Z-INF Report

University of Bergen

Department of Informatics

Digital Forensic Acquisition of mobile phones

Author:
Oskar Krystian Michalski

Supervisor:
Øyvind Ytrehus

September 6, 2024

Abstract

Gunnar Alendal demonstrates the dilemma that Law Enforcement (LE) encounters in the context of acquiring forensic information in order to assist crime investigations. The information accumulated in our devices can be of great relevance when it comes to mitigating potential terrorist attacks and similar. Therefore bypassing security implementations successfully and acquiring data stored on our devices is not only a relevant topic for outlaws but for LE as well. G.Alendal showcases the dilemma of LE needing to take the role as a "hacker" or "attacker" to achieve (in my own words) "greater good". Throughout his five papers he presents important research around Digital Forensic Acquisition (DFA).

Alendal introduces the outlines which the LE must consider when performing a DFA. G.Alendal discusses some of the challenges in acquiring un-encrypted user data with respect to modern day security implementations. Furthermore he showcases how performing analysis of the system at target can yield powerful information about the system vulnerabilities. The approach he employees is very similar to the starting point for DFA, where no information is known about the target in advance. With this approach he demonstrates how researching the attack surface can yield many different attack vectors that was not even considered during the implementation of security on target systems. In addition to that he presents different stages of gathering information where one stage can yield detailed attack plan against the targeted system and another, yield some alternative approaches for the attack saving resources and time.

List of Figures

Contents

Chapter 1

Introduction

1.1 Motivation

Technology has developed impressively fast these past few years, and suddenly our mobile device has become our tool for socializing, organizing, and planning. Nowadays it might just be that our devices might know more about us than we know about ourselves. The information about what we do, who we are and who we are with, is now gathered on one device making a "picture" of our entity physically acquirable. This creates dangers and concerns around our private life that must be handled. The growing danger of private life being exposed to unwanted eyes, has put pressure on vendors that distribute such devices. Their response is to develop counter measures on our "precious" devices in order to keep our privacy sacred and safe from unwanted outsiders. These "countermeasures" are referenced to as security implementations which become progressively more and more intricate and sophisticated. However information about us as a person is not only a target for outlaws, but also for the LE themselves.

In order to handle the diversity of possible attack scenarios on our private life, stored on our mobile device, vendors have attempted to develop a more complex and intricate security implementations to prevent vulnerabilities from emerging. It is said to be in our human nature to make mistakes. But mistakes in security implementations are still unacceptable and can have great consequences upon mobile device consumers if exploited by unwanted actors.

Consequently vendors of devices have put a lot of effort in developing methods that narrow down the error area and prevent code designers from writing vulnerable code. Employing different security methods and making them interact with each other, makes the security development process even more sophisticated. In consequence making vulnerabilities to appear in more obscure circumstances that are not as easy to detect as before.

In my opinion; if vendors would be more willing to cooperate with LE, the area of digital forensic acquisition would not be as greatly developed as it currently is. The need of LE to outrun the security implementations that are in constant development, gives LE a constant motivation to stay ahead in the development race. In addition, that could lead to more structure in the DFA, considering the vague restrains that LE must follow when performing DFA. Cooperation between LE and the vendors of mobile device would provide a more strict frame for the LE to operate within. That is because gathered information could be exchanged in a controlled manner

between the two parties. Meanwhile the way of discovering security vulnerabilities and misusing them in order to gain unrestricted access to the device, leaves much more space for the LE to gather and keep information that may serve other purposes then assumed.

1.2 Objective

The primary goal of this thesis is to emphasize some of the essential matters addressed by Gunnar Alendal in his research "Digital Forensic Acquisition of mobile phones in the Era of Mandatory Security" [1]. Where some of the most essential challenges presented by Gunnar Alendal, are showcasing the gap between intended and achieved security in systems. Alendal stresses the importance of LE pursuing further development of new Digital Forensic methods. Meanwhile this thesis will also attempt to address the third perspective on the topic presented by G.Alendal. Thus my personal insights and interpretations of the topics will be included.

1.3 Organization

The is organized as follows. Chapter 2 will introduce key concepts required for further discussion. Meanwhile the rest of the Chapters will present the different topics presented by Gunnar Alendal. These Chapters will be divided into a Introduction, a Topic, and a Conclusion -section. Where the Introduction section will introduce a very generalised overview of the topic as well as its purpose. While the Topic section will be to some extent a more in depth presentation of Alendals work including my personal assumptions and interpretations of presented research. At last the Conclusion section in each Chapter will provide conclusions considering both Alendals work as well as my own insights.

1.4 Outcome

This thesis seeks to showcase the challenges that DFA needs to counter, how performing Digital Forensic Acquisition be somewhat controversial when considered as an attack against user privacy on modern day mobile devices. Throughout this Paper I will together with Alendals work, cast some more light on the difference between intended security and achieved security, by emphasizing the danger this gap brings to modern day cryptography.

Chapter 2

Required Background

2.1 The general outline of DFA

During a criminal investigation, personal information of a potential criminal may make the difference between life and death. Therefore, the ability of performing an successful Digital Forensic Acquisition in an efficient way is of great importance. However as Alendal declared, the fluctuating nature of security vulnerabilities makes it difficult for the process to be effectively planned.Efficiency and success depends on time and resources as Gunnar A. explains. Therefore instead of providing a specific step-by-step guide he rather offers a general scheme of the whole process performed during DFA, revealing a relevant pattern of the Law Enforcement to follow.

The first step is in my opinion the most relevant step and requires a more in depth introduction of the sub tasks that are considered. The process starts with seizing the target device and discovering the different approach alternatives (or better described as attack vectors). Alendal highlights how essential the elements considered in this step are. Factors like the state of the seized device which can be either powered on or off, or choosing which memory to target, as well as what information to look for, are of great importance. All of these aspects must be considered during the preparation for the rest of the process. The complexity of the rest of the process can depend highly on whether volatile or long-term memory is found relevant for the investigation, or if the owner of the device have unlocked the device before it was seized. All these can determine whether the DFA will be successful or not.

The rest of the process is not as relevant as the first part therefore it is explained in a more compact and general way. The rest of the process is implementing the chosen approach and obtaining relevant and human readable information from the source, in order to further analyze and process the gathered information. Finally for the relevant data to be presented to other sections of the LE.

2.2 Ethics and Legal matters to consider

Today, the purpose of DFA is considered as gaining unauthorized access and extract personal data from a suspect's device. These methods would be controversial if employed by anyone other than law enforcement. During discussion in a paper G.Alendal describes the requirement for an legal Digital Forensic Acquisition to be described as "forensically sound". G.Alendal also explains how the lack of cooperation from vendors forces law enforcement to adopt more offensive approaches to obtain crucial investigation material from target system. This often involves exploiting undisclosed vulnerabilities that the vendors themselves are unaware of. The fact that law enforcement possesses knowledge of these vulnerabilities and chooses not to share them with vendors raises ethical concerns. From the law enforcement perspective, keeping this knowledge as a secret is the key to using this vulnerability in bypassing otherwise unavoidable security measures. However, not disclosing these vulnerabilities to vendors means they remain unpatched, potentially exposing the system to exploitation by malicious actors. But whether this area is a responsibility for the LE or not is outside of this papers scope therefore it will not be discussed any further.

2.3 Introduction to papers

G. Alendal notes that the inspiration for this research topic originated from a project involving self-encrypting hard drives. During their study of "Western Digital HW encryption drives," they discovered several errors in the implementation of the algorithms and security measures, which completely compromised the system's security layer.

The results of the studies motivated Alendal to address the original question. In what way can security vulnerability discovery and exploitation contribute to the improvement of digital forensic acquisition? This led to three additional main research questions that Alendal attempts to answer in the following papers.

The research questions shown in Figure 2.1, are as follows. He aims to address the first research question "How can modern security measures be bypassed by exploiting security vulnerabilities?" in "Forensic Acquisition - Analysis and circumvention of Samsung secure boot enforced common criteria mode" (first paper) and "Chip Chop - smashing the mobile phone secure chip for fun and digital forensics" (fifth paper). In the introduction he also refers to his sixth paper "Breaking Android Security by Abusing Implicit HW trust" which is not included in this report. The second research question, "Can we identify potential feature attack surfaces useful for digital forensic acquisition?" is addressed in "Exploiting Vendor-Defined Messages in the USB Power Delivery Protocol" (second paper), "Leveraging the USB Power Delivery Implementations for Digital Forensic Acquisition." (third paper) and "Digital Forensic Acquisition Kill Chain" (fourth paper). The third and last research question "How can digital forensic acquisition draw benefit from published security vulnerabilities?" is also discussed in the fifth paper mentioned previously.

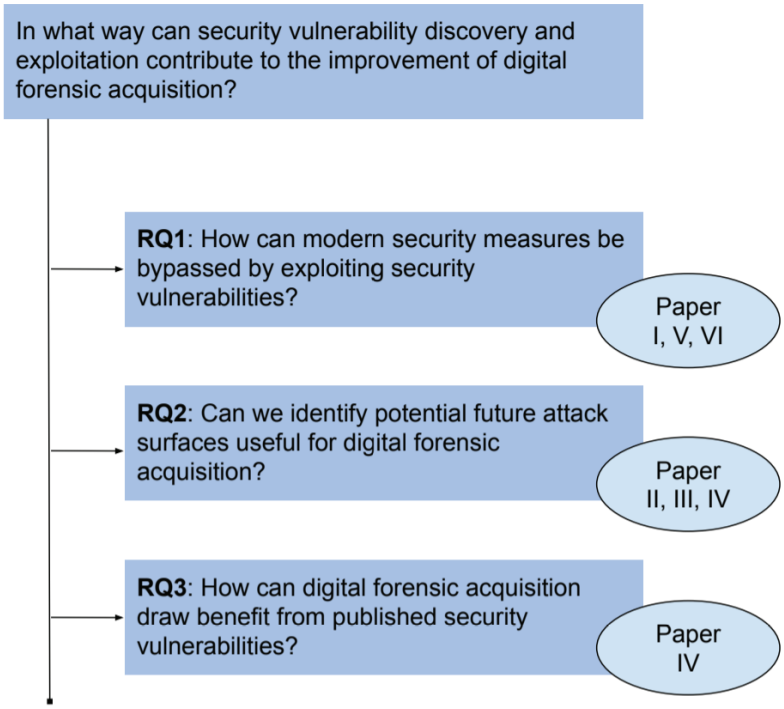

Figure 2.1: Papers addressing research questions respectively.

