Evil Never Sleeps: When Wireless Malware Stays On After Turning Off iPhones

Jiska Classen
Secure Mobile Networking Lab, TU Darmstadt
Germany
jclassen@seemoo.de

Alexander Heinrich
Secure Mobile Networking Lab, TU Darmstadt
Germany
aheinrich@seemoo.de

Robert Reith
Secure Mobile Networking Lab, TU Darmstadt
Germany
rreith@seemoo.de

Matthias Hollick
Secure Mobile Networking Lab, TU Darmstadt
Germany
mhollick@seemoo.de

ABSTRACT

When an iPhone is turned off, most wireless chips stay on. For instance, upon user-initiated shutdown, the iPhone remains locatable via the Find My network. If the battery runs low, the iPhone shuts down automatically and enters a power reserve mode. Yet, users can still access credit cards, student passes, and other items in their Wallet. We analyze how Apple implements these standalone wireless features, working while iOS is not running, and determine their security boundaries. On recent iPhones, Bluetooth, Near Field Communication (NFC), and Ultra-wideband (UWB) keep running after power off, and all three wireless chips have direct access to the secure element. As a practical example what this means to security, we demonstrate the possibility to load malware onto a Bluetooth chip that is executed while the iPhone is off.

CCS CONCEPTS

• **Security and privacy** → **Mobile and wireless security**; **Software reverse engineering**.

KEYWORDS

Malware, Find My, Digital Car Key, Express Mode, Low-Power Mode, Secure Element, Near Field Communication, Ultra-wideband, Bluetooth

ACM Reference Format:
Jiska Classen, Alexander Heinrich, Robert Reith, and Matthias Hollick. 2022. Evil Never Sleeps: When Wireless Malware Stays On After Turning Off iPhones. In *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '22), May 16–19, 2022, San Antonio, TX, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3507657.3528547

1 INTRODUCTION

Wireless chips on the iPhone can run in a so-called Low-Power Mode (LPM). Note that LPM described in this paper is different from the energy saving mode indicated by a yellow battery icon. In LPM, the iPhone does not react to tapping the screen or shaking. This mode is either activated when the user switches off their phone or when iOS shuts down automatically due to low battery. If the iPhone was switched off by the user, it turns on when pressing the power button. In battery-low power reserve mode, pressing the power button only activates the screen for a few seconds. It indicates the low battery and lists the currently active LPM features, as shown in Figure 1.

LPM features increase the user's security, safety, and convenience in most situations. Find My is available after user-initiated and battery-low shutdown. A user who loses their iPhone while it is turned off can locate it using the Bluetooth-based Find My network [26]. Express Mode supports selected student, travel, and credit cards, as well as digital keys to be used faster without additional authentication by the user. Upon battery-low shutdown, these cards and keys can be used for up to 5 h [8]. Initially, this only required the NFC to stay on. Instead, the new Digital Car Key (DCK) 3.0 protocol uses Bluetooth and UWB, and it also supports Express Mode in power reserve [4]. Express Mode protects users against locking themselves out of their cars and homes and being unable to make payments.

LPM support is implemented in hardware. The Power Management Unit (PMU) can turn on chips individually. The Bluetooth and UWB chips are hardwired to the Secure Element (SE) in the NFC chip, storing secrets that should be available in LPM. Since LPM support is implemented in hardware, it cannot be removed by changing software components. As a result, on modern iPhones, wireless

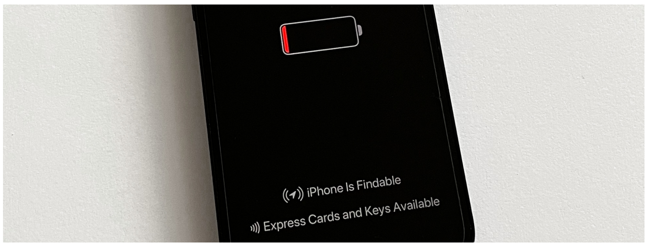


Figure 1: Find My and Express Cards in power reserve mode.

chips can no longer be trusted to be turned off after shutdown. This poses a new threat model. Previous work only considered that journalists are not safe against espionage when enabling airplane mode in case their smartphones were compromised [30]. LPM is significantly more stealthy than a fake power off that only disables the screen. Even though the National Security Agency (NSA) used fake power off on smart TVs for espionage [57], high battery drainage would be noticeable and could reveal such implants on mobile devices. We show that LPM is a relevant attack surface that has to be considered by high-value targets such as journalists, or that can be weaponized to build wireless malware operating on shutdown iPhones.

We are the first to analyze Apple's LPM features. We reverse-engineer multiple wireless daemons and firmware components to systematically analyze LPM. Our key contributions are:

- We perform a security analysis of new LPM features introduced in iOS 15,
- we identify flaws in the Find My LPM implementation, limiting the total advertisement time to 24 h even after reboot,
- we design, implement, and open-source tooling to analyze and modify Bluetooth firmware on recent iPhones, and
- we demonstrate that Bluetooth LPM firmware can be modified to run malware.

We responsibly disclosed all issues to Apple. They read the paper prior to publication but had no feedback on the paper's contents. The tooling required for firmware analysis and modification on modern iPhones is available as part of the *InternalBlue* [1] and *Frankenstein* [2] repositories.

The remainder of this paper is structured as follows. Apple describes how NFC LPM is supposed to work, which we summarize in Section 2. Then, we explain undocumented LPM Bluetooth and UWB features in Section 3. Based on this knowledge, we analyze threats posed by LPM in Section 4. In Section 5, we demonstrate that it is possible to modify the Bluetooth firmware loaded while the chip is in LPM. Our reverse-engineering methods are described in Section 6 to ensure reproducibility of our results. We discuss related work in Section 7 and conclude our paper in Section 8.

2 NFC LOW-POWER MODE

From a hardware perspective, enabling Bluetooth, UWB, or other chips like NFC while the remaining iPhone is off are very similar. Bluetooth and UWB LPM features are undocumented and have to the best of our knowledge not been investigated, yet. However, Apple's platform security guide [3] documents an NFC LPM feature called Express Card, introduced in iOS 12. Hardware components supporting Express Cards and enforcing security boundaries are shown in Figure 2. We describe these in the following.

2.1 Security Boundaries

iOS implements secure wireless payments on an NFC chip with a Secure Element (SE). The SE is certified based on Common Criteria and runs a JavaCard platform [3, p. 140]. This platform executes applets, e.g., Apple Pay [3, p. 141], DCK [3, p. 169], or other third-party applets. During setup, applets are personalized. Personalization can

[1]https://github.com/seemoo-lab/internalblue
[2]https://github.com/seemoo-lab/frankenstein

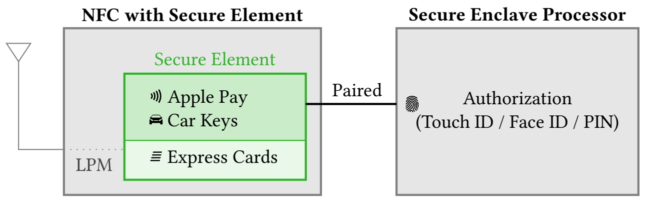


Figure 2: Secure Element (SE) and Secure Enclave Processor (SEP) usage as documented by Apple.

have additional security boundaries, e.g., no credit card information is stored on a payment applet but a revokable identifier known to the payment network [3, p. 145]. Applets, including their secrets, are separated from each other. They are stored within the SE and do not leave it. During a payment process or similar wireless transactions, the SE within the NFC chip replies directly without forwarding this data to iOS [3, p. 141]. Thus, even if iOS or apps are compromised, credit cards and other keys cannot be stolen from the SE [44].

The SE is connected to the Secure Enclave Processor (SEP). The SEP authorizes payments by enforcing that the user authenticates using Touch ID, Face ID, or a passcode [3, p. 141]. Note that the SEP does not store data within the SE but uses a separate secure storage component [3, p. 15]. The SE and SEP are paired in-factory and, thus, can communicate encrypted and authenticated [3, p. 144].

2.2 Express Cards and Power Reserve

The Wallet app allows configuring selected NFC credit, travel, and student cards as well as keys for Express Mode. An Express Card does no longer require authorization via the SEP [3, p. 153f], enabling fast and convenient payment without unlocking the iPhone. Possessing the iPhone is the only authorization in Express Mode. Skipping authorization means that neither the SEP nor iOS are required to complete an NFC transaction, and NFC can operate standalone using SE applets.

