SYSTEM LEVEL SECURITY TESTING ON EMBEDDED HARDWARE AND FIRMWARE: A STUDY AND ANALYSIS

Asmita Asmita, Banafsheh Saber Latibari
aasmita@ucdavis.edu, bsaberlatibari@ucdavis.edu

September 23, 2023

Abstract

Our daily life is full of interaction with Embedded Systems (ES). The current growth of application of embedded systems in different domains, especially, their usage in military domain attracts a lot of attention towards their security issues. Embedded systems are also the core part of internet of things (IoT) ecosystem. IoT devices are everywhere, and so is its security threat. In this study, we did a detailed survey of security issues related to embedded systems, targeting firmware attacks and hardware attacks. This work presents a brief survey on these two security issues of the embedded devices, followed by analysis and a brief discussion on future scope. The main aim is to draw the attention of readers towards the importance of embedded security by providing examples of impacts, research, challenges and ideas emphasizing the need of the hour to contribute towards embedded security research.

1 Introduction

Embedded devices are the main part of the computation systems [1]. They have been the main target of the attackers recently especially because of the immense growth of IoT devices, of which, embedded system is a core part. In case of resource constrained embedded devices, generally, security is neglected during development. Hence, they become an easy target for attackers causing significant impacts. Though, the secure designing of generic software has been strengthen a lot but when it comes to embedded devices, its security testing is far behind. Even during the testing phase of these embedded devices, testing is mainly related to functional and quality testing but not security. Some companies perform penetration testing on the device after production, but security is rarely taken care during the development stage of embedded devices. Various applications of embedded system and increase in their security issues motivated us to do a study in this domain. This paper aims to provide readers an overview of embedded firmware and hardware attacks, impact, existing tools and research in terms of embedded security assessment. It also provides a brief discussion on the requirement of future works and challenges involved; so that readers could get to know the significance of research in this domain and would inspire them to contribute towards security of embedded systems.

In section 2 we talk about background and motivation. Section 3 gives some example on embedded firmware and hardware attacks. In 4 we talk about the available tools for the firmware and hardware security testing and assessment. In section 5 we discuss some of the new research direction and topics in the both hardware and firmware security assessment. Sections 6 and 7 talk about the existing challenges, analysis and future work respectively. Finally, section 8 concludes the report.

2 Motivation & Background

This section provides readers, the motivation behind this study, followed by the relevant background information related to embedded systems and corresponding details related to firmware and hardware.

As discussed in the introduction, with the increase in IoT devices, attacks on embedded systems have grown by leaps and bounds as embedded systems are core part of IoT ecosystem. Apart from that, embedded systems are core of various sectors including automotive, power grids, nuclear, space, medical, consumer, industrial, etc. With the increase in the security of generic software within the organizations, attackers are targeting these embedded devices to get into the system and create massive impacts that have been discussed in section 3. Generally, when attackers are outside the supply chain, they get access to the deployed system and its corresponding components. Hence, exploiting system level vulnerabilities become a low hanging target for the attackers.

Though, there are a lot of testing tools used within the development and manufacturing stage. These tools are mostly for quality and functionality testing but not very robust towards security testing. The increasing number of embedded attacks motivated us to perform an in-depth study of existing scenarios when it comes to testing of embedded systems just before the deployment stage, i.e. at the system level after the production when the device is ready to be sent to consumers. There are already certain survey papers that have been conducted prior. For example, in [2] authors have provided detailed survey and taxonomy of all the tools for static and dynamic analysis of firmware. It also discusses in depth about the challenges involved, various analysis approaches, along with detailed and comparative discussion of all the tools. In [3], authors have provided a detail survey regarding firmware attacks during its update. Authors have discussed in detail, existing firmware update methods, challenges, tools and techniques for testing to enhance the security. In [4], authors have provided details about the tools and works for firmware vulnerability detection along with proposed method for authentication bypass detection. In [5], detailed survey about fuzzing techniques, existing tools and challenges have been provided. These survey, provide the details regarding different aspects of embedded security. Inspired by these, we performed this detailed study to bring a broader view in this paper where we discuss about both hardware and the firmware attacks, tools, techniques, researches and future scope at one place that could help reader to connect things and get big picture visualization of importance of embedded security and hence create awareness that significant contribution is further required for robust security testing of embedded devices.

