Amazon Echo Dot or the Reverberating Secrets of IoT Devices

Dennis Giese
dgiese@ccs.neu.edu
Northeastern University

Guevara Noubir
g.noubir@northeastern.edu
Northeastern University

ABSTRACT

Smart speakers, such as the Amazon Echo Dot, are very popular and routinely trusted with private and sensitive information. Yet, little is known about their security and potential attack vectors. We develop and synthesize a set of IoT forensics techniques, apply them to reverse engineer the hardware and software of the Amazon Echo Dot, and demonstrate its lacking protections of private user data. An adversary with physical access to such devices (e.g., purchasing a used one) can retrieve sensitive information such as Wi-Fi credentials, the physical location of (previous) owners, and cyber-physical devices (e.g., cameras, door locks). We show that such information, including all previous passwords and tokens, remains on the flash memory, even after a factory reset. This is due to the wear-leveling algorithms of the flash memory and lack of encryption. We identify and discuss the design flaws in the storage of sensitive information and the process of de-provisioning used devices. We demonstrate the practical feasibility of such attacks on 86 used devices purchased on eBay and flea markets. Finally, we propose secure design alternatives and mitigation techniques.

CCS CONCEPTS

• **Security and privacy** → **Usability in security and privacy**; **Systems security**; **Embedded systems security**;

KEYWORDS

Internet of Things, privacy, forensics, reverse engineering

ACM Reference Format:
Dennis Giese and Guevara Noubir. 2021. Amazon Echo Dot or the Reverberating Secrets of IoT Devices. In *Proceedings of Conference on Security and Privacy in Wireless and Mobile Networks, Abu Dhabi, United Arab Emirates, June 28–July 2, 2021 (WiSec '21),* 12 pages.
https://doi.org/10.1145/3448300.3467820

1 INTRODUCTION

Virtual assistant-based systems like Amazon Echo and Google Home gained increasing popularity over the last few years, and are expected to become more important in the future [2]. These systems are used in daily life to control smart home environments, order products, issue queries or organize someone's life. Recent research analyzed the communications channel of such devices revealing the potential for manipulation and privacy leaks [20, 35]. However, little is knows about their internal hardware/software design, operation and privacy mechanisms due to the security protections put in place by the manufacturers. Furthermore, the first Amazon Echo was released in 2014. Since then, several generations of products are integrating Amazons Alexa functionality. New generations might encourage the users to retire older devices, sell them, or give them away. Additionally, devices break over time are discarded. Due to their nature, such IoT devices may contain private information about the user, sensitive log files, and may even give access to the user's Amazon account. Before giving away or discarding devices, it seems obvious that the owner needs to factory reset their devices in order to delete all critical data. However, there are users who are lacking technical knowledge to reset their devices properly. For broken devices, this is not possible at all. Also, the reliability of the factory reset is unknown. The user's expects that the vendor of such devices will try to protect private data.

In this paper, we develop and synthesize a set of hardware/software techniques for mobile and embedded devices into a systematic method for analyzing smart-speakers, and IoT devices in general. We apply these techniques to reverse engineer and analyze the privacy risk of used Amazon Echo devices and the security measures the vendor implemented. We purchased 86 used Amazon Echo Dot devices over 16 months, from sources like eBay and flea markets. Additionally, we analyzed new and "certified refurbished" devices.

While there are several Amazon Echo models, we focus on Echo Dot's as they are the most affordable and common devices. Our focus is on the third generation, released end of 2018. We analyse the effectiveness of factory resets and investigate the implemented protections on user data. We show that private information, including all previous passwords and tokens, remains on the flash memory, even after a factory reset. This is due to wear-leveling algorithms of the flash memory and lack of encryption. We identify and discuss the design flaws in the storage of sensitive information and the process of de-provisioning used devices. Additionally, we apply our forensics techniques to other manufacturers and discover similar vulnerabilities. These results can be found in 8. Therefore, demonstrating that our method for reverse engineering will help other researchers for reverse engineering a variety of IoT devices. Finally, we propose secure design alternatives, and short-term mitigation techniques effective against adversaries with hardware attack capabilities. Our contributions are summarized as follows:

- Systematic and comprehensive reverse engineering of Amazon Echo Dot hardware and software platforms.
- Data extraction techniques of user data even after factory resets exploiting the hardware architecture, lack of encryption, and side effects of wear-leveling.
- Demonstrating the feasibility of breaching the users privacy by reconnecting used devices to the cloud.

WiSec '21, June 28–July 2, 2021, Abu Dhabi, United Arab Emirates

ACM ISBN 978-1-4503-8349-3/21/06…$15.00
https://doi.org/10.1145/3448300.3467820

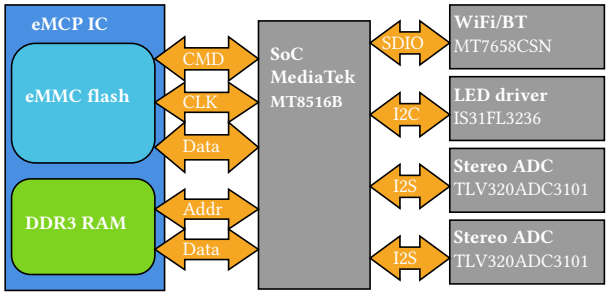


Figure 1: Echo Dot 3rd Gen logical schematic

- Providing an advanced method to instrument access to eMCP flash without repeated soldering.
- A privacy-preserving study of 86 used devices revealing most users do not adequately reset their devices.
- Responsible disclosure to Amazon
- Generalization to findings to other smart speakers.
- Mitigation and discussion of trade-offs between usability, security, and hardware requirements.

2 THE AMAZON ECHO PLATFORM

While Amazon sells several models of Echo devices, the underlying hardware and software platform remains unchanged to a large extent. We decided to focus on Amazon Echo Dots due to their popularity and ubiquity in homes. Their main distinguishing characteristic is the lack of display or advanced user interface. Instead, the user interaction is through buttons, LEDs, and voice input and output. Our hardware teardown and software architecture analysis, of over 98 used and new, Echo Dot devices reveals the following.

SoC/CPU: The central component of Amazon's Echo Dots is the System on a Chip (SoC). Third (3rd Gen) generation devices rely on Mediatek MT8516B SoC, which is a 64-bit quad-core ARM device. The MT8516B is designed specifically for voice assistant devices [28]. The SoCs integrate *ARM TrustZone* support, a key hardware component to secure ARM-based platforms, e.g., to provide Secure Boot or store keys securely. There is little technical information publicly available for these SoCs.

RAM and Flash Storage: In most of the analyzed devices, the RAM and eMMC-flash are integrated in a single IC, the embedded Multi-Chip Package (eMCP) (See Figure 2). The eMCPs come in different models and are sourced from multiple manufacturers, depending on the generation, availability and cost of chips. Known vendors of eMCPs are Micron, Samsung and SKhynix. For Gen 3 (FCC ID: 2AOAG-3668), we found the size of the flash memory varying between 4 and 8 GBytes. We also found that there is a new PCB revision of generation 3 devices (FCC ID: 2ARIW-2778, 2ASD2-7483) with separate RAM ICs and raw NAND flash.

Connectivity and interfaces: Echo Dots have several communications and connectivity peripherals. They support Bluetooth and dual-band Wi-Fi. Generation 3 devices use MediaTek MT7658CSN ICs, combining Bluetooth and Wi-Fi. For interfacing, they integrate LEDs, Buttons, ADCs, a microphone array, and speakers. The PCB has USB and UART debug interfaces as test-points (See Figure 3).

Operating System: Echo Dots run Fire OS, which is a customized Android operating system [10]. Generation 3 uses Android 7.1.2 (Nougat). The flash memory contains two copies of the Android

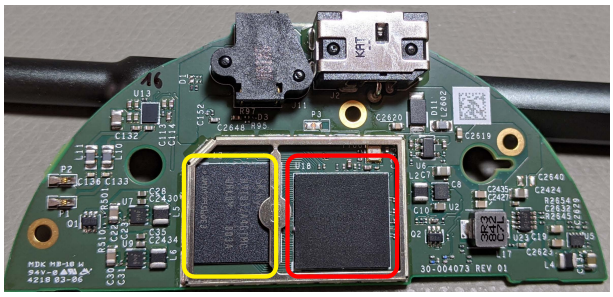


Figure 2: Echo Dot 3rd Gen PCB: SOC (red) & eMCP (yellow)

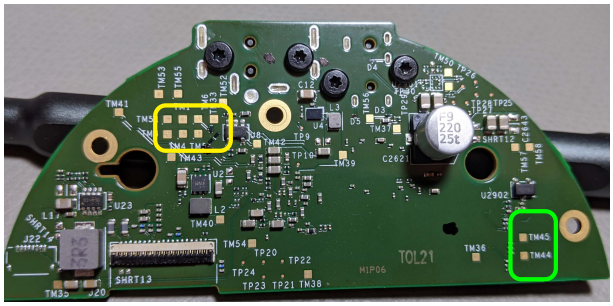


Figure 3: Back side of Echo Dot 3rd Gen PCB with position of USB Debug pins (Yellow) and UART (Green)

root partition (active and passive copy), the kernel image and the Trusted Execution Environment (TEE) firmware. The operating system has *SELinux* enabled and in "enforcing" mode.

