

6.2.
Discrepancies in the Systems Receiving Firmware vs. Software Security Patches

Though all the EFI vulnerabilities discussed in section 6.4 were eventually patched by Apple, the way in which this seems to have been achieved appears somewhat arbitrary and differs significantly depending on the combination of the Mac model and OS version.

For example, older iMac models from the iMac11,1 range did not appear to have received any updated EFI firmware until it was bundled with either the OS X 10.11.1 update, or applied with the EFI Security Update 2015-002. These two updates were released in October 2015 and contained patches for the Thunderstrike vulnerability. That same patch had been made previously available for newer Mac models back in June 2015, meaning certain models of Mac were left vulnerable almost four months longer than others.

Older and no longer shipping models received their EFI updates much later, or not at all, vs. newer models that were still available. This could make more sense if the systems not receiving EFI security patches were also no longer supported by the OS versions that Apple continues to provide software security patches for, but this is not the case.

Overall, our analysis of the security patches identifies the following 16 models of Mac systems that are still supported from the perspective of security updates for the OS and built-in applications, but that also appear to no longer receive security updates for their EFI firmware.

iMac	iMac7,1 iMac8,1 iMac9,1 iMac10,1
MacBook	MacBook5,1 MacBook5,2
MacbookAir	MacBookAir2,1
MacBookPro	MacBookPro3,1 MacBookPro4,1 MacBookPro5,1 MacBookPro5,2 MacBookPro5,3 MacBookPro5,4
Macmini	N/A
MacPro	MacPro3,1 MacPro4,1 MacPro5,1

These delays and lack of EFI updates would make more sense if those older models also no longer supported the then-current shipping OS, but this is not the case. All of the “delayed” older models were fully capable of running the current OS at the time, yet did not receive EFI firmware updates (if they received them at all) until a later Security Update release, or updates to the OS itself.

This creates a situation where there are models of Macs that are secure and patched against known security issues from the perspective of their software, but are still vulnerable and out of date in terms of their firmware - or, to put it another way, they are software secure but firmware vulnerable.

There were other unexplained omissions from the list of Mac models that received EFI firmware updates as well. Certain models that, at the time of writing, had supported every shipping version of the OS had either not received any EFI firmware updates or only started receiving them much later. Some examples include:

- Mid-2010 MacBook 13" (MacBook7,1)
- Mid-2010 MacBook Pro 17" (MacBookPro6,1)
- Mid-2010 MacBook Pro 13" (MacBookPro7,1)
- Early 2009 Mac Mini (Macmini3,1)

The Mid-2010 MacBook 13" (MacBook7,1) did not appear to have received any EFI firmware updates until macOS 10.12 Sierra shipped in September 2016. Since this is a model that first shipped in 2010, it would be reasonable to expect that some, if not all, of the EFI firmware patches that shipped between January 2015 and September 2016 were applicable to its firmware, even though it does not have any PCI-connected peripheral ports.

Patch Protection Against ThunderStrike 1 & 2

You may recall that while the first Thunderstrike vulnerability used PCI-attached peripherals to force loading option ROM code that attacked the EFI firmware, later vulnerabilities did not require physical malicious hardware to be present, thus potentially making the 2010-era 13" MacBook vulnerable.

The Mid-2010 17" MacBook Pro did not receive an EFI firmware update until EFI Update 2015-002 was released, providing protection against Thunderstrike 2. Since the earlier EFI Update 2015-001 patched the hardware-based Thunderstrike 1 vulnerability, this means there was a gap in coverage of about four months during which MacBookPro6,1 was vulnerable.

In comparison, the id-2010 13" MacBook Pro went without any EFI firmware updates until macOS 10.12.4 when a MacBookPro7,1-specific update appeared. Lacking any evidence of earlier EFI firmware updates for this model, we must conclude that the Mid-2010 13" MacBook Pro was vulnerable to any vulnerabilities starting with Thunderstrike 1 and continuing until 10.12.4 patched the PCI DMA attack by Ulf Frisk.

It is unclear if this update also provided "true up" patches for other historical EFI vulnerabilities or if it only patched the DMA attack vulnerability. We would be more inclined to assume that these models were simply not affected by any of the vulnerabilities found over the course of 2015 and 2016, but since they have now also started receiving regular EFI firmware updates, the reason for this late inclusion might very well be because of security concerns that arose later.

6.3.
Lack of Visibility, User Alerts & Mac Model/OS Inventory

Compounding these issues further is the lack of visibility into the discrepancy between the security support provided to software vs. firmware, as well as no available data detailing which Mac models on which OSs will receive firmware security patches. Compared to software updates, there is a lack of alerts provided to notify a user of the need for firmware updates, or that the currently running EFI firmware is out of date.

Additionally, there is a lack of readily-available information describing the known security issues any particular EFI version may be vulnerable to; this means users and admins are rarely aware of the risks associated with running unpatched EFI firmware.

The state and visibility of firmware security updates appears significantly lacking when compared to the security updates of OS and application software.

As a result, Apple infrastructure is exposed to the risk of a compromise due to a variety of public exploits that target EFI firmware.

In addition, Apple arbitrarily and gradually phases out security updates for older OS versions, causing those versions to quietly miss out on important EFI firmware updates. The lack of a reliable and published roadmap for current and past OS version support makes it hard to establish an enforceable OS deprecation timeline for most IT organizations. This may inadvertently soothe some into thinking that their older models running older OS versions are still fully supported. Apple's only standing advice for anyone managing Mac models and OS versions of any age is to always run current hardware and software.

6.4.
Anomalies in Firmware Versions Suggest Incorrect Firmware May Have Shipped With Security Updates

As we were analyzing the various OS updates and building up our lookup tables for which versions of EFI firmware shipped for which Mac models, we encountered an unexpected situation where it seems that Security Update 2017-001 (released March 27 2017) for both OS X 10.10.x (Yosemite) and 10.11.x (El Capitan) were released containing EFI binaries that were older than the EFI binaries released with the previous OS updates (Security Update 2016-007 for Yosemite and Security Update 2016-003 for El Capitan).

MD5 hashes of the EFI files contained in the 2017-001 updates show that they match the versions of firmware that were released in security updates from October 2016. Tables 4 and 5 in **Appendix A** provide the breakdown of EFI firmware that shipped with the 2017-001 security updates and in the previous two security updates.

In total, there were 24 EFI binaries that shipped with Security Update 2017-001 for Yosemite (Build 14F2315) that were older versions than shipped with Security Update 2016-007 for Yosemite (Build 14F2109). There were also 23 EFI binaries that shipped with Security Update 2017-001 for El Capitan (Build 15G1421) that were older versions than shipped with Security Update 2016-003 for El Capitan (Build 15G1217).

After this discovery, we wanted to confirm whether there was ever a situation in which this might have caused an unintended downgrade of the EFI firmware of any of the supported Macs. Our analysis showed that in normal use, neither of the EFI firmware updater tools **`MultiUpdater.efi`** or **`efiupdater`** would have caused older versions of EFI to be reflashed on a system as the pre-boot EFI update environment independently checks the version of the new firmware image and will not allow downgrades.

Even if the updater applications were used with non-standard options to force an older EFI firmware binary to be set for installation (or if the whole EFI update process was done manually), security mechanisms within the EFI update pre-boot environment itself prevent a firmware downgrade from occurring. As such, we do not believe a widespread downgrade of EFI firmware was likely to have occurred. This means that the EFI binaries shipping with Security Update 2017-001 look like dead weight and would not serve any purpose to an end user, which further supports the assumption that the presence of the old EFI firmwares in the two 2017-001 security updates were an error by Apple in the packing of the updates. As of the time of writing, this is an error that has still not been addressed by Apple.

Upon the release of Security Update 2017-002 (15 May 2017), these regressions were fixed for OS X 10.11 (El Capitan) with EFI binaries of a completely new version being released, however Security Update 2017-002 for 10.10 (Yosemite) did not address the version issues, as it released without any EFI binaries at all. This creates a confusing situation whereby there are 27 Mac models that would be running more up-to-date firmware if they were at OS update 2016-006 rather than at OS update 2017-002, but would obviously be running out-of-date software.

While an interesting anomaly in and of itself, the fact that Apple shipped what appears to be the incorrect versions of EFI firmware in a security update also raises some concerns around the QA release process that goes into the production release of security updates, as well as the internal coordination between the engineering teams responsible for the Mac software and firmware. This further goes to show the asymmetric relationships that exist in terms of software and firmware security, depending on complex intersections of specific versions of hardware and software.

6.5.
Public EFI Vulnerabilities & Associated Patch Releases

Tables 6, 7, 8 & 9 in **Appendix A** show the updates released that address the range of known public EFI vulnerabilities that have been either explicitly called out as being fixed in Apple's release notes or that were verified as being patched through testing by vulnerability authors (it is assumed that any updates released subsequently to the versions cited as addressing a vulnerability also contain the fix, although, at this time, this has not been independently verified through reverse engineering or comprehensive testing for exploitability).

To our knowledge, there is no currently available dataset that maps Apple's EFI versions to the vulnerabilities that impact them. As part of this work, we will be releasing the data and APIs to make those queries simple to make.