2.1 Embedded Systems

In this section we briefly explain embedded systems and their features. An embedded system is similar to a computer system that is programmed with a specific task. Because of their application these system have some specific features. Depending on the application, they should be real time. Moreover these systems should be low power and also compact. There are different types of embedded systems including, stand-alone, networked, real-time, and mobile. They are present everywhere, for example, washing machines, HVAC, refrigerators, microwave, fitness tracker, medical instruments, industrial instruments, automotive, all smart IoT devices, etc. Embedded devices have two main parts hardware and firmware. Ascher Opler coined the term firmware in 1967. As per Wikipedia, it's a type of software that gives the low-level control for a device's specific hardware. It's stored in non-volatile memory like ROM, Flash. It could be either OS based, non-os based i.e, bare-metal, or custom OS/ RTOS based. They are the core logic that control the whole embedded systems. Hardware is specific to the kind of application for which the embedded system is designed for. Apart from micro-controllers and microprocessors, sensors, actuators, peripherals, debug points, external memory, input/output, etc are core parts of embedded systems. We will further give brief info regarding security aspects of embedded hardware and firmware.

2.2 Firmware Attacks

Firmware is the main logic of the device. Sometimes, it's proprietary to specific vendors. Having access to firmware, provides attacker plethora of ways to exploit the existing firmware vulnerabilities in the device. There are different ways to extract the firmware, one is via hardware attack via debug ports that would be discussed in the hardware section. Other ways include downloading the firmware if available on the vendor's website, sniffing it during firmware updates and extracting it via unauthorized remote access to the device due to implementation vulnerability/ unauthorized access via opened network ports in the device. If the firmware is encrypted properly with secure management of keys, it makes the attack harder for the attacker.

Once the access to firmware is achieved, attacker hunts for the vulnerability in the device via firmware reverse engineering and analysis. Figure 1 shows the attack surface for firmware, that's generally targeted for exploitation. There are two types of firmware analysis, static analysis and dynamic analysis. Static analysis does not require the firmware to be in running state, attacker statically reverse and analyse it to find info about the device like architecture, file system, internal details to hunt for vulnerabilities like presence of hard-coded credentials, insecure functions, vulnerable software components, exploitable services and overall system details. The information/vulnerability retrieved from static analysis is used during dynamic analysis to exploit the existing vulnerability and disrupt system behavior. Dynamic analysis requires the firmware to be in running state. The crux in this scenario is that attackers cannot always have the access to the physical device to perform dynamic analysis. Despite of this obstacle, there are different existing emulation techniques that would be discussed in the later section of this paper to emulate the device and run firmware even in the absence of actual hardware to perform dynamic analysis. Attacks like command injection [6], remote code execution (RCE) [7], buffer overflow [8], returned-oriented programming (ROP) [9] exploits the system implementation vulnerabilities. Figure 2 provides the common attack types in embedded firmware. Dynamic analysis confirms the exploitation probability of such vulnerabilities that paves way for the actual

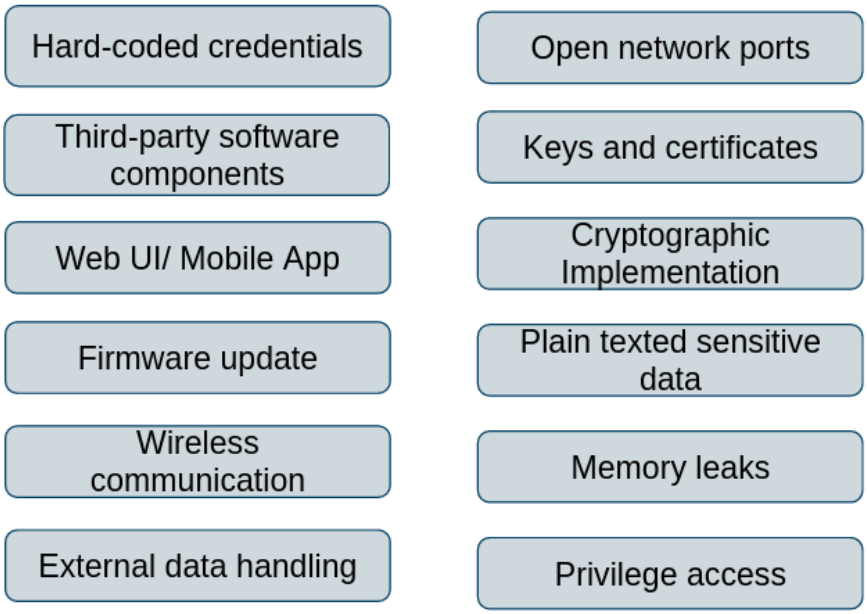

Figure 1: Common attack vectors for firmware attack

attack execution on the device. Though such attacks have been more common in web, generic software based systems. But, they become the low hanging fruits when it comes to embedded firmware. Due to resource constraint and negligence during development like improper input sanitization, insecure memory management, insecure function, third party vulnerable software components, insecure updates, hard-coded sensitive data and similar logical and implementation vulnerabilities present in firmware make it prone to such attacks.