File systems: The Android operating system root partitions are stored on an Ext4 file system, which is not encrypted, but integrity protected by *dm-verity*. For the 3rd Gen devices, the operating system uses around 800 MByte. The user data is stored on a separate user partition which is formatted with Ext4. This partition is neither encrypted nor integrity protected.

Manufacturer configuration: This is written onto the device at manufacturing and is device-specific. We identified the special "Boot2" boot partition that has the partition signature "beefdeed2.1" and used as NVRAM. It contains the model number, product ID, serial number, Wi-Fi MAC address, Bluetooth MAC address, and region. This storage is also used for bootloader configuration and control. It also contains Device Hardware Authentication (DHA) credentials, which consist of a private key and a certificate. The private key is stored in an encrypted form in the *KeyBox* (KB) partition and is decrypted by a *TrustZone* application. The certificate, which is signed by Amazon's Device Certificate Authority (CA), is stored in the "persist" partition in plaintext.

Secure Boot: Echo devices use the Android Verified Boot concept [33]. On Power-ON, the Boot ROM (stored on the SoC) is the first code to execute. It verifies and loads the Pre-loader from the flash storage. The verification of Boot ROM integrity is done using the manufacturer's RSA public key. The hash of the public key is burned in the eFuses of the SoC. This Pre-loader sets up the basic hardware and does the first security checks. Then, it loads the signed ARM Trusted Firmware (ATF), verifies and executes it in *TrustZone Exception Level* (EL3). ARM's 64-bits architecture defines four exception levels (EL0-EL3) with increasing privileges to access registers, memory, and hardware features. EL3 being the

highest and only level that can change security state. In this step, the *Little Kernel* (LK) is verified and loaded into memory. The anti-rollback protection is also performed in this step. It checks (e.g., Replay-protected-memory-block (RPMB)) the currently installed version against the minimum expected version to prevent malicious rollback to potentially insecure earlier versions. However, not all analyzed devices utilize RPMB, as we noticed that RPMB can be empty and not provisioned. In the next step, the LK verifies the signed Linux kernel image and loads it. The Linux kernel then mounts the root partition and ensures its integrity using *dm_verity*. After boot, the serial connections are disabled. The access over USB is limited as the bootloader is locked and no open services in Android are listening on USB. We were able to observe and verify the boot process by analysis of the boot log via UART.

Apps: The Fire OS core is similar to a standard Android OS. The system contains standard tools as *busybox*, *iptables*, *ip6tables*, *logcat*, *dnsmasq*, *hostapd* and *wpa-supplicant*. *Dnsmasq* is used as a DNS resolver. A list of DNS names and the respective IP addresses of Amazon services is stored on the user data partition. *Hostapd* is used to configure and deploy an unprovisioned device's Wi-Fi access point. *WPA-supplicant* is responsible for connecting to configured access points after provisioning. We found that it creates its configuration files on the user data partition in the folder "*misc/wifi/*". Here the file "*wpa_supplicant.conf*" contains the Wi-Fi credentials, such as the SSID and PSK. If multiple wireless networks are configured, we discovered that this file stores all of them. In the same folder, there is the file "*networkHistory.txt*" which contains a logfile of all connections to wireless networks, including the BSSID (MAC address of the Wi-Fi AP). This file can also contain a list of surrounding Wi-Fi APs including their BSSID, frequency and signal strength. Copies of the "*wpa_supplicant.conf*" and "*networkHistory.txt*" can be also found in the free space of the EXT4 file system.

The logical functionality of the device is implemented using Android apps. Their purpose ranging from connectivity apps to device management, telemetry, cloud interaction and voice recognition. The system partition contains basic voice models for the wake-up word detection, but user-specific voice models are stored on the user data partition in the folder "*data/securedStorageLocation/models*". We noticed that these models may differ for different owners of devices. We found the usage of "Pryon" framework in the wake-up word detection [6]. Here the speech is converted to text and matched against a list of regular expressions.

The device often uses SQLite on the user partition for storing user data. Fire OS uses the Android Account Storage and some user information, e.g., the owner name is stored in "*accounts.db*" and "*0.xml*" located in "*system/users/0/*". The device identity and token for Amazon Device Messaging [11] are stored in the folder "*com.amazon.device.messaging*". The central storage for log files is "*data/system/dropbox*", including kernel, event and debug logs.

Provisioning process: An unprovisioned device will enter the setup mode after booting up. This is indicated through a rotating yellow light ring on the device and by the announcement to use the Alexa app to set up the device. At this point, the device has created an open Wi-Fi access point with a unique SSID in the form of "Amazon-XXXX". The user is asked to open the Alexa app to log into the Amazon Account and enter the credentials of user's home Wi-Fi. The devices only support WPA/WPA2 PSK for private customers. The app then connects to the open configuration Wi-Fi access point of the device and transmits the credentials. The device then connects itself to the assigned Wi-Fi network and to the cloud. If the connection succeeds, the device appears in the owner's Alexa app and can be configured from there. Upon provisioning firmware updates are installed automatically. By default, the Wi-Fi credentials are also uploaded to the Amazon Cloud. To prevent this, the user needs to actively opt-out in the setup process. Echo devices also support Amazon Zero-Touch Setup, which is also called Wi-Fi simple setup [9]. Here, the device is linked in the cloud backend to the owner's Amazon account at the time of the purchase. The requirement for this is an already existing and active Amazon device, the provisioner devices, in proximity. This device does not need to belong to the same owner. When the device is powered on for the first time, it will try to connect to a hidden Wi-Fi SSID of the provisioner device and connect through it to the Amazon backend. Using this channel, it will configure its Wi-Fi access and connect to Amazon cloud. The configuration is done over a secure channel which is protected by usage of the DHA credentials [12]. As the DHA credentials are protected by *TrustZone* and are used for the authentication, an impersonation of the device or Man-in-the-Middle (MitM) attack is not trivial.

Unprovisioning process: Amazon Echo devices can be reset via the Alexa app, Amazon website, or device. In the app or website, the user selects the device and issues a factory reset command. This de-registers the cloud binding and issues a factory reset command to the device. If the device is connected to the Internet, it reboots and performs a reset. After this, the device will be in setup mode ready for provisioning.

A user also has the possibility to reset the device locally by pressing buttons in a certain sequence. The exact buttons and sequence differs between generations. 3rd Gen devices require the press of the "Action" button for 25 seconds. The devices will signal the process by turning the light ring orange. In our tests, we noticed that the device enters the "setup mode" after 10-15 seconds and turns the light ring orange. However, it did not trigger a factory reset, but instead a Wi-Fi reconfiguration. While the device was in Wi-Fi reconfiguration mode, a power off and power on would reconnect the device to the network again without actually resetting it. This behavior misleads users to believing that the factory reset is successful and release the buttons too early.

Depending on whether the device is connected to the Internet, a local factory reset will de-register the cloud binding. A device reset without an Internet connection, e.g., outside of the configured Wi-Fi, will remain associated to the user account and the cloud binding won't be de-registered. A user need to manually delete the association of a device over the Alexa app or Amazon website. This can be problematic as detailed in section 5.3.

Vulnerabilities: To this day there were no vulnerabilities or rooting methods reported for Amazon Echo Dots (2nd and 3rd Gen) [4]. However, there are public root methods for other Amazon devices, like the Fire tablets [3][5][8]. These methods abuse a vulnerability which enables to replace the *LK* with a modified version and still pass the verification process. We found the same exploits and methods can be likely adopted using eMMC access for Amazon Echo Dot devices as they use the same SoC and similar firmware.

In future research this can be analyzed as a way to potentially bypass the Secure OS boot process and execute modified software.

3 NAND/EMMC FLASH FORENSIC

We first provide an overview of various flash-based storage systems and their key characteristics, then present the techniques we developed for data extraction from Echo Dot. We assume access over the device software itself is not possible.