Chapter 3

Paper I: Forensic Acquisition - Analysis and circumvention of Samsung secure boot enforced common criteria mode.

3.1 Introduction

3.1.1 The Challenge and motivation for the chosen approach.

Alendal presents how methods of acquiring information that were sufficient in earlier years has become invalid, which Alendal stresses is a great argument stating that DFA must keep improving their capabilities in DFA in order to "keep up" in this ever changing world of security. Modern day security standards has now increased to a degree where simply being in possession of a target device is not sufficient to gain investigation materials from that device.

The main attack vector employed in Paper I requires physical access to the device, and is based on interacting with the security mechanisms in order to find any potential misconfiguration that causes a vulnerability on the running device. I would describe Alendals approach as being similar to what I would call "poking the system with a stick".

Due to vendors desire to reserve their systems from such vulnerabilities, finding an appropriate attack vector means dealing with multiple security layers implemented to defend the system from such errors. In the introduction Alendal explains that a common approach is to attempt to install a code that creates a runtime vulnerability when executed on target system. The countermeasures used by vendors to protect their systems from such attack is separating code into sections. Such that different code that is responsible for different tasks runs independently from each other. (A code that boots up the device should be separated from the code that is run when the device is already on and so on up to a fully operating system). This method is called Secure Boot which is supposed to protect the system from forged boot code and ensure a safe start up that executes only the original code written by the vendors of the device. In order to preserve this secure boot, authentication of code is used.

3.1.2 The approach

Alendal highlights how despite many attempts, providing a security of the system that considers the interaction of third party devices is a difficult task, and therefore is not always achieved. Managing third parties communication and access rights to the device is one of challenges in security. MDM (Mobile Device Management) is an attempted counter step that efforts to address the challenge of handling the communication between the device and other "extension devices".

In the papers, Alendal discusses how Samsung utilizes Common Criteria (CC) mode, a security feature designed to enhance device protection against unauthorized access from extension devices. Enabling this mode blocks access to the device's firmware update mechanisms, which G. Alendal identifies as a primary method for law enforcement to acquire data. Consequently, enabling Common Criteria mode introduces an additional challenge that Digital Forensic Acquisition (DFA) must address. Therefor the main approach of Paper I by G.Alendal is to reverse engineering of CC mode and revealing its security vulnerabilities that can be utilized to circumvent the CC mode for further forensic acquisition.

Alendals stresses how disabling CC mode will give the opportunity of using other existing methods and third party device to increase the attack surface and increase the possibility of discovering further vulnerabilities on the device. Hence the main goal of Alendals approach is to disable the CC mode. This concludes the motivation and main goal of Paper I.

3.1.3 CC mode

Alendal explains how CC mode only allows Firmware Over the Air (FOTA) to be installed on the device, in order to protect the device from attackers trying to install unauthorized firmware with physical access to the device. Which as mentioned earlier is the main method of data gathering practiced by the DFA. By my understanding, in order to be able to physically “push” firmware updates to the device Alendal claims that one must use ODIN mode that is implemented on Samsung devices, which allows the attacker to install firmware from an USB connection. CC mode blocks both the ODIN mode and any attempt to boot an unofficial boot image already stored on the system. This defines the scope of interest concerning CC mode, and other information about Common Criteria that is not relevant for ODIN mode will not be further discussed.

Tightening the scope even more, G.Alendal describes how different versions of Samsung devices can make use of different SoC (System on-a-chip). Where the devices that will be of interest in this Paper is those devices that are based on the “Exynos SoC”, these are devices like Galaxy s6 and Galaxy s7. Where the boot loader called SBOOT is responsible for the running of the native starting code. Alendal stresses how documentation and source code of SBOOT is not publicly available since its Samsung’s proprietary code. Which creates further complications that must be solved during this paper.

The goal of analysis is to understand how the SBOOT knows that CC mode is enabled, and further how it is used to limit access to certain features of the device. The analysis tries to comprehend how the CC mode configuration is stored and how SBOOT interacts with the configuration, potentially revealing how it can influence the state of the configuration (influencing the configuration to be turned

on or off). Ultimately leading to SBOOT being responsible for not only enforcing the configuration but also changing its settings.

Eventually Alendals research team managed to locate a vulnerability in the SBOOT of mentioned Samsung devices and develop an exploit made entirely out of python code with the use of a keystone assembler framework. Keystone assemblers are mainly used for ensuring stability, functionality and integrity of the overall system that they are working on, which is why I assume that Alendals team used this assembler to run the python coded exploit on the target device. He explains that the main goal of the exploit was to disable CC mode and gain access to the ODIN mode in order to permit the installation of unofficial boot images on the device ultimately granting an unlimited access to the device which allows downloading unencrypted user data from the device.

3.2 Topic

3.2.1 Locating the appropriate attack vector

The boot loader on the target Samsung device is directly communicating with CC mode and is able to disable and enable the ODIN mode in order to prevent uploading unauthorized firmware for sidestepping the centralized control. Which makes the boot loader the main target that makes it possible to disable the CC mode.

In order to understand this connection deeper G.Alendal explains the depths of Samsung secure boot. Boot loader on Samsung is designed to exclusively run secure and signed code from the start up to the fully functioning android system. The codes that are supposed to run at start are signed by an SSBK. Some public parts of the key are stored in "BootROM" (Boot loaders Read-Only Memory) and every code that is stored in there is checked before execution whether it contains the required Samsung signature.

The process can be described in the following way. The first boot loader is initialized and executes its task before checking if the next boot loader (BL2) is a valid boot loader, and if it is then the current boot loader runs the next one and finishes its job. Ultimately separating each code execution from each other, this method is implemented to minimize the impact of an security breach, and enforce the principle of least privilege which is fundamental for implementing robust and secure systems.

Enabling CC mode on the device leads to several system changes. Specifically, CC mode enforces FIPS-validated cryptography, disables USB connectivity in recovery mode, and permits only Firmware Over the Air (FOTA) updates. This mode is automatically activated by the Samsung Common Criteria APK.

The scope of the analysis is narrowed down to the on/off state of CC mode, focusing on its interaction with SBOOT. The analysis examines how enabling CC mode affects SBOOT and how SBOOT enforces ODIN blocking on devices.

There are two methods for updating the firmware: through FOTA or directly via ODIN mode. Protecting ODIN mode from unauthorized access is crucial because disabling CC mode is possible through ODIN access. Conversely, installing firmware updates via FOTA is designed to be secure against external threats, as it involves online communication between the device and Samsung firmware servers.

3.2.2 Targeting the PARAM partition

To understand the impact of CC mode on SBOOT, where this setting is stored, and how to exploit it, G. Alendal employed static reverse engineering of SBOOT and dynamic reverse engineering of an SBOOT exploit. The analysis revealed that the CC mode setting is stored in flash memory, specifically in the PARAM data partition. Which is a storage partition that is not effected by the reboot.

After deeper analysis the researchers managed to understand how referencing to this PARAM partition works, which was crucial information in order to be able to manipulate the data stored on the partition. The PARAM partition might appear harmless due to its primary contents, such as JPG files with the Samsung logo. However, it also contains critical files like MDM settings, the current system status ("Samsung Official" or "Custom"), and CC mode settings. Accessing this information involves referencing the PARAM partition block size, which is 512 bytes in this case, and counting backwards to the correct pointer. Within these 512 bytes, there is a 64-byte block that includes the encrypted and signed CC settings.

When the system loads these settings, the data is read and decrypted using the following function:

```
S_CC_decrypt(the 64-byte encrypted cc setting, 0-initialized output buffer)
```

Where the 0-initialized output buffer decides where the output of the function will be stored. An important observation is that this function does not require a key parameter, suggesting that the key is likely embedded within the function or retrieved by it. This function utilizes a whitebox AES cipher, meaning the AES key is not disclosed during either encryption or decryption. Consequently, the "S CC decrypt()" function can be used as a decryption oracle. However, the fact that SBOOT does not call "S CC encrypt" led researchers to conclude that the configuration of CC mode is written by the Android environment and not by SBOOT. SBOOT's role is limited to querying the configuration execution. This conclusion is supported by the discovery of an "encrypt/decrypt oracle" in a native Android library, specifically in the "/system/lib64/libSecurityManagerNative.so" file.

Gunnar Alendal implies that by using encryption/decryption oracles, one might be able to read and write the CC configuration file from both SBOOT and Android environments if one controls the execution and has full access to the PARAM partition.

The reasoning behind this approach is that, once the CC configuration file is located, it can be decrypted using the decryption oracle, modified, and then re-encrypted using the encryption oracle before being placed back into its location on the PARAM partition. However, a critical comment on this approach is that the integrity of the file and its status as a secure file could be compromised. Samsung's integrity protocols implemented to maintain security in "secure boot" should detect such simple changes in the code. This detection can be accomplished by logging activity related to this file or using more advanced integrity methods that are not implemented by the encryption and decryption algorithm.

After decrypting the CC mode file found in the PARAM partition, its contents can be read. The structure of the file is as follows: "timg | N0CC/FF0C," where the N0CC byte indicates an enabled state and the FF0C byte indicates a disabled state.

SBOOT executes the code for CC mode and MDM settings very early in the boot process, setting a global flag to "active" if they are enabled. An important aspect to note is that MDM settings and CC mode can interplay, influencing each other's functionality.

G. Alendal underscores the importance of the precise timing for checking the CC flag during boot-up. The device continually performs various security checks, intertwining them with error checks, which can create potential vulnerabilities. In some cases, certain security features might not be checked, depending on the types of errors encountered.

The function "S boot enter download mode" is called multiple times during boot-up. This function checks whether it is permissible to enter ODIN mode. If the device is allowed to enter ODIN mode, it calls the function "S USB mode enter" with the parameter "0", indicating ODIN mode, which then switches the device to ODIN mode.