2.3 Hardware Attacks

Hardware attacks can happen during different steps of a device life cycle [10]. The security threat can be the result of the design flaw, system side effects and design modification. Understanding different type of hardware security threats and being aware of them can help us to find various types of countermeasures to avoid them. In this section we will explain different types of the hardware attacks on embedded devices.

2.3.1 The DRAM Attack or Rowhammer Vulnerability

These types of attacks are targeting private data in dynamic random access memory (DRAM) [11,12]. They are based on the theory that after powering off the memory, DRAM data persists for a short amount of time. By cooling off the DRAM and reducing the current, this time can be extended. So, the attacker can move the DRAM to another system and then read the data. Rowhammer based attacks are using the fact that the DRAM data can be altered by accessing nearby data. In this method the attacker can alter the data maliciously by putting his data in nearby DRAM cell and changing it.

2.3.2 Cache Attack

In this type of attack [13] the attacker exploits cache information leakage. Cache is shared among different processes and every process data will remain inside the cache after execution for a while. The attacker can extract the data access information and access to confidential data.

2.3.3 Side Channel Attacks

In hardware attack there is possibility of different types of side-channel attacks like timing, power and electromagnetic and photonic. This section covers this type of attacks.

- **Timing Channel-** In this type of attacks the attacker observes the execution time of the process. In other word they use the small differences in execution time to extract information. This type of attack is caused by sharing hardware resources between different software [13].

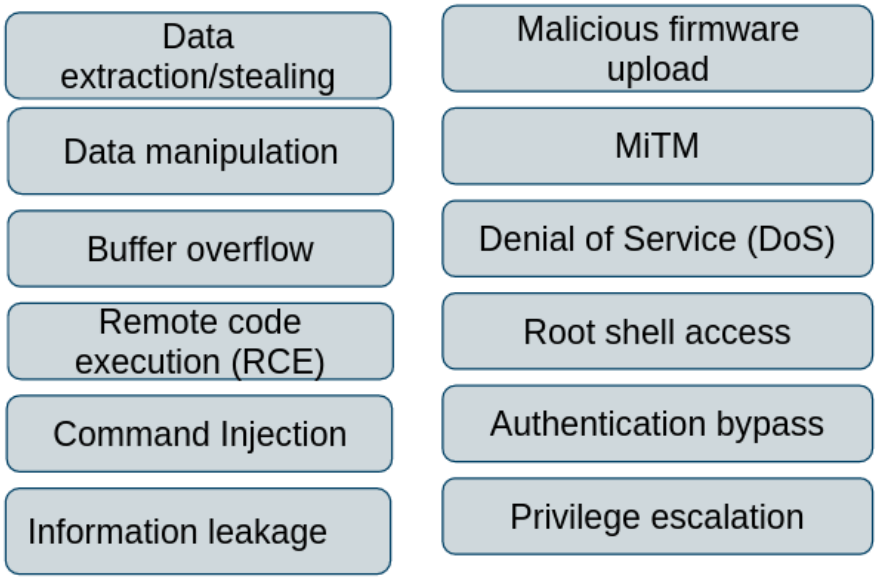

Figure 2: Attacks on embedded firmware

- **Power Channel-** In this type of attacks [14,15] the attacker tries to extract information by measuring the switching power traces of the embedded system components and then he finds the critical data by using mathematical analysis on the extracted information. These types of attacks are based on the fact that transit power traces of a chip shows its internal switching pattern. One of the common techniques that attackers are using to do power attack is simple power analysis (SPA) [14]. This type of attack is possible when the attacker is familiar to the internal structure of the design. Differential power analysis (DPA) [14] is one of the common techniques that are used by the attackers. In this attack the attacker measures the power consumption traces of the victim device during several time steps by applying several inputs. Then he divides the resulting power traces into two groups. Finally he calculates the differences between two subsets and the results show the possibility of the information leakage in the design.
- **Electromagnetic and Photonic Channel-** Unintentional electromagnetic (EM) radiation from electronic devices [16] leaks information of the critical devices. In these works [17,18] they show practically how to measure the EM information leakage.

2.3.4 Fault Injection

In a fault attack or fault injection method, the attacker disrupts the normal execution of the design by intentionally injecting faults. By doing this type of attack the attackers can access the secret information of the design like key bits [19]. This attack is a severe threat to systems. In [20] the attacker flips the processor status flags and bypasses the authentication process. Fault attacks are mainly used to attack the crypto-system but it is not limited to them.

2.3.5 Hardware Trojan

Hardware Trojan (HT) is a malicious modification of the design and can be classified by the following characteristics physical activation, and action. The first feature describes the architecture, the structure, and the placement of the Trojan design. The second feature is described by the activation characteristics, which comprise external, internal, or physical options or a combination [21].