Once an iPhone runs out of battery and if the user has an Express Card, the iPhone shuts down but keeps the NFC chip powered for up to 5 h [8]. This enables the user to use their transit and credit cards as well as car keys until they can charge their iPhone. During this period, the display indicates Express Cards are enabled when pressing the power button, as shown in Figure 1. Express Cards are only enabled in power reserve mode. A user-initiated shutdown also powers off NFC [3, p. 28].

3 BLUETOOTH AND UWB LPM

iOS 15 introduces two new LPM features: *(i)* Find My, Apple's Bluetooth Low Energy (BLE)-based offline finding network, and *(ii)* Digital Car Key (DCK) 3.0 support, which uses UWB for a secure distance measurement. Thus, also the Bluetooth and the UWB chip are able to operate standalone while iOS is powered off. These capabilities are undocumented and have not been researched before. To improve readability, we explain our reverse engineering process later in Section 6, and only describe how these modes work in the following.

3.1 Find My Bluetooth Module

3.1.1 Working Principle. Find My is an offline finding network, which has been reverse-engineered in detail [26]. It locates iPhones, AirTags, and other Apple devices while they are not connected to the Internet. To this end, offline devices in lost mode regularly send BLE advertisements. Devices with a network connection scan for these advertisements and report lost devices' locations to the legitimate owners.

The Find My protocol is end-to-end encrypted, anonymous, and privacy preserving. This security guarantee is based on a device-specific master beacon key. The master beacon key is synchronized with the user's iCloud keychain, allowing access as long as the user can log in to iCloud. The keychain is also stored locally, and access to its secret key requires a roundtrip through the SEP [3, p. 96]. Based on the master beacon key, a rolling sequence of private/public key pairs is generated. This sequence is fixed forever, and each key is only valid during its time slot. Linking public keys in this sequence to each other requires knowledge of the master beacon key. The device broadcasts the current public key as BLE advertisement, and a portion of the key also sets the MAC address to a random value. Any Internet-connected device that observes a Find My BLE advertisement can encrypt its current approximate location with the public key and report it to Apple. Only the legitimate owner of the offline device possesses the matching private key to decrypt the location report. The location report does not include the reporter's identity. Public keys roll every 15 min on iPhones, which limits smartphone tracking via BLE advertisements.

AirTags work similar to Find My on iPhones. When they lose the Bluetooth connection to the owner's iPhone, they start sending advertisements. On AirTags, the advertisement public key only rolls every 24 h. This way, it can be detected when an AirTag follows the same person, and an anti-stalking warning can be displayed on the tracked person's smartphone [25]. However, a longer rolling time limits the legitimate user's privacy. A 24 h time interval might be a technical requirement on AirTags, which do not have a precise time source. Drifting away from a shorter 15 min window would mean that they could no longer be found when querying Apple's servers.

3.1.2 Low-Power Mode Setup. When Find My is supported after power off, this is shown as part of the shutdown dialogue. Users can change this setting during each manual power off procedure as depicted in Figure 3. If the iPhone runs out of battery and enters power reserve mode, Find My is enabled automatically, similar to Express Cards in the NFC chip.

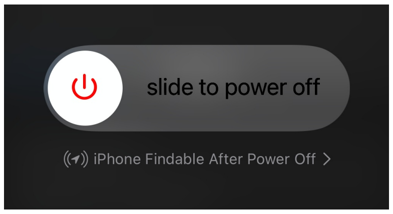


Figure 3: Find My shutdown dialogue on iOS 15.

LPM support requires a Bluetooth firmware that can send BLE advertisements while iOS is off. Broadcom Bluetooth chips can either be configured to interact with a host, such as iOS, or to run as standalone application, such as IoT devices. When entering LPM, the iOS Bluetooth stack is terminated and the Bluetooth chip is reset. Then, multiple Host Controller Interface (HCI) commands configure Find My parameters. The last HCI command disables HCI, thereby stopping all communication with iOS. The firmware starts a standalone Find My advertisement thread. The PMU is advised to keep the Bluetooth chip on despite shutting iOS down.

The Find My configuration HCI commands are very flexible. They allow setting multiple public keys for a short and a long rotation interval. The rotation interval duration and the total number of keys are variable parameters. As of iOS 15.3, 96 short interval (15 min) and 0 long interval (24 h) keys are set. Thus, the standalone Bluetooth app can only send Find My advertisements for up to 24 h. In power reserve mode, entered due to low-battery shutdown, Find My stops even earlier, after slightly more than 5 h. This property is opaque to users, who might expect that their iPhone remains findable for a longer period of time.

Only configuring a limited amount of short interval public keys has multiple security advantages. Advertisement linkability is reduced by the short rotation time, which increases privacy [41]. This also means that the iPhone will not trigger anti-stalking features, thereby hiding a potentially lost iPhone from thieves. Most importantly, the master beacon key stays within the keychain and is not shared with the SE, Bluetooth chip, or Bluetooth daemon.

3.2 Digital Car Key 3.0 Bluetooth and UWB Modules

3.2.1 UWB as NFC Successor for Car Keys. In addition to NFC cards, iOS also supports car keys. Since they are digital, they can have other properties than regular car keys. For example, they can be individualized per user, including speed limits for young drivers, and shared between iCloud users [4, 10]. While having DCK support on iOS is convenient for users, this also adds new attack vectors to car theft. On the implementation side, the main risk is on car manufacturers—a compromised DCK or failures in its implementation can be used to steal a car but not to steal an iPhone. Unlike digital payments, physical car theft cannot be undone. This risk motivated car manufacturers to move from DCK 2.0, which is based on NFC, to DCK 3.0.

Even though the acronym Near Field Communication (NFC) suggests that only two nearby communication parties can perform transactions successfully, NFC and older car key technologies are prone to relay attacks [21, 49]. In a relay attack, signals are forwarded over a larger distance than intended. Relay attacks pose a significant threat to applications like DCKs, because they allow car access and theft while the user is not nearby. Short distances can be reduced by amplifying an existing signal. Long distances require decoding a signal into packets and forwarding these. The first practical low-cost tool for performing packet-based NFC relay attacks has been published in 2011 [22], and similar tools supporting up-to-date NFC technologies are still maintained [31, 35].

While the DCK 2.0 specification relies on NFC [11], DCK 3.0 adds support for secure ranging based on UWB [12]. Even though

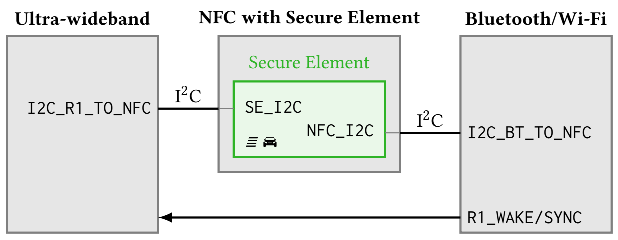


Figure 4: New Secure Element (SE) and inter-chip interfaces present in the iPhone 11, 12, and 13 series.

UWB was marketed to support secure ranging, the UWB distance measuring fundamentals are flawed [47], and low-cost distance shortening attacks against Apple's UWB chip of up to 12 m were demonstrated in practice [37]. We assume that these flaws were unknown during the DCK 3.0 design phase, and UWB still provides better ranging security than NFC.

3.2.2 Working Principle. The DCK 3.0 protocol uses both, BLE and UWB. BLE sets up the initial connection and authentication. Then, UWB is used for fine ranging secured with a symmetric key, but without data transfer. Thus, BLE and UWB both require access to secret key material. The DCK 3.0 specification describing precise steps of the BLE and UWB protocols is exclusively available to members of the Car Connectivity Consortium. Marketing materials only contain rudimentary information [10–12, 49]. The most detailed information was released in a presentation by Apple, which confirms that DCK 3.0 is supported in power reserve mode [4].

Apple prepared hardware components for DCK 3.0 since the iPhone 11. On iPhones with DCK 3.0 support, the SE is hardwired to the Bluetooth and UWB chips, as shown in Figure 4. Thus, similar to NFC transactions using the SE, replies generated by the SE are sent directly over-the-air without being shared to iOS. This ensures the same security boundaries as DCK 2.0, while adding support for secure ranging.