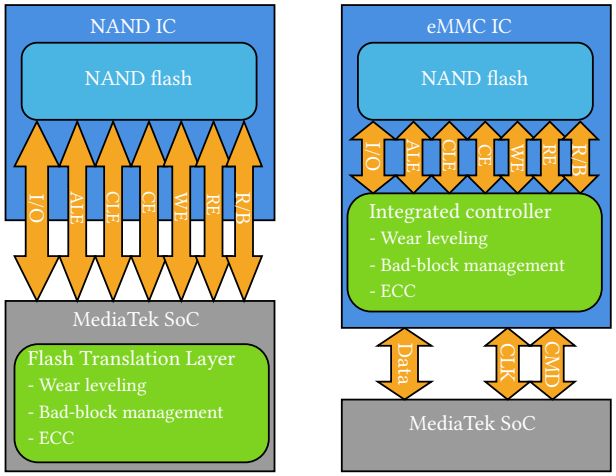
3.1 Overview of Flash-Based Storage

Flash memory is widely used in embedded devices as non-volatile memory because it provides an easy way to store data on a small area and due to its power efficiency. Flash generally comes as NAND or NOR flash [36]. For IoT, and more general, embedded devices usually NAND flash is used due to its cost and high density. NAND flash can come in different packages and types.

NAND characteristics: NAND flash is typically organized in planes, blocks and pages. A page is the smallest unit and typically has a data size of 512 Bytes for 1 Gbit to 4 Gbit NAND flashes, and 2048 Bytes and more for bigger sized flashes. In addition to the data size, the page contains also a "out-of-band" (OOB) area which is used for management and error correction codes (ECC). Primary usage of the management is wear-leveling and bad block management. Multiple pages are organized in blocks, for example, a block can contain 64 to 256 pages. The exact number of pages per block and size of the OOB area are vendor- and flash- specific. NAND flash has a typical endurance rating of 100,000 writes per block. NAND flash by nature is a fast and cheap storage solution but is also unstable [19]. Reads and writes produce bit errors which need to be corrected by ECC. NAND flash is written to at page level. To be able to write to a page, it needs to be empty. A page is empty, if all physical bits are set to "1". In order to erase data, NAND flash requires the whole block to be erased. The process of erasing NAND flash has implications for wear-leveling. Instead of deleting a block every time information on a page being changed, the information is written to a new block and the old page is marked as invalid. This spreads the wear over the whole flash over time. However, the actual information still remains in the page. This data might get deleted at some point by the garbage collection of the flash controller. This behavior prolongs the life of the flash memory but creates the issue of the secure deletion of sensitive data. The general problem of data remanence has been known for a long time [24].

Unmanaged NAND: The simplest type of NAND flash is unmanaged NAND, which can also be called raw NAND flash. Standards describe features, interfaces and packages [23]. The management is done by using a Flash Translation Layer (FTL) in the OS as part of the SoC itself. The FTL takes care of the wear-leveling, ECC, and bad-block management. The exact implementation depends on the used controller and can be delegated to the OS. The controller is connected to the flash by 8 or 16 data lanes, and typically 7 control lanes. Unmanaged NAND flash is usually cheaper. Due to its connection, the SoC or OS can access the raw pages of the flash. The connection between SoC and NAND IC is illustrated in Figure 4a.

Managed NAND: Another kind of NAND flash is managed NAND. In the context of embedded devices typically *Embedded Multimedia Card(eMMC)* and *Embedded Multi-Chip Package(eMCP)* ICs are



(a) Unmanaged flash (NAND) **(b)** Managed flash (eMMC)

Figure 4: Comparing the connection between raw NAND and eMMC flash

used. eMMC ICs typically use FBGA-153 packages, while eMCP use FBGA-221. eMCP ICs are a combination of eMMC flash with DRAM on one chip, reducing the number of parts on the PCB. By eMMC, we will denote both eMMC and eMCP ICs. In contrast to unmanaged NAND, eMMC ICs contain a flash controller embedded on the chip, which acts as an interface between the SoC and the NAND flash. This controller takes care of the wear leveling, ECC, and bad-block management internally. The supported features of eMMC memory itself are defined in the JEDEC standards. For eMMC ICs which support the eMMC standard 4.4 and newer [13], the wear leveling mechanism offers features as *TRIM*, *Discard* and *Background*. These features improve the performance of the flash and effectiveness of the garbage collector [16]. Additionally, the standard defines security features like *Secure Trim*, *Secure Erase* and *Replay Protected Memory Block (RPMB)*. Version 4.5 of the eMMC standard introduced the *Sanitize* command for secure data removal [14]. All of eMMC flashes we encountered supported at least eMMC version 4.5. Interestingly, none of the devices used the *Secure Trim* or *Sanitize* functions. The electrical interface between SoC and eMMC is also defined in the JEDEC standards and consists typically of 2 control lanes and 1 or more data lanes [14, p. 189]. A connection between SoC and eMMC ICs is illustrated in Figure 4b. In our analysis, we observed the actual implementation of the controller features and flash management being highly depending on the manufacturer and model of the IC. However, no managed NAND exposes the raw NAND flash to the SoC or host directly.

NAND Forensics: Direct access to unmanaged NAND flash is standardized and tools are available. However, the interpretation of the data requires knowledge about the used wear-leveling mechanism, ECC and XOR method [30].

The mechanism might be different even for the same NAND IC as it is dependent on the type of the used FTL. There are methods to identify the used ECC [39] and XOR method [37].

For eMMC flash the controller prevents direct access to the (physical) raw flash. In contrast to unmanaged NAND the access to the logical information does not require any additional knowledge about the flash management, as the eMMC controller manages it transparently. Most eMMC ICs have test pads, which bypass the

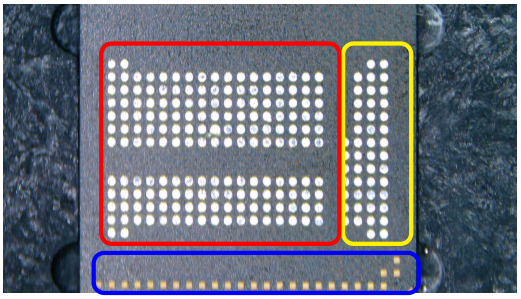


Figure 5: Pin layout of Hynix BGA221 eMCP ICs, with DDR RAM (red), eMMC (yellow) and NAND test pads (blue)

controller [34]. This access method is used by commercial companies and tools to recover data from eMMC flash. Examples for the layout of pads of an eMCP IC can be found in figure 5.

For extraction and analysis, we use different methods, depending on the reset state of the device. For devices that have not been factory reset, we can use the In-System-Programming (ISP) method. Otherwise, we use the Chip-Off method, which is more invasive. Additionally, we developed an advanced Chip-Off method.

3.2 Data Extraction using ISP

It is not necessary to remove the eMCP IC when the device is still provisioned. Instead, it is possible to access the flash using test-points and traces, a method widely used for In-system programming (ISP). To be able to use the method, it is required that we reverse engineer the PCB to find points where to attach to the flash signals. The schematics of the PCBs are not public. By removing the eMCP IC and the SoC, we identified the traces for CMD, CLK and DAT0. While they are easily accessible and known for Gen2 devices [25], the DAT lines for 3rd Gen are using buried vias and were not published yet. Buried vias are more difficult to trace and to access. We developed a method to access that line using a conductive needle. For access to the flash it is not required to find all DAT lines. Instead, DAT0 is sufficient [14, p.8] [21]. In order to use the ISP method, it is required that the SoC is kept in a reset state to not interfere with the eMMC access. This can be achieved with various methods, we chose to connect the crystal to the ground. This disables the clock for the SoC and stops the SoC. The Vias for DAT lines are shown on Figure 6. Figure 7 shows the position for the test points of CMD and CLK. We can access the flash memory without removing any chip by only using conductive needles.

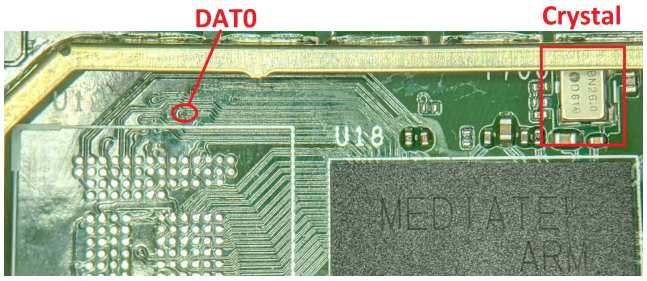


Figure 6: ISP points for Echo Dot 3rd Gen on PCB front side

3.3 Chip Removal and Re-soldering (Chip-Off)

The NAND interface pads of an eMCP IC are not used and not connected for normal operation. Therefore, they are not accessible

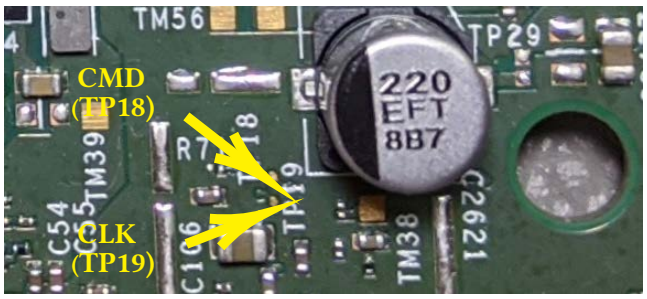


Figure 7: ISP points for Echo Dot 3rd Gen on PCB back side

when the chip is soldered to the PCB. It is required to detach the chip from the PCB to use an external adapter to dump its contents. This process is similar to traditional flash forensics [15].