Beyond the data surrounding which EFI firmware versions patch which vulnerabilities, the collation of these vulnerability-specific datasets themselves allows further analysis, which presents some interesting results that are discussed below in more detail. In tables 6, 7, 8 & 9 in **Appendix A**, anomalies are highlighted with green text.

Thunderstrike 1 (CVE-2014-4498[27]) patches - This patch looks relatively straightforward. All systems receiving the patch receive it evenly across the board, regardless of whether they are running 10.10, 10.11, or 10.12.

In all, there were 47 modern Mac models capable of running 10.10, 10.11 or 10.12 that did not appear to receive a patch against the Thunderstrike 1 vulnerability:

iMac	iMac7,1, iMac8,1, iMac9,1, iMac10,1, iMac11,1, iMac11,2, iMac11,3, iMac12,1, iMac12,2, iMac13,1, iMac13,2
MacBook	MacBook5,2, MacBook6,1, MacBook7,1
MacBookAir	MacBookAir2,1, MacBookAir3,1, MacBookAir3,2, MacBookAir4,1, MacBookAir4,2, MacBookAir5,1, MacBookAir5,2
MacBookPro	MacBookPro3,1, MacBookPro4,1, MacBookPro5,1, MacBookPro5,2, MacBookPro5,3, MacBookPro5,4, MacBookPro5,5, MacBookPro6,1, MacBookPro6,2, MacBookPro7,1, MacBookPro8,1, MacBookPro8,2, MacBookPro8,3, MacBookPro9,1, MacBookPro9,2
Macmini	Macmini3,1, Macmini4,1, Macmini5,1, Macmini5,2, Macmini5,3
MacPro	MacPro3,1, MacPro4,1, MacPro5,1, and MacPro6,1

Thunderstrike 2 (CVE-2015-3692,[28] CVE-2015-3693[29]) patches - This patch set is also consistent in the versions of EFI that are being patched across all OS versions that contained the EFI updates. Exceptions worth noting would be iMac15,1 and MacBook8,1 that did not have updated firmware shipped in Security Update 2015-005 for 10.9 or 10.8. This was most likely because these models were released with OS X 10.10.x installed and so are not considered supported for 10.9 or 10.8.

In all, there were 31 modern Mac models capable of running 10.10, 10.11 or 10.12 that did not appear to receive a patch against the Thunderstrike 2 vulnerability:

iMac	iMac7,1, iMac8,1, iMac9,1, iMac10,1, iMac11,1, iMac11,2, iMac11,3, iMac12,1
MacBook	MacBook5,1, MacBook5,2, MacBook6,1, MacBook7,1
MacbookAir	MacBookAir2,1, MacBookAir3,1, MacBookAir3,2
MacBookPro	MacBookPro3,1, MacBookPro4,1, MacBookPro5,1, MacBookPro5,2, MacBookPro5,3, MacBookPro5,4, MacBookPro5,5, MacBookPro6,1, MacBookPro6,2, MacBookPro7,1
Macmini	Macmini3,1, Macmini4,1
MacPro	MacPro3,1, MacPro4,1, and MacPro5,1

CVE-2015-7035[30] patches - This patch set had quite a number of anomalies focused on the uneven distribution of EFI updates across the range of OS and security patches that should have contained them. A summary of the anomalies is given below:

- No EFI patches for this vulnerability were released for systems running OS X 10.9.x.
- Certain models of Mac seem to have had EFI firmware patches missing from Security Update 2015-004 for 10.10 and only updated firmware included in 10.11.1 and EFI Security Update 2015-002. The models in question are MacBookAir4,1, MacBookAir4,2, MacBookPro8,1, MacBookPro8,2, MacBookPro8,3, Macmini5,1, Macmini5,2, Macmini5,3. The absence of firmware updates for them in Security Update 2015-004 for 10.10 is in contrast to the many other models receiving patches for CVE-2015-7035 across the board, looking at the **heatmap** (duo.sc/EFI-heatmap) helps make these anomalies clear.
- iMac16,2 does not have an EFI update included in 10.11.1 whereas iMac17,2 does. This model of Mac was released only a short time before the CVE-2015-7035 patch and so it is conceivable that the EFI firmware it shipped with from the factory was already patched against the vulnerability, though this raises the question as to why that wasn't also the case for iMac17,1.
- Another anomaly is the absence of EFI patches in the EFI Security Update 2015-002 for the following models: iMac15,1, MacBook8,1, MacBookAir7,1, MacBookAir7,2, MacBookPro11,4, MacBookPro11,5 and MacMini7,1. These models break the pattern of other closely related models that ship firmware patches in EFI Security Update 2015-002 and is also noteworthy for the fact that all of these models had firmware patches contained in EFI Security Update 2015-001.

In all, there were 25 modern Mac models capable of running 10.10, 10.11 or 10.12 that did not appear to receive a patch against the CVE-2015-7035 vulnerability:

iMac	iMac7,1, iMac8,1, iMac9,1, iMac10,1
MacBook	MacBookAir2,1, MacBookAir3,1, MacBookAir3,2
MacbookAir	MacBookAir2,1, MacBookAir3,1, MacBookAir3,2
MacBookPro	MacBookPro3,1, MacBookPro4,1, MacBookPro5,1, MacBookPro5,2, MacBookPro5,3, MacBookPro5,4, MacBookPro5,5, MacBookPro7,1
Macmini	Macmini3,1, Macmini4,1
MacPro	MacPro3,1, MacPro4,1, and MacPro5,1

DMA attack (CVE-2016-7585[31]) patches - This patch set contained a number of anomalies relating to the combinations of Mac hardware and OS version that see patches for the vulnerability being shipped.

- Singular in its occurrence was the anomaly of iMac16,2 that seems to be missing a patch for 10.11 even though it receives a patch for 10.12 and 10.10

- Distinct as compared to the other security patches discussed so far was the number of models that received EFI security updates that had never seen them before. However, they only received the fix for CVE-2016-7585 if they were running 10.12; if they were running OS versions 10.11 or 10.10, they remained vulnerable. The models in questions are: MacBook7,1, MacBookPro7,1, and MacMini4,1.

- The patch set also exhibited a characteristic not seen in other EFI vulnerability patches where the version numbers of the EFI firmware were different depending on which software patch contained them. While these different version numbers are not necessarily vulnerabilities in and of themselves, it is an artifact of the inconsistencies observed in the way EFI firmware security patches are being provided across different OS versions.

In all, there were 22 modern Mac models capable of running 10.10, 10.11 or 10.12 that did not appear to receive a patch against the CVE-2016-7585 vulnerability:

iMac	iMac7,1, iMac8,1, iMac9,1, iMac10,1
MacBook	MacBook5,1, MacBook5,2, MacBook6,1
MacbookAir	MacBookAir2,1, MacBookAir3,1, MacBookAir3,2
MacBookPro	MacBookPro3,1, MacBookPro4,1, MacBookPro5,1, MacBookPro5,2, MacBookPro5,3, MacBookPro5,4, MacBookPro5,5
Macmini	Macmini3,1
MacPro	MacPro3,1, MacPro4,1, and MacPro5,1

Additionally, these 22 models did not receive security patches for any of the four public vulnerabilities discussed above.

Overall, the level of inconsistency that seems to be present with regards to which systems are receiving EFI patches for known public vulnerabilities is concerning due to the likelihood of systems remaining inadvertently unpatched and vulnerable with end users and admins possibly having a false sense of security in thinking their systems had been patched. This **heatmap** shows a more graphical overview of which systems did or did not receive a patch for a given vulnerability at the time of initial release.

We would encourage Apple and other vendors to be both explicit and transparent with the systems that are, and are **not**, receiving EFI firmware patches for known public vulnerabilities in order to provide the users of systems full visibility to the state of their systems and any associated risks.

6.6. Incorrect CVE Assigned to Unused EFI Functions

A smaller finding discovered as part the research was the incorrect CVE ID given in the release notes for Apple's OS update for OS X El Capitan 10.11.1 and related security updates for Yosemite and Mavericks.

The discrepancy was that in the Apple security release notes,[32] the CVE attribution for this issue was listed as CVE-2015-4860.[33] This was incorrect as it refers to an entirely unrelated Oracle Java vulnerability. The correct CVE that should have been referenced for the vulnerability is CVE-2015-7035.

We raised this as a Radar item that got assigned ID 32995209.[34]

6.7.
Miscellaneous & Interesting Anomalies in the Dataset

This section just captures a few other observations made during the research, and while they are not as concerning as some of the other findings above, they are worth noting and could warrant more investigation or explanation in the future.

Analyzing the released firmware updates and looking at the highest version available for each Mac model seen in **Table 10** in Appendix A reveals some interesting anomalies (note that due to a range of reasons discussed elsewhere in the paper, the highest numbered version of EFI firmware does not always relate to the most recently released).

A collection of high-level observations from this data include:

- Mac model Mac11,1 seems to have never received a newer EFI firmware version since the standalone 2015-002 EFI update.
- Mac model MacBookPro6,1 saw no EFI updates for an OS version since the standalone 2015-002 EFI update until the 10.12.4 update released in March 2017. After that, two month delay until Security Update 2017-002 (El Capitan) was released in May 2017.
- There are six systems where the most recent version of EFI firmware shipping with OS X 10.11 (El Capitan) do not match the most recent versions of EFI firmware shipping with macOS 10.12 (Sierra). These systems are iMac14,1, iMac14,2, iMac14,3, MacBookAir6,1, MacBookAir7,1 and MacMini7,1.
- With the exception of iMac11,1, any system running OS X 10.10 (Yosemite) runs older EFI firmware than the same system running OS X 10.11 (El Capitan) or macOS 10.12 (Sierra).
- Given the possibility of the EFI firmware version regression already discussed in section 6.4, the highest versions of EFI firmware for Yosemite systems were released in early December 2016.