3.2.3 Low-Power Mode Setup. When the iPhone shuts down due to low battery, and a car key is configured, UWB Express Mode is enabled. This mode is very similar to the NFC Express Mode.

Initial DCK 3.0 protocol steps only require BLE. Hence, UWB can go to sleep, but a BLE application keeps running. The Bluetooth stack is terminated and a standalone application is started on the Bluetooth chip, similar to the Find My LPM. Both applications, DCK and Find My, run in the same thread. Applications are only initialized and executed when certain flags are set, meaning that the same firmware image can be used even if only one application is active. The DCK application scans for other BLE devices, runs a Generic Attribute (GATT) service for data exchange over BLE, and retrieves key material from the SE using the Inter-Integrated Circuit (I^2C) protocol.

In contrast to the Bluetooth chip, the UWB chip is sleeping. The daemon responsible for UWB sends a special LPM enabling and configuration packet to the chip. Once secure ranging is needed, the Bluetooth chip sends a wake signal over the hardwired connection shown in Figure 4. Timing between Bluetooth and UWB is synchronized to reduce active receiver times to the expected incoming packet window [4]. The combination of both saves a lot of power on the UWB chip, since it is sleeping most of the time, and even when active only listens within very limited time windows.

Table 1: Wireless chips with actively used LPM support.

Series	NFC + SE	LPM	Bluetooth/Wi-Fi	UWB	LPM
iPhone XR	NXP SN100	✓	BCM4347B1	–	–
iPhone Xs	NXP SN100	✓	BCM4377B2	–	–
iPhone 11	NXP SN200	✓	BCM4378B1	r1p0	✓
iPhone SE 2020	NXP SN200	✓	BCM4378B1	–	–
iPhone 12	NXP SN210	✓	BCM4387C2	r1p1	✓
iPhone 13	NXP SN210	✓	BCM4387C2	r1p2	✓

3.3 Supported Devices

Bluetooth and UWB LPM support are mainly driven by DCK 3.0 integration. Only iPhones with DCK 3.0 support also have Bluetooth and UWB LPM, as listed in Table 1. The Bluetooth chip in the iPhone SE 2020 is the same as in the iPhone 11. Find My could run standalone on this chip while powered by the PMU. However, as of iOS 15.2, the firmware on the iPhone SE 2020 is missing the LPM thread. It is unclear if this feature is missing intentionally or if a firmware supporting only Find My without DCK 3.0 is still under development. Apple might add Find My LPM support to further devices in the future.

4 SECURITY ANALYSIS

In this section, we analyze security of LPM features following a layered approach. First, we check if LPM applications are working as intended to ensure the user's security and safety, and then analyze the impact of LPM on firmware and hardware security.

4.1 Applications

4.1.1 Adversary Model. On the application layer, we assume an attacker that did not manipulate the firmware or software on the iPhone. Instead, they want to compromise or use LPM features, e.g., disable Find My to steal an iPhone or use Express Cards and Keys to steal money or physical assets. There is a usability aspect, for example, if the legitimate user does not understand technological limitations of LPM features, theft becomes more likely.

4.1.2 Find My. When observing Find My, we found that LPM does not hold its security promises to users. It stops sending advertisements way earlier than expected and, in many cases, does not send advertisements at all. In the following, we explain under which conditions advertisements are sent and which problems occurred during testing.

Maximum LPM Time Period. Only 96 advertisements each broadcasted for 15 min are configured during user-initiated shutdown. After 24 h, all advertisements are transmitted. This is not explained to the user, who might think their iPhone remains findable for multiple days. The current shutdown dialogue suggests that Find My after power off is a strong security feature. Disabling even requires a passcode as of iOS 15.3 as additional anti-theft measure, as shown in Figure 5. Users might permanently lose their iPhone when relying too much upon the Find My after power off feature. The maximum time period could be extended by configuring more advertisements, cycling advertisements only after 24 h, or by storing the master beacon key within an SE applet that the Bluetooth chip can query to generate more advertisements.

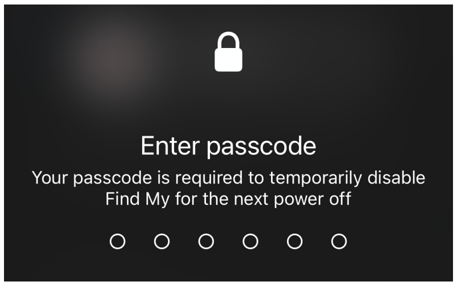
Upon battery-low shutdown, Find My is only active for slightly more than 5 h. This is similar to Express Mode [8] but not documented at all. The battery should not be fully discharged to prevent hardware damage. Thus, at some point, the iPhone has to switch from power reserve to power off.

Find My Stops Before First Unlock After 24 h. On power off, advertisements for the next 24 h are additionally stored locally on disk. Even if Find My was disabled on power off, advertisements are still cached. Once the iPhone boots, it restores a Find My token from an NVRAM storage [38, p. 14ff], which survives reboots. The Find My token is then used to decrypt the cached advertisements, making them accessible to the Bluetooth daemon before first unlock [3, p. 98]. When 24 h expire, the iPhone stops sending advertisements. This is irrespective of previous actions, the iPhone could have been in LPM before or rebooted directly. Since advertisements are fixed to a scheduled transmission time, this period cannot be extended.

If the iPhone has an Internet connection before first unlock, it connects to Apple's servers for reporting its location. Many conditions cause the iPhone to be offline before first unlock. Wi-Fi keys are only available after first unlock [3, p. 97], and a cellular connection fails if the user has set a SIM PIN or the SIM card was removed. Thus, a thief can be sure that Find My is disabled after 24 h, even if they power on the iPhone.

Failures Initializing Find My LPM. Even when the user interface shows that Find My is enabled after power off, this is sometimes not the case. For example, Find My advertisement generation might fail. We observed this error multiple times on a device with iOS 15.0.1. If Find My LPM setup fails during shutdown, no warnings are shown to the user.

Unintended Boot. When connected a power cable is plugged in during user-initiated power off and removed later, the iPhone boots after a couple of hours. We observed this behavior on two iPhone 12 and an iPhone 13 on iOS 15.3 or lower with Find My enabled. If no power cable is plugged in during power off, the iPhone does not boot automatically. On an iPhone 12 on iOS 14, which does not support Find My LPM, no unintended boots occurred. Due to this issue, the user might find their iPhone with an empty battery when Find My is active, despite switching it off.


Enter passcode

Your passcode is required to temporarily disable Find My for the next power off

Figure 5: Passcode dialogue to disable Find My on power off.

Advertisement Accuracy. We observed a standard deviation of 19 s in the 15 min rotation windows when capturing advertisements of an iPhone in LPM for 24 h. This deviation could become an issue when an iPhone would support LPM for more than 24 h. When the advertisements are outside of the search window, the Find My servers will no longer find the device.

4.1.3 Express Cards and Keys. The obvious threat for Express Cards and Keys is that a user's credit card or car key is used without further authentication when their iPhone is stolen. To this end, the user must actively enable Express Mode once for each item in the Wallet application. Limitations, such as unsupported cards or shutdown after 5 h are publicly documented [8]. Thus, we assume that most users understand the functionality and risks of Express Mode.

4.2 Firmware

4.2.1 Adversary Model. On the firmware layer, we assume an attacker with privileged firmware access. For example, the attacker could be able to send custom commands to the firmware via the operating system, modify the firmware image, or gain code execution over-the-air. This is a strong precondition, which we discuss first. Then, we discuss what an attacker could do once firmware is compromised.

4.2.2 Tampering with Firmware. An attacker has different possibilities to gain firmware control, which also vary in their preconditions.

Local Firmware Modification. An attacker with system-level access could modify firmware of any component that supports LPM. This way, they maintain (limited) control of the iPhone even when the user powers it off. This might be interesting for persistent exploits used against high-value targets, such as journalists [30].