This process takes around 20-30 minutes per device, in addition to the time needed for data retrieval. Every time when an event occurs, e.g., when the device performs a factory reset, this needs to be monitored and the process needs to be repeated. The reballing, placement and soldering of the IC requires precision, as the correct connection of the RAM is required for the operation of the device.

3.4 Hybrid Chip-Off for eMCP

One disadvantage of the Chip-Off method is that the removal of remaining solder potentially causes damages to the solder-mask of the PCB or eMCP IC. Additionally, both components are exposed to thermal stress in the process. With every iteration the risk of breaking PCB pads or creating shorts under the IC increases. It is difficult to keep the eMCP chip externally, as DRAM has strict requirements for the electrical interface and signaling quality. An external eMMC chip is possible, however, the RPMB used by some devices, prevents the use of a replacement chip in such cases.
We developed a method that allows us to repeatedly access the contents of the flash without soldering every time. For this, we use a donor eMCP chip for the RAM and we use the eMMC portion of the original eMCP chip externally. As a preparation, the original eMCP chip is removed and the eMMC portion of the pads on the PCB are masked. Then the donor eMCP chip is soldered onto the PCB. The masking prevents that the eMMC portion of the donor chip is electrically connected to the SoC. We use an adapter for the original eMCP chip and connect the eMMC pads via the previously described ISP points. The connection is illustrated in Figure 8. This method enables us the easy observation of data which have been written through the eMMC interface and its representation in raw NAND. The risk of damaging components in each iteration is minimized. This method is faster in comparison to the traditional chip removal. Additionally, this method is very powerful as the communication between the SoC and eMMC can be intercepted and even modified. This can be beneficial for future research with IoT devices which use eMCP ICs.

3.5 Reading raw NAND from eMMC/eMCP ICs

Reading data from raw NAND of eMMC/eMCP ICs can be very challenging as the implementation of the controller defines how the data is stored on the flash memory. The manufacturers of ICs

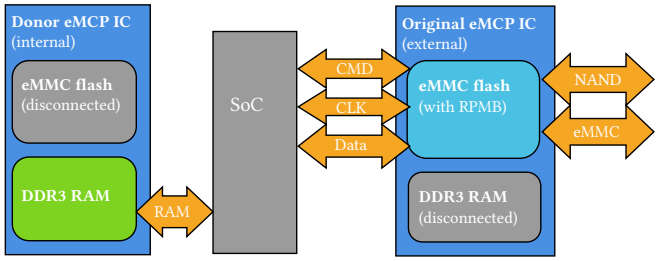


Figure 8: Hybrid Chip-Off method

do not publish any information about the internal functionality or the layout of the chips. No universal standard seems to exist for controllers to manage the data. For instance, we encountered data that has been scrambled using an XOR operation with an unknown pattern by the controller. Furthermore, the exact method of logical to physical addressing is unspecified.

The first challenge is to connect to the raw NAND by bypassing the controller. We used a logic analyzer to find the pin-out and applied observations about the behavior of generic NAND from [30]. At power on the controller runs several operations on the NAND flash. The first operation is "ID Read", which is a standard operation and returns the 5 Bytes long NAND ID code. By observing this operation, it is possible to distinguish the control pins (CLE, CE, WE, ALE, RE) and most of the IO pins. The pads we found for Hynix eMCP ICs can be found in Figure 9.

The second challenge is to determine the geometry of the NAND, such as the page and block size. Some of this information is available online, like in the Linux MTD subsystem documentation [1]. For a unknown geometry, we determine the page size by first assuming a big page (e.g., 20 KBytes) and then read a few pages. Each of these pages likely contains multiple pages. By aligning distinct structures, like the OOB area, it is possible to determine the page size. The entropy of the OOB area is noticeably lower in comparison to random data. The same approach can be used to determine the block size and the number of blocks. For this paper, we only want to detect the presence of information, therefore we ignore the error correction and the exact addressing of the data.

The last challenge is the unknown XOR pattern. The exact XOR mechanism depends on the manufacturer. For the analyzed ICs it was not publicly available. Our approach was to fill the flash with zeros multiple times over the eMMC interface. Additionally, we changed a single bit in a block of zeros which had the size of a page and filled the flash with it. The goal is to ensure that as many pages as possible are filled with zeros. Later we would revert the changed single bit. In case of Hynix eMMC the resulting flash dump could be used as the XOR key in order to unscramble data contents of other dumps. While this method worked for the Hynix eMMC, it may not work for other vendors and models. A more efficient, but more complex approach consists of extracting the eMMC controller firmware and the scrambling keys [15, 39]. However, such an approach is out of scope of this paper, as it is not necessary to reach our objective.

4 EFFECTS OF WEAR-LEVELING

In order to analyse the effectiveness of factory resets, we needed to estimate the likelihood that a particular information is deleted.

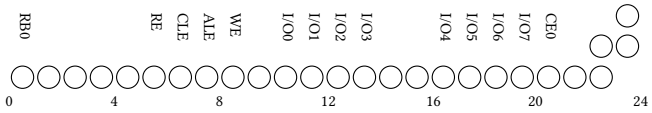


Figure 9: Hynix eMCP NAND test pads

We developed a method to estimate the probability that a piece of information is still present after deletion or factory resets. The goal was to replicate the behavior of the analyzed devices.

Selection of IC: We chose the eMCP which we found most frequently in 3rd Gen devices. We are aware that different models and versions of flash might behave differently. The eMCP "Hynix H9TQ32A4GTMCUR-KUM" is an IC which combines 4 GBytes DDR3-RAM and 4 GBytes eMMC flash in a BGA-221 package. The page size of the flash is 9040 Bytes. One block consists of 256 pages and the flash has 2112 blocks in total. The boot areas, the RPMB and the user area contain 3825672 KBytes in total. The raw NAND equals 4773120 KBytes in size, which includes the OOB area. For our experiment, we used new ICs, which were initially empty.

Creation of a test image: We assumed that the typical Wi-Fi credentials are smaller than 512 Bytes. For our experiment, we have chosen a size of 310 Bytes for an example *wpa_supplicant* configuration with valid Wi-Fi settings. As the base image we used a copy of an Amazon Echo Dot 3rd Gen's flash memory, where the data partition (1.5 GBytes) was zeroed. We assume further, that a device cannot delete the non-user data partitions, e.g., system or bootloader. All non-user data partitions combined have a size of 2.2 GBytes in total, 0.6 GBytes being empty (filled with "00").

Process of writing and erasing: In order to keep track of information, we created flash dumps after every step. These dumps consisted of the data in the raw NAND and data which was accessed via the eMMC interface.

Using blank eMCP ICs, the base image was written to the chip. An initial dump was created. Then the data partition was initially formatted with "mkfs.ext4", followed by another dump. The drive and file system caches have been disabled. To detect data, we used a similar approach as in [26, p.49 ff] and [38]. A unique pattern of 310 Bytes was written in 100 new files. These files are defined as "static". After each file 1 MByte of random data was written in another file to ensure that the unique pattern is spread over multiple blocks. This should also simulate the other files which are usually stored on the data partition.

In a second step 100 additional files were created . This time these files were modified and an additional line with a pattern was added. This simulates the wear-leveling in case of modification of files, like changes of configuration files. These files are defined as "dynamic". After the creation of the test files, a dump was created, which acted as the baseline for the following experiments.

To simulate a factory reset, the data partition was formatted under Linux with "mkfs.ext4". The raw NAND and eMMC dump was compared to the baseline. Then, in every iteration 200 files with 1 MByte size were written to the data partition, while the existing files were deleted. After each iteration a dump is created. Between iterations, the chip remained powered on for 15 minutes in order to give time for the garbage collection algorithm of the controller and to replicate a life-cycle of a device.