6.7.1. Mac Models With Multiple Observed Versions in Production but No Observed EFI Updates

An analysis of the real world dataset alongside the EFI firmware updates released with the OS patches shows a number of Mac models that have only been observed with either a singular, or small number of deployed versions. Single versions of observed EFIs strongly indicate that those Mac models have never received a field update to their EFI and are running the firmware that they left the factory with.

Two or three observed versions that are very close in build number could indicate that either there were a small number of different firmware versions that were used on shipping Macs, or that there were field updates - but they were before the release of OS X 10.10 and were not part of our analysis. If the latter is the case, then the EFI firmware of those models has gone many years without seeing an in-field update and are likely open to multiple vulnerabilities.

Table 11 in Appendix A shows the models that had no EFI updates between OS versions 10.10.0 to 10.13.1 and where the real world data shows only one, two or three EFI versions observed in the wild. All of the models here are capable of running 10.11 or newer, older models were not included as part of this analysis.

6.7.2.
An Anomaly in EFI Firmware Updates for Macbook Pros

There were three models of Macbook Pros (MBP111, MBP114 & MBP121) where, for Security Update 2016-003 (El Capitan), the firmware in the OS update did not appear to have their versions increased in the same way that EFI's for other related Macbook Pros did. Table B shows the observed anomalies:

Update	Date	MBP101	MBP102	MBP111	MBP112	MBP114	MBP121
2016-007 (Yosemite)	2016/12/13	00EE B0B	0106 B0B	0138 B18	0138 B18	0172 B10	0167 B18
2016-003 (El Capitan)	2016/12/13	00EE B0B	0106 B0B	**0138 B17**	0138 B18	**0172 B09**	**0167 B17**
10.12.2 (Sierra)	2016/12/13	00EE B0B	0106 B0B	0138 B18	0138 B18	0172 B10z	0167 B18

Table B *- EFI firmware versions released in Security Update 2016-003 (El Capitan) that did not appear to receive an update while other Macbook Pro models did*

Table B shows in green the three models of Macbook Pro with versions of firmware that did not seem to receive updates in the OS update package despite other Macbook Pro models receiving an incremental update for the same 2016-003 (El Capitan) security update, and the same Macbook Pro models also receiving updated firmware in the updates for 10.12.2 and 2016-007 (Yosemite).

This data tends to suggest that there could have been another `FirmwareUpdate.pkg` QA issue where the three Macbook Pros erroneously didn't receive the firmware update they should have; it also could have been intentional on Apple's part. It's impossible for us to know, but it is interesting nonetheless.

6.7.3.
EFI Update Filename Anomalies

Some EFI update filenames seem to flip the endianness of the numbers used in the file naming of the build number in OS update 10.12.4 released in March 2017. The differently-named EFI updates were confirmed to be identical through comparing the SHA-256 hashes on the files themselves.

For example, Mac model iMac12,1 had an EFI update named 'IM121_0047_**B29**' in OS update 10.12.4, and it converted back to 'IM121_0047_**29B**' in OS update 10.12.5. While this is not a security issue in and of itself, it does raise questions about the level of automation and associated QA processes involved in the release of new EFI updates. Inconsistencies in the versioning of updates quickly leads to worries about inconsistencies elsewhere that may be less visible. Aside from any security issues, naming convention discrepancies like this can make tooling that inspects things like versions more of a pain to write as there has to be unique exceptions for such anomalies.

The full list of systems where we observed this EFI file name change is: iMac11,2, iMac12,1, MacBook7,1, and MacBookPro6,1.

7.0

Mitigation

While the age of Macs and the OS running on them as used by an organization in combination with the management tools in use can make for a complicated picture when it comes to keeping track of the actual deployed versions of EFI firmware, the ways to mitigate out-of-date versions are straightforward:

- Always deploy the full update package as released by Apple, do not remove separate packages from the bundle updater. Some Mac sysadmins will separate the OS and EFI firmware updates for historical purposes but this should no longer be necessary.

- When possible, deploy Combo OS updates instead of Delta updates. After the initial .1 point update to macOS, Apple releases both Combo and Delta updates. Mac sysadmins in the field report better update success rates when choosing the Combo update.

- Verify that endpoints received the expected EFI firmware version as shipped with the OS update that was applied. Refer to the **Research Methodology** section for directions on how to find the versions of EFI firmware as part of an OS or security update or make use of the EFIgy tooling we have released along with this paper.[35]

- Since Apple no longer provides separate EFI firmware updaters through their Support website, it is imperative that any endpoints that did not receive the expected EFI firmware update are scheduled to re-install their most recent OS or security update.

- Although faster, it is important to keep in mind that when using imaging workflows instead of file-based installations of the OS, there is a significant chance that the target Mac will not receive EFI firmware updates. Imaging workflows simply write the blocks of a precreated disk image to disk and unless the post-imagine workflow is followed up with an installation of the appropriate EFI firmware update for the just deployed OS version, the endpoint firmware may go out of date. This includes imaging via Target Disk Mode and NetBoot or NetInstall-based imaging workflows.

- As a general rule of thumb, always run the latest version of macOS (10.12 at the time of writing). While Apple has historically provided security updates for at least the two previous OS versions, they typically do not contain all the security patches that ship for the current OS version and this seems increasingly true for EFI firmware updates.

8.0

Conclusion

This research has shone a light on some of the ways in which security patch support provided by Apple for firmware is quite different than the security patch support they provide for software. Our findings highlighted five main areas of note regarding Apple's EFI security:

1. **You can be software secure but firmware vulnerable**. The EFI firmware security patch support does not map 1:1 to software security patch support provided by Apple. As a result, you can be unknowingly running systems that are fully up to date for OS and applications, but years out of date in terms of EFI firmware, leaving your Mac vulnerable to publicly disclosed vulnerabilities and exploitation.

2. **There seems to be something interfering with the way bundled EFI firmware updates are getting installed, leading to systems running old EFI versions.** We are not able to give an exact reason why, but there are significant discrepancies between the firmware version that is actually running on real world production systems and the version that is expected to be running, given the OS build. This means that even if your Mac is still receiving security patch support, there is a non-trivial chance that your system is not running the latest version, even though you thought it was installed.

3. **The release QA on the FirmwareUpdate bundles is concerning** - The presence of what looks like version regressions on the included EFI firmware bundles for recent security updates is very surprising to find. Additionally, the seemingly erroneous absence of security-specific EFI updates for some models of hardware is even more worrying. The fact that it has not been detected and fixed by Apple retrospectively is more surprising still, as is the fact that the following 2017-002 security update only rectified the issue for El Capitan systems and not for Yosemite.

4. **There is very little visibility to the state of EFI firmware security for Apple systems.** From a variety of perspectives, access to information about EFI firmware security is hard or impossible to find. There are no published timelines for how long EFI firmware will be supported for firmware patches, or any lists of which systems are no longer going to receive firmware updates, despite continuing to receive software security updates. Visibility is also lacking for the admins and users of Mac systems who are not notified when their systems are running out-of-date firmware, nor are they able to find out in a direct fashion which vulnerabilities apply to their current version of EFI firmware.

5. **Mac sysadmins too often ignore the importance of EFI firmware updates, or actively remove them due to past issues with their deployment.** The process of applying EFI firmware updates used to be a laborious process that required hands-on interaction by IT support staff. Due to this, many Mac sysadmins over time decided to remove or disable the deployment of EFI firmware updates alongside OS or security updates, deciding to "deal with it" as needed. While at one point this may have been an acceptable workaround if combined with diligent manual application of EFI firmware updates, the current automated process of deploying EFI firmware updates alongside OS or security updates should be followed. Despite the possible gaps in coverage as described by this paper, it is still vastly superior to either not applying them at all or only at certain larger "catch-up" intervals or when a specific new EFI vulnerability is announced and patched.

As was noted at the start of this report, while Apple systems were the subject of this study, we fully expect the same issues, and in all likelihood far worse issues, with Wintel PC systems. We chose the Apple ecosystem to study as it's a far more controlled environment and therefore easier to analyze and arrive upon conclusions. The far more heterogeneous nature of the PC ecosystem likely exacerbates the issues related to visibility, QA, and availability of EFI firmware security patches. It would certainly be an area we will look into researching at a future date.

The advent of UEFI brought with it a far more modern pre-boot environment and finally put an end to the many years of legacy workarounds that had to be applied to the aging IBM BIOS 'standard,' providing a common, uniform and higher-level platform to innovate upon. However, that uniformity and accessibility also opened the door to far more generic and useful pre-boot environment attack opportunities. Much of the publicly available evidence suggests it has been an active, lucrative area of security research.