As shown in figure 3.1, during boot-up, when "S boot enter download mode" is called, a reason is passed as a parameter, and the rest of the code runs afterward. At the end of the function, there is a check to see if "S CC MODE isSet" is set. This check occurs after the first if-else block runs. If the reason equals 6, the else block leads to calling "S USB mode enter(0)" without performing the "S CC MODE check".

```
S_boot_enter_download_mode(int reason){
  if (reason == 1) {
    S_draw_image("warning_L.jpg");
      if (S_user_cancel()){
        S_reboot_device();
      }
      else {
        S_draw_image("download_L.jpg");
      }
  }
  else {
    if (reason == 6) {
      S_draw_image("download_error_L.jpg");
      S_USB_mode_enter(0);
    }
    S_draw_image("download_L.jpg");
    if (reason == 3) {
      // SUD mode ..
    }
    if (reason != 4) {
      goto error();
    }
  }
  //...
  if (S_CC_MODE_isSet()) {
    S_screen_print(
        "DOWNLOAD_IS_BLOCKED_BY_CC_MODE");
    S_sleep(1000);
    S_power_off_device();
  }
  S_USB_mode_enter(0);
}
```

Figure 3.1: High level code interpretation developed by Alendals team through reverse engineering

During startup, there is a set of environment parameters that influence the boot process. One relevant mode is "REBOOT MODE", which signals to SBOOT the type of boot to perform, based on the parameter passed: (0) for normal startup, (1) for ODIN mode, (2) for upload mode, and (4) for recovery mode.

Although there are many options to influence the startup, this article focuses on one particular option related to CC mode, specifically the "DN ERROR" function.

The process by which "DN ERROR" leads to ODIN mode is as follows: Starting with "S s5p check download", which returns 0 only if it should not switch to ODIN mode. This path would lead to "S boot enter download mode" returning a zero integer (indicating no ODIN mode). This should then proceed to calling "S enc get", which retrieves the integer value of "DN ERROR". If this variable is set, there is a call to "S boot enter download mode" with the parameter 6.

This method allows the device to enter ODIN mode even when CC mode is enabled.

3.2.3 Brief MDM setting discussion

The MDM setting performs a check if it should prevent all firmware upgrades of any other partition than the SBOOT itself. The MDM setting differs from CC mode by performing an additional "enabled/disabled check" when the device is in ODIN mode. The MDM setting itself is an unencrypted setting that is stored in the PARAM Partition, where any overwriting of the variable leads to disabling MDM setting which will not limit access of the ODIN mode anymore. However, since the MDM check is performed also when in ODIN mode, it must be taken into consideration during the procedure.

3.2.4 Unauthorized Disabling of the CC mode

Alendal presents three different approaches to disable the CC mode without proper authorization.

Modifying the PARAM Partition By obtaining Read/Write access to the PARAM Partition and utilizing the WAES encryption/decryption oracle, one can locate the CC mode setting, decrypt it, modify it, re-encrypt it, and store it back in the PARAM Partition. This process will effectively set the CC mode to the desired option, such as “disabled.” This can be achieved by overwriting the targeted data with zeroes, either through physical access to the underlying flash storage or by using ODIN mode. However if this is done through ODIN mode, access will be blocked if the MDM setting is enabled. Luckily, physical access to the flash storage allows one to overwrite the MDM setting, thereby enabling the subsequent overwriting of the CC mode.

SBOOT exploitation Another way used to disable the CC mode by Alendal involves using the "home-developed" exploit created by Alendals team, as a result of vulnerability analysis of reverse engineered SBOOT code. The exploit causes SBOOT to ignore the CC mode setting which can be achieved in several ways.

One method to interrupt SBOOT’s setup flow and make it disregard the CC mode setting is to patch the code so that SBOOT calls "S boot enter download

mode" with the parameter 6, triggering the emergency ODIN mode which is discussed more in-depth in the Paper. This mode can then be used to overwrite the PARAM partition.

Another approach is to patch the "S CC MODE isSet" function to always return false. This can be done by either modifying the function to return 0 or by changing the global variable that indicates this mode to always be zero.

Furthermore, it is generally accepted that once one has access to uploading patches, even with MDM settings enabled, it is typically possible to bypass any security measures by disabling them, consequently implying a success security breach.

Setting DN ERROR By accessing the SBOOT console using a USB connector and an "RS232-to-USB" serial converter, one can edit the environment variable "DN ERROR" and force the path to open an emergency ODIN mode by executing the following commands. "setenv DN ERROR 2" which sets the environment variable of DN ERROR to two, "saveenv" which saves the changes and "reset" which restarts the system and boots it up with the new environment variable saved.

This will reboot the device in normal mode and force it to check the "DN ERROR" value, which will subsequently lead to the activation of emergency ODIN mode. This enables the possibility of modifying the PARAM partition as discussed earlier.

However, this attack can be prevented if the MDM setting is enabled, as the MDM flag is checked even when ODIN mode is entered. Consequently, only changes to SBOOT are allowed in this scenario.

3.3 Conclusion

Performing the attacks discussed in this paper increases the attack surface available for forensic acquisition. This expands the possibility of direct access to the storage or RAM partition and allows for unsigned firmware updates through ODIN mode, depending on the Factory Reset Protection and Rollback Protection settings on the device. Using the SBOOT exploit can bypass both of these settings if they are enabled.

This attack highlights the importance of a secure boot loader that maintains a chain of trust in code execution. Compromising SBOOT's security disrupts this chain of trust, leading to complete anarchy and a lack of trust in the normal world of code execution. Alendal stresses how unexplored vulnerabilities can be investigated by disabling CC mode on the device and researching how the MDM setting can be bypassed with such attacker access.

I would like to emphasize that even the most sophisticated and meticulously implemented security measures can reveal vulnerabilities when subjected to detailed analysis and comprehension. This phenomenon underscores the inherent complexity and challenges associated with developing robust security systems.

Security measures often involve intricate algorithms and protocols designed to safeguard sensitive data and prevent unauthorized access. However, these measures are typically implemented in real-world systems where perfect security is nearly impossible to achieve. The interplay between various components, user behaviors, and potential attack vectors can introduce unforeseen weaknesses.

Chapter 4

Paper II: Forensic Acquisition - Exploiting Vendor-Defined Messages in the USB Power Delivery Protocol.

4.1 Introduction

The USB Power Delivery protocol is designed to enable devices to negotiate power exchange methods. While it uses standard commands, it also includes special vendor-specific commands within the SCSI protocol. These undocumented commands can potentially lead to firmware updates, memory dumps, and even enable backdoors on devices. This paper focuses on these vendor-defined messages, aiming to exploit them to identify commands that can be leveraged in digital forensic investigations to acquire data stored on devices. There are two types of storage of interest for Digital Forensic Acquisition (DFA): RAM (volatile storage) and long-term storage with a well-structured file system.

RAM storage employs encryption methods that prevent data from being understood even if it can be obtained. This is achieved through encryption keys, which are themselves protected by additional encryption keys that tie the data to the device's long-term storage. The data required to decrypt this information is typically safeguarded by security features like TrustZone, which rely on tamper-proof hardware. Therefore, exploiting device vulnerabilities to increase the attack surface or implementing backdoors is necessary to access the required data.

G. Alendal emphasizes that any data acquisition method should be thoroughly researched and tested across different devices. The USB Power Delivery protocol is one such method that aims to expand the attack surface on the target device. It helps identify hidden features and security vulnerabilities that can be exploited to facilitate data acquisition.

In essence, the USB Power Delivery protocol is an inter-device communication channel that makes negotiation of the optimal power delivery method between two devices possible, the possibility of such communication can be an potential exploit. This report presents a black-box testing approach to reveal proprietary messages that could be used in digital forensic investigations to acquire data stored on devices supporting the protocol.

The USB Power Delivery protocol creates a highly vulnerable yet under-researched attack surface. Alendal points out that this should be of particular interest to the security analysts and law enforcement agencies. To underscore his point, Alendal mentions the existence of emulators designed to research these undocumented commands on various devices and the dangers they pose if the system is not properly armed to protect itself from those emulators that might assist the attacker in locating vulnerabilities.

The USB Power Delivery protocol is made for enabling a cross-vendor-device communication method to negotiate power trade configuration and is specifically for devices using Type-C input. This protocol makes a both way communication possible. This protocol makes use of three different communication messages, which makes the negotiation of power source configuration possible.

1. Control Messages
    - Short messages
    - Does not require data exchange
2. Data Messages
    - Contains data objects that are transmitted between devices.
3. Extended Messages
    - Data messages with larger data payloads

Depicted in Figure 4.1 is the structure of a data message used in USB PD protocol, where the grayed out areas are the areas that are handled the transport layer protocol which is outside of this Papers scope of interest.

Figure 4.1: Structure of a simple data message used in USB PD protocol

The USB Power Delivery protocol requires both ports (the sink which is the power receiving device and the source that provides power) to establish a contract that defines the configuration of power exchange. Once this contract is established, various types of messages are exchanged. Among these are vendor-defined messages, which can be either (1) structured or (2) unstructured. Structured messages come with some documentation on how to use them, whereas unstructured messages are implemented by vendors on what Alendal describes as an "ad hoc basis" which makes it undocumented and complicates further comprehension of their nature.

Vendor messages are a type of data message, and they are limited in size. The maximum data a message can contain is limited to six vendor data objects (VDOs)

plus a 32-bit header. Each VDO contains a 32-bit value. This protocol requires vendors to include a 16-bit USB-IF number, assigned to the vendor's ID, in each vendor message to prevent conflicts between vendors. The USB-IF number is an identification method provided by the developers of the Universal Serial Bus to ensure security and integrity for "vendor edit" access for devices.

Thus, an editor with an appropriate USB-IF-assigned vendor ID can implement any command containing up to six additional VDOs in one vendor-defined message. The command is the second part of the message header and can be any 15-bit value.

4.2 Topic

4.2.1 Methodology

Alendal adopts a black box approach due to the varied and undocumented nature of unstructured vendor-defined messages across numerous vendors using the USB Power Delivery protocol.

The methodology involves setting up a device to act as the source, establishing a connection with the test device, and checking for vendor-defined messages. While a detailed description of this method is beyond the scope of this paper, a general overview of the attack methodology is provided.

Although the approach may appear simple, it is complex. One of the initial challenges is accurately guessing the vendor ID of the device of interest. Without the correct vendor ID, the device will not respond to a vendor-defined message, even if the command itself is correct.