As of iOS 15, LPM-supported chips are NFC, UWB, and Bluetooth. An attacker does not need to change kernel behavior and can use existing drivers to enable LPM on these chips, and only needs to modify the chip's firmware. The NFC chip is the only wireless chip on an iPhone that has encrypted and signed firmware. Even though attempts were made to bypass the secure bootloader, these were not successful [14]. The firmware of the UWB chip also has secure boot but is only signed and not encrypted [16, 18]. However, the Bluetooth firmware is neither signed nor encrypted. Since the Bluetooth chip does not have secure boot enabled, it misses a root of trust that could be used to verify the firmware it loads. In Section 5, we show that it is possible to create malware that runs on iPhone 13 Bluetooth chips, even if the phone is powered off.

Remote Code Execution. Attackers without system-level access could try to gain code execution on an LPM-enabled chip over the air. In the past, various vulnerabilities have been disclosed for the Bluetooth chip series used in iPhones [24, 40, 46]. Such vulnerabilities could also exist in other wireless chips.

Minimizing functionality of a wireless stack also reduces its attack surface [58]. The Find My module only transmits advertisements over Bluetooth but does not receive data. However, NFC Express Mode as well as Bluetooth and UWB DCK 3.0 allow data exchange. Apple already minimizes the attack surface by only enabling these features on demand.

4.2.3 Reconfiguring Firmware Options. Even if all firmware would be protected against manipulation, an attacker with system-level access can still send custom commands to chips. The Bluetooth daemon configures Find My LPM on shutdown with vendor-specific HCI commands. These commands allow a very fine-grained configuration, including advertisement rotation intervals and contents. With these commands, an attacker, for example, could set every second advertisement to a public key that matches their account. Afterward, the attacker could locate the victim device, but the legitimate user would still see their iPhone with a recent location in Find My.

This specific attack is not relevant as of now, because the Bluetooth firmware on iPhones can be modified. If Bluetooth chips on the next generation of iPhones would support secure boot, this attack surface would become relevant. The Apple Watch, which runs an iOS derivative, has a different Bluetooth chip supporting secure boot [28]. If Find My LPM would be added to the Watch, fine-grained configurability would introduce the same issues.

4.2.4 Direct Secure Element Access via Firmware. Bluetooth and UWB LPM were introduced to support car keys, which store their data in an SE applet. Transactions between wireless components and the SE should be hidden from iOS [3, p. 141], ensuring security even if iOS or applications are compromised. For example, a malicious application that can access the SE could relay payment transactions [44]. Since the Bluetooth firmware can be manipulated, an additional SE interface is directly exposed to iOS. This weakens the trust model for car keys and other secret information in the SE.

Additional interfaces influence the overall security level provided by the SE. It is unclear if the SE still holds the same security standards, because the interface features, applet implementations, and intended security boundaries are not public.

4.3 Hardware

4.3.1 Adversary Model. On the hardware layer, we assume that neither attackers nor legitimate users would manipulate the hardware. We analyze which components could be stealthily powered on while the iPhone is off and which applications attackers could build. We discuss if users could detect such LPM applications.

4.3.2 Suitable Chips. Any chip that is controlled by the Power Management Unit (PMU) could be active in LPM. This property holds for most chips in an iPhone but still does not make every chip ideal to be operating in LPM. Since LPM should not significantly drain the battery, suitable chips should either be standalone or combinations of a few battery-friendly chips. Moreover, these chips should have some way to interact with the user or the outside world.

Even though most chips on an iPhone are connected to the PMU, they are also connected to the Application Processor (AP) or the Always-on Processor (AoP). The AP is the main processor running iOS. The AoP is a co-processor running Apple's RTKitOS, an embedded Real-Time Operating System (RTOS) [38, p. 20ff]. While the AoP is always active and waits for hardware events and triggers, the AP can go to sleep, significantly reducing power consumption. Drivers in the iOS kernel running on the AP often have a companion running on the AoP. This hardware architecture requires most chips to either interact with the AoP, which then routes information to the AP, or chips talk directly to the AP. Neither the AoP nor the AP should be running in LPM, because they consume too much energy.

All wireless chips are connected to the PMU and could operate standalone. This includes the Wi-Fi chip and the cellular baseband, which are currently not used for LPM applications. The display is connected to the PMU. In power reserve mode, the display only shows a static screen. The AP sets an image on the display when entering LPM, but wireless chips cannot access the display. Buttons are connected to the PMU, such that a button press can boot or reset the iPhone from any state including LPM. Audio is routed via the AoP, meaning that it is not possible to build a hidden microphone by recording speech and forwarding it to a wireless chip in LPM.

4.3.3 Detection and Deactivation. Detecting that chips were running an LPM firmware or completely disabling LPM is not possible with current iOS on-board tools. Additional monitoring hardware is needed to detect wireless transmissions in LPM.

Logging. The original LPM firmware images support logging. The UWB chip has a storage for crash logs, including DCK, which is read out after system boot during chip initialization by iOS. The Bluetooth LPM firmware has multiple logging statements, depending on logging flags, which could be stored on the SE. Logs on this level are intended for iOS system developers and cannot be extracted without jailbreak. Thus, users cannot determine if LPM firmware was running and which actions it performed, even on a system that is not compromised.

Disabling LPM. LPM support is implemented in the hardware. Even if iOS would remove LPM features in the future, these basic hardware features could be activated on a compromised system.

A possibility to remove permanent support in future hardware revisions would be a physical switch that disconnects the battery. Apple already has similar physical switches in their hardware, for example, the microphone on MacBooks is disconnected when the lid is closed [3, p. 27].

Other Protection. Users who are concerned about being spied on by an iPhone that is off can install a transmission monitoring device, which can be mounted on the back of an iPhone 6 and wiretaps wireless chip connections on the mainboard [30]. Such a device would even detect LPM malware. This solution would need to be modified for modern iPhones, if applicable at all. The mainboard has two stacked and soldered layers since the iPhone 11 [32]. The desoldering process has a higher risk of damaging the iPhone than on older models.

Another option would be to put the smartphone into a Faraday bag. This solution is also considered in [30] but small holes and gaps can lead to significant leakage.

5 BLUETOOTH FIRMWARE MODIFICATION

Recent iPhones introduced a new chip interface, including a new Bluetooth firmware patch format. We show how we can analyze the chip's ROM and RAM nonetheless, and that the new patch format is unsigned. Based on this, we demonstrate how to build custom firmware and malware.

5.1 Dumping and Loading Firmware

5.1.1 Firmware Dumps. We dump the complete firmware for an initial analysis. Broadcom Bluetooth chips store most of their firmware in ROM. A full dump including the ROM is required for understanding the firmware. The ROM is readable with a vendor-specific Host Controller Interface (HCI) command supported by all Broadcom chips, including the one in the iPhone 13. Apple's Core Bluetooth framework does not allow sending HCI commands to the chip [5].

On a jailbroken iPhone, we use *InternalBlue* to send such HCI commands [40]. It attaches itself to the Bluetooth subsystem to send HCI commands and interpret received events. An iOS-specific behavior of the current InternalBlue version is that the Bluetooth daemon is detached, and we can exclusively access the chip without interfering with other system components. The initial InternalBlue iOS package only supported legacy chips connected via UART, and we added support for the newer PCIe driver. With this modification, we dump the ROM of the latest generation of Bluetooth chips. For analysis purposes, we dump the RAM in different states.

5.1.2 Firmware Patching with Patchram. Firmware is patched temporarily in RAM. Broadcom calls this mechanism *Patchram*, and it has been documented within the InternalBlue project [40]. Any 4-byte value in ROM can be temporarily mapped to another 4-byte value. On ARM, a function call requires a 4-byte instruction. Thus, a function in ROM can be overwritten with a function call to an address in RAM. For this purpose, RAM is writable and executable. Modern Broadcom chips have 256 Patchram slots [46].

Legacy chips are connected via UART, and there is no other transport layer than sending the whole firmware patch using small HCI commands with a maximum size of 255 bytes. Broadcom's legacy `.hcd` patch format contains a sequence of HCI commands, which directly write to RAM and configure the Patchram [39].

Since the iPhone Xs, the Bluetooth chip is connected via PCIe. This allows for transferring one large `.bin` patch file at once. In the following, we reverse-engineer this new format, enabling us to install custom patches on modern iPhones.