As the pre-boot environment becomes increasingly like a full OS in and of its own, it must be likewise be treated like a full OS in terms of the security support and attention applied to it. This attention goes beyond just releasing well QA'd EFI patches - it extends to the use of appropriate user and admin notifications to message the security status of the firmware alongside easy-to-apply remedial actions.

Overall, we are pleased to have been able to share what we learned with you alongside the datasets, APIs and tools[36] that will help provide better visibility into the security state of your Apple Mac fleet's EFI security.

References

[1] "Specifications and Tools | Unified Extensible Firmware ... - UEFI Forum." **http://www.uefi.org/specsandtesttools**. Accessed 1 Sep. 2017.

[2] "Beyond BIOS | Intel® Software." **https://software.intel.com/en-us/articles/beyond-bios**. Accessed 1 Sep. 2017.

[3] "IntroBIOS - Open Security Training." 14 Oct. 2015, **http://opensecuritytraining.info/IntroBIOS.html**. Accessed 1 Sep. 2017.

[4] "GitHub - advanced-threat-research/firmware-security-training." **https://github.com/advanced-threat-research/firmware-security-training**. Accessed 1 Sep. 2017.

[5] "BIOS Necromancy: Utilizing "Dead Code" for BIOS Attacks - LegbaCore." 14 Oct. 2015, **http://www.legbacore.com/Research_files/BIOSNecromancy.pdf**. Accessed 1 Sep. 2017.

[6] "assurance - Black Hat." **https://media.blackhat.com/bh-us-12/Briefings/Loukas_K/BH_US_12_LoukasK_De_Mysteriis_Dom_Jobsivs_Slides.pdf**. Accessed 1 Sep. 2017.

[7] "Thunderstrike 31c3 - Trammell Hudson's Projects." 15 Feb. 2015, **https://trmm.net/Thunderstrike_31c3**. Accessed 1 Sep. 2017.

[8] "Thunderstrike FAQ - Trammell Hudson's Projects." 31 Jan. 2015, **https://trmm.net/Thunderstrike_FAQ**. Accessed 1 Sep. 2017.

[9] "apple-sa-2015-01-27-4 - Apple - Lists.apple.com." **https://lists.apple.com/archives/security-announce/2015/Jan/msg00003.html**. Accessed 1 Sep. 2017.

[10] "CVE - CVE-2014-4498." **https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-4498**. Accessed 1 Sep. 2017.

[11] "Thunderstrike2 details - Trammell Hudson's Projects." 14 Aug. 2015, **https://trmm.net/Thunderstrike2_details**. Accessed 1 Sep. 2017.

[12] "CVE - CVE-2015-3692." **https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-3692**. Accessed 1 Sep. 2017.

[13] "CVE - CVE-2015-3693." **https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-3693**. Accessed 1 Sep. 2017.

[14] "WikiLeaks - Sonic Screwdriver." **https://wikileaks.org/vault7/document/SonicScrewdriver_1p0/**. Accessed 1 Sep. 2017.

[15] "WikiLeaks - DerStarke v1.4." **https://wikileaks.org/vault7/document/DerStarke_v1_4_DOC/**. Accessed 1 Sep. 2017.

[16] "presentations/DEFCON-24-Ulf-Frisk-Direct-Memory-Attack-the-Kernel ...." **https://github.com/ufrisk/presentations/blob/master/DEFCON-24-Ulf-Frisk-Direct-Memory-Attack-the-Kernel-Final.pdf**. Accessed 1 Sep. 2017.

[17] "Security | DMA | Hacking: macOS FileVault2 Password Retrieval." 15 Dec. 2016, **http://blog.frizk.net/2016/12/filevault-password-retrieval.html**. Accessed 1 Sep. 2017.

[18] "About EFI and SMC firmware updates for Intel-based ... - Apple Support." 4 May. 2016, **https://support.apple.com/en-us/HT201518**. Accessed 1 Sep. 2017.

[19] "bless Man Page - macOS - apple.com." **https://developer.apple.com/legacy/library/documentation/Darwin/Reference/ManPages/man8/bless.8.html**. Accessed 1 Sep. 2017.

[20] "Pike's Universum" **https://pikeralpha.wordpress.com**. Accessed 1 Sep. 2017.

[21] "Apple Hardware Test Download Links" **https://github.com/upekkha/AppleHardwareTest**. Accessed 1 Sep. 2017.

[22] "Customizing hardware model filters for netboot" **https://macops.ca/customizing-hardware-model-filters-for-netboot**. Accessed 1 Sep. 2017.

[23] "GitHub - LongSoft/UEFITool: UEFI firmware image viewer and editor." **https://github.com/LongSoft/UEFITool**. Accessed 1 Sep. 2017.

[24] "CharlesSoft — Pacifist" **https://www.charlessoft.com/cgi-bin/pacifist_download.cgi?type=zip**. Accessed 1 Sep. 2017.

[25] "EFIgy." **https://github.com/duo-labs/EFIgy** . Accessed 20 Sep. 2017.

[26] "Osquery." **https://osquery.io/**. Accessed 1 Sep. 2017.

[27] "CVE - CVE-2014-4498." **https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-4498**. Accessed 1 Sep. 2017.

[28] "CVE - CVE-2015-3692." **https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-3692**. Accessed 1 Sep. 2017.

[29] "CVE - CVE-2015-3693." **https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-3693**. Accessed 1 Sep. 2017.

[30] "CVE - CVE-2015-7035." **https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-7035**. Accessed 1 Sep. 2017.

[31] "CVE - CVE-2016-7585." **https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-7585**. Accessed 1 Sep. 2017.

[32] "About the security content of OS X El Capitan 10.11.1 ... - Apple Support." 30 Jun. 2017, **https://support.apple.com/en-us/HT205375**. Accessed 1 Sep. 2017.

[33] "CVE - CVE-2015-4860." **https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-4860**. Accessed 1 Sep. 2017.

[34] "rdar://32995209: Incorrect CVE listed in HT205375 ... - Open Radar." 26 Jun. 2017, **https://openradar.appspot.com/32995209**. Accessed 1 Sep. 2017.

[35] "EFIgy." **https://github.com/duo-labs/EFIgy** . Accessed 20 Sep. 2017.

[36] "EFIgy." **https://github.com/duo-labs/EFIgy** . Accessed 20 Sep. 2017.

Appendix A

Below are a series of tables related to the analysis in the preceding sections.

Table 1.
EFI Model ID to Apple Board ID Mapping
A complete table of all known EFI model to Board ID mappings is as follows:

EFI Model ID	Board ID
IM141	Mac-031B6874CF7F642A
IM142	Mac-27ADBB7B4CEE8E61
IM143	Mac-77EB7D7DAF985301
IM144	Mac-81E3E92DD6088272
IM151	Mac-42FD25EABCABB274, Mac-FA842E06C61E91C5
IM161	Mac-A369DDC4E67F1C45
IM162	Mac-FFE5EF870D7BA81A
IM171	Mac-DB15BD556843C820, Mac-B809C3757DA9BB8D, Mac-65CE76090165799A
IM181	Mac-4B682C642B45593E
IM183	Mac-77F17D7DA9285301, Mac-BE088AF8C5EB4FA2
MB101	Mac-EE2EBD4B90B839A8
MB81	Mac-BE0E8AC46FE800CC, Mac-F305150B0C7DEEEF
MB91	Mac-9AE82516C7C6B903
MBA61	Mac-35C1E88140C3E6CF, Mac-7DF21CB3ED6977E5
MBA71	Mac-9F18E312C5C2BF0B, Mac-937CB26E2E02BB01
MBP111	Mac-189A3D4F975D5FFC, Mac-D1FF70AF6D8C849A
MBP112	Mac-3CBD00234E554E41, Mac-2BD1B31983FE1663
MBP114	Mac-06F11FD93F0323C5, Mac-06F11F11946D27C5
MBP121	Mac-E43C1C25D4880AD6
MBP131	Mac-473D31EABEB93F9B
MBP132	Mac-66E35819EE2D0D05, Mac-1BDAB09B689867E2
MBP133	Mac-A5C67F76ED83108C
MBP141	Mac-B4831CEBD52A0C4C
MBP142	Mac-CAD6701F7CEA0921
MBP143	Mac-551B86E5744E2388
MM71	Mac-35C5E08120C7EEAF

Table 2.

Mapping of Mac Models Using EFI Firmware Files Named for Other Mac Models

Table showing the aliases between models of Mac that use firmware versions named after other Mac models. These firmware files contain the board-ID used to identify all of the systems they are compatible with, regardless of the filename.