After completion of the experiment, we counted the remaining

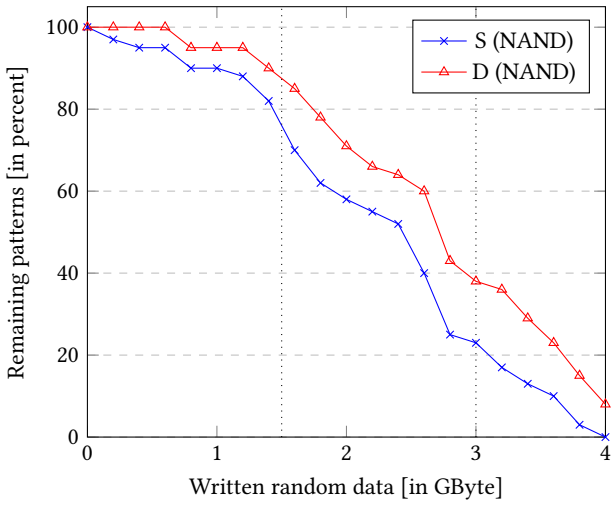


Figure 10: Remaining patterns after appending random data, with reading via direct NAND interface (averaged)

unique patterns in each dump by using fuzzy matching. Due the nature of NAND flash individual bit errors are likely and by ignoring ECC, the tool needed to be tolerant to bit errors. Additionally, we repeated each NAND flash dump multiple times to further increase our accuracy. We verified the non-changed partitions in each iteration over the eMMC interface and made sure they were not changed. The results of our experiment are shown in Figure 10.

Interpretation: Erased data continues to exist in the raw NAND flash for a long duration. Multiple copies of patterns could be found, especially for the "dynamic" files. This can be explained by the wear-leveling mechanism which tries to avoid erasure of blocks. *TRIM* and *Discard* being enabled seem not to ensure that data gets deleted. We noticed that the garbage collector does not immediately delete blocks with erased data. The amount of deleted blocks increases over hours when the device is idle, but it is not deterministic. This behavior is especially problematic for factory reset devices, as the previous owner would power off the device shortly after the reset, meaning garbage collection likely has not yet erased all the private data. Our experiments match the observations for SSDs in [38] and [29]. We conclude that formatting the file system and overwriting it with random data does not reliably delete the data from the physical flash.

5 ATTACKING USED ECHO DOT DEVICES

Our attacks on Echo devices require physical access. In our attacker model the attacker can get physical access to a device by obtaining a used device, e.g., by purchasing them on eBay or flea markets, or by finding broken devices in the trash. The attacker does not need sophisticated tools or knowledge. The required tools can be obtained for less than 100 USD, which however only allows the extraction of still provisioned devices. As devices can be resold after the extraction, the attack will be financially neutral. The data can be used by the attacker for various crimes (e.g fraud) or used for pranks. Also, such data can be collected and sold, e.g. to malicious persons in a particular region or proximity of a victim.

5.1 Availability of used devices

There are many different sources for used devices. An attacker can obtain used/broken devices locally, e.g. by e-waste or flea markets. Another source are eBay or Facebook market place. We observed the availability of used devices on eBay over multiple months in 2020 and 2021. Each week 300-400 used devices were offered on the platform in North America. The amount of "broken" or "for parts' devices varied between 10-30 per week. Due to the implementation of Facebook's market place, it is difficult to determine the exact number of offerings. However, the number was in the thousands. Obtaining devices by mail has the advantage, that the senders name and address might be known to the attacker.

5.2 States of used devices

Used or broken devices can come in different states, which have an implication of at risk information.

(State 1) Device not reset and still connected to the cloud: In this state the device was not reset by the previous owner and still has existing cloud binding. The Wi-Fi credentials are still present on the device and the data partition contains pieces of the previous owner's information. The account owner can see the device in the Alexa app. If this device can be connected to the Internet, it will have access to the previous owner's Amazon account. For an attacker this is the best scenario, as it offers them the broadest and easiest access to a victim's data.

(State 2) Device not reset, but cloud binding deleted: This state is similar to the previous one, however here the previous owner deleted the device in the Alexa app. While the device still contains the Wi-Fi credentials and other private information, the cloud binding is invalid. If such a device gets connected to the Internet, it will not have access to the previous owner's Amazon account.

(State 3) Device reset, but cloud binding still exists: This scenario is rare, as it requires that the device reset happened while the device did not have an active Internet connection. The data partition was wiped, but the Amazon cloud is not aware that the device has been reset. When restoring this information, the device can be reconnected to the cloud and to the previous owner's account.

(State 4) Device reset and cloud binding deleted: When a device is reset and has an active Internet connection, the cloud binding will be invalidated. If an attacker can restore the information on the device, it is not possible to access the Amazon account. Other information can be still extracted but that requires more effort.

(State 5) Device was never used: If a device has never been used, it does not contain any private information. Devices that are still sealed or have never been powered on can be sold as "used". This is the worst scenario for an attacker, as there is no data to gain. However, if the device was linked to a victims Amazon account when purchased and it can be provisioned using Amazon Zero-Touch Setup, the attacker could have a fully provisioned device.

5.3 Extraction of data of Echo devices

In preparation for the case study, we provisioned *ourselves* 6 new Amazon Echo Dot devices with test accounts at different locations and different Wi-Fi APs. Over a period of multiple weeks, we used these. We paired different smart home and Bluetooth devices with the provisioned devices. Directly after this period, we disassembled

the devices and extracted the flash content. For these provisioned devices we confirmed that the data was accessible, and we could retain the Wi-Fi credentials. To test our assumption regarding the cloud connection, we moved the reassembled device to a different location and created a Wi-Fi access point with the extracted credentials. Our assumption was, that the device would not require an additional setup when connected at a different location and Wi-Fi access point with a different MAC address. We confirmed that the device connected successfully, and we were able to issue voice commands to the device. When asked "Alexa, Who am I?", the device would return the previous owner's name. The re-connection to the spoofed access point did not produce a notice in the Alexa app nor a notification by email. The requests are logged under "Activity" in the Alexa app, but they can be deleted via voice commands. We were able to control smart home devices, query package delivery dates, create orders, get music lists and use the "drop-in" feature. If a calendar or contact list was linked to the Amazon account, it was also possible to access it. The exact amount of functionality depends on the features and skills the previous owner had used.
Before and after a factory reset the raw NAND flash was extracted from our provisioned devices using the Chip-Off method. Additionally, we created a dump using the eMMC interface. To find information in the resulting dumps, we had to develop a method to identify interesting information.

5.4 Identification of information

As we provisioned the devices ourselves, we knew what kind of information the device could potentially store. We created a list of keywords which would be used to look for this specific information. We divided the keywords in four different categories: information about the owner, Wi-Fi related information, information about paired devices and geographical information. The individual keywords per category can be found in Table 1.

The dumps of the flash were searched for the keywords. On the pre-reset and post-reset eMMC image we used the forensic tool *Autopsy*, which supports keyword search on images [7]. The NAND dumps were analyzed manually. The name of the Amazon account owner was found multiple times. In all analyzed devices multiple complete copies with contents of the "*wpa_supplicant.conf*" file were found. The number of found artifacts varied between 22 and 37 copies. Additionally, we found artifacts of log-files in raw NAND and the eMMC dump. We confirmed that the EXT4 data partition has been deleted. All artifacts were spread over the NAND flash and were not stored at the same location.
By obtaining the positions of the keywords, we identified the files and places where privacy related information is stored. For example, we observed that we would find Wi-Fi credentials close to strings like "ssid=". Having this information enables us to detect the presence of information in unknown memory dumps in an automated way. This is important to automatically analyze real users' devices in a privacy-preserving way as discussed later.

5.5 Retrieval of address via voice commands

While we found out that it is not directly possible to ask Alexa for the previous owner's address, we were able to find the approximate location by asking indirect questions. When asked about restaurants, Alexa will return restaurant names, addresses and distances in proximity of the address. The same applies to grocery stores, post offices, public libraries, public transport stops and other locations. Using this information, it is possible to narrow down the approximate position. In experiments in city locations (100.000 residents) we were able to get an accuracy of approximately 150 meters. The best estimation was possible if Alexa has been asked about nearby grocery stores. We expect that the accuracy might differ for different types of areas.

5.6 Retrieval of owners address via log files

The Wi-Fi credentials enable the attacker to connect to the previous owner's Wi-Fi. With the SSID and the stored Wi-Fi access point MAC addresses, it is possible to search for the previous position using search engines like WiGLE [1]. The more precise Google Geolocation API [22] requires at least two MAC addresses for the position retrieval. Due to the network connection log-files and Wi-Fi survey log-files, it is possible to find two or more MAC addresses of the surroundings of the previous owner. In some cases, a device owner used two or more Wi-Fi access points with the same SSID (e.g., for a larger home). Some Wi-Fi credentials might contain personal or location information. Interestingly, we found that kernel logs censor the SSID, while event logs censor the BSSID (but not SSID).