This complexity is further exacerbated by the timeout periods of various protocols involved in establishing a power configuration contract between devices. A power charging configuration contract must be established before any other messages can be sent. Consequently, any software attempting to guess the correct vendor ID may be rendered ineffective by these timeouts.

By capturing the device's response to the "Discover Identity" command, one might obtain valuable information to forge a valid vendor ID and command. Alendal outlines two main approaches: targeting skews in the device's response timing and testing for responses other than "GoodCRC" from the device. Due to the lack of sufficiently accurate technology to detect timing skews, the research focused on the second approach.

4.2.2 Analyzing the results

When analyzing responses in the USB PD protocol the researcher considered not getting any response as a form of response, indicating that the device (sink) might not support the USB Power Delivery protocol. According to the protocol documentation, the source should send a prompt with the supported revision versions of the protocol, and the sink should respond with the highest version it supports from those provided.

After explicit contract negotiation, a Discover Identity message was sent to the test device to obtain a USB-IF vendor ID. When the explicit contract was successfully negotiated and the protocol was confirmed to work correctly, the next step in the research was to identify potential unstructured vendor-defined messages available

on the device. Any response from the device other than GoodCRC was considered an attempted response to the command sent.

Without delving into the specifics of the brute force attempts or analyzing the device responses, the results indicated that very few test devices responded to the brute force tests.

Observations from the tests showed that the four vendor data objects (VDOs) sent in response to the same vendor message with different vendor IDs varied in a pseudo-random order. It was deduced that setting the vendor use to 0x0002 corresponded to an initialization command. Repeating the message with different vendor uses resulted in different VDOs, which researchers concluded must involve some form of encryption or obfuscation. The reasoning was as follows.

Observation 1: Sending two identical messages yields the same results. Any randomization of the four VDOs sent with vendor use = 0x0002 seems to result in a random reply VDO when compared with messages using vendor use = 0x0003.

Observation 2: This behavior instantiates an initialization vector or a key transmission in response to the message sent with vendor use = 0x0002.

Concluding, encryption is in place, and this encryption may be a symmetric cipher.

Additionally, a device called the Samsung Anyway S103 was employed to demonstrate the functionality of an exploit. This exploit used a console interface provided by the device boot loader, typically used for debugging. The exploit targeted bypassing certain security features of the Anyway device, demonstrating the significance of establishing attack surfaces. An additional factory test feature was enabled.

The Samsung Anyway S103 was used as the source, and the Samsung Galaxy S9 was selected as the sink. Observations of the communication showed that no explicit contract was negotiated before vendor-defined messages were sent.

Repeating the experiment with a micro-USB Samsung device allowed access to the same console as the Samsung Anyway S103, without the factory device's assistance. The same message exchange was successful, reinforcing the belief that the first four VDOs in the vendor use = 0x0002 reply were crucial initialization vectors.

Alendal stresses that using the received VDOs as potential keys or initialization vectors with some symmetric ciphers and vendor use = 0x0003 did not yield any positive results.

4.3 Conclusion

This method employs a black box approach, where the target device is unaware of the testing process. The results indicate that at least one of the test devices responds to forged messages, revealing some proprietary USB Power Delivery protocol message information. This raises the possibility of exposing initialization vectors or symmetric keys during negotiation.

Another significant aspect of this research is the ability of the researchers to enable factory device features on the test device, thereby exposing a wide range of attack surfaces. This was demonstrated on a test Samsung device, where the researchers successfully opened a console on the device, allowing vendor-level communication without using the factory device.

This source-sink relationship could potentially be used to transfer data out of a device via the USB Power Delivery protocol. Future research should focus on this potential vulnerability, as it remains largely unexplored. G. Alendal offers several suggestions for improving this testing and obtaining more accurate feedback. One notable suggestion is the use of a third party device to measure the response time for different VDOs.

Chapter 5

Paper III: Leveraging the USB Power Delivery Implementation for Digital Forensic Acquisition.

5.1 Introduction

When acquiring a device with potential forensic value, it is essential to consider the entire attack surface. Alendal stresses that the newly introduced USB Power Delivery Protocol is particularly important and warrants further exploration due to its undocumented nature and independence from the device's primary security systems. This protocol is managed by dedicated hardware beyond the control of the device's operating system.

This paper presents a black-box approach for discovering vulnerabilities in the USB Power Delivery Protocol on Apple devices. The approach involves reverse engineering Apple-specific communications and carefully analyzing the firmware of the USB Power Delivery hardware.

Acquiring data from a seized device can be challenging today, as direct access to the device's storage is not feasible. Instead, one must consider user-credential encrypted data, especially if the device is in the After First Unlock (AFU) state. Exploiting this state requires identifying interfaces that can help bypass security implementations and enable for information gathering. Externally-accessible interfaces is particularly desirable due to their accessibility and the avoidance of physically opening the device.

G.Alendal includes that research has shown how likelihood for code errors increase corollary to code density. The USB Power Delivery protocol has already a large code base. It is implemented on multiple different platforms and is designed to be able to interact with many other software which enlarges the possibility of unconsidered circumstances creating a vulnerability in the system.

Alendal questions the integrity of the chip implementing the protocol and the device's system. He suggests that flaws in the chip's implementation could lead to "evil maid attacks," where the firmware in a chip implementing USB Power Delivery protocol could be manipulated to assist the attacker in gaining crucial information about weaknesses in the system. He stresses the importance and challenge of extracting firmware, noting that the protocol and its implementations have vulnerabilities that could be exploited through hidden features and implicit trust relationships.

Researching the USB Power Delivery protocol is challenging because the source code documentation is kept confidential. Therefore, reverse engineering is one of the few methods available to researchers for locating vulnerabilities. This process is computationally demanding, involving both static and dynamic methods to produce human-readable assembly code, followed by decompilation to obtain pseudo high-level source code. Static reverse engineering analyzes the code without execution, while dynamic reverse engineering involves interacting with and debugging the code during execution.

Another method of gathering information about the firmware is retrieving it from general iOS updates for Apple devices, specifically by unpacking and investigating the iOS updates distributed in .ipsw archives.

Understanding the code is often complex due to limited information about the hardware, architecture, memory layout, and interfaces. Since the code is not intended to be human-readable, it rarely includes error messages, making static reverse engineering infeasible. Therefore, dynamic reverse engineering, which executes the code and observes firmware behavior using embedded tools, is necessary. However, reproducing a pseudo high-level source code that is human-readable is challenging and often incomplete, serving mainly to provide a greater understanding of the code.

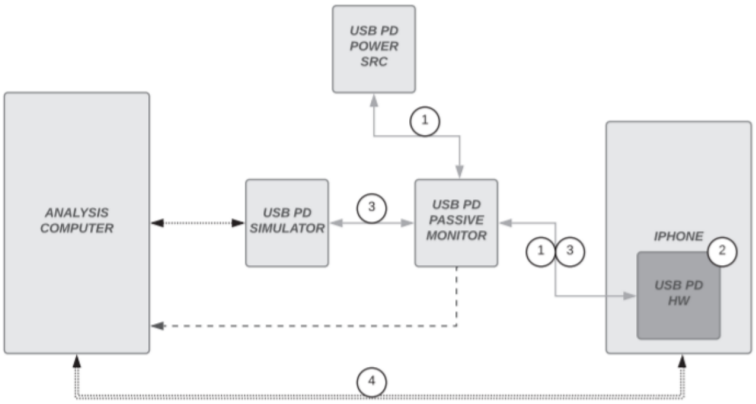

Figure 5.1: This is the chosen setup that was used to conduct the attack that employed reverse engineering.

The first step involves gathering information from open sources. Researchers used a generic USB PD passive monitoring tool to observe device functionality when connected to other Apple devices versus other vendors (as shown in Figure 5.1). This method examines behavior beyond intended functionality, offering weak indications of proprietary vendor code. However, passive monitoring of USB PD communication covers only a small portion of the protocol and some vendor-specific messages, prompting researchers to extract the firmware from the USB PD chip in the selected Apple device and reproduce pseudo code through decompilation.

A brute-force, trial-and-error method was employed by sending forged messages and observing the device's behavior. After numerous trials and gaining a deeper understanding of the components, firmware, and undocumented vendor-specific functionalities, researchers attempted to reconstruct the device's behavior based in the results. Jailbreaking the device was one of the methods used in this process.

5.2 Topic

The devices researched included an iPhone X, iPhone 8, and iPhone 8 Plus. While some hardware design information was available from vendors, mapping how data is stored in the hardware was not possible. Helpful insights included details about the microcontroller, which employs additional flash storage for firmware, and the system interface used to communicate with other peripherals, like Apple's system-on-chip.

Another useful feature highlighted by G. Alendal is the Serial Wire Debug (SWD) access, which can be utilized via a JTAG interface. JTAG, short for Joint Test Action Group, is an industry-standard device used for the development and verification of printed circuit boards. Although potentially useful for reverse engineering, JTAG was not employed to gain interface access in this research.

Using a commercial USB Power Delivery analyzer, researchers determined whether the devices supported the USB Power Delivery protocol. This was crucial because many iPhones do not have this protocol enabled. The analyzer also revealed the revision specification, which indicates the supported messages for the given device.

When the target iPhone X was connected to a power source and the information gathered was inspected, vendor-specific communication was uncovered. The communication began with an unstructured VDM, signaling that further proprietary communication would also be unstructured. The target device was then asked for its device ID, after which the power source initiated the Apple-specific protocol, which will be discussed later.

Important USB Power Delivery firmware files can be located in iOS updates, which are regularly released by Apple to update the operating system and support on-board peripherals. The file of interest is named "USB-C HPM,X.BIN", where 'X' denotes the number of firmware files included. Although these firmware files are very similar in size with minor binary differences, each has its own product ID (PID). These PIDs respond to a structured Discover Identity message in the Power Delivery protocol (VDM), enabling other devices connected via USB PD to identify the iPhone model.

These firmware files are instrumental in discovering vulnerabilities. G. Alendal notes that because the firmware code is very similar across different PIDs, a vulnerability found in one firmware can be exploited in other iPhones with similar PIDs. This broadens the applicability of forensic data acquisition methods. Developing one reverse engineering method for a specific device may thus be applicable to several other models, saving time and resources.