2.3.6 Attack Surface

Attackers can use different hardware parts of an embedded device to extract information. Figure 3 shows the attack surface.

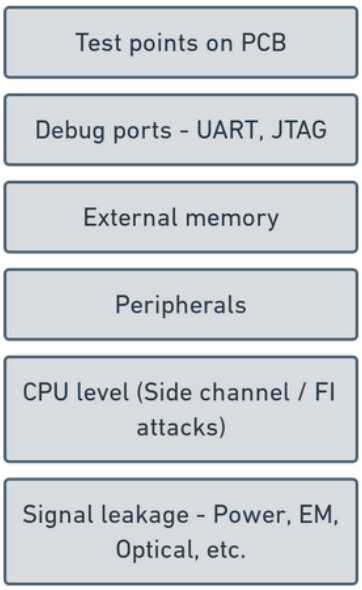

Figure 3: Hardware attack surface.

3 Attacks on Embedded Devices

Hardware and firmware are the core of embedded devices. Attacks targeting these cores could have major impact on the overall system. In this section, examples of embedded hardware and firmware attacks have been presented demonstrating the significant impacts. Having the knowledge about these attacks also helps readers to understand the kind of attacks that can take place and the reasons behind them.

3.1 Firmware Attacks

The background section covered various types of firmware attacks. This section would cover some practical examples of firmware attacks that could help the readers to visualize the impact in the real world scenario. There have been numerous command injection and buffer overflow based exploitation on firmwares that could be referred at National Vulnerability Database (NVD) website [22]. Many embedded systems come with user interface via web or mobile applications to exchange data to and from the external world. These are often targeted by attackers to execute their exploitation. Embedded exploitation case study by Ang Cui, Michael Costello and Salvatore J. Stolfo [23] provides a detailed analysis on the exploitation of Laser jet printer firmware update to inject malware by exploiting the vulnerable functionality. In 80.4% cases, authors identified third-party libraries with known vulnerabilities in the analyzed firmwares. A recent bug finding in Media Tek chips that has globally affected 37% of all IoT devices and smartphones [24] exploited heap based buffer overflow in the audio DSP component to perform privilege escalation. It could allow attacker to execute malicious payload inside DSP firmware that could be exploited to eavesdrop on user, as DSP has access to audio data flow. It occurred because of out of bounds write [25] due to an incorrect bounds check. A remote code execution impacted many modern Netgear Small Offices/Home Offices (SOHO) devices [26]. It was the vulnerability in the third party component Circle used in many Netgear devices for parental control services. Circle runs the daemon by default even when the parental control services are not enabled. The daemon updates the circled daemon and the database but the database update from Netgear are in plain text and unsigned allowing attacker to perform man in the middle attack and manipulate the update with malicious payload execution that can also be exploited to enter organization network. As provided in detail on NVD website, it's associated CVE-2021-40847 [27] have Common Vulnerability Scoring System (CVSS) score [28] of 8.1. Such attacks have major impact on SOHO devices that have proliferated significantly to connect to corporate networks. Another example of RCE attack was on surveillance cameras from 70 different vendors [29] because of vulnerable implementation of HTTP server.It impacted huge number of products that used same implementation of HTTP server providing attacker the unauthorized access to the devices. There have been numerous similar vulnerability findings in embedded firmware that make them prone to exploitation.

Tesla hack, "TBONE – A zero-click exploit for Tesla MCUs" [30] found the privilege escalation exploit to upload a new WiFi firmware, that can turn it into an access point that could be used to exploit other Tesla cars in the proximity , making it wormable. It exploited the vulnerability founded in the ConnMan

daemon to gain remote code execution. Ripple20 [31] is a series of 19 zero day vulnerabilities found in a third party TCP/IP library. This vulnerable library exists in various IoT and embedded devices, including medical, home, enterprise, networking, retail, aviation, power grids, etc. impacting huge number of devices. These vulnerabilities include RCE, information leak, denial of service (DoS), out of bound writes, improper handling of length parameter, etc. Most of these embedded devices in IoT ecosystem uses wireless protocols like bluetooth, zigbee, WiFi, etc. In order to implement these protocol stack, most of the time already built in libraries/SDKs (Software development kits) are used. The vulnerability in these libraries/SDKs could impact large number of devices simultaneously. Sweyntooth [32] research discovered a collection of bluetooth low energy (BLE) vulnerabilities across the various BLE software development kits (SDKs). It exposed the BLE implementation flaws. These vulnerabilities include crashes, deadlock, buffer overflows, security bypass, etc. Another attack on Philips Hue smart bulbs [33] exploited the heap-based overflow in the implementation of the zigbee wireless protocol. It resulted in remote code execution that sent remote command to bulb via zigbee, compromising the complete target network. Mirai botnet attack [34] exploited the insecure IoT devices with open Telnet ports and weak/default credentials to build a botnet by pushing malware onto them. It impacted the website of renowned security journalist Brian Krebs. It is one of the largest distributed denial of service attacks (DDoS) to date. It caused massive internet outage at the US east coast. Tesla keyless entry fob attack [35] exploited the over the air (OTA) firmware update implementation. The implementation lacked code signing and verification in combination with other vulnerabilities. That could enable attacker to rewrite the firmware of a key fob via Bluetooth connection, take full control of it, and use it to steal a Tesla Model X. A detailed analysis regarding it has been provided by the team at Synopsis [36]. Thus such attacks exploiting the vulnerabilities in embedded firmware could have significant impact. In a March 2021 report, Microsoft reported that only 30% of 1000 interviewed businesses allocated funds for firmware protection in their budgets despite the fact that 80% of them experienced at least one firmware attack over the past two years [37].