6 CASE STUDY OF USED DEVICES

For our case study, we bought over 86 used Amazon Echo Dot devices from private sellers on eBay and flea markets. To have a comparison to different countries, we purchased devices from the US, Germany and the Netherlands. Devices that one can find on platforms like eBay can have different states. Some of the devices are sold as used devices even though they never have been used and are still in their original package. These devices never contained any user information and are therefore irrelevant for our analysis. For the purchase, we focus on devices which are described as "used". As an additional test group, we purchased 6 "certified refurbished" devices from Amazon directly, which are in stock periodically. All our purchases were made between July 2019 and November 2020.

6.1 Privacy-preserving analysis process

As we are aware that some devices might contain private information, we developed a process to maintain the user's privacy and not use, manipulate, or reveal private information.
Upon arrival, the devices were labeled and inspected. No personal information of the sender/seller is associated with the device.
For devices which have been purchased at a flea market, we did not have any information about the sellers.
The devices are powered-on in order to check if they are still provisioned. This can be determined if the "yellow ring" comes on after approximately 30 seconds after the device boots. If the ring does not turn orange, the wake-up word "Alexa" is said and if the device responds that it has trouble connecting to the network, the device is marked as "no reset". At this point the assumption is that the device still contains data, but it is unknown if the cloud binding is still active. Due to ethical and legal considerations, there is no simple way to verify if a device is still connected to a user account.

[1] https://wigle.net/

As we did not associate a device to a seller, we cannot ask specific questions. For this reason, we did not verify this.
If the "orange ring" comes up, the device has been reset to factory settings or has never been provisioned. Therefore, it is not known if there are traces of information on the device or if there is still a cloud binding remaining (e.g., reset outside the range of the Wi-Fi).
In the next step, the device is disassembled, as we did not find a trivial method of dumping the flash memory from outside the device. For that the bottom rubber plate is heated up to loosen up the adhesive and to remove it. This plate hides the screws. The device can be disassembled, and the PCB removed. The ICs inside are of metal shielding cans which can be easily opened. For provisioned and non-provisioned eMMC based devices the next step differs:
If the device is still provisioned, we can use ISP to create a dump of the flash memory. A script confirms the presence of Wi-Fi credentials and other information. Private information, like owner name or Wi-Fi passwords, are not extracted. As the devices were not associated to a particular seller, we have no association between SSIDs / MAC addresses and a particular person.
For devices which are not provisioned, or which are using a NAND flash IC, we use the Chip-Off approach as the deleted data is not accessible through the eMMC interface. Here we use the methods described in Section 3. Depending on the flash type, we can use the test pins of the eMMC or directly read the NAND flash using a flash reader. After creating the dump, a script automatically checks for traces of Wi-Fi credentials and owner information.
In case of no traces, we assumed that the device was never provisioned. This assumption also was based on the condition of the device (e.g., sealed package, unused power supply, etc.). When traces of Wi-Fi credentials are found, the *number* of findings was counted and saved. Additionally, the SSID and MAC addresses are extracted and saved. No further information is extracted.
For devices which were sold as broken we tried the ISP approach first and if that approach failed (e.g., the device was too damaged), we used the Chip-Off approach.
The correctness of the proposed methods was tested and confirmed with our own Amazon Echo Dot 3rd Gen, which was in use for over 1 year and was moved multiple times.

6.2 Results

The results of the case study are summarized in Table 2. We used the described analysis method to process all the devices we purchased. Our main goal was to preserve the user's privacy.
Not reset devices: A surprising number of devices (61%) were not reset by the previous owners. Due to the setup of our experiment, we had no possibility of asking the previous owners any questions. It is unclear how many of the identified Wi-Fi credentials are still valid. Also, we were not able to check if the devices are still bound to the Amazon account. After interpretation of the data, we see three potential explanations for the high number of devices which have not been reset: missing user knowledge, missing awareness and unclear reset instructions. From our own experience, the reset procedure of Echo Dot devices can be misleading. When tried by ourselves, we were deluded by the Wi-Fi setup mode, which is triggered after 10 seconds. After power-cycling the device had not been reset and we needed to press the button for a longer period.

Table 1: Keywords for detection of privacy related information

Category	Keywords	found?
Owner	Owner name, username, email address	✓
Wi-Fi networks	SSID, PSK, BSSID, SSIDs in proximity	✓
Paired devices	MAC addresses and names	✓
Geographical	Address, GPS coordinates	✘

Table 2: Device states of purchased devices in case study

Source		Total	Provisioned (S1),(S2)	Reset (S3),(S4)	New (S5)
eBay - working	US	38	24 63%	13	1
eBay - broken	US	13	13 100%	0	0
local flea markets	US	9	6 66%	2	1
eBay - working	DE	16	9 56%	5	2
eBay - working	NL	10	5 50%	4	1

Broken devices: All the devices sold as broken still contained all the data. However, this can be explained by the inability of the user to reset the device when it is not powering on. The majority of the devices had a broken power supply or bad power connector. Only in two cases the actual PCB was broken, of which both indicated water damage. We were not able to purchase a broken device with a partial defect. It might be possible that users would reset a device which has a broken speaker or some other minor defect. It is unclear if the previous owners were aware of the data which is stored on the device. However, in a situation with a broken device the choices are limited: The device can be destroyed, thrown away or sold.
Recovered traces in reset devices: Even for devices which have been reset by previous owners, we were still able to recover traces. This means that a customer cannot rely on the private data being deleted even if the customer follows the reset procedures. This is especially concerning, as the user has no way of confirming that all data has been erased.
Refurbished devices: We purchased six certified refurbished devices in total directly from Amazon. These devices were not always available and we assume that they are sold in batches. We noticed that newer refurbished devices have the new revision of the PCB, which uses NAND flash instead of eMMC. All of the devices appeared to be unused, however were missing the serial number sticker on the bottom side of the device. We did not find any traces on the flash memory of the devices, neither on the eMMC nor the NAND based revisions. Our assumption is that all of them are open box returns, which have been refurbished. Another explanation could be that all devices were opened and the PCB was replaced. This also explains the missing serial number sticker as the rubber base cover would have been replaced.
Ratio of reset devices over time: Over a period of 16 months we did not observe a significant change in the ratio between provisioned and reset devices. However, this observation is limited by the amount of devices which have been purchased on average per month and due to the fact that the previous owner might have stopped to use the device for a period of time before selling it.

7 MITIGATIONS

Our analysis has revealed the weaknesses of protections of privacy related data in smart speakers. The implementation of security features in devices like smart speakers or other IoT devices

is challenging, as this kind of devices has to meet particular user expectations. Features like the requirement for user passwords or PINs at power-up are likely unacceptable. While it might be more difficult to establish the same level of security as for smart phones, there are mitigations which increase the security and can likely be implemented easily. In our opinion the devices should keep the user data safe in all circumstances, independent of whether the device has been reset properly or not.

7.1 Usage of eMMC security features

As discussed in part 3.1, the eMMC standard supports secure deletion of information. However, none of the analyzed devices were using these features. While this only applies to eMMC based devices, the *Sanitize* or *SecureTRIM* commands can be used when the user issues a reset to the user data partition. If the eMMC flash adhered to the standard, the data would be erased from the physical memory. While this would meet the users expectation for a factory reset, it would not protect the data if the device was not reset. This mitigation can be implemented easily by the vendors and would not have negative impact for the users. A limitation is however, that this operation is time intensive and will be interrupted if the user disconnects the device from the power supply. Also the *Sanitize* is blocking, so the flash cannot be used while the data is being erased.

7.2 Wear-leveling aware erasing of data

A device that uses raw NAND flash has potentially more low level control over the physical data stored on the NAND. If the OS is aware of the physical position of the data, it can erase the data securely and make sure that no other copies exist. This feature needs to be implemented in the FTL, e.g., in case of Linux this would be the MTD subsystem. This mitigation is not applicable everywhere as in some cases the integrated controller in the SoC has the control over the NAND flash. This is mainly done for performance reasons, especially as ECC computation is complex. We encountered this issue with MediaTek based SoCs. In this case the OS would not have full access to the raw NAND flash. Also, this mitigation would not protect the user data if the device is not reset properly.

7.3 User data encryption

Our proposed mitigation is the encryption of the user data partition, which would solve multiple problems: First, a physical attack on a provisioned device cannot extract user data and credentials in a simple fashion anymore as a data dump would only contain encrypted information to which an attacker needs to retrieve the respective key first. This would protect the user credentials even if a reset was not possible nor performed. Second, most of the issues with wear-leveling are mitigated as all blocks are stored encrypted. The identification and reassembly of such blocks becomes very difficult. Also, the correct identification and reconstruction of traces of a deleted key is in our opinion not possible or very unlikely. We believe that this mitigation can be implemented in firmware updates and be enabled even for already provisioned devices. However, implementations might have different challenges and implications.