Mac Model	Model of Mac EFI Firmware Used	Notes
IM113	IM112	
IM122	IM121	
IM132	IM131	
MBA32	MBA31	
MBA42	MBA41	
MBA52	MBA51	
MBA62	MBA61	
MBA72	MBA71	
MBP113	MBP112	
MBP115	MBP114	
MBP54	MBP53	MP53 is one of the firmwares that never appears to have been updated in the field, meaning MBP54 has never seen an update either
MBP62	MBP61	
MBP82	MBP81	
MBP83	MBP81	
MBP92	MBP91	
MM52	MM51	
MM53	MM51	
MM62	MM61	
IM182	IM183	Only a single datapoint for this model, but interesting to see IM182 use firmware named for IM183 and not IM181. All other aliased firmware seem to use firmware from models of a lower ID

Table 3.
Breakdown of Systems Running Different-Than-Expected EFI Versions, Based on OS Version

Mac Model	% Running Older-Than-Expected EFI Version	Raw Count of Systems Running Older EFI	Total Count of Systems Running Older EFI
iMac16,2	43.0%	941	2190
MacBookPro13,2	34.8%	114	328
MacBookPro13,1	28.5%	39	137
MacBookPro13,3	24.8%	78	314
MacBookPro8,2	14.9%	89	598
MacBookPro8,1	11.9%	59	498
Macmini3,1	11.5%	6	52
Macmini6,1	6.7%	13	194
iMac16,1	5.2%	15	287
MacBookAir6,1	5.0%	29	586
MacBook9,1	4.9%	10	206
Macmini7,1	4.8%	50	1035
MacBookAir4,1	4.4%	6	138
MacBookPro8,3	4.4%	3	69
iMac13,1	4.1%	86	2119
MacBookAir6,2	3.6%	81	2244
Macmini5,2	3.0%	4	135
MacBookPro9,1	2.8%	40	1419
iMac12,1	2.8%	75	2725
MacBookAir5,2	2.6%	16	619
MacBookPro9,2	2.4%	55	2257
MacBookPro12,1	2.3%	116	5048
MacBookPro11,2	2.3%	38	1659
MacBookAir4,2	2.2%	7	313
iMac17,1	2.2%	47	2103
MacPro6,1	2.2%	13	582
MacBookAir7,1	2.1%	19	894
MacBookPro11,1	1.9%	28	1458
MacBookPro11,4	1.8%	30	1635

Mac Model	% Running Older-Than-Expected EFI Version	Raw Count of Systems Running Older EFI	Total Count of Systems Running Older EFI
iMac11,3	1.7%	6	350
MacBook8,1	1.6%	5	309
MacBookPro11,3	1.6%	17	1080
iMac14,1	1.5%	42	2781
iMac14,4	1.4%	5	347
Macmini6,2	1.4%	8	563
MacBookAir5,1	1.3%	3	227
MacBookPro10,1	1.2%	16	1336
iMac14,2	1.1%	39	3602
iMac15,1	1.0%	11	1064
iMac11,2	0.8%	6	749
MacBookPro10,2	0.8%	4	529
iMac13,2	0.4%	9	2099
iMac14,3	0.1%	4	4644
iMac11,1	0.0%	0	1009
iMac12,2	0.0%	0	1547
MacBook5,1	0.0%	0	37
MacBook7,1	0.0%	0	10
MacBookAir3,1	0.0%	0	58
MacBookPro5,5	0.0%	0	80
MacBookPro6,1	0.0%	0	14
MacBookPro7,1	0.0%	0	237
Macmini4,1	0.0%	0	155
Macmini5,1	0.0%	0	44
Macmini5,3	0.0%	0	31

Table 4.

EFI Firmware for Security Update 2017-001, OS X 10.11

This table reflects EFI firmware contained in the Security Update 2017-001 for OS X 10.11 (El Capitan), showing the one-to-one match between its EFI binary versions and the versions shipped with the security update two versions prior.

Mac Model	**Security Update 2017-001 (10.11)** [Released March 27, 2017]	**Security Update 2016-003 (10.11)** [Released Dec 13, 2016]	**Security Update 2016-002 (10.11)** [Released Oct 24, 2016]
IM121	0047 23B	0047 25B	0047 23B
IM131	010A B09	010A B0A	010A B09
IM141	0118 B13	0118 B14	0118 B13
IM142	0118 B13	0118 B14	0118 B13
IM143	0118 B13	0118 B14	0118 B13
IM144	0179 B13	0179 B14	0179 B13
IM151	0207 B06	0207 B08	0207 B06
IM161	0207 B03	0207 B04	0207 B03
MB81	0164 B14	0164 B19	0164 B14
MB91	0154 B05	0154 B09	0154 B05
MBA41	077 B14	077 B15	077 B14
MBA51	00EF B04	00EF B05	00EF B04
MBA61	0099 B22	0099 B23	0099 B22
MBA71	0166 B12	0166 B13	0166 B12
MBP81	0047 2CB	0047 2DB	0047 2CB
MBP91	00D3 B0D	00D3 B0E	00D3 B0D
MBP101	00EE B0A	00EE B0B	00EE B0A
MBP102	0106 B0A	0106 B0B	0106 B0A
MBP112	0138 B17	0138 B18	0138 B17
MM51	0077 B14	0077 B15	0077 B14
MM61	0106 B0A	0106 B0B	0106 B0A
MM71	0220 B07	0220 B08	0220 B07
MP61	0116 B17	0116 B21	0116 B17

Table 5.

EFI Firmware for Security Update 2017-001 for OS X 10.10

This table reflects EFI firmware contained in the Security Update 2017-001 for OS X 10.10 (El Capitan), showing the one-to-one match between its EFI binary versions and the versions shipped with the security update two versions prior.

Mac Model	Security Update 2017-001 (10.10) [Released March 27, 2017]	Security Update 2016-007 (10.10) [Released Dec 13, 2016]	Security Update 2016-006 (10.10) [Released Oct 24, 2016]
IM121	0047 23B	0047 25B	0047 23B
IM131	010A B09	010A B0A	010A B09
IM141	0118 B12	0118 B14	0118 B12
IM142	0118 B12	0118 B14	0118 B12
IM143	0118 B12	0118 B14	0118 B12
IM144	0179 B12	0179 B14	0179 B12
IM151	0207 B05	0207 B08	0207 B05
MB81	0164 B09	0164 B19	0164 B09
MBA41	077 B14	077 B15	077 B14
MBA51	00EF B04	00EF B05	00EF B04
MBA61	0099 B20	0099 B23	0099 B20
MBA71	0166 B08	0166 B13	0166 B08
MBP81	0047 2CB	0047 2DB	0047 2CB
MBP91	00D3 B0C	00D3 B0E	00D3 B0C
MBP101	00EE B0A	00EE B0B	00EE B0A
MBP102	0106 B0A	0106 B0B	0106 B0A
MBP111	0138 B16	0138 B18	0138 B16
MBP112	0138 B16	0138 B18	0138 B16
MBP114	0172 B06	0172 B10	0172 B06
MBP121	0167 B14	0167 B18	0167 B14
MM51	0077 B14	0077 B15	0077 B14
MM61	0106 B0A	0106 B0B	0106 B0A
MM71	0220 B06	0220 B08	0220 B06
MP61	0116 B16	0116 B21	0116 B16

Table 6.
EFI Versions Patched for the CVE-2014-4498 Vulnerability and Observed Anomalies

Vulnerability / CVE :	Thunderstrike 1
CVE Number(s):	CVE-2014-4498
Updates Containing Patch:	10.10.2 Security Update 2015-001 (10.8, 10.9)
Date of Update(s):	2015/01/27

Mac Model	EFI Version	Update(s) Containing EFI
iMac14,1	IM141_0118_B09	10.10.2 Security Update 2015-001 (10.8, 10.9)
iMac14,2	IM142_0118_B09	10.10.2 Security Update 2015-001 (10.8, 10.9)
iMac14,3	IM143_0118_B09	10.10.2 Security Update 2015-001 (10.8, 10.9)
iMac14,4	IM144_0179_B08	10.10.2 Security Update 2015-001 (10.8, 10.9)
iMac15,1	IM151_0207_B01	10.10.2 Security Update 2015-001 (10.8, 10.9)
MacBookAir6,1	MBA61_0099_B18	10.10.2 Security Update 2015-001 (10.8, 10.9)
MacBookAir6,2	MBA61_0099_B18	10.10.2 Security Update 2015-001 (10.8, 10.9)
MacBookPro10,1	MBP101_00EE_B07	10.10.2 Security Update 2015-001 (10.8, 10.9)
MacBookPro10,2	MBP102_0106_B07	10.10.2 Security Update 2015-001 (10.8, 10.9)
MacBookPro11,1	MBP111_0138_B14	10.10.2 Security Update 2015-001 (10.8, 10.9)
MacBookPro11,2	MBP112_0138_B14	10.10.2 Security Update 2015-001 (10.8, 10.9)
MacBookPro11,3	MBP112_0138_B14	10.10.2 Security Update 2015-001 (10.8, 10.9)
Macmini7,1	MM71_0220_B01	10.10.2 Security Update 2015-001 (10.8, 10.9)
MacPro6,1	MP61_0116_B11	10.10.2 Security Update 2015-001 (10.8, 10.9)
MacPro6,2	MP61_0116_B11	10.10.2 Security Update 2015-001 (10.8, 10.9)

Table 7.
EFI Versions Patched for CVE-2015-3692 & CVE-2015-3692 and Observed Anomalies

Vulnerability:	Thunderstrike 2
CVE Number(s):	CVE-2015-3692 CVE-2015-3693
Updates Containing Patch:	10.10.4 Security Update 2015-005 (10.8, 10.9) EFI Security Update 2015-001
Date of Update:	2015/06/30 2015/06/30 (EFI update)