Hence, the firmware attacks exploits different loopholes present in the system that could be summarized as:

- Weak and easy to guess credentials
- Hard-coded sensitive information and credentials in plain text
- Insecure open network ports
- Insecure function usage
- Wireless data transfer in plain text
- Vulnerable third party software components, libraries and SDKs
- Improper input sanitization and bound checks
- Improper memory management
- Lack of secure coding and best practises
- Insecure firmware update
- Lack of proper signature and authentication verification
- Vulnerable implementation of cryptographic algorithms, key management and wireless protocols

The list highlights some of the targeted points that attackers tend to exploit. The examples of firmware attacks presented was aimed to provide readers a brief overview of the types of attacks, the reason behind them and their implications and to make them aware regarding such attack possibilities and firmware vulnerabilities. In next section, we would cover the hardware related attacks on embedded devices followed by existing tools and methods to perform testing with respect to such vulnerabilities to mitigate such attacks.

3.2 Hardware Attacks

In this section, we explain some of the practical hardware attacks on embedded devices. [38] presents evidence of hardware Trojan. The compromised microprocessor in the Syrian Radar system enabled Israeli jets to bomb suspected nuclear unhampered. The inserted "kill switch" in a microprocessor used by french defense contractors enables the french to disable military equipment that falls into enemy hands. Another example of a hardware attack is Cisco routers imported from china. Counterfeit Cisco routers may cause unexpected failures in American networks [39]. [40] Shows that power analysis attacks can extract secret information

from smart cards. Smart cards have a lot of applications like banking, mobile, and electronic signature. In all of these devices, power analysis is an approach for an attack.

Autonomous vehicles are very popular these days and are the main focus of a lot of research groups. Tesla uses the autopilot system that enables autonomous cruises on freeways. Although the autopilot system has a lot of applications, it could be a source of the attack. In this work, they show that by doing a cache side-channel attack the location of the vehicle can be detected [41]. [42] summarizes attacks on Intel SGX. They divide the attacks into 7 categories. A cache attack is one of the hardware attacks that is performed on SGX. In this work, [43] they reverse-engineered Apple's BootROM and they design a toolkit to do hardware security experiments on Apple's SOC. By using the designed toolkit they do a cache timing attack on Apple A10 Fusion SoC. This attack reduces the security of the OpenSSL AES-128. This paper [44] shows possibility of fault attack on AMD GPU. Using Fault attack they can recover AES keys in minutes. The attack uses voltage-frequency scaling features of commercial GPUs. This paper [45] is a survey on the possible hardware attacks on mobile devices.

4 Existing tools and frameworks

The attack scenarios presented so far with respect to embedded firmware and hardware illustrate the significant need of robust tools and techniques to identify such vulnerabilities before it can be exploited by the attackers. Though, the development life cycle of product comprise of testing stage, but most of the time the testing stage focuses mostly on functional and quality testing not security specific. After the product development and before deploying it in market, some vendors go for the product penetration testing. Such testing is done at the system level considering the attackers point of view. Moreover, it's highly recommended to include security since the initial stage of the product lifecycle but that's not always followed by the vendors because of time and resource constraints. In order to perform such testing, several tools and frameworks exist in the market to assist security researchers and penetration testers to perform security analysis. In this section, such existing tools and frameworks will be covered.