5.1.3 Firmware Patch Image Upload. Most communication between iOS and the Bluetooth chip is handled by the Bluetooth daemon. For selected low-level tasks, the Bluetooth daemon calls `BlueTool`. `BlueTool` allows sending HCI commands to the chip without interfering with the stack, similar to InternalBlue, and additionally allows loading firmware patches. `BlueTool` can be instrumented via Cross-Process Communication (XPC) calls from other daemons, used interactively via a command line interface, or execute scripts. It is an Apple-internal tool without documentation, and the help output is incomplete. We can load a PCIe firmware as follows:

```
power off
device -D
bcm -w /tmp/path/to/firmware.bin
```

Using this method, we can apply firmware patches of arbitrary iOS versions to the chip, no matter which iOS version a jailbroken research iPhone has. iOS system logs indicate if a firmware patch image could be loaded or not.

5.1.4 Firmware Patch Image Format. RAM dumps enable us to determine address mappings of patches. We can modify arbitrary parts of the patch file and see if the chip accepts it. This leads us to the format shown in Figure 6. The patch is only verified with Cyclic Redundancy Checks (CRCs) but not signed by Broadcom.

Two Patchram areas in RAM contain writable and executable code regions. The regular RAM region used by threads to store data is located in between.

A patch configuration data area in Type Length Value (TLV) format defines how patches are applied. It contains basic information like a human-readable patch name, hardware configuration settings, as well as 4-byte patch entry instructions that overwrite the ROM. The format is very similar to the one defined within *WICED Studio 6.2* `.hdf` files [20]. The configuration data section starts with the string `BRCMcfgS`. Each patch entry starts with the byte sequence `0x0110`, is 15 byte long, and contains the ROM address as well as the new 4-byte instruction. A more detailed description of analyzing and patching configuration data is provided in our blog post [15].

5.2 Analyzing and Modifying Firmware

5.2.1 Patch Analysis Tooling. With these new insights, we can load any patch file on top of an existing InternalBlue ROM dump for static reverse engineering. We modify existing patch files and load them onto a running iOS device. We create three scripts to assist us in this process: *(1)* A standalone Python script, which can extract patch regions and also reassemble a patch file including correct CRCs, and *(2)* an IDA Pro script, which can parse patch configuration data, and *(3)* another IDA Pro script, which names functions calling the abort handler.

Applications of such tools are very flexible. Broadcom Bluetooth has been patched in the past to increase BLE reliability [48], enable the chip to send and receive ZigBee [13], and test against security issues in the Bluetooth specification [1, 2]. In the following, we analyze if the firmware can be modified and loaded into iOS, and also analyze the internals of the LPM module.

5.2.2 Malware Patch Example. Preventing firmware tampering during runtime is an important security barrier. For example, it prevents the Bluetooth daemon from directly observing and modifying communication between the SE and the Bluetooth chip when a DCK 3.0 is used. It also hardens against other inter-chip escalation strategies, e.g., executing code in Wi-Fi via Bluetooth [17].

iOS can write to the executable RAM regions using a vendor-specific HCI command. This is a legacy feature required for applying

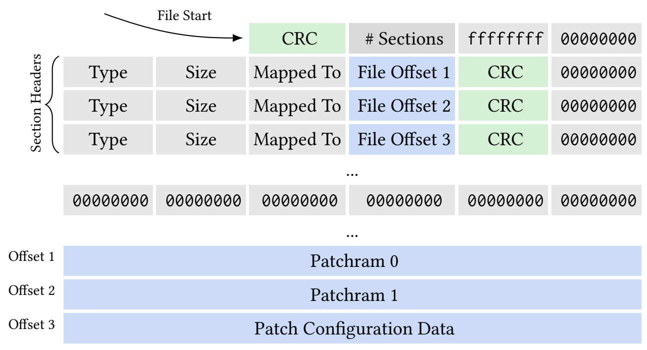


Figure 6: New Bluetooth firmware `.bin` patch format, introduced with the iPhone Xs PCIe Bluetooth chip interface.

.hcd files, still present in the iPhone 13 Bluetooth chip. The chip protects itself from modification during runtime with a patch that disables this HCI command. We alter this patch to demonstrate that firmware modification is still possible.

HCI commands are handled in the function `bt_hci_ingress_HandleCommand`. This function has an abort handler and is already named by our IDA Pro script. One of the functions it calls is an HCI command filter. The filter has a very specific pattern and compares HCI opcodes. The opcode for writing to RAM is `0xfc4c`. Opcodes are compared after subtracting `0xfc00`. On an iPhone 12 with a patch level of iOS 15.2, a part of this patch looks as follows:

```
patch1:002c57d2    cmp   r2, #0x2e ; Download Minidriver
patch1:002c57d4    beq   command_disallowed
patch1:002c57d8    cmp   r2, #0x4c ; Write RAM
patch1:002c57da    beq   command_disallowed
```

After replacing the `cmp r2, #0x4c` instruction, repacking the firmware patch image with the correct CRCs, and loading it with `BlueTool`, we can write to the Bluetooth chip RAM during runtime. As of January 2022, there is no iOS 15.2 jailbreak available. Thus, we load modified iOS 15.2 firmware to an iPhone SE 2020 with iOS 14.8 and an iPhone 12 with iOS 14.2.1 to confirm our patches work on the chips in the last three generations of iPhones (see Table 1).

5.2.3 Custom Bluetooth LPM Applications. Using the same methods, an attacker could alter the LPM application thread to embed malware. For instance, they could advise the iPhone to send the legitimate owner's Find My advertisements and additionally send advertisements that make the iPhone locatable for the attacker. Instead of changing existing functionality, they could also add completely new features.

Broadcom Bluetooth chips can run a standalone application without being connected to a host. This configuration is very common for IoT devices—a Bluetooth application like a BLE thermal sensor service can run on a single chip. WICED Studio provides a C wrapper for creating custom standalone applications [20]. Standalone applications run in a thread called *MPAF*. Apple's LPM application runs in the same thread. Even though WICED Studio is for Cypress chips, their code base is similar to Broadcom chips [23], and we can use it as a base for reverse-engineering and function naming.

The application thread executes functions from a queue. Functions can be added by calling `mpaf_thread_PostMessage(void *function)`. For example, Apple's LPM application starts by enqueueing a function that initializes the application thread, and later on, a state machine decides if Find My advertisements or scanning for cars should be activated. Functions can also be executed after a timer expires, using `mpaf_timer_start(timer *p_tle, int seconds)` with a struct that contains a callback function. For example, timers are used to cycle Find My advertisements. Knowing these semantics of the application thread, an attacker can add custom LPM features.

Providing a full analysis of the LPM firmware and its modules is out of scope of the paper. We reverse-engineered a lot of internals, such as Find My advertisements including their storage and scheduling, the GATT service for DCK 3.0 and the I^2C connection to the SE. The manually reverse-engineered symbols for the iPhone 12 and 13 firmware, including more than checked 2000 function names, are published within the *Frankenstein* repository.

6 REVERSE ENGINEERING METHODS

The reverse-engineering techniques we used are also applicable when analyzing other functionality close to hardware on an iPhone. In the following, we document them, which also ensures reproducibility of our results.

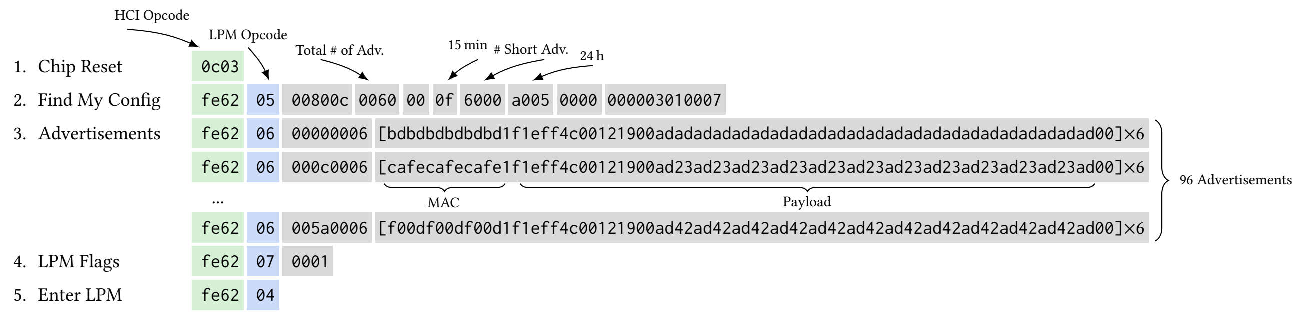
6.1 Dynamic Analysis

Dynamic analysis close to hardware requires access to physical devices. As of now, there is no full iOS emulator, and even commercial solutions mock up the lower hardware layers [19]. Ideally, the devices are on different iOS versions and have different hardware revisions. Using outdated iOS versions is required to use jailbreaks [34, 56], however, devices might behave differently without jailbreak and on the latest iOS release. The initial efforts to build a proper testing lab meeting these conditions is nonetheless worth the efforts.