Additionally, the codebase supports binary diffing, a method of comparing two binary files. This significantly reduces the time and resources needed to discover potential security patches across devices. Downloading different iOS updates can be used to analyze and detect changes to the USB Power Delivery firmware. The research showed that the SHA-1 hash remains unchanged from version to version unless there is a USB Power Delivery firmware update. This linearity in hash can be observed in Figure 5.2

The device architecture used in this experiment is known to be ARM little-endian, which is a "lowest-level" code, meaning the code interacts directly with the physical parts of the device. To understand this type of code, it is necessary to make assumptions for effective reverse engineering.

In their reverse engineering approach, the team successfully reconstructed and

iOS	Filename	sha1sum(USB-C_HPM,4.bin)
13.4	iPhone_5.5_P3_13.4_17E255_Restore.ipsw	9767A86F62ABDC8C1046F4D807CC30DAB99A4693
13.3.1	iPhone_5.5_P3_13.3.1_17D50_Restore.ipsw	273A80375FE8FEC09D498221BE472958B818582F
13.3	iPhone_5.5_P3_13.3_17C54_Restore.ipsw	273A80375FE8FEC09D498221BE472958B818582F
13.2.3	iPhone_5.5_P3_13.2.3_17B111_Restore.ipsw	273A80375FE8FEC09D498221BE472958B818582F
13.2.2	iPhone_5.5_P3_13.2.2_17B102_Restore.ipsw	273A80375FE8FEC09D498221BE472958B818582F
13.1.3	iPhone_5.5_P3_13.1.3_17A878_Restore	273A80375FE8FEC09D498221BE472958B818582F
12.4.1	iPhone_5.5_P3_12.4.1_16G102_Restore	79CB8220D2C6F5917C1C11ED7B4BF733E3C9B1C8
12.3	iPhone_5.5_P3_12.3_16F156_Restore.ipsw	79CB8220D2C6F5917C1C11ED7B4BF733E3C9B1C8
12.2	iPhone_5.5_P3_12.2_16E227_Restore.ipsw	79CB8220D2C6F5917C1C11ED7B4BF733E3C9B1C8
12.0	iPhone_5.5_P3_12.0_16A366_Restore.ipsw	B374072044A97669A688A49E1723C55E9973A851
11.4.1	iPhone_5.5_P3_11.0_11.4.1_15G77_Restore.ipsw	1E20D8B4D54D6C092DA9B668A53AAAE81ABFA3EE
11.0	iPhone10,5_11.0_15A372_Restore.ipsw	1E20D8B4D54D6C092DA9B668A53AAAE81ABFA3EE

Figure 5.2: Multiple different versions, hashes to the same SHA256 checksum

Spec	Index	Time	Role	Message	Data
v2.0	182	0:21.801.787	Source:DFP	[3]Source_Cap	61 17 F0 90 01 08 EA 21 1F CC
v2.0	189	0:21.803.817	Sink:UFP	[0]Request	42 10 F0 C0 03 13 BC 0F E8 2B
v2.0	197	0:21.805.451	Source:DFP	[4]Accept	63 09 3F 92 D5 76
v2.0	203	0:21.834.345	Source:DFP	[5]PS_RDY	66 0B 56 07 AC E5
v2.0	238	0:24.825.555	Source:DFP	[1]VDM:DiscIdentity	6F 13 01 80 00 FF 16 62 AB 1B
v2.0	245	0:24.827.511	Sink:UFP	[2]VDM:DiscIdentity	4F 44 41 80 00 FF AC 05 00 54 00 00 00 00 00 21 7D 16 94 99 07 82
v2.0	255	0:24.831.372	Source:DFP	[2]VDM:DiscSVID	6F 15 02 80 00 FF 58 38 5E 86
v2.0	262	0:24.833.320	Sink:UFP	[3]VDM:DiscSVID	4F 26 42 80 00 FF 00 00 AC 05 F6 20 C2 26
v2.0	270	0:24.837.322	Source:DFP	[3]VDM:DiscMode	6F 17 03 80 AC 05 BA E4 F8 1B
v2.0	277	0:24.839.154	Sink:UFP	[4]VDM:DiscMode	4F 28 43 80 AC 05 02 00 00 00 72 AD 21 96
v2.0	285	0:24.844.365	Source:DFP	[4]VDM:EnterMode	6F 19 04 81 AC 05 55 08 DD 38
v2.0	292	0:24.846.477	Sink:UFP	[5]VDM:EnterMode	4F 1A 44 81 AC 05 8E 2F C5 E3
v2.0	299	0:24.850.260	Source:DFP	[5]VDM:Unstructured	6F 1B 05 00 AC 05 E7 4D 56 1A
v2.0	307	0:24.851.919	Sink:UFP	[6]VDM:Unstructured	4F 1C 15 00 AC 05 5E C3 C3 FF
v2.0	315	0:24.853.357	Sink:UFP	[7]VDM:Attention	4F 3E 06 81 AC 05 02 01 AC 05 00 00 00 00 DC 69 C4 D9
v2.0	324	0:24.856.927	Source:DFP	[6]VDM:Unstructured	6F 3D 02 01 AC 05 00 00 00 00 06 00 00 20 12 5E E4 81
v2.0	333	0:24.858.774	Sink:UFP	[0]VDM:Unstructured	4F 10 12 00 AC 05 E6 16 E4 A7
v2.0	340	0:24.860.209	Sink:UFP	[1]VDM:Attention	4F 32 06 81 AC 05 02 01 AC 05 04 00 00 00 70 47 1C CC
v2.0	349	0:24.863.748	Source:DFP	[7]VDM:Unstructured	6F 3F 02 01 AC 05 04 00 00 00 00 02 08 00 D1 C4 AA B0
v2.0	358	0:24.865.628	Sink:UFP	[2]VDM:Unstructured	4F 14 12 00 AC 05 26 B0 64 52

Figure 5.3: Captured messages of the USB PD

understood all "Apple-specific" messages, which helped them comprehend the unstructured VDMs supported by the firmware. Thanks to the fact that all "Apple-specific" unstructured VDMs are handled by the same handler function, the research team was able to reconstruct the unstructured VDM messages. The handler function, which also manages user input, may be a potential target for vulnerability analysis. Any error in handling USB PD messages could be an attack vector, as G. Alendal explains.

With the use of Figure 4.1 , one can observe clues about Apple-specific information being exchanged. By dissecting the raw data of unstructured VDMs, the team determined the byte locations of the corresponding USB PD message header, VDM header, CRC message, and finally the VDM header (little-endian). The VDM header revealed the VID and VDM type 0 (unstructured type). After sending a dissected Identity Discover message, a response containing the Apple VID and an undocumented command was received. G. Alendal states that for each undocumented command, a handler function can be identified in the associated firmware file "USB C HPM,4.bin" and disassembled. This handler function helps understand the purpose and consequences of any given command.

Message 315, sent from the sink to the source, occurs only when communicating with an Apple device. Additionally, the iPhone requests various data from the Apple power source, such as the serial number, device name, and manufacturer information.

Using "checkra1n," the researchers gained root access to the iPhone, which enabled the researchers to gain information to recover and understand unique Apple VDMs used in Apples PD protocol. The researchers were able to obtained the content of the iPhone I/O Registry, allowing for further interpretation of exchanged data. G. Alendal concludes that by using the discovered handler function in the firmware, one can identify and understand all the implemented as well as unstructured vendor protocol messages.

Controlling the supported messages and the ability to communicate with the iPhone hardware makes it possible to discover and exploit security vulnerabilities, such as direct code execution on the iPhone hardware. The protocol's poor integrity checks enlarge the attack surface, allowing messages sent between devices to be intercepted and modified at will.

The "USB-C HPM,X.bin" files are unsigned and are not verified during installation or runtime. After jailbreaking the device, researchers were able to update the "USB-C HPM,5.bin" file located on the device without requiring user credentials. This modification can be observed by monitoring the communication between the Apple power supply and the iPhone with the modified firmware, resulting in a change in the PID. This change is a response to the Discover Identity VDM message. The discovery shows that it is fully possible to modify the USB Power Delivery firmware and perform a rollback to an older, more vulnerable system. G. Alendal highlights the need to research this vulnerability further due to the unpredictable nature of how the file might be modified, suggesting that it could potentially propagate to other iPhones.

5.3 Conclusion

The methodology discussed in this paper exploits the implicit trust between the USB Power Delivery protocol and the vendors of different devices. The performed analysis of the implementation details of a USB PD device can be applied to a wide range of different vendors. Information gathering, as conducted in this paper, supports firmware reverse engineering, side-channel analysis, and attack development. Using a simulation tool, the researchers were able to simulate sending and receiving arbitrary messages, which facilitated the reverse engineering process and exploitation development.

This black-box approach, which involved reverse engineering firmware to create disassembled code, enables both manual and automated vulnerability analyses. Additionally, using techniques that help identify differences between patches can also increase the possibility of locating an vulnerability. Rollbacks of the USB PD protocol are also possible on jailbroken devices without requiring user credentials, due to the lack of firmware signatures and rollback protection mechanisms in this protocol.

Alendal states that future research will continue to focus on vulnerability discovery, including simulating the extracted firmware to advance "fuzzing" techniques for identifying vulnerabilities.

Chapter 6

Paper IV: Digital Forensic Acquisition Kill Chain - Analysis and Demonstration.

6.1 Introduction

The goal of this paper is to demonstrate how the Kill Chain is inteded to be used and how it can help to increase the performance of a DFA. G. Alendal describes the Generic Digital Forensic Acquisition model as consisting of four phases: Seizure, Acquisition, Analysis, and Reporting. The Seizure phase involves obtaining the device, Acquisition focuses on retrieving data from the seized device, Analysis entails examining the gathered information, and Reporting involves presenting the findings of the processed information. As mentioned earlier, vendors like Samsung and Apple prioritize data encryption to secure devices against both local and remote attacks, making it challenging to access data without user credentials.

Reliance on vendor assistance is not a viable option, as law enforcement has faced resistance from various vendors. Alendal cites multiple instances where law enforcement requests for assistance in bypassing security have been denied. One notable case is when Apple refused the FBI's request to create a backdoor to access information during a terrorism investigation. This scenario forces law enforcement to develop new methods and adopt a more offensive approach to acquiring data from devices, including analyzing and discovering n-day and 0-day vulnerabilities.