4.1 Firmware

Testing of embedded firmware can be done both at development stage and pre-deployment stage. There are certain existing tools that performs source code review complaint with secure code guidelines. A list of source code security analyzer tools [46] have been provided by NIST to identify software vulnerabilities. Similarly secure source code analysis tool list has been provided by OWASP [47]. These tools support various languages and designed for generic softwares. Embedded firmwares are mainly written in C/C++ for architectures including ARM, MIPS, RISC V comprising of bare-metal, Linux operating system (OS) or other real time operating systems (RTOS). The existing source code analyzers are not sufficient to identify major vulnerabilities. Another important factor is that after the source code is compiled, a lot of changes occur at the binary level, hence testing overall system firmware at binary level is equally important. Also, in case when vendors use third party libraries, they don't have the access to the source code just binary, it requires the need for proper firmware binary analysis. Apart from that, there are existing firmwares that have been developed long before without proper maintenance of source code. In order to test the security of such firmwares, binary security analysis is required. Moreover, considering practical attack scenarios, attackers get the access to the firmware binary, not the source code. Hence security testing at binary level is more closer to attackers approach. Below, we will cover some of the existing methods, tools and frameworks used for firmware binary analysis.

As explained before in the Firmware Attack section, there are two types of binary analysis, static and dynamic. In order to perform testing on the binary, the analyzer needs to test considering both static and the dynamic parts. Some of the tools used for static and dynamic testing would be discussed here. Binwalk [48] is one of the most famous tools used for firmware extraction, and analysis. It can be used to test if the firmware binary can be extracted. Firmware extraction is the fundamental step that is required to perform any attack. If the developed binary can be easily extracted, it becomes an easy target for the attacker. Hence, testing using this tool would help analyzer to identify this weakness. This tool cannot be used in case of non-OS based firmware. After extraction, static analysis is performed to traverse through the extracted file system to identify any sensitive files, credentials, configurations, keys and certificates. Firmwalker [49] is a bash script based tool that can search the extracted file system to automatically identify sensitive information and files to accelerate static analysis. The extracted file system contains lot of ELF (Executable and linkable format) binaries that are further analyzed using disassembler and decompiler tools. Ghidra [50], IDA Pro [51], Radare [52] are some the famous tools for binary reverse engineering. Even in case of bare metal and RTOS based binaries where Binwalk is not useful, these tools are used to perform binary analysis and find implementation related vulnerabilities like memory leak, insecure functions, possibility of

buffer overflow and RCE based attacks, hard-coded credentials, etc. There are tools that assist in hunting for existing Common Vulnerability Enumeration (CVEs) and Common Weakness Enumerations (CWEs). Cve-bin-tool [53], cwe-checker [54], and cve-search [55] are some examples of such tools. The tools for static analysis discussed so far have individual unique features that is performed separately. There are some tools that are combination of all these features at one platform where as an analyzer, one just needs to upload the binary, those tools does everything in the back-end and provide all the result on a single platform, user is not required to worry about different tools for different features. Some examples of such tools are : Binare.io [56], EXPLIoT firmware auditor [57], FACT tool [58], Emba [59]. Out of these, FACT and Emba are open source. Though, these tools are useful for static analysis of firmware but each one of them possess certain limitations including improper support for non-OS based firmwares; inaccuracy, do not support dynamic analysis, complicated back-end installation processes, and similar other limitations that researchers are trying to solve using machine learning. More about it would be discussed in 5 section. Now, some tools related to dynamic analysis would be discussed.

Emulation and fuzzing are two important aspects of dynamic analysis. Though, the dynamic analysis can be performed directly on the physical device. But performing dynamic analysis on actual device possess the risk of bricking/damaging it. Also, testing has to be delayed until the device production is ready. Having emulated environment for testing provides a lot of flexibility. There are some existing tools to perform binary emulation and assist in dynamic analysis. Qemu [60] is an open source emulator and a virtualizer. It supports both user mode and system emulation. Unicorn [61] framework is build on top of qemu but it focuses particularly on CPU operations at the instruction level. Apart from that it supports various other features including dynamic instrumentation, flexibility, thread-safe, etc. Qiling [62] is another binary emulation framework built on top of Unicorn. Apart from just emulation, it supports other features including reverse engineering, binary instrumentation, dynamic on-the-fly patching, supporting applications dependant on OS, etc. There are several other similar tools apart from the one discussed in this subsection to assist for firmware dynamic and static analysis. There are certain limitations that still persist including hardware limitations during emulation, code coverage during analysis, proper method for inclusion of symbolic execution, and many others. In the next subsection, the exiting tools and frameworks with respect to hardware security assessment would be discussed followed by the ongoing and existing research to solve some of the existing limitations.

4.2 Hardware

Researchers tried to propose hardware frameworks to attack or check the security of the designs. In [63] they propose a common reverse engineering scoring system (CRESS) that helps to conduct the analysis of the hardware attack. Users can use the proposed web tool to rate and examine different attack scenarios. In [64] they propose a framework with the ability to inject multiple faults. The framework takes the circuit implementation of the cipher and fault model as inputs. [65] proposes a framework to automate Trojan insertion. The framework receives the design netlist as input. In [66] they design a platform to perform attack and also they proposed a new method to perform fault attack. Chip Whisperer is an open-source platform for side-channel power analysis.