Mac Model	EFI Version	Update(s) Containing EFI
iMac12,1	IM121_0047_21B	10.10.4, Security Update 2015-005 EFI Update 2015-001
iMac12,2	IM121_0047_21B	10.10.4, Security Update 2015-005 EFI Update 2015-001
iMac13,1	IM131_010A_B08	10.10.4, Security Update 2015-005 EFI Update 2015-001
iMac13,2	IM131_010A_B08	10.10.4, Security Update 2015-005 EFI Update 2015-001
iMac14,1	IM141_0118_B11	10.10.4, Security Update 2015-005 EFI Update 2015-001
iMac14,2	IM142_0118_B11	10.10.4, Secuity Update 2015-005 EFI Update 2015-001
iMac14,3	IM143_0118_B11	10.10.4, Security Update 2015-005 EFI Update 2015-001
iMac14,4	IM144_0179_B10	10.10.4, Security Update 2015-005 EFI Update 2015-001
iMac15,1	IM151_0207_B03	10.10.4, EFI Update 2015-001
MacBook8,1	MB81_0164_B06	10.10.4, EFI Update 2015-001
MacBookAir4,1	MBA41_0077_B12	10.10.4, Security Update2015-005 EFI Update 2015-001
MacBookAir4,2	MBA41_0077_B12	10.10.4, Security Update2015-005 EFI Update 2015-001
MacBookAir5,1	MBA51_00EF_B03	10.10.4, Security Update 2015-005 EFI Update 2015-001
MacBookAir5,2	MBA51_00EF_B03	10.10.4, Security Update 2015-005 EFI Update 2015-001
MacBookAir6,1	MBA61_0099_B19	10.10.4, Security Update 2015-005 EFI Update 2015-001
MacBookAir6,2	MBA61_0099_B19	10.10.4, Security Update 2015-005 EFI Update 2015-001

Mac Model	EFI Version	Update(s) Containing EFI
MacBookAir7,1	MBA71_0166_B06	10.10.4, Security Update 2015-005 EFI Update 2015-001
MacBookPro8,1	MBP81_0047_2AB	10.10.4, Security Update 2015-005 EFI Update 2015-001
MacBookPro8,2	MBP81_0047_2AB	10.10.4, Security Update 2015-005 EFI Update 2015-001
MacBookPro8,3	MBP81_0047_2AB	10.10.4, Security Update 2015-005 EFI Update 2015-001
MacBookPro9,1	MBP91_00D3_B0B	10.10.4, Security Update 2015-005 EFI Update 2015-001
MacBookPro9,2	MBP91_00D3_B0B	10.10.4, Security Update 2015-005 EFI Update 2015-001
MacBookPro10,1	MBP101_00EE_B09	10.10.4, Security Update 2015-005 EFI Update 2015-001
MacBookPro10,2	MBP102_0106_B08	10.10.4, Security Update 2015-005 EFI Update 2015-001
MacBookPro11,1	MBP111_0138_B15	10.10.4, Security Update 2015-005 EFI Update 2015-001
MacBookPro11,2	MBP112_0138_B15	10.10.4, Security Update 2015-005 EFI Update 2015-001
MacBookPro11,3	MBP112_0138_B15	10.10.4, Security Update 2015-005 EFI Update 2015-001
MacBookPro11,4	MBP114_0172_B04	10.10.4, Security Update 2015-005 EFI Update 2015-001
MacBookPro12,1	MBP121_0167_B07	10.10.4, Security Update2015-005 EFI Update 2015-001
Macmini5,1	MM51_0077_B12	10.10.4, Security Update2015-005 EFI Update 2015-001
Macmini5,2	MM51_0077_B12	10.10.4, Security Update2015-005 EFI Update 2015-001
Macmini5,3	MM51_0077_B12	10.10.4, Security Update2015-005 EFI Update 2015-001
Macmini6,1	MM61_0106_B08	10.10.4, Security Update2015-005 EFI Update 2015-001
Macmini6,2	MM61_0106_B08	10.10.4, Security Update2015-005 EFI Update 2015-001
Macmini7,1	MM71_0220_B03	10.10.4, Security Update2015-005 EFI Update 2015-001
MacPro6,1	MP61_0116_B15	10.10.4, Security Update2015-005 EFI Update 2015-001

Table 8.
EFI Versions Patching the CVE-2015-7035 Vulnerability and Observed Anomalies

Vulnerability:	A local authenticated attacker may be able to execute arbitrary code with the privileges of system firmware, potentially allowing for persistent firmware level rootkits, bypassing of Secure Boot, or permanently DoS'ing the platform.
CVE Number(s):	**CVE-2015-7035**
Update Containing Patch:	10.11.1 Security Update 2015-004 (10.10) EFI Security Update 2015-002
Date of Update:	**2015/10/21**

Mac Model	EFI Version	Update(s) Containing EFI
iMac11,1	IM111_0034_04B	10.11.1 Security Update 2015-004 EFI Security Update 2015-002
iMac11,2	IM112_0057_03B	10.11.1 Security Update 2015-004 EFI Security Update 2015-002
iMac11,3	IM112_0057_03B	10.11.1 Security Update 2015-004 EFI Security Update 2015-002
iMac12,1	IM121_0047_21B	10.11.1 Security Update 2015-004 EFI Security Update 2015-002
iMac12,2	IM121_0047_B21	10.11.1 Security Update 2015-004 EFI Security Update 2015-002
iMac13,1	IM131_010A_B09	10.11.1 Security Update 2015-004 EFI Security Update 2015-002
iMac13,2	IM131_010A_B09	10.11.1 Security Update 2015-004 EFI Security Update 2015-002
iMac13,1	IM131_010A_B09	10.11.1 Security Update 2015-004 EFI Security Update 2015-002
iMac14,1	IM141_0118_B12	10.11.1 Security Update 2015-004 EFI Security Update 2015-002
iMac14,2	IM142_0118_B12	10.11.1 Security Update 2015-004 EFI Security Update 2015-002

Mac Model	EFI Version	Update(s) Containing EFI
iMac14,3	IM143_0118_B12	10.11.1 Security Update 2015-004 EFI Security Update 2015-002
iMac14,4	IM144_0179_B12	10.11.1 Security Update 2015-004 EFI Security Update 2015-002
iMac15,1	IM151_0207_B05	10.11.1 Security Update 2015-004
iMac16,1	IM161_0207_B01	10.11.1
iMac17,1	IM171_0105_B04	10.11.1
MacBook8,1	MB81_0164_B09	10.11.1 Security Update 2015-004
MacBook8,2	MB81_0164_B09	10.11.1 Security Update 2015-004
MacBook8,3	MB81_0164_B09	10.11.1 Security Update 2015-004
MacBookAir4,1	MBA41_0077_B12	10.11.1 EFI Security Update 2015-002
MacBookAir4,2	MBA41_0077_B12	10.11.1 EFI Security Update 2015-002
MacBookAir5,1	MBA51_00EF_B04	10.11.1 Security Update 2015-004 EFI Security Update 2015-002
MacBookAir5,2	MBA51_00EF_B04	10.11.1 Security Update 2015-004 EFI Security Update 2015-002
MacBookAir6,1	MBA61_0099_B20	10.11.1 Security Update 2015-004 EFI Security Update 2015-002
MacBookAir6,2	MBA61_0099_B20	10.11.1 Security Update 2015-004 EFI Security Update 2015-002
MacBookAir7,1	MBA71_0166_B08	10.11.1 Security Update 2015-004
MacBookAir7,2	MBA71_0166_B08	10.11.1 Security Update 2015-004
MacBookPro6,1	MBP61_0057_11B	10.11.1 Security Update 2015-004 EFI Security Update 2015-002
MacBookPro6,2	MBP61_0057_11B	10.11.1 Security Update 2015-004 EFI Security Update 2015-002
MacBookPro8,1	MBP81_0047_2AB	10.11.1 EFI Security Update 2015-002
MacBookPro8,2	MBP81_0047_2AB	10.11.1 EFI Security Update 2015-002

Mac Model	EFI Version	Update(s) Containing EFI
MacBookPro8,3	MBP81_0047_2AB	10.11.1 EFI Security Update 2015-002
MacBookPro9,1	MBP91_00D3_B0C	10.11.1 Security Update 2015-004 EFI Security Update 2015-002
MacBookPro9,2	MBP91_00D3_B0C	10.11.1 Security Update 2015-004 EFI Security Update 2015-002
MacBookPro10,1	MBP101_00EE_B0A	10.11.1 Security Update 2015-004 EFI Security Update 2015-002
MacBookPro10,2	MBP102_0106_B0A	10.11.1 Security Update 2015-004 EFI Security Update 2015-002
MacBookPro11,1	MBP111_0138_B16	10.11.1 Security Update 2015-004 EFI Security Update 2015-002
MacBookPro11,2	MBP112_0138_B16	10.11.1 Security Update 2015-004 EFI Security Update 2015-002
MacBookPro11,3	MBP112_0138_B16	10.11.1 Security Update 2015-004 EFI Security Update 2015-002
MacBookPro11,4	MBP114_0172_B06	10.11.1 Security Update 2015-004
MacBookPro11,5	MBP114_0172_B06	10.11.1 Security Update 2015-004
MacBookPro12,1	MBP121_0167_B14	10.11.1 Security Update 2015-004
Macmini5,1	MM51_0077_B12	10.11.1 EFI Security Update 2015-002
Macmini5,2	MM51_0077_B12	10.11.1 EFI Security Update 2015-002
Macmini5,3	MM51_0077_B12	10.11.1 EFI Security Update 2015-002
Macmini6,1	MM61_0106_B0A	10.11.1 Security Update 2015-004 EFI Security Update 2015-002
Macmini6,2	MM61_0106_B0A	10.11.1 Security Update 2015-004 EFI Security Update 2015-002
Macmini7,1	MM71_0220_B06	10.11.1 Security Update 2015-004
MacPro6,1	MP61_0116_B16	10.11.1 Security Update 2015-004 EFI Security Update 2015-002