Performance impact: All analyzed devices had multiple cores. We believe that such devices have enough computation resources to perform file system encryption without performance issues. However, this might does not apply universally to all IoT devices and smart speakers. In cases of computation limits, the most critical user data could be stored encrypted, decrypted in the boot phase and then stored in memory.

Implementation: All of the analyzed devices use an Android based operating system. Android supports encryption of the user data partition [32]. The partition could be automatically unlocked at boot-up. In case of a factory reset, the encryption keys should be deleted and regenerated.

Key storage: In order to be able to unlock the partition, the keys need to be stored on the device. Depending on the storage, the key might be subject to wear-leveling. In case of a factory reset, the device needs to ensure that the keys gets deleted. A way to ensure this could be to use the previously mentioned ways to erase data. Due to the rather small size of the key such an operation would be very fast. Another problem is that the user data must remain safe even if the device has not been reset and the key has not been deleted. One way to achieve this would be to store the key where it cannot be accessed, e.g. the efuses in the SoC. While this would prevent a simple dump of the key from the flash memory, it would also mean that the key is static. If the key can be extracted in the future, it might be used to decrypt old data. The devices we encountered had SoCs which used `ARM TrustZone` for key storage and integrity protection. Since the previously mentioned approach suffers from major drawbacks, we propose to use `ARM TrustZone` to store the encryption key for the user data partition and to bind it to the SoC instead. This would ensure that the SoC needs to be present in order to unlock the user data partition. Additionally, we recommend that the `RPMB` is used.

Limitations: We encountered vendors which encrypted the user data partition with `LUKS` and used `ARM TrustZone` to store the key. However, we were still able to access the user data by abusing vulnerabilities in the firmware. While the data is protected at rest, it can be accessed by the OS when the device is powered on. We noticed that some devices were also subject to downgrading attacks, in particular as the `RPMB` was not used.

8 COMPARISON TO OTHER DEVICES AND VENDORS

In order to gain an understanding of whether the found issues were exclusively related to Amazon Echo Dots, we analyzed other Amazon products and comparable devices. For this analysis we selected devices depending on availability and based on a similar price range. We purchased a small number of used devices, therefore the results are not representative. We used the same method as for the Amazon Echo Dots. In our analysis we found very similar issues.

Amazon Echo Show 5: The Echo Show 5 (H23K37) was released in Q2 2019 by Amazon. In comparison to the Echo Dot it has a 5.5 inch touch-screen display, which can be used for user interaction. Such interactions can be using the integrated browser, retrieving recipes or watching videos in Amazon Prime Video. A front facing camera allows video calls. All the other features of the Echo Dot, like the Alexa virtual assistant, are supported. The devices we analyzed had the FCC ID 2ARIV-2425. The device uses the MediaTek MT8163V, the same SoC as the Echo Dot 2nd Gen. It supports Wi-Fi and

Bluetooth. The system has a separate 1 GByte DDR3 RAM and an 8 GByte eMMC flash memory.

Amazon uses again FireOS 6 (based on Android 7.1.2 Nougat) for this device. However, the user can not install apps from the Amazon Appstore or Google Play. Instead, only Alexa skills can be installed. The integrity of the OS is protected by `dm-verity` but the partition containing user data is not encrypted. The user can configure the Wi-Fi or trigger a factory-reset using the touch screen. We noticed that a change of the Wi-Fi configuration requires an additional confirmation by the user by requiring to login with the Amazon account credentials. In our experiment we were able to extract the same amount of data as for the Amazon Echo Dot.

Due to the possibility of user interaction, we were also able to extract user data from the browser. We purchased two used and two broken devices. Both used devices were reset, but we were able to find traces of the previous owner's data. The broken devices still contained all user data.

Google Home Mini: The Google Home Mini (H0A) is a direct competitor to the Amazon Echo Dots and is a smart speaker that uses the Google assistant. We analyzed the model with the FCC ID A4RH0A, which was released in late 2017. It is based on the Marvell 88DE3006 SoC, which is an ARM Cortex-A7 Dual-Core. The device contains 512 MByte of DDR3 RAM and 256 MB of NAND flash. Compared to the Amazon Echo Dots the device offers similar connectivity, such as Wi-Fi and Bluetooth, which is provided by the Marvell Avastar 88W8887 IC.

The Google Home Mini is based on Android. Similar to the Amazon Echo Dot it also uses integrity protection of the OS by employing `dm-verity`. We found that the configuration data, like Wi-Fi credentials and tokens, are stored unencrypted on the device. The device has a reset button on the bottom side of the case. However, the button is hidden under the rubber floor plate and is not directly obvious for users. We purchased five used devices and found that all of the devices were not reset by the previous owners.

Amazon FireTV 3rd Gen: Amazon FireTV (AFTN) is a line of digital media players which offer access to Amazon services, such as Amazon Prime Video, on a TV. Alexa is integrated as a virtual assistant and can be controlled by voice using the FireTV remote. The device which we analyzed was the FireTV 3rd Gen, which has the FCC ID 2ALBL-1731. It was released in 2017 and is based on the Amlogic S905Z (Octa-core ARM Cortex-A53-based SoC). It contains 2 GByte dedicated DDR3 RAM and 8 GByte eMMC flash memory.

The operating system in use is FireOS 6 which is based on Android 7.1.2 Nougat. The user can install apps from the Amazon Appstore. For example, apps like Netflix or Firefox can be installed and used. Similar to the previously discussed Amazon Echo devices, the OS is protected using using `dm-verity`, and the user data partition is not encrypted. This enabled us to extract private data from a device which was not reset once again. Due to the possibility of installing apps, we were also able to extract browser histories, saved credentials or access tokens to third-party apps, like Netflix. We purchased two used, working devices and one broken device. One of the working devices was reset, but we were able to detect traces of private data. The other two devices still contained all private information.

Xiaomi Smart AI speaker with display: The Xiaomi Smart AI speaker (LX04) is very similar to the Amazon Echo Show 5 and was released in 2019. It has a 4 inch touch-screen display. The device is geared towards the Mainland China market and not officially marketed to different regions. It supports Xiaomi's own virtual assistant, "XiaoAI" and can also be used to control any Mijia smart home devices. The SoC is a MediaTek MT8163. The device has 1 GByte of DDR3 RAM and 8 GByte of eMMC flash.

The operating system is based on Android 8.1.0 Oreo. The user cannot install custom applications. However, the system contains a browser and E-Mail client. The integrity of the OS is protected by `dm-verity`. In contrast to the previously described devices, the device contains encrypted partitions. While it is not possible to access the user data via the Chip-Off approach, we were still able to use a vulnerability in the firmware to downgrade the firmware, and then extract the keys and information. We did not purchase used devices of this model.

9 RELATED WORK

Reverse engineering Amazon Echo devices has been a topic of interest since their first generations. In 2016 Clinton et al. [18] did a hardware analysis of the *Linux-based* Amazon Echo (1st Gen). It had few security features. Hyde and Moran [25] forensically examined in 2017 the Amazon Echo Dot (1st and 2nd Gen), Amazon Echo (1st Gen) and the Amazon app. They used UART and ISP to access the devices. They retrieved Wi-Fi information from the Amazon Echo via ISP and other information from Amazon apps. Li et al. [27] proposed a forensic analysis model and used extracted artifacts from the Amazon Echo and the Alexa app as a use case to demonstrate their model.

Chung et al. [17] explored in 2017 the communication of the Amazon app and the cloud. Their focus was on the artifacts and databases of the Amazon app. Early devices had limited protections providing easy access for reverse engineering and data. Since then Amazon increased the security of the Echo Dot devices. More recently, Pawlaszczyk et al. [31] examined the Amazon Echo Dot Gen3 and the Alexa app. The authors used a destructive Chip-off method to remove the eMCP IC and specialized forensics equipment to access the eMMC flash.

All previously mentioned work only uses provisioned devices. To our knowledge our work is the first on several dimensions, (1) demonstrates the retrieval of private users data from even factory reset devices (exploiting wear-leveling), (2) demonstrates the ability to hijack a previous user Amazon account, (3) is non-destructive and systematic, and (4) applied to a relatively large set of user devices revealing, poor users security practices, and usable security limitations. We responsibly disclosed our results to Amazon. The company reproduced and confirmed our findings. This applies also for newer models. Currently Amazon is still working on integrating a mitigation into the devices.