There is also possibility of attack in gate level. There are a lot of research on this topic also. Here we mention two important attack tools in gate level. This [67] work for the first time, propose an algorithm based on satisfiability checking (SAT). Using the proposed tool the attacker can decrypt an encrypted design netlist. The golden circuit and the obfuscated circuit netlist are the inputs to the tool. The output of the attack would be the key bits. The work in [68] proposes RANE that is an open source CAD based tool for attacking the secured design. RANE uses two approaches SAT and formal verification to attack a design. So it has the ability to receive input also in HDL format.

5 Ongoing research

This section discusses about some of the ongoing researches to improve the security testing tools and techniques for embedded hardware and firmware. This section provides reader, understanding about the latest approaches being taken to solve some of the existing challenges.

5.1 Firmware

This subsection talks about the ongoing research with respect to firmware testing tools and methods. The work, "Toward the Analysis of Embedded Firmware through Automated Re-hosting" [69] tries to overcome the hardware related limitations during emulation for dynamic analysis. In this work authors have proposed an automated re-hosting platform called, "PRETENDER". In this case, at first the interaction between the

original hardware and the firmware is first observed. Then, the observations are used to automatically create the models of hardware peripherals using machine learning engine. These models replaces the hardware requirements with software only components making it possible for firmware emulation and analysis without any further hardware dependencies. The condition in this case is that, at the initial stage the testing team should have access to original hardware and firmware to capture the interaction observations. Though, considering manufacturers have hardware, they could just use one to perform the initial stage, later once the observations are captured, they could perform any number of dynamic testing without putting at risk other hardwares during testing. But, more better research is required in the scenario when actual hardware and the firmware is not available during testing. As discussed in the 4 section, there are tools for emulation and dynamic analysis even incorporating symbolic execution for more code coverage but all are limited by hardware constraints. One of the research works on FirmAE [70] tries to solve this problem using heuristic search and fixing the hardware related issues at the run-time but it's designed targeting only emulation and analysis of embedded web interfaces.

Apart from that, fuzzing is an important aspect of dynamic analysis, ongoing research work to improve fuzzing techniques include EM-fuzz [71] and FIRMCORN [72]. EM-fuzz is designed specifically to identify memory related vulnerabilities by performing fuzzing. It integrates fuzzing with real time memory checking rather than traditional code coverage approach. The integration of real time memory checking has significantly improved the performance of existing fuzzers to test for memory related vulnerabilities. FIRMCORN is a vulnerability-oriented fuzzing tool, targets for more faster, and accurate emulation and fuzzing along with tries to resolve the hardware constrains during emulation. It uses heuristic algorithm to skip unnecessary function during fuzzing and hook hardware dependency for fuzzing stability and possess a vulnerable code search algorithm for firmware vulnerability identification. Researchers of FIRMCORN have also mentioned the challenges that are yet to be resolved. Hence, designing a robust framework for emulating and performing the dynamic firmware testing needs a lot more research work. In terms of improving the inaccuracy in existing tools for both static and dynamic analysis, incorporation of machine learning is being looked forward for better results as covered in research works [73] [74] [75].

Another research work, ARMORY [76], tries to assist the identification of hardware attacks like fault injection (FI) via software method. Identification of FI based implementation vulnerability is different from other software based vulnerability. As discussed in the "Hardware attack" section, FI intends to change the normal flow of program. It could either corrupt data, skip or modify instruction, or bypass any security checks. Hence, its identification is highly dependant on specific architecture. Hence, in this case the generic firmware analysis tools cannot be applied. ARMORY is designed specific for exhaustive fault simulation on binaries of the ARM-M class. It is based on M-ulator simulator to simulate combinations of various custom fault models on a given binary. Hence, there are several other research works that are trying to solve the security issues in embedded system by providing robust and feasible automated testing tools with static and/or dynamic testing capabilities.

5.2 Hardware

In this subsection, some of the recent hardware security research on embedded systems will be discussed. This work [77] presents the hardware-based voltage glitching attack against a fully-fledged Intel CPU for the first time. VoltPillager injects messages on the Serial Voltage Identification bus between the CPU and the voltage regulator on the motherboard that allows the attacker to control CPU voltage. VoltPillager has two main components: 1) firmware of the Teensy 4.0. 2)PC-side software that controls the device. This work [78] shows that GPUs are also vulnerable to side-channel attacks. The security of GPUs is important since they are processing sensitive data. The applications that are using a GPU at the same time share a lot of resources so this way there would be a possibility of a side-channel attack. In this work, they reversed engineered the Nvidia GPU scheduling and characterized the colocation opportunities for concurrent application. They demonstrate some side-channel attack models that cover both graphics and computational stack. The first attack model is a Graphics spy attacking a graphics victim in this attack the attacker exploits the graphic stack using OpenGL or WebGL. The second attack is a CUDA spy attacking a CUDA victim and the third attack is a cross-stack attack that is a CUDA spy attacking a graphics victim, or vice versa. In the last two attacks, the GPU is accessible to the attacker using CUDA or OpenCL.