6.1.1 System Log Analysis. iOS system logs are very verbose and show how daemons interact with each other. For example, they tell which chips go into LPM on shutdown, with which HCI commands the Find My LPM module is initialized, and that Find My restores beacons from NVRAM before first unlock.

System logs can be extracted from any iOS device, even if not jailbroken. Apple's debug profiles increase log verbosity [6]. In addition to the Bluetooth profile, we also installed the Location Services profile, which increases verbosity on some Find My and ranging-related messages. Logs can be collected up to three days in retrospect, depending on the log type, by running a *sysdiagnose*. Pressing all buttons until the iPhone vibrates triggers a sysdiagnose, which is then located a few moments later in *Settings* → *Privacy* → *Analytics Data*. After unpacking on macOS, the `system_logs.logarchive` can be read with the Console app. This method allows to collect logs including the initial boot process and the very last messages logged during shutdown.

For our analysis, we created logs under various conditions. For example, we looked into shutdown logs with Find My enabled and disabled, with and without Express Cards, user-initiated shutdown versus battery-low shutdown, and combinations of those. In many cases, these logs are so verbose that it is hard to find relevant messages. On a device that is actively used, these logs contain multiple millions of messages just for three days. Messages can be filtered by daemons and log message contents. The most relevant daemons during our research were `bluetoothd` for Bluetooth, `nearbyd` for UWB, `nfcd` for NFC, `seserviced` for the SE and `findmydeviced` for Find My. Naming of features in log messages is often inconsistent. For example, LPM is also called *LPEM* or *LEPM*, and the DCK protocol is internally named *Alisha*. This turns log analysis into a time-consuming and manual process. Also, not everything implemented within a daemon is logged, and often, conditions under which something interesting is logged are hard to reach. This is especially the case for the DCK protocol, since the cars supporting DCK 3.0 are yet to be released. Still, there are some hints, such as the UWB chip that attempts to enter LPM on power off, and the `seserviced` checking if UWB Express is active on battery-low shutdown when other LPM features are disabled.



```
1. Chip Reset        0c03
2. Find My Config    fe62 05 00800c 0060 00 0f 6000 a005 0000 000003010007
3. Advertisements    fe62 06 00000006 [bdbdbdbdbdbd1f1eff4c00121900adadadadadadadadadadadadadadadadadadadadadadad00]×6
                     fe62 06 000c0006 [cafecafecafe1f1eff4c00121900ad23ad23ad23ad23ad23ad23ad23ad23ad23ad23ad23ad00]×6
   ...
                     fe62 06 005a0006 [f00df00df00d1f1eff4c00121900ad42ad42ad42ad42ad42ad42ad42ad42ad42ad42ad42ad00]×6
4. LPM Flags         fe62 07 0001
5. Enter LPM         fe62 04
```

Figure 7: Configuration of Find My LPM with HCI commands.

6.1.2 Find My Advertisement Capturing. Information obtained via log messages should be further analyzed to be confirmed. For wireless protocols, signals and their timings can be captured and compared to logs to see if anything is missing. System logs are not available in LPM, which is the focus of our research.

One example for missing information in logs is Find My. Based on log messages, only 80 Find My advertisements are set up during power off via HCI commands. As we found later, 96 advertisements are configured, however, HCI commands are truncated in the logs. The logs do not show how often advertisements are cycled.

We create a Python script running on Linux collecting Find My advertisements. This script allows us to see when an iPhone starts or stops sending advertisements. Even if other devices are around, we can assign Bluetooth MAC addresses to devices by comparing them to system logs. This script assisted us in finding the issues in Section 4.1.2. Besides these issues, Find My works very reliable.

6.1.3 Bluetooth Firmware Instrumentation. Even though not all iOS versions are jailbreakable, we can backport specific features for analysis. For example, we can use this to load Bluetooth firmware with LPM support onto a jailbroken device, as explained in Section 5. Then, we analyze it dynamically with InternalBlue [40].

First of all, we can send all HCI commands that the Bluetooth daemon sends when entering LPM as well. These commands are also shown in Figure 7. When we send the last command, it terminates communication with the host. Thus, we cannot execute it during dynamic analysis of the LPM module. However, we can send all other commands and then take a RAM dump. This way, we can see in which memory area advertisements and other configuration is stored and look for functions that reference these. We can then analyze if some of these values are used as timers or conditional flags. For example, the HCI command `0xfe62 05 [19b config]` contains a 24 h timer as minutes in reverse byte order, which is the total time of advertisements, as well as the total number of advertisements, and the 15 min rotation interval. Thus, we know that these are configurable values.

6.2 Static Analysis

Static analysis is often more time-consuming than dynamic analysis. It is still an important technique, because not everything shows up in logs, and not all functions are called during runtime. Thus, it can reveal hidden features and shows significantly more details.

6.2.1 Hardware Schematics. There is a large community sharing smartphone schematics, mainly driven by the need to repair iPhones and other devices independent from vendors. Hence, many schematics are leaked and publicly available on the Internet.

We use these schematics to confirm that the SE is hardwired to UWB and Bluetooth in Section 3.2. Schematics allow us tracing connections between other chips to estimate threats by LPM in Section 4.3. The schematics also contain the interfaces' protocols, e.g., PCIe or I^2C. This further helps when reverse-engineering firmware and drivers implementing these protocols.

6.2.2 System Daemon and Binary Analysis on iOS. In the following, we first explain how to obtain binaries for iOS, because not all steps for these are documented comprehensively for recent versions, and then explain how to analyze them.

Obtaining Binaries. Analyzing different iOS versions allows to compare features across different software and hardware revisions. Even without a physical iPhone, iOS system updates can be downloaded [33], unpacked as `.zip`, and the included `.dmg` partitions can be mounted.

Recent iOS versions have a Dynamically Loaded Library (DYLD) shared cache [7], containing a lot of proprietary functionality also used by daemons, which is one 4 GB large binary blob. The packing format changes regularly, and, thus, is often not fully supported by free reverse engineering tools. After connecting an iPhone to Xcode, an unpacked format containing separate libraries can be found in the Xcode `iOS DeviceSupport` folder.

The kernel cache contains the iOS kernel and its drivers. On recent iOS versions, it is no longer encrypted and can be unpacked with `img4tool` [53]. Sometimes, Apple forgets to strip symbols from the kernel cache. For example, Over-the-Air (OTA) updates [52] for all iPhones and further devices between iOS 15.0 and iOS 15.1 Beta 3 contain a kernel cache with symbols. Names in driver interfaces reveal hardware capabilities that are otherwise difficult to reverse-engineer.

Analysis. Similar to the approach when analyzing logs, relevant binaries and kernel drivers can be identified by searching for strings. Log messages seen during dynamic analysis can be searched in binaries. Interesting binaries can then be loaded into a disassembler or decompiler [9, 29, 42]. Often, these tools do not find all references, and logs can be used to reconstruct call graphs. Even a short analysis can uncover additional features and code paths not reached when capturing logs.

6.2.3 *UWB Firmware.*

In addition to Bluetooth firmware, we also analyzed the UWB firmware. It is stored in a file called `ftab.bin`, which contains firmware for a high-level AP on the UWB chip as well as a second firmware for a low-level Digital Signal Processor (DSP) [18]. The UWB firmware is under active development and features change regularly. New features are sometimes added for testing and removed later.

Early versions of the iPhone 11 UWB firmware had enhanced logging. These versions also contained debug prints for communication with the SE. A lot of testing and logging functionality was removed in later versions. *Alisha*, the code name for the DCK 3.0 protocol, started appearing in the firmware in iOS 14.3, and LPM support for UWB was added to `nearbyd` in the same release. These examples show that looking into multiple versions to understand new feature integration is imperative.