ACKNOWLEDGMENTS

This work was partially supported by grants NCAE-Cyber Research Program, and NSF/DGE-1661532. We would also like to thank Cameron Kennedy and Erik Uhlmann for their valuable input, advice and for the interesting discussions.

REFERENCES

[1] 2011. NAND Flash Table. (Jul 2011). http://www.linux-mtd.infradead.org/nand-data/nanddata.html [Online; accessed 7. July. 2020].

[2] 2019. Juniper Estimates 3.25 Billion Voice Assistants Are in Use Today, Google Has About 30% of Them - Voicebot.ai. (Feb 2019). shorturl.at/uHNOP [Online; accessed 4. Aug. 2020].

[3] 2020. 2017 Fire HD 10: Unbricking from anti-rollback. (Aug 2020). https://forum.xda-developers.com/hd8-hd10/development/2017-fire-hd-10-unbricking-anti-rollback-t3896616 [Online; accessed 31. Aug. 2020].

[4] 2020. Amazon Echo Dot : CVE security vulnerabilities, versions and detailed reports. (Aug 2020). https://www.cvedetails.com/product/46475/Amazon-Echo-Dot.html?vendor_id=12126 [Online; accessed 4. Aug. 2020].

[5] 2020. Fire HD 8 (2018 ONLY) unbrick, downgrade, unlock & root. (Aug 2020). https://forum.xda-developers.com/hd8-hd10/orig-development/fire-hd-8-2018-downgrade-unlock-root-t3894256 [Online; accessed 31. Aug. 2020].

[6] 2020. Pryon. https://www.pryon.com/. (Oct 2020). [Online; accessed 1. Oct. 2020].

[7] 2020. Sleuthkit Autopsy features. (Aug 2020). http://www.sleuthkit.org/autopsy/features.php [Online; accessed 1. Aug. 2020].

[8] 2020. [UNLOCK][ROOT][TWRP][UNBRICK] Fire HD 8 2017 (douglas). (Aug 2020). https://forum.xda-developers.com/hd8-hd10/orig-development/unlock-fire-hd-8-2017-douglas-t3962846 [Online; accessed 31. Aug. 2020].

[9] Amazon. 2020. Amazon.com Help: Amazon Frustration-Free Setup Frequently Asked Questions. (2020). https://www.amazon.com/gp/help/customer/display.html?nodeId=GMPKVYDBR223TRPY [Online; accessed 15. Mar 2020].

[10] Amazon. 2020. Fire OS 6 for Fire Tablets | Fire Tablets. (2020). https://developer.amazon.com/docs/fire-tablets/fire-os-6.html [Online; accessed 02. May. 2020].

[11] Amazon. 2020. Overview of Amazon Device Messaging | Amazon Device Messaging. (2020). https://developer.amazon.com/docs/adm/overview.html [Online; accessed 02. May. 2020].

[12] Amazon. 2020. Understanding Frustration-Free Setup | Frustration-Free Setup. (2020). https://developer.amazon.com/docs/frustration-free-setup/understanding-ffs.html [Online; accessed 15. Mar 2020].

[13] JEDEC Solid State Technology Association. 2010. Embedded MultiMediaCard (eMMC) eMMC/Card Product Standard, High Capacity, including Reliable Write, Boot, Sleep Modes, Dual Data Rate, Multiple Partitions Supports, Security Enhancement, Background Operation and High Priority Interrupt (MMCA, 4.41). *JESD84-A441* (2010).

[14] JEDEC Solid State Technology Association. 2012. Embedded Multimedia Card (eMMC), Electrical Standard 4.51. *JESD84-B451* (2012).

[15] Marcel Breeuwsma, Martien De Jongh, Coert Klaver, Ronald Van Der Knijff, and Mark Roeloffs. 2007. Forensic data recovery from flash memory. *Small Scale Digital Device Forensics Journal* 1, 1 (2007), 1–17.

[16] Byungjo Kim, Dong Hyun Kang, Changwoo Min, and Young Ik Eom. 2014. Understanding implications of trim, discard, and background command for eMMC storage device. In *2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE)*. 709–710.

[17] Hyunji Chung, Jungheum Park, and Sangjin Lee. 2017. Digital forensic approaches for Amazon Alexa ecosystem. *Digital Investigation* 22 (2017), S15 – S25. https://doi.org/10.1016/j.diin.2017.06.010

[18] Ike Clinton, Lance Cook, and Shankar Banik. 2016. A survey of various methods for analyzing the amazon echo. *The Citadel, The Military College of South Carolina* (2016).

[19] Jim Cooke. 2007. The inconvenient truths of NAND flash memory. *Flash Memory Summit* 3, 3 (2007), 3–1.

[20] Daniel J. Dubois, Roman Kolcun, Anna Maria Mandalari, Muhammad Talha Paracha, David Choffnes, and Hamed Haddadi. 01 Oct. 2020. When Speakers Are All Ears: Characterizing Misactivations of IoT Smart Speakers. *Proceedings on Privacy Enhancing Technologies* 2020, 4 (01 Oct. 2020), 255 – 276.

[21] Amir Etemadieh, CJ Heres, and Khoa Hoan. 2017. Hacking Hardware With A $10 SD Card Reader. *Blackhat US* (2017).

[22] Google. 2020. Overview | Geolocation API | Google Developers. (2020). https://developers.google.com/maps/documentation/geolocation/overview [Online; accessed 02. May. 2020].

[23] Open NAND Flash Interface Working Group. 2020. Open NAND Flash Interface Specification. *ONFI 4.2* (2020).

[24] Peter Gutmann. 2001. Data Remanence in Semiconductor Devices.. In *USENIX Security Symposium*. 39–54.

[25] Jessica Hyde and Brian Moran. 2017. Alexa, are you Skynet. *SANS Digital Forensics and Incident Response Summit* (2017).

[26] Magnus Larsson. 2015. Sanitization of embedded network devices: Investigation of vendor's factory reset procedure. (2015).

[27] S. Li, K. R. Choo, Q. Sun, W. J. Buchanan, and J. Cao. 2019. IoT Forensics: Amazon Echo as a Use Case. *IEEE Internet of Things Journal* 6, 4 (Aug 2019), 6487–6497. https://doi.org/10.1109/JIOT.2019.2906946

[28] MediaTek. 2020. MediaTek 8516 Datasheet. (2020). https://www.mediatek.com/products/tablets/mt8516 [Online; accessed 02. May. 2020].

[29] Alastair Nisbet, Scott Lawrence, and Matthew Ruff. 2013. A forensic analysis and comparison of solid state drive data retention with trim enabled file systems. (2013).

[30] Jeong Wook Oh. 2014. Reverse engineering flash memory for fun and benefit. *Blackhat US* (2014).

[31] D Pawlaszczyk, J Friese, and C Hummert. 2019. "Alexa, tell me..."-A forensic examination of the Amazon Echo Dot 3 rd Generation. (2019).

[32] Android Open Source Project. 2020. Full-Disk Encryption | Android Open Source Project. (2020). https://source.android.com/security/encryption/full-disk [Online; accessed 02. May. 2020].

[33] Android Open Source Project. 2020. Verified Boot | Android Open Source Project. (2020). https://source.android.com/security/verifiedboot [Online; accessed 02. May. 2020].

[34] Rusolut. 2018. eMMC CHIPS. DATA RECOVERY BEYOND CONTROLLER. (2018). https://rusolut.com/wp-content/uploads/2018/10/eMMCvsNAND.pdf BelkaDay - Belkasoft Digital Forensic Conference 2018, Prague, Czech Republic.

[35] Lea Schönherr, Maximilian Golla, Thorsten Eisenhofer, Jan Wiele, Dorothea Kolossa, and Thorsten Holz. 2020. Unacceptable, where is my privacy? Exploring Accidental Triggers of Smart Speakers. (2020). arXiv:cs.CR/2008.00508

[36] Arie Tal. 2002. Two flash technologies compared: NOR vs NAND. *White Paper of M-Systems* (2002).

[37] Jan Peter van Zandwijk. 2015. A mathematical approach to NAND flash-memory descrambling and decoding. *Digital Investigation* 12 (2015), 41 – 52. https://doi.org/10.1016/j.diin.2015.01.003

[38] Michael Wei, Laura M. Grupp, Frederick E. Spada, and Steven Swanson. 2011. Reliably Erasing Data from Flash-Based Solid State Drives. In *Proceedings of the 9th USENIX Conference on File and Stroage Technologies (FAST'11)*. USENIX Association, USA, 8.

[39] Li Zhang, Yu an Tan, and Qi kun Zhang. 2012. Identification of NAND flash ECC algorithms in mobile devices. *Digital Investigation* 9, 1 (2012), 34 – 48. https://doi.org/10.1016/j.diin.2012.04.001