The notion of law enforcement leveraging vulnerabilities that are not known to the vendors (refereed to as n-day vulnerabilities) is particularly concerning, as it requires them to assume the role of an attacker to obtain necessary information. The practice of acquiring and withholding discovered vulnerabilities for their own use raises significant ethical issues. Alendal states that this paper does not take a stance on the ethical dilemma but instead introduces a method for acquiring forensically sound information using the "Digital Forensic Acquisition Kill Chain," based on the "Intrusion Kill Chain."

The "Intrusion Kill Chain" is described as "... a systematic process of targeting and engaging an adversary to achieve the desired security effects." The Digital Forensic version of this approach aims to "target electronic devices using offensive techniques to facilitate digital forensic acquisition." Law enforcement, with the authority to seize devices, can control target device updates and exploit both publicly

known n-day vulnerabilities and undiscovered 0-day vulnerabilities.

Historically, courts have required a standardized outline for court-accepted Digital Forensic Acquisition. Montasari provided this with his standardized model of Digital Forensic Acquisition, which ensures that law enforcement adheres to established guidelines, even if they are very generic.

6.2 Topic

6.2.1 Putting words upon the unspeakable

The goal of the "Digital Forensic Acquisition Kill Chain" is to create a generic yet precise framework for law enforcement to develop new acquisition methods using offensive techniques. It aims to "...improve the performance and success rate of time-constrained, case-motivated development of digital forensic acquisition methods..." in the short term. In the long term, it aims to establish a process for developing and improving other digital forensic acquisition methods that are not as case-sensitive, considering trends in consumer technology adoption.

G. Alendal describes a model of the intrusion kill chain outlined by Eric M. Hutchins [2], where the goal is make vendors of the devices aware of the process used by attackers to prevent such simple attack scenarios from happening. Specifically, Alendal mentions a security implementation method that detects intrusions by referring to previously known intrusion patterns to mitigate attacks before they occur. This method is known as "intelligence-driven computer network defense."

These phases specify the process of acquiring "intelligence" on a potential target surface in order to be able to penetrate a specific security countermeasure. This "Kill Chain Paradigm" has proven to be practical for identifying unwanted threatening behavior from adversaries.

Hutchins' Intrusion Kill Chain encompasses six phases: find, fix, track, target, engage, and assess. This model improves existing doctrines by specifying new phases of targeting. The correlated Kill Chain methodology includes reconnaissance, weaponization, delivery, exploitation, installation, command and control, and actions on objectives.

The goal of this methodology mapping is to provide proactive intrusion detection that detects and mitigates threats before they materialize. However, in this paper, the method is used to describe how to leverage the intrusion kill chain to facilitate offensive actions in digital forensic acquisition scenarios, considering modern defensive mechanisms.

The Digital Forensic Acquisition Kill Chain consists of nine phases, most of which are well known. The first five are of great importance and may require a methodological approach, while the last four deal with more operational issues. Alendal identifies two main scenarios motivating the Digital Forensic Acquisition Kill Chain. Where "Case-Motivated Scenarios" are scenarios that is relevant for specific cases, and "Case-Independent Scenarios" are those that are relevant for the DFA in general:

1. Case-Motivated Scenarios

    - In these scenarios, a specific device is targeted. Alendal explains that DFA is often case-driven, and new needs frequently arise without the

required methodology to fulfill them. This Kill Chain attempts to provide a guide for addressing these needs, creating a process to acquire necessary information legally and systematically.

2. Case-Independent Scenarios

    - These require a general outline for developing new information acquisition methods that address a "class of challenges." These scenarios may depend on the related vendor being challenged, utilizing gathered information about the higher-level layer before employing the "case-motivated kill chain."

Both scenarios require a generic kill chain outline to guide the development of methods for acquiring desired information.

6.2.2 Kill Chain Discussion

Alendal discusses the different phases and what is involved in their scope.

Reconnaissance phase. This phase involves narrowing down the targeted devices and gather relevant information, including details that could be missused to penetrate the target system and estimate the likelihood of a successful attack.

1. Case-Motivated Scenarios

    - This step should be concise and brief since the target is often already known, and the necessary information is typically gathered in advance.

2. Case-Independent Scenarios

    - In these scenarios, this phase is more open-ended. Gathering information and locating vulnerable targets is not as straightforward and requires a broader approach.

Identification phase. This phase outlines how the approach to acquiring relevant information from targets is often performed in a bottom-up manner. The second step involves identifying relevant sources of information to target and then identifying the layers that protect the targeted information.

Alendal points out that relevant information for digital forensics is often stored in volatile memory, which would not survive a power cycle. He notes that addressing this issue would divert from the main discussion of the kill chain, and therefore, it is not further explored.

Surveillance and Vulnerability Research. During this step, existing vulnerabilities, techniques, tools, and services are investigated to find potential attack vectors.

Alendal introduces sub-phases to address efficiency issues during the research process. One sub-phase focuses on gathering an overall vulnerability assessment, keeping a broad perspective and concentrating on various approaches and their potential success, as well as the challenges associated with these methods and attack

surfaces. The second sub-phase focuses on finding relevant tools to overcome the challenges identified in the first sub-phase and delves deeper into the methods presented in that step.

The main goal of this division into sub-phases is to ensure proper resource allocation. The first sub-phase, which identifies overall challenges and potential methods, is more certain and may not require as many resources. In contrast, the second sub-phase, which is more uncertain, is often time- and resource-intensive.

As an example of how one kill chain can spawn multiple others, discovering a vulnerability can lead to a new kill chain that attempts to exploit this vulnerability. This new kill chain requires the same phases used during the general digital forensic kill chain process.

Weaponization phase. This phase involves developing and implementing one of the identified exploits, which, as Alendal notes, is often challenging and may initially seem "unrealizable." Therefore, this step progresses from developing a proof of concept to creating a production-quality exploit that has a high probability of success. The most critical aspect of this step is to ensure that the method is "forensically sound and complies with the law and established digital forensic standards."

The Delivery phase. The Delivery phase focuses on discovering and utilizing channels to execute the weaponized exploit. After preparing the weapon e.g. a program, the next step is to find how this program will be installed on target software. Potential channels include USB, SPI, JTAG, I2C, Wi-Fi, Bluetooth, Near Field Communication (NFC).

Exploitation phase. During the exploitation phase, one must adapt to challenges that may arise while implementing the developed methodology. This step considers aspects such as the device's state, legality, special requirements, and any new challenges that were previously not considered.

Installation phase. This phase primarily focuses on maintaining an reasonable access to data, such that the acquisition only makes use of relevant information and does not cross any boundaries of personal life of the suspect outside its jurisdiction. The evidence of what data was processed during the DFA of the device is the digital fingerprint that is left after the acquisition. Since the state of the device is used as evidence in court, it is crucial to document the state and any enabled steps or features for legal purposes.

Command and Control phase. This is the phase when the attacker has managed to gain control over the device, potentially with an open shell and root access, further information should be gathered to identify relevant targets. Alendal emphasizes that this may result in the initiation of a new kill chain.

Actions on the objective. This is the final phase and its main goal is to execute the final goal which performs the digital acquisition of data valuable from the device or service.

6.2.3 Provided example of an Case-Motivated Kill Chain

This section provides an example of the desired Kill Chain process used in Digital Forensic Acquisition. The main goal of law enforcement is to extract data from a broadband router seized at a crime scene. The data is suspected to include log files with network activity that might be relevant to the investigation, specifically targeting log files that contain logs of access history of connected devices during a certain period that might be of relevance.

1. Reconnaissance
    - Analyzes focused on the open discussion forums about the firmware employed on the target router.
    - Information about patches was gathered and used in the investigation
2. Identification
    - Flash memory was the target information storage which stored log-data.
    - Challenge during this Data Acquisition is that many low-ended embedded devices, including this router stores log files in volatile short term storage. Considering the fact that volatile storage is only volatile and is completely cleaned out after restart this means that valuable data might be lost if the device is powered off or restarted.
        - (This challenge spawns a separate kill chain which targets the RAM data. This idea was considered to contain more valuable information and therefore was further pursued during the investigation.)
3. Surveillance and Vulnerability Research
    - Sub-Phase 1
        - Acquires the information about RAM and its possibly exploitable vulnerabilities.
        - Vendors Patch report was considered as a potential vulnerability information source. Older firmware files and change logs were available and therefore this method was pursued. With the possibility for the change logs to contain some information about a potential security issue.
            * Different versions were analyzed but it was taken in consideration that although those versions could include more vulnerabilities, rollback is not possible since the target data would not survive a restart.
        - Previous firmware version was discovered to have major security flaws, which was patched in the newer version.
    - Sub-Phase 2
        - The more demanding sub phase included identifying and analyzing the difference between the two version (the installed one, and the one that required the patch.).

        * This process involves unpacking and comparing the two versions, which might be resource intensive.
    - This Phase discovered how the security flaw that was addressed by the patch was a shell console that could be opened during the login process by simply entering ctrl + z. This spawned a shell console with root access to the router. Thus this phase resulted in the rediscovery of the vulnerability and a potential vulnerable target, while the vulnerability still remained to be triggered.

4. Weaponization

    - Triggering this vulnerability was shown to be rather simple. This led to making the weaponization phase the phase that involves discovering how the console might be accessed.
    - This was done via an UART interface on the circuit board of the router.
    - The results included that this could be done without powering off the device which would result in losing the valuable volatile data.

5. Delivery

    - This was performed by sending the ctrl+z signal over an attached serial console.

6. Exploitation

    - During this phase the risk and likelihood of failure must be addressed in order to prevent ad hoc decisions.
    - The reason behind this, is the fact of how each action performed on the device might consequently affect the RAM data. Therefore a bare minimum footprint must be maintained.

7. Installation

    - Persistent access was acquired which did not require other tools being installed.

8. Command and Control

    - In order to preserve a minimum footprint only few carefully inspected commands was executed from the shell.

9. Actions on the Objective

    - The goal was acquired were the access to the RAM was successfully granted.

6.3 Conclusion

As mentioned in subsection 6.2.1, Alendal aims to develop a "generic yet precise" outline for Digital Forensic Acquisition to follow during investigations. However, creating a general step-by-step guide for such a volatile process is challenging. Cases can differ, circumstances may change, and law enforcement may need to make more "ad hoc" decisions than desired. These factors can influence how the DFA process is conducted. What is commendable about Alendal's work is his focus on creating an overall outline of procedures for different stages of forensic acquisition, rather than a rigid plan.