7 RELATED WORK

Wireless Firmware on iPhones. InternalBlue looked into Broadcom's Bluetooth firmware before [40]. It does not support the new firmware patch format and did not analyze LPM features. Additionally, Broadcom's Bluetooth firmware was tested with fuzzing [46]. Apple's UWB chip was analyzed with regards to basic driver and firmware functionality [16, 18, 45], as well as practical distance shortening attacks [37]. The NFC chip was not reverse engineered so far. A very similar NFC chip by NXP has been analyzed and a security issue in the bootloader was found [14]. The researcher also tried attacking the SN100 chip, but the bootloader is different and not affected. Common Criteria certification reports of NXP's NFC chips are publicly available. The SE component in the SN200 NFC chip and the full SN100 NFC chip were tested, but no exploitable vulnerabilities were found [54, 55]. Furthermore, the JavaCard Operating System running on the SN200 and SN210 chips was evaluated [43].

Wireless Services on iOS. Various wireless services and their underlying daemons and protocols have been analyzed on iOS. This includes Find My [26], AirDrop [51], multiple Continuity services [50], and Magic Pairing [28]. Security of the iOS Bluetooth cellular baseband daemons has been tested with fuzzing [27, 36].

Inter-Chip Attacks. The direct connection between the SE and multiple wireless chips enables potential inter-chip privilege escalation attacks. A practical attack that enables code execution in Wi-Fi via Bluetooth has been demonstrated to work on recent iPhones [17].

Transmission Monitoring. Airplane mode might not disable all wireless transmissions on a smartphone under surveillance. Thus, Huang and Snowden developed a separate hardware-based device, which they attached to the mainboard of an iPhone 6 to monitor wireless chips [30]. This generation of smartphones neither considered LPM nor supported UWB, adding a different attack angle and requiring upgrades to the existing hardware.

8 CONCLUSION

The current LPM implementation on Apple iPhones is opaque and adds new threats. Since LPM support is based on the iPhone's hardware, it cannot be removed with system updates. Thus, it has a long-lasting effect on the overall iOS security model. To the best of our knowledge, we are the first who looked into undocumented LPM features introduced in iOS 15 and uncover various issues.

Design of LPM features seems to be mostly driven by functionality, without considering threats outside of the intended applications. Find My after power off turns shutdown iPhones into tracking devices by design, and the implementation within the Bluetooth firmware is not secured against manipulation. Tracking properties could stealthily be changed by attackers with system-level access. Furthermore, modern car key support requires UWB in LPM. Bluetooth and UWB are now hardwired to the SE, used to store car keys and other secrets. Given that Bluetooth firmware can be manipulated, this exposes SE interfaces to iOS. However, the SE is specifically meant to protect secrets under the condition that iOS and applications running on it could be compromised.

Even though LPM applications increase security and safety in many use cases, Apple should add a hardware-based switch to disconnect the battery. This would improve the situation for privacy-concerned users and surveillance targets like journalists.

AVAILABILITY

Our tools for modifying and analyzing recent Broadcom firmware are published within the *InternalBlue* and *Frankenstein* repositories. This includes a standalone firmware patcher, IDA Pro scripts for interpreting firmware patch files and naming functions by debug strings, and symbols for the iPhone 12 and 13 firmware.

ACKNOWLEDGMENTS

We thank Matthew Green for initially rising the question how Find My after power off works, Hector Martin for pointing us at iPhone schematics to confirm what is possible in LPM, Fabian Freyer for the discussions about LPM, Florian Kosterhon for proofreading this paper and comparing UWB firmware versions, and Apple for reading our paper prior to publication.

This work has been funded by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE.

REFERENCES

[1] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper Rasmussen. 2020. BIAS: Bluetooth Impersonation AttackS. In *IEEE Symposium on Security and Privacy*.
[2] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper B. Rasmussen. 2019. The KNOB is Broken: Exploiting Low Entropy in the Encryption Key Negotiation Of Bluetooth BR/EDR. https://www.usenix.org/conference/usenixsecurity19/presentation/antonioli. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 1047–1061.
[3] Apple. 2021. Apple Platform Security. https://manuals.info.apple.com/MANUALS/1000/MA1902/en_US/apple-platform-security-guide.pdf.
[4] Apple. 2021. Explore UWB-based car keys. https://developer.apple.com/videos/play/wwdc2021/10084/.
[5] Apple. 2022. Core Bluetoot | Apple Developer Documentation. https://developer.apple.com/documentation/corebluetooth.
[6] Apple. 2022. Bug Reporting—Profiles and Logs. https://developer.apple.com/bug-reporting/profiles-and-logs/.
[7] Apple. 2022. Overview of Dynamic Libraries. https://developer.apple.com/library/archive/documentation/DeveloperTools/Conceptual/DynamicLibraries/100-Articles/OverviewOfDynamicLibraries.html.
[8] Apple. 2022. Use Express Mode with cards, passes, and keys in Apple Wallet. https://support.apple.com/en-us/HT212171.
[9] Binary Ninja. 2022. A New Type of Reversing Platform. https://binary.ninja.
[10] BMW. 2022. https://www.bmw.de/de/topics/service-zubehoer/bmw-connecteddrive/digital-key.html.

[11] Car Connectivity Consortium. 2020. Digital Key – The Future of Vehicle Access, Whitepaper. https://global-carconnectivity.org/wp-content/uploads/2020/04/CCC_Digital_Key_2.0.pdf.

[12] Car Connectivity Consortium. 2021. Car Connectivity Consortium Publishes Digital Key Release 3.0. https://carconnectivity.org/press-release/car-connectivity-consortium-publishes-digital-key-release-3-0/.

[13] Romain Cayre, Florent Galtier, Guillaume Auriol, Vincent Nicomette, Mohamed Kaâniche, and Géraldine Marconato. 2021. WazaBee: attacking Zigbee networks by diverting Bluetooth Low Energy chips. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*.

[14] Christopher Wade. 2021. https://www.pentestpartners.com/security-blog/breaking-the-nfc-chips-in-tens-of-millions-of-smart-phones-and-a-few-pos-systems/.

[15] Jiska Classen. 2021. https://naehrdine.blogspot.com/2021/01/broadcom-bluetooth-unpatching.html.

[16] Jiska Classen, Fabian Freyer, and Thomas Roth. 2021. Over the Air-Tag: shenanigans with the most over-engineered keyfinder. Presentation at hardwear.io Netherlands 2021, https://hardwear.io/netherlands-2021/speakers/jiska-and-fabian-and-stacksmashing.php.

[17] Jiska Classen, Francesco Gringoli, Michael Hermann, and Matthias Hollick. 2022. Attacks on Wireless Coexistence: Exploiting Cross-Technology Performance Features for Inter-Chip Privilege Escalation. (2022).

[18] Jiska Classen and Alexander Heinrich. 2021. Wibbly Wobbly, Timey Wimey – What's Really Inside Apple's U1 Chip. Presentation at Black Hat USA 2021.

[19] Corellium. 2022. Device configuration features. https://www.corellium.com/platform/features.

[20] Cypress Semiconductor Corporation. 2022. WICED Software. https://www.cypress.com/products/wiced-software.

[21] Aurélien Francillon, Boris Danev, and Srdjan Capkun. 2011. Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA, 6th February - 9th February 2011*. The Internet Society. https://www.ndss-symposium.org/ndss2011/relay-attacks-on-passive-keyless-entry-and-start-systems-in-modern-cars

[22] Lishoy Francis, Gerhard P. Hancke, Keith Mayes, and Konstantinos Markantonakis. 2011. Practical Relay Attack on Contactless Transactions by Using NFC Mobile Phones. *IACR Cryptol. ePrint Arch.* (2011), 618. http://eprint.iacr.org/2011/618

[23] Jan Friebertshäuser, Florian Kosterhon, Jiska Classen, and Matthias Hollick. 2021. Polypyus–The Firmware Historian. *Workshop on Binary Analysis Research (BAR) 2021* (Feb 2021).

[24] Matheus E Garbelini, Sudipta Chattopadhyay, Vaibhav Bedi, Sumei Sun, and Ernest Kurniawan. 2021. BrakTooth: Causing Havoc on Bluetooth Link Manager. https://asset-group.github.io/disclosures/braktooth/.