In this paper, [79] they focus on the security issues in smartphones. They propose "BetaLogger" which is an android application to show that there is the possibility of information leakage through smartphone hardware sensors. The proposed application tries to infer the typed text on a smartphone keyboard using language modeling. BetaLogger has two main phases, in the first phase the text vectors are given to the ML model for label prediction and in the second phase, the sequence generator module generates the output sentences. In [80] they propose a framework for the security of IoT devices. The proposed framework tries to gather side-channel information like timing. To create a fingerprint for the IoT device they gathered

execution time under normal and attack scenarios. Their idea is based on the fact that in an attack situation the execution time would be more because resources like memory, CPU, and the I/O will be used more. The proposed approach is composed of two main parts. The first part is data collection that involves the required hardware setup for the side channel leakage data collection. The second part is the machine learning algorithm. In the first part, the hardware setup emulates the IoT device and it considers its design, storage, and computational resources. They did multiple attacks using "Kali Linux" on the Raspberry Pi when it runs Amazon's Alexa. Using this setup they collect data into scenarios when the system is working normally and when there is an attack on it. In the second part, at first, they do data cleaning and feature extraction. They labeled data based on the system condition that they collected, "Attack" or "No Attack". Using the collected data they trained a machine learning model that predicts the presence of an attack on the device or not.

The concern of this [81] work is the Static Random Access Memory (SRAM) security issue. Although there are several works on the Trojan designs in logic circuits, trojan insertion in the embedded SRAM arrays is a new hardware security issue. In this work, they propose a new hardware Trojan that escapes from the industry-standard post-manufacturing tests. In this attack scenario, the Trojan is inserted before chip fabrication by an untrusted foundry. They classify SRAM Trojans into two categories. The classification is based on the effective payload defect type. The Trojan type one is Resistive Short/Bridge and the Trojan type two is Resistive Open. In type one, the Trojans can be designed to cause resistive shorts between one circuit node and voltage sources or short between two nodes. In type two, to cause an open circuit they implement variations of one Trojan type that breaks one pull-up path in a cell. The inserted Trojan in the SRAM array can easily corrupt the data and cause functional failure.

6 Challenges

Performing security assessment of embedded systems and designing automated tools for the same involves a lot of challenges. First of all, hardware and firmware of embedded systems are tightly coupled to specific set of application and design. There is no global standard, design diversity depending on application, vendor to vendor variation in protocols and backend framework and implementation, etc make it challenging to design a common platform to test everything. Moreover, different embedded systems use different architectures, hence different instruction sets, that makes challenging for analysis.

Apart from that, the inaccuracy and limitations of existing tools makes it challenging for robust testing. Hardware specification limitation during firmware assessment is another challenge. The diversity in embedded system including hardware types, firmware types, communications, protocols, etc makes it challenging to address all security vulnerabilities. Moreover, dealing with improvement on individual features in itself involve huge time and resource, accompanied with lack of awareness towards embedded security becomes huge obstacle.

Other challenges that we should consider is test time and also cost. The proposed test method should have reasonable test time so the product have logical time to market. Moreover, the cost of test method also limits the ability of the foundry to do the complete testing on some products because the cost of final product depends on the production process so we should find a logical ratio between test and final product cost. Plus, in case of hardware attack like inserting hardware Trojan there is possibility that the attack is not detectable by the foundry like Trojan insertion on embedded memory. So, we need more powerful test tool in case of the hardware.

7 Analysis and future works

The detailed study and survey discussed so far regarding the attacks, existing tools and ongoing researches with respect to embedded firmware and hardware security provides the in-depth view in this domain. This section discusses about a few analysis that can be deduced from the study so far, followed by the discussion of future work scope.

7.1 Firmware

It has been observed that there are certain common attack vectors that attackers target for performing firmware attack. Figure 1 shows the common attack vectors that are usually targeted. Despite of the known attack vectors, the attacks have been repeated over the years. Attacks shown in figure 2 are commonly found in firmware. These attacks could be looked up in detail at CVE and CWE websites. Though, the attack vectors and attack categories are well known but the identification of its presence in the firmware is challenging. After the study of different tools and these attacks, certain common features for testing tool