Although the notion of law enforcement needing to "become the attacker" to gain investigation-relevant information might be controversial, it is necessary. The exponentially growing complexity of security mechanisms threatens the feasibility of performing successful DFA to aid criminal investigations.

Chapter 7

Paper V: Chip Chop - Smashing the Mobile Phone Secure Chip for Fun and Digital Forensic.

7.1 Introduction

7.1.1 Compromising the Secure Element, Analysis and Implications.

The SE has been one of the most challenging hardware-enforced security methods for Digital Forensic Acquisition (DFA). The embedded Secure Element is designed to withstand a fully compromised system and aims to minimize the attack surface for both the system it is implemented in and itself. This includes keeping its functionality as simple as possible to reduce the likelihood of implementation errors.

This paper presents how researchers, with limited resources, managed to compromise the eSE implemented by Samsung. By "compromise," we refer to the complete defeat of the Secure Element's security goals, resulting in the total loss of its features and trust. This research highlights "the gap between intended and achieved security" and introduces the methodology used, addressing the major consequences of even small vulnerabilities.

Alendal demonstrates how the concepts of Trusted Computing have driven DFA towards a more offensive approach, focusing on locating and exploiting vulnerabilities in systems. Trusted Computing involves standalone reliable hardware devices that perform as intended. Various vendors implement this concept, using hardware-assisted separation within the system to create trusted, tamper-proof, and secure environments for critical security components. One application of such a system includes the separation of root trust, implying that the system's trusted root does not necessarily have the same level of trust as the SE element.

The eSE is designed to operate almost entirely independently of the system it is embedded in, providing protection against side-channel attacks. The eSE controls its own processor, RAM, and storage.

This security approach introduces a single point of failure, meaning that although the attack surface is small, any security vulnerability could lead to the complete compromise of the system. To verify that the eSE performs as intended, devices are certified with a CC EAL certificate. Specifically, the compromised certificate in this

paper is the Common Criteria EAL 5+ certificate.

The attack discussed in this paper is a remote attack on a state-of-the-art eSE hardware used by Samsung. This attack does not require physical access and targets the logical interface, bypassing the eSE's security and facilitating the DFA of user data. It works on BFU (Before First Unlock) state devices without any user credentials.

This paper asserts that placing all security trust in a single, albeit very secure, hardware component is not an optimal security solution. It introduces the risk of total system compromise due to a single implementation error. The severity of these findings is significant, especially given that the trust in the eSE implemented by Samsung is expected to provide a high level of security, as required for protocols like the mobile eID solution used in Germany. The research concludes that despite such certification, the robustness of the security implementation cannot be guaranteed. Additionally, it reveals that this vulnerability cannot be patched in already shipped Samsung mobile devices.

7.1.2 Secure Element background knowledge.

The tested Secure Element (eSE) was one of the newer versions available during this research. This eSE is a single-chip solution, soldered directly to the circuit board of the mobile phone.

The Embedded Secure Element in question holds the CC EAL 5+ certificate, which is supposed to guarantee that the device works as intended. This certificate comes with two other documents: the Security Target document, which describes the SE and its security requirements, and a second document that outlines the intended protection profile. This protection profile is described as "generic and not specific to the S3K250A," which is the SE chip.

The main security goals outlined in these documents are to maintain the integrity and confidentiality of user data and to ensure the correct operation of the services provided by the eSE. Specifically, an attacker should not be able to alter stored data, read any eSE data, or influence the operation of the eSE.

The certification itself acknowledges that the product may not be completely vulnerability-proof, as achieving a completely error-free system is almost impossible. Therefore, efforts are made to minimize the likelihood of errors as much as possible. One such effort is the implementation of the Common Criteria Advanced methodical vulnerability analysis, a vulnerability assessment aimed at identifying potential vulnerabilities. This assessment is ranked from 1 (sufficient) to 5 (strongest security promise). The "AVA VAN.5" level mentioned in the certificate for the eSE indicates that the highest level of vulnerability analysis was used to assess the device's security.

The following attacks assume physical control over the chip, targeting an isolated hardware component like the eSE. Two main attack vectors are present: the logical interface between the eSE and the host system (the Rich Execution Environment, or REE), and potential side-channel attacks on the eSE. Communication between the secure element and the device is managed by the Application Protocol Data Units (APDU), originally a communication protocol for smart cards. More specific information about the process is illustrated in the figure 7.1.

Attacks on the physical separation implementation have been previously ad-

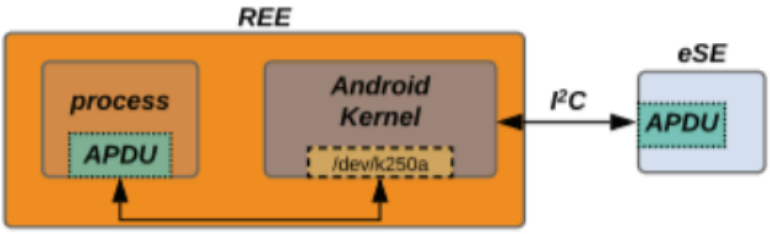

Figure 7.1: Description of the communicating components of REE with the eSE.

dressed. Mr. Anderson [3], whom G. Alendal refers to, has identified two types of attacks in this scenario: those involving physical access and logical attacks. Attacks involving physical access are referred to as local attacks and include many side-channel attacks. In contrast, logical attacks are referred to as remote attacks and target the logical interface.

G. Alendal explains how Mr.Anderson conducted a cryptographic API attack that exposed design flaws, leading to information leaks via the API. This information could then be used to perform a brute force attack on the embedded DES keys. Alendal also mentions several other researchers who have conducted attacks on isolated chips. Some of these attacks use oracles to leak information without requiring knowledge of the target code.

Another significant attack is the remote buffer overflow attack conducted by Mr.Bittau [4], which did not require knowledge of the target binaries. Bittau improved the typical method of employing gadgets for Return-Oriented Programming (ROP) attacks. His technique, called Blind ROP, can be used to attack closed-source and unknown implementations with the help of leaked information.

Numerous attacks have demonstrated the existence and severity of vulnerabilities in TrustZone implementations. These attacks highlight the importance of design and coding quality for security and the significant consequences that can arise if these aspects are not properly managed.

7.2 Topic

The attack conducted in this research combines Bittau's BROP approach and Lee's Dark-ROP to gather the necessary information for a successful attack. The primary difference between a BROP attack and a Dark-ROP attack lies in their detectability. BROP attacks rely on feedback from crashes caused by the attacker, making them loud and easily detectable. In contrast, Dark-ROP is a stealthier, more sophisticated variation of the ROP attack, requiring more knowledge about the target system and its security mechanisms to avoid detection.

This attack does not require prior knowledge of the binaries, as gathering information is part of the attack process. The researchers used true Return-Oriented Programming to construct an image of the target system.

The research team describes the attack in the following steps:

1. Gather as much information as possible about the security measures and functionality of the secure element.

2. Gain knowledge about potential attack vectors on the eSE.

3. Identify at least one information-leaking oracle to locate a 0-day vulnerability in the eSE.

4. Exploit the discovered vulnerability to breach the system's confidentiality or integrity. In this context, breaching confidentiality means obtaining unauthorized information, while breaching integrity involves tampering with or manipulating data in the system.

The attack is based on accessing the logical interface of the chip, which was possible due to exposed Android Kernel access. This access allowed communication with the APDU of the chip, exposing all functionalities of the eSE hardware. The researchers operated as if they were a high-privilege binary process, providing numerous attack surface opportunities. They then set up a tool called "chip breaker" to gain more attack surface by gathering additional system information. This program ran with root privileges via a root shell spawned by the attacking system. To spawn this type of shell, the attacking system must be connected to the target system via a cable or network.

The success of this approach depends on how stable the established control over the system will be. Another method introduced by the researchers involves infecting a program with system privileges to communicate with the eSE with high privileges. Since the attack can be performed remotely, it opens up more side-channel attack possibilities. Gaining elevated privileges can be achieved in several ways without triggering a user data wipe, which is crucial. This elevated access can be used, for example, to break the secure boot of the device and introduce attacker code onto the target device.

The information gathering phase started on some already acquired information sources that was to great use when gaining information of the target system. The different information sources are stated as follows.

1. CC EAL certification document.

2. A privileged android service process that communicates with the eSE using the APDU-based logical interface. The program was called "hermesd". The only process in REE that has this ability of communicating with the eSE.

3. Libraries for vendors that support the program and implements very low level communication with the eSE. The files that the "hermesd" was accessing.

The researchers were able to find a container file which held the images of the boot up of the eSE. In which some files were encrypted while others were not.

The Logical eSE Interface Attack Vector. The target of the attack was the APDU handlers located on the chip, intended for oracle purposes and potentially leading to an input validation vulnerability. By observing the "hermesd" process communicating with the APDU of the eSE, the researchers gathered crucial information for reverse-engineering a REE library. This revealed the communication logic between the REE (hermesd) and the eSE.

Alendal was able to locate multiple handlers by analyzing and brute-forcing the eSE. These handlers could be used as oracles that either return a response or an

error. One assumed danger of brute-forcing the eSE with different message pairs is the possibility of triggering unwanted behaviors, such as a "Factory Reset."

Brute-forcing the APDU on the eSE allowed the researchers to discover multiple handlers, enabling them to communicate with the eSE using its own protocol.

When performing a "black-box" attack, some "blind attack" methods must be employed, which can yield information about the target system. This often involves "poking the system with a stick" using oracles that provide information through error messages or response returns. These oracles are a promising source of information leakage.

However, separating the eSE from the rest of the system makes access to such oracles challenging since they exist on a separate and protected system. Therefore, the researchers targeted the logical interface oracles, which were the only available option due to the chip's isolation. The identified oracles were as follows:

Oracle 1, was the lack of response, often caused by a crash or an error in message interpretation. Oracle 2, was the logical interfaces, where oracles that read or write data can be manipulated to gain additional information or alter existing data.

Using a customized tool for communication with the eSE, the researchers obtained buffer size information. The leaked information revealed details about the fixed size of messages within the eSE, which were assumed to be 32 bytes, while the length of a "secret" was said to range between 1 and 256 bytes.

The second oracle provided information about user data locations and similar pointers, which were valuable for the reverse engineering process and exploitation.