[25] Alexander Heinrich, Niklas Bittner, and Matthias Hollick. 2022. AirGuard – Protecting Android Users From Stalking Attacks By Apple Find My Devices. In *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. ACM.

[26] Alexander Heinrich, Milan Stute, Tim Kornhuber, and Matthias Hollick. 2021. Who Can Find My Devices? Security and Privacy of Apple's Crowd-Sourced Bluetooth Location Tracking System. *Proceedings on Privacy Enhancing Technologies* 3 (2021), 227–245.

[27] Dennis Heinze, Jiska Classen, and Matthias Hollick. 2020. ToothPicker: Apple Picking in the iOS Bluetooth Stack. In *14th USENIX Workshop on Offensive Technologies (WOOT 20)*. USENIX Association. https://www.usenix.org/conference/woot20/presentation/heinze

[28] Dennis Heinze, Jiska Classen, and Felix Rohrbach. 2020. MagicPairing: Apple's Take on Securing Bluetooth Peripherals. In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks* (Linz, Austria) *(WiSec '20)*. Association for Computing Machinery, New York, NY, USA, 111–121. https://doi.org/10.1145/3395351.3399343

[29] Hex-Rays. 2022. IDA Pro. https://hex-rays.com/ida-pro/.

[30] bunnie Huang and Edward Snowden. 2016. Against the Law: Countering Lawful Abuses of Digital Surveillance. https://www.tjoe.org/pub/direct-radio-introspection. *The Journal of Open Engineering* (21 7 2016). https://doi.org/10.21428/12268 https://www.tjoe.org/pub/direct-radio-introspection.

[31] Iceman. 2022. Proxmark3. https://github.com/RfidResearchGroup/proxmark3.

[32] iFixit. 2019. iPhone 11 Teardown. https://www.ifixit.com/Teardown/iPhone+11+Teardown/126192.

[33] Just a Penguin. 2022. IPSW Downloads. https://ipsw.me.

[34] Kim Jong Cracks. 2022. checkra1n—iPhone 5s – iPhone X, iOS 12.3 and up. https://checkra.in/.

[35] Steffen Klee, Alexandros Roussos, Max Maass, and Matthias Hollick. 2020. NFCGate: Opening the Door for NFC Security Research with a Smartphone-Based Toolkit. In *14th USENIX Workshop on Offensive Technologies (WOOT 20)*. USENIX Association. https://www.usenix.org/conference/woot20/presentation/klee

[36] Tobias Kröll, Stephan Kleber, Frank Kargl, Matthias Hollick, and Jiska Classen. 2021. ARIstoteles–Dissecting Apple's Baseband Interface. In *European Symposium on Research in Computer Security*. Springer, 133–151.

[37] Patrick Leu, Giovanni Camurati, Alexander Heinrich, Marc Roeschlin, Claudio Anliker, Matthias Hollick, Srdjan Capkun, and Jiska Classen. 2021. Ghost Peak: Practical Distance Reduction Attacks Against HRP UWB Ranging. https://securepositioning.com/ghost-peak/.

[38] Jonathan Levin. 2020. **OS Internals, Volume II, Kernel Mode*. North Castle, NY.

[39] Dennis Mantz. 2018. *InternalBlue - A Bluetooth Experimentation Framework Based on Mobile Device Reverse Engineering*. Master thesis. TU Darmstadt. Supervised by Matthias Schulz and Jiska Classen.

[40] Dennis Mantz, Jiska Classen, Matthias Schulz, and Matthias Hollick. 2019. InternalBlue - Bluetooth Binary Patching and Experimentation Framework. In *The 17th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '19)*. https://doi.org/10.1145/3307334.3326089

[41] Travis Mayberry, Ellis Fenske, Dane Brown, Jeremy Martin, Christine Fossaceca, Erik C. Rye, Sam Teplov, and Lucas Foppe. 2021. Who Tracks the Trackers? Circumventing Apple's Anti-Tracking Alerts in the Find My Network. In *Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society* (Virtual Event, Republic of Korea) *(WPES '21)*. Association for Computing Machinery, New York, NY, USA, 181–186. https://doi.org/10.1145/3463676.3485616

[42] National Security Agency. 2022. Ghidra. https://ghidra-sre.org/.

[43] NXP Semiconductors. 2020. NXP JCOP6.x on SN200.C04, Secure Element, Product Evaluation Document. https://www.commoncriteriaportal.org/files/epfiles/nscib-cc-235773_2-st-lite.pdf.

[44] Michael Roland, Josef Langer, and Josef Scharinger. 2012. Relay Attacks on Secure Element-Enabled Mobile Devices. In *Information Security and Privacy Research*, Dimitris Gritzalis, Steven Furnell, and Marianthi Theoharidou (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–12.

[45] Thomas Roth, Fabian Freyer, Matthias Hollick, and Jiska Classen. 2022. AirTag of the Clones: Shenanigans with Liberated Item Finders. (Aug. 2022).

[46] Jan Ruge, Jiska Classen, Francesco Gringoli, and Matthias Hollick. 2020. Frankenstein: Advanced Wireless Fuzzing to Exploit New Bluetooth Escalation Targets. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 19–36. https://www.usenix.org/conference/usenixsecurity20/presentation/ruge

[47] Mridula Singh, Marc Roeschlin, Ezzat Zalzala, Patrick Leu, and Srdjan Čapkun. 2021. Security Analysis of IEEE 802.15. 4z/HRP UWB Time-of-Flight Distance Measurement. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. 227–237.

[48] Michael Spörk, Jiska Classen, Carlo Alberto Boano, Matthias Hollick, and Kay Römer. 2020. Improving the Reliability of Bluetooth Low Energy Connections.. In *EWSN*. 144–155.

[49] Markus Staeblein. 2021. 3 Reasons Why CCC Digital Key Release 3.0 Is Cause for Excitement. https://www.nxp.com/company/blog/3-reasons-why-ccc-digital-key-release-3-0-is-cause-for-excitement:BL-CCC-DIGITAL-KEY.

[50] Milan Stute, Alexander Heinrich, Jannik Lorenz, and Matthias Hollick. 2021. Disrupting Continuity of Apple's Wireless Ecosystem Security: New Tracking, DoS, and MitM Attacks on iOS and macOS Through Bluetooth Low Energy, AWDL, and Wi-Fi. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 3917–3934. https://www.usenix.org/conference/usenixsecurity21/presentation/stute

[51] Milan Stute, Sashank Narain, Alex Mariotto, Alexander Heinrich, David Kreitschmann, Guevara Noubir, and Matthias Hollick. 2019. A Billion Open Interfaces for Eve and Mallory: MitM, DoS, and Tracking Attacks on iOS and macOS Through Apple Wireless Direct Link. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 37–54. https://www.usenix.org/conference/usenixsecurity19/presentation/stute

[52] The iPhone Wiki. 2022. OTA Updates. https://www.theiphonewiki.com/wiki/OTA_Updates.

[53] tihmstar. 2022. `img4tool`. https://github.com/tihmstar/img4tool.

[54] TÜV Rheinland Nederland B.V. 2019. Certification Report, SN200 Series - Secure Element with Crypto Library SN200_SE B1.1 C04. https://commoncriteriaportal.org/files/epfiles/NSCIB-CC-217812-CR.pdf.

[55] TÜV Rheinland Nederland B.V. 2021. Certification Report, NXP JCOP 5.2 on SN100.C58 Secure Element. https://www.commoncriteriaportal.org/files/epfiles/NSCIB-CC-0023577-CR3.pdf.

[56] unc0ver. 2022. The moast advanced jailbreak tool. https://unc0ver.dev.

[57] Wikileaks. 2014. Weeping Angel (Extending) Engineering Notes. https://wikileaks.org/ciav7p1/cms/page_12353643.html.

[58] Jianliang Wu, Ruoyu Wu, Daniele Antonioli, Mathias Payer, Nils Ole Tippenhauer, Dongyan Xu, Dave (Jing) Tian, and Antonio Bianchi. 2021. LIGHTBLUE: Automatic Profile-Aware Debloating of Bluetooth Stacks. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 339–356. https://www.usenix.org/conference/usenixsecurity21/presentation/wu-jianliang