Table 9.
EFI Versions Patching the CVE-2016-7585 Vulnerability and Observed Anomalies

Vulnerability:	DMA Attack
CVE Number(s):	CVE-2016-7585
Update Containing Patch:	10.12.4 Security Update 2017-001 (10.11, 10.10)
Date of Update:	2017/03/27

Mac Model	EFI Version	Update(s) Containing EFI
iMac11,1	IM111_0034_04B	10.12.4 Security Update 2017-001 (10.11, 10.10)
iMac11,2	IM112_0057_B09	10.12.4
	IM112_0057_03B	**Security Update 2017-001 (10.11, 10.10)**
iMac11,3	IM112_0057_B09	10.12.4
	IM112_0057_03B	**Security Update 2017-001 (10.11, 10.10)**
iMac12,1	IM121_0047_B29	10.12.4
	IM121_0047_B23	**Security Update 2017-001 (10.11)**
	IM121_0047_23B	**Security Update 2017-001 (10.10)**
iMac12,2	IM121_0047_B29	10.12.4
	IM121_0047_B23	**Security Update 2017-001 (10.11, 10.10)**
iMac13,1	IM131_010A_B11	10.12.4
	IM131_010A_B09	**Security Update 2017-001 (10.11, 10.10)**
iMac13,2	IM131_010A_B11	10.12.4
	IM131_010A_B09	**Security Update 2017-001 (10.11, 10.10)**
iMac14,1	IM141_0118_B20	10.12.4
	IM141_0118_B13	**Security Update 2017-001 (10.11)**
	IM141_0118_B12	**Security Update 2017-001 (10.10)**
iMac14,2	IM142_0118_B20	10.12.4
	IM141_0118_B13	**Security Update 2017-001 (10.11)**
	IM141_0118_B12	**Security Update 2017-001 (10.10)**
iMac14,3	IM143_0118_B20	10.12.4
	IM141_0118_B13	**Security Update 2017-001 (10.11)**
	IM141_0118_B12	**Security Update 2017-001 (10.10)**
iMac14,4	IM144_0179_B21	10.12.4

Mac Model	EFI Version	Update(s) Containing EFI
	IM141_0118_B13	Security Update 2017-001 (10.11)
	IM141_0118_B12	Security Update 2017-001 (10.10)
iMac15,1	IM151_0207_B16	10.12.4
	IM151_0207_B06	Security Update 2017-001 (10.11)
	IM151_0207_B05	Security Update 2017-001 (10.10)
iMac16,1	IM161_0207_B11	10.12.4
	IM161_0207_B03	Security Update 2017-001 (10.11)
iMac16,2	IM162_0207_B11	10.12.4
iMac17,1	IM171_0105_B20	10.12.4
	IM171_0105_B08	Security Update 2017-001 (10.11)
MacBook7,1	MB71_0039_B15	10.12.4
MacBook8,1	MB81_0164_B25	10.12.4
	MB81_0164_B14	Security Update 2017-001 (10.11)
	MB81_0164_B09	Security Update 2017-001 (10.10)
MacBook9,1	MB91_0154_B17	10.12.4
	MB91_0154_B05	Security Update 2017-001 (10.11)
MacBookAir3,1	MBA31_0061_B0E	10.12.4
MacBookAir4,1	MBA41_0077_B1B	10.12.4
	MBA41_0077_B14	Security Update 2017-001 (10.11, 10.10)
MacBookAir4,2	MBA41_0077_B1B	10.12.4
	MBA41_0077_B14	Security Update 2017-001 (10.11, 10.10)
MacBookAir5,1	MBA51_00EF_B0C	10.12.4
	MBA51_00EF_B04	Security Update 2017-001 (10.11, 10.10)
MacBookAir5,2	MBA51_00EF_B0C	10.12.4
	MBA51_00EF_B04	Security Update 2017-001 (10.11, 10.10)
MacBookAir6,1	MBA61_0099_B33	10.12.4
	MBA61_0099_B22	Security Update 2017-001 (10.11)
	MBA61_0099_B20	Security Update 2017-001 (10.10)
MacBookAir6,2	MBA61_0099_B33	10.12.4
	MBA61_0099_B22	Security Update 2017-001 (10.11)
	MBA61_0099_B20	Security Update 2017-001 (10.10)
MacBookAir7,1	MBA71_0166_B19	10.12.4
	MBA71_0166_B12	Security Update 2017-001 (10.11)

Mac Model	EFI Version	Update(s) Containing EFI
	MBA71_0166_B08	Security Update 2017-001 (10.10)
MacBookAir7,2	MBA71_0166_B19	10.12.4
	MBA71_0166_B12	Security Update 2017-001 (10.11)
	MBA71_0166_B08	Security Update 2017-001 (10.10)
MacBookPro6,1	MBP61_0057_B17	10.12.4
	MBP61_0057_11B	Security Update 2017-001 (10.11, 10.10)
MacBookPro6,2	MBP61_0057_B17	10.12.4
	MBP61_0057_11B	Security Update 2017-001 (10.11, 10.10)
MacBookPro7,1	MBP71_0039_B15	10.12.4
MacBookPro8,1	MBP81_0047_32B	10.12.4
	MBP81_0047_2CB	Security Update 2017-001 (10.11, 10.10)
MacBookPro8,2	MBP81_0047_32B	10.12.4
	MBP81_0047_2CB	Security Update 2017-001 (10.11, 10.10)
MacBookPro8,3	MBP81_0047_32B	10.12.4
	MBP81_0047_2CB	Security Update 2017-001 (10.11, 10.10)
MacBookPro9,1	MBP91_00D3_B15	10.12.4
	MBP91_00D3_B0D	Security Update 2017-001 (10.11)
	MBP91_00D3_B0C	Security Update 2017-001 (10.10)
MacBookPro9,2	MBP91_00D3_B15	10.12.4
	MBP91_00D3_B0D	Security Update 2017-001 (10.11)
	MBP91_00D3_B0C	Security Update 2017-001 (10.10)
MacBookPro10,1	MBP101_00EE_B12	10.12.4
	MBP101_00EE_B0A	Security Update 2017-001 (10.11, 10.10)
MacBookPro10,2	MBP102_0106_B12	10.12.4
	MBP101_00EE_B0A	Security Update 2017-001 (10.11, 10.10)
MacBookPro11,1	MBP111_0138_B25	10.12.4
	MBP111_0138_B17	Security Update 2017-001 (10.11)
	MBP111_0138_B16	Security Update 2017-001 (10.10)
MacBookPro11,2	MBP112_0138_B25	10.12.4
	MBP111_0138_B17	Security Update 2017-001 (10.11)
	MBP111_0138_B16	Security Update 2017-001 (10.10)
MacBookPro11,3	MBP112_0138_B25	10.12.4
	MBP111_0138_B17	Security Update 2017-001 (10.11)

Mac Model	EFI Version	Update(s) Containing EFI
	MBP111_0138_B16	Security Update 2017-001 (10.10)
MacBookPro11,4	MBP114_0172_B16	10.12.4
	MBP114_0172_B09	Security Update 2017-001 (10.11)
	MBP114_0172_B06	Security Update 2017-001 (10.10)
MacBookPro11,5	MBP114_0172_B16	10.12.4
	MBP114_0172_B09	Security Update 2017-001 (10.11)
	MBP114_0172_B06	Security Update 2017-001 (10.10)
MacBookPro12,1	MBP121_0167_B24	10.12.4
	MBP121_0167_B17	Security Update 2017-001 (10.11)
	MBP121_0167_B14	Security Update 2017-001 (10.10)
MacBookPro13,1	MBP131_0205_B15	10.12.4
MacBookPro13,2	MBP132_0226_B15	10.12.4
MacBookPro13,3	MBP133_0226_B15	10.12.4
Macmini4,1	MM41_0042_B09	10.12.4
Macmini5,1	MM51_0077_B1B	10.12.4
	MM51_0077_B14	Security Update 2017-001 (10.11, 10.10)
Macmini5,2	MM51_0077_B1B	10.12.4
	MM51_0077_B14	Security Update 2017-001 (10.11, 10.10)
Macmini5,3	MM51_0077_B1B	10.12.4
	MM51_0077_B14	Security Update 2017-001 (10.11, 10.10)
Macmini6,1	MM61_0106_B12	10.12.4
	MM61_0106_B0A	Security Update 2017-001 (10.11, 10.10)
Macmini6,2	MM61_0106_B12	10.12.4
	MM61_0106_B0A	Security Update 2017-001 (10.11, 10.10)
Macmini7,1	MM71_0220_B14	10.12.4
	MM71_0220_B07	Security Update 2017-001 (10.11)
	MM71_0220_B06	Security Update 2017-001 (10.10)
MacPro6,1	MP61_0116_B25	10.12.4
	MP61_0116_B17	Security Update 2017-001 (10.11)
	MP61_0116_B16	Security Update 2017-001 (10.10)

Table 10.
Highest Released Versions of EFI Firmware,
Segmented by Major OS Version and Mac Model

	10.10.x	10.11.x	10.12.x	10.13.x
IM101			00CF 00B	00CF 00B
IM111	0034 04B	0034 04B	0037 00B	0037 00B
IM112	0057 03B	0057 09B	005B 00B	005B 00B
IM121	0047 25B	0047 29B	004D 00B	004D 00B
IM131	010A B0A	010A B11	010F B00	010F B00
IM141	0118 B14	0118 B20	0122 B00	0123 B00
IM142	0118 B14	0118 B20	0122 B00	0123 B00
IM143	0118 B14	0118 B20	0122 B00	0123 B00
IM144	0179 B14	0179 B21	0183 B00	0183 B00
IM151	0207 B08	0207 B16	0211 B00	0211 B00
IM161	-	0207 B11	0212 B00	0212 B00
IM162	-	0207 B11	0212 B00	0212 B00
IM171	-	0105 B20	0110 B00	0110 B00
IM181	-	-	0151 B00	0151 B00
IM183	-	-	0151 B00	0151 B00
MB61	-	-	00CB 00B	00CB 00B
MB71	-	-	003D 00B	003D 00B
MB81	0164 B19	0164 B25	0168 B00	0168 B00
MB91	-	0154 B17	0159 B00	0159 B00
MB101	-	-	0154 B00	0154 B00
MBA31			0067 00B	0067 00B
MBA41	0077 B15	0077 B1B	007B B00	007B B00
MBA51	00EF B05	00EF B0C	00F4 B00	00F4 B00
MBA61	0099 B23	0099 B33	0103 B00	0103 B00
MBA71	0166 B13	0166 B19	0171 B00	0171 B00
MBP61	0057 11B	0057 17B	005A 00B	005A 00B
MBP71	-	-	003D 00B	003D 00B
MBP81	0047 2DB	0047 32B	004D 00B	004D 00B
MBP91	00D3 B0E	00D3 B15	00D7 B00	00D7 B00
MBP101	00EE B0B	00EE B12	00F2 B00	00F2 B00
MBP102	0106 B0B	0106 B12	010B B00	010B B00
MBP111	0138 B18	0138 B25	0142 B00	0142 B00
MBP112	0138 B18	0138 B25	0142 B00	0142 B00
MBP114	0172 B10	0172 B16	0177 B00	0177 B00
MBP121	0167 B18	0167 B24	0171 B00	0171 B00
MBP131	-	-	0212 B00	0212 B00
MBP132	-	-	0233 B00	0233 B00
MBP133	-	-	0233 B00	0233 B00
MBP141	-	-	0167 B00	0167 B00
MBP142	-	-	0167 B00	0167 B00
MBP143	-	-	0167 B00	0167 B00
MM41	-	-	0045 00B	0045 00B
MM51	0077 B15	0077 B1B	007B B00	007B B00
MM61	0106 B0B	0106 B12	010B B00	010B B00
MM71	0220 B08	0220 B14	0224 B00	0226 B00
MP51	-	-	0084 00B	0084 00B
MP61	0116 B21	0116 B25	0120 B00	0120 B00

Table 11.

Potentially Vulnerable Mac Models With Low Build Numbers

This table lists Mac models from the real world dataset that have only been observed with one, two or three updates with low build numbers. This suggests they haven't been updated from the versions of firmware they were originally shipped with from the factory - making them likely to contain unpatched vulnerabilities.

Mac Model	EFI Versions Observed in the Real World Data		
IM101	IM101.00CC.B00		
IM71	IM71.007A.B00	IM71.007A.B01	IM71.007A.B03
IM81	IM81.00C1.B00		
IM91	IM91.008D.B00	IM91.008D.B04	IM91.008D.B08
MB51	MB51.007D.B03	MB51.0073.B06	
MB52	MB52.0088.B06		
MB61	MB61.00C8.B00		
MBA21	MBA21.0075.B03	MBA21.0075.B05	
MBP31	MBP31.0070.B07		
MBP41	MBP41.00C1.B03		
MBP51	MBP51.007E.B05	MBP51.007E.B06	
MBP52	MBP52.008E.B05		
MBP53	MBP53.00AC.B03		
MBP55	MBP55.00AC.B03		
MM31	MM31.00AD.B00	MM31.0081.B06	
MP31	MP31.006C.B02	MP31.006C.B05	
MP41	MP41.0081.B04	MP41.0081.B07	MP41.0081.B08
MP51	MP51.007F.B00	MP51.007F.B01	MP51.007F.B03

Table 12.

Heat Map of EFI Updates for Public Vulnerabilities

(duo.sc/EFI-heatmap)

Appendix B

The disclosure timeline related to the findings of our research and our communications with Apple are below, Duo Labs' responsible disclosure policy and further information can be found here **duo.com/labs/disclosure**.

Date	Action
June 26, 2017	Radar item 32995209 **raised** citing the incorrect CVE being cited for issue APPLE-SA-2015-10-21-4 in the release notes of OS X El Capitan 10.11.1 and Security Update 2015-007.
June 28, 2017	Initial contact with Apple sending a PDF summary of the research findings as well as offers of discussing the problems and extra context.
July 14, 2017	Follow up email sent to Apple contacts enquiring if Apple had any questions or concerns we may be able to help with. We also requested technical clarification on a point within the 10.13 beta 3 release note.
August 2, 2017	Follow up email sent to Apple contacts enquiring if Apple had any questions or concerns we may be able to help with. August 31, 2017 was given as the date after which our findings would no longer be private.
September 5, 2017	Follow up email sent to Apple contacts enquiring if Apple had any questions or concerns we may be able to help with. Informed Apple that a presentation of our research would be publicly released at Ekoparty on September 29, 2017 in Buenos Aires.
	Draft version of the research paper sent to our Apple contacts to allow them to see its content and respond with any questions before it's released publicly.
	Apple product security team reached out to ask for further technical details; the draft research paper was sent over to them.
September 11, 2017	Follow up email sent to Apple product security team enquiring if Apple had any questions or concerns we may be able to help with.
September 15, 2017	Follow up email sent to Apple product security team enquiring if Apple had any questions or concerns we may be able to help with.
September 16, 2017	Response from Apple product security team acknowledging receipt and setting up a phone call to discuss on September 19, 2017.
September 19, 2017	Phone call with the Apple product security team discussing the research and the content of the paper.
September 29, 2017	First public release of this paper and presentation discussing the work given at Ekoparty. 90+ days since initially contacting Apple with details of the research findings.

Appendix C

Document Versions

Date	Version	Notes
Sept 29 2017	1.0	Initial release of the paper at Ekoparty 13
Nov 30 2017	1.1	Updates to cover macOS 10.13, 10.13.1, OS X 10.12.6 Security Update 2017-001, and 10.11.6 Security Update 2017-004. Release for Blackhat EU 2017

About the Authors

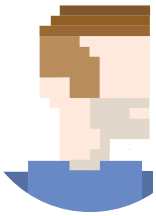
Pepijn Bruienne

@bruienne

Research & Development Engineer

Pepijn Bruienne is an R&D Engineer at Duo Security and a former long-time Mac Admin who recently made the jump from administering Macs to breaking them in order to better protect them for his employer's customers.

Prior to that he worked for the University of Michigan as a senior Mac operations and development specialist, at Cengage Learning as a Senior Mac systems administrator and various other smaller Mac-based shops in a darker past. He has written a number of FOSS tools for Mac admins and contributed to a number of other projects as well.



Rich Smith

@iodboi

Director of Research & Development

Rich Smith is the Director of R&D for Duo Labs, supporting the advanced security research agenda for Duo Security. Prior to joining Duo, Rich was Director of Security at Etsy, co-founder of Icelandic red team startup Syndis, and has held various roles on security teams at Immunity Inc., Kyrus, Morgan Stanley, and HP Labs among others.

Rich has worked professionally in the security space since the late 90s covering a range of activities including building security organizations, security consulting, penetration testing, red teaming, offensive research, and developing exploits and attack tooling. More recently, Rich co-authored a new book for O'Reilly titled Agile Application Security: Enabling Security in a Continuous Delivery Pipeline. He has worked in both the public and private sectors in the U.S., Europe, and Scandinavia, and currently spends most of his time bouncing between Detroit, Reykjavik and NYC.

Our mission is to protect your mission.

Experience advanced two-factor authentication, endpoint visibility, custom user policies & more with your free 30 day trial.

Try it today at **duo.com**.

Duo Security makes security painless, so you can focus on what's important. Our scalable, cloud-based **Trusted Access** platform addresses security threats before they become a problem, by verifying the identity of your users and the health of their devices before they connect to the applications you want them to access.

Thousands of organizations worldwide use Duo, including Facebook, Toyota, Panasonic and MIT. Duo is backed by Google Ventures, True Ventures, Radar Partners, Redpoint Ventures and Benchmark. We're located from coast to coast and across the sea.

Follow @duosec and @duo_labs on Twitter.

The Trusted Access Company

duo.com