Combining the two oracles indicated error handling evidence (this was called Oracle 1) when the APDU was prompted with a buffer size greater than 84 bytes (this was called Oracle 2). It was concluded that this was overwriting important stack pointers. The researchers implemented a simple brute force attack with message lengths between 84 and 88 bytes. Successfully guessing the stack layout of some APDU stack data completed the vulnerability discovery step. Inspecting Oracle 1 (the behavior of responses from eSE) showed that the system suffered from a simple stack buffer overflow error. This, combined with knowledge about the system's "storing" functions, helped deduce what kind of data was overwritten.

Figure 7.2 illustrates how the buffer overflow attack was conducted using the "APDU writeWeaver" function, which utilized one of the handlers (IWEA) to write a tuple containing a challenge, a secret, and a footer to the system. Filling the slots with data to overwrite 100 bytes allowed the possibility of overwriting values in the R5-R7 registers stored in the next 16 bytes on the stack. The last 4 bytes would then overwrite the "LR register," which, if done correctly, would prompt the attacker system with valuable information.

Arbitrary flash and RAM read. The buffer overflow attack described in Figure 7.2 can be used to read flash and RAM memory by utilizing a special ROP gadget. This gadget takes an attacker-controlled message as input and returns 16 bytes of data. Due to the predictable nature of storage handling in the eSE, this was achieved by forging valid R4-R7 inputs, which could successfully overwrite the designated slots.

The researchers employed this methodology to gain access to the ROP gadget. By manipulating the message with valid R4-R7 and PC data, they were able to access the ROP gadget.

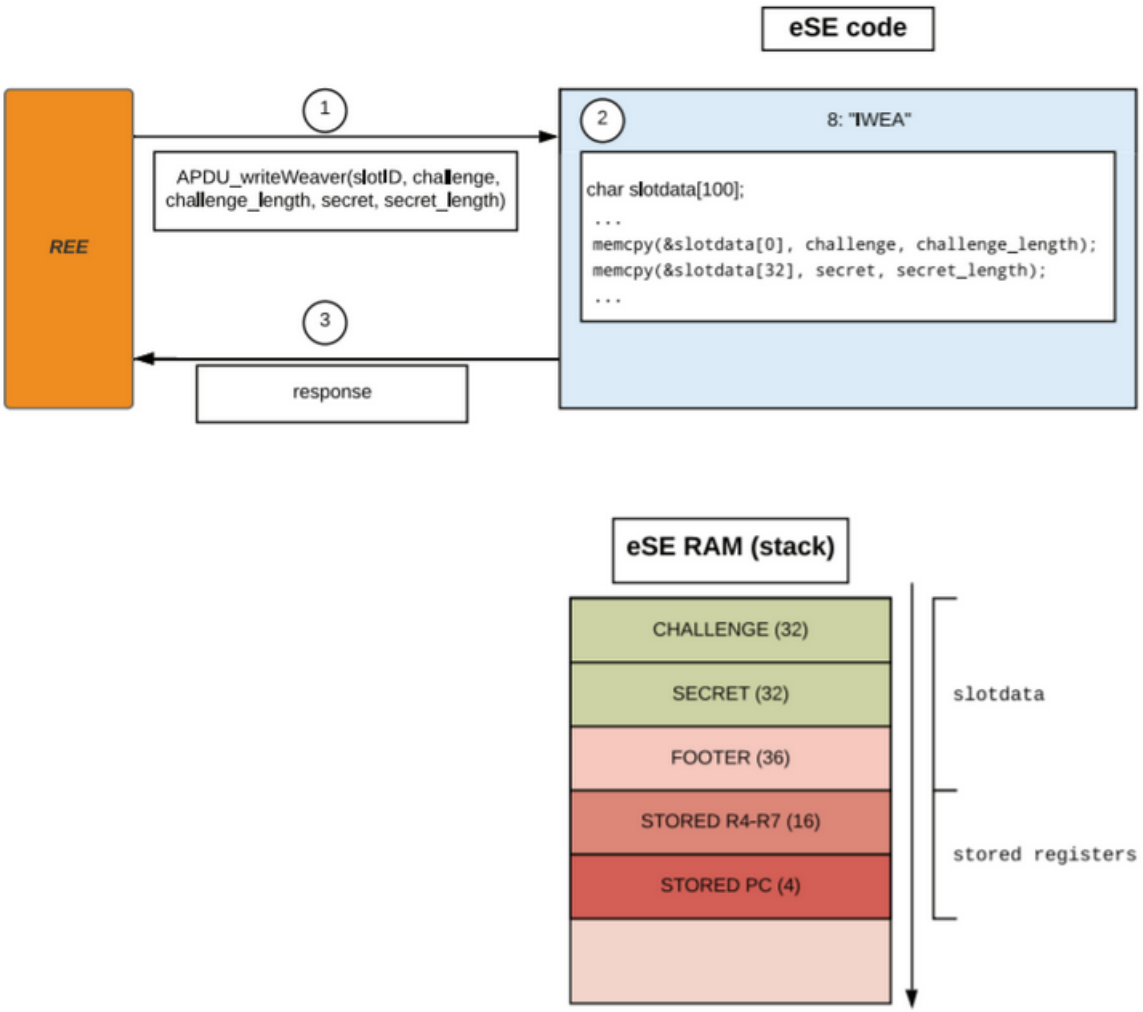

Figure 7.2: eSE attack with aid of APDU write weaver

```
MOVS    R0, #0x10   ; size to read
STR     R7, [R4]    ; Store address
STR     R0, [R4,#4] ; Store size
MOVS    R0, #0x90   ; SW1
STRB    R0, [R4,#8] ; Store SW1
MOV     R0, R5      ; SW2
STRB    R5, [R4,#9] ; Store SW2
POP     {R1-R7,PC}  ; pop and return
```

Figure 7.3: Display of how the ROP gadget can be used to read data from the flash and RAM of the eSE by setting the R7 addresses to the desired address

It is demonstrated in Figure7.3 how this gadget can be used to read data from the flash and RAM of the eSE. By setting the R7 address to the desired location, the gadget reads the 16 bytes it is prompted with. This gadget can be iteratively used to read the entire flash and RAM storage of the eSE, ultimately revealing the full layout of the eSE storage.

Arbitrary code execution. The "APDU writeWeaver" function used to access this gadget can also execute attacker-provided code, as there is no "no execute" protection over the stack memory for this handler. These functionalities, when acquired by an "outsider," could be used to gain full control of the eSE hardware, providing both read and write access.

Persistence. The code images are poorly protected, relying on CRC32 and SHA256 hash verification for security. However, these hashes are stored in the flash memory as well. This means that by gaining read and write access to the system, these hashes can be modified to match and verify arbitrary hashed code. Additionally, the researchers were able to modify the BOOT image without any signature verification, resulting in a complete breach of the chain of trust on the eSE.

With the full capability to dump 100% of the storage and read all the secure storage information on the eSE, the researchers were able to reverse engineer the boot images entirely. These images contained an embedded eSE AES key and initialization vector used for encryption within the eSE. Any attacker with knowledge of these keys can decrypt all past and future firmware update messages for the eSE.

This compromise of the eSE is not easily reparable. Since the attacker can decode any future firmware updates, they also have access to any new keys sent to the compromised system, rendering the system completely vulnerable and irreparable. This research also opens the door to other side-channel attacks and the creative use of logical interface access possibilities.

7.2.1 Attack Discussion

The attack performed by the researchers fully compromises the eSE, effectively nullifying its purpose. Alendal states that this attack was possible due to the lack of privilege separation and the absence of security measures such as stack canaries and "No Execution" (NX) logic for different processes running on the eSE. He further explains that this attack also causes malfunctions in other security features provided by the Android system.

Alendal demonstrates that even if a seized device is in the Before First Unlock (BFU) state, user credentials can be retrieved through post-attack access, allowing entry to the Credential Encrypted file system, which stores sensitive user data. The handler (IWEA) used in previous sections is intended to have high authority, serving as the communication link between the security keys and the Android system. Since this handler was compromised and its purpose misused, the stored keys accessible by this handler were also compromised.

Alendal states that user credentials are transformed, verified, and stored within the eSE system. By bypassing the verification step, the researchers managed to perform a brute force approach for the remaining steps, retrieving everything needed to forge the user credentials' password.

This attack demonstrates its power, showing that it can target seized devices even in the BFU state or when completely powered off, granting full access to all storage on a seized mobile device.

This research underscores how both new and old attack approaches can still be effective against modern security measures. Starting with no knowledge of the target system and gathering information through oracle leakages can yield data that should not be accessible. This highlights the importance of considering that information not supposed to be known to non-vendor users may indeed be accessible when implementing security measures.

This attack violates all security goals of the secure element, compromising its integrity, confidentiality, and availability. The attack is relatively easy to perform and has a broad attack surface, considering it can be executed wirelessly. A well-implemented attack could be difficult to detect, making it hard to ascertain if the firmware of the secure element has been tampered with. The only way to fix already shipped devices would be to implement a new secure element, which is deemed unreasonable.

7.3 Conclusion

This attack was performed to demonstrate a methodological approach for developing an attack using publicly available documentation and analysis. The attack enables the execution of shell code and can be carried out with very limited resources, which is significant as it allows the attacker to read and write from the system storage.

Therefore, securing the system against even the slightest possibility of a buffer overflow error is crucial to eliminate attack surfaces for the attacker.

This research highlights "the gap between intended security and achieved security," illustrating that the certificate used to prove the solution's security only indicates that some "unidentified amount of effort" was made to secure the device, but does not guarantee that the device is free from errors.

Bibliography

[1] G. Alendal, *Digital Forensic Acquisition of mobile phones in the Era of Mandatory Security.* Gjøvik: NTNU, 2022.

[2] R. M. A. P. Eric M. Hutchins, Michael J. Cloppert, *Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains*, vol 1 ed. Lockheed Martin Corporation: Lockheed Martin Corporation, 2011.

[3] R. Anderson, *Cryptography and Competition Policy – Issues with 'Trusted Computing'.* Cambridge University, 2003.

[4] A. Bittau, A. Belay, A. Mashtizadeh, D. Mazières, and D. Boneh, "Hacking blind," in *2014 IEEE Symposium on Security and Privacy*, 2014, pp. 227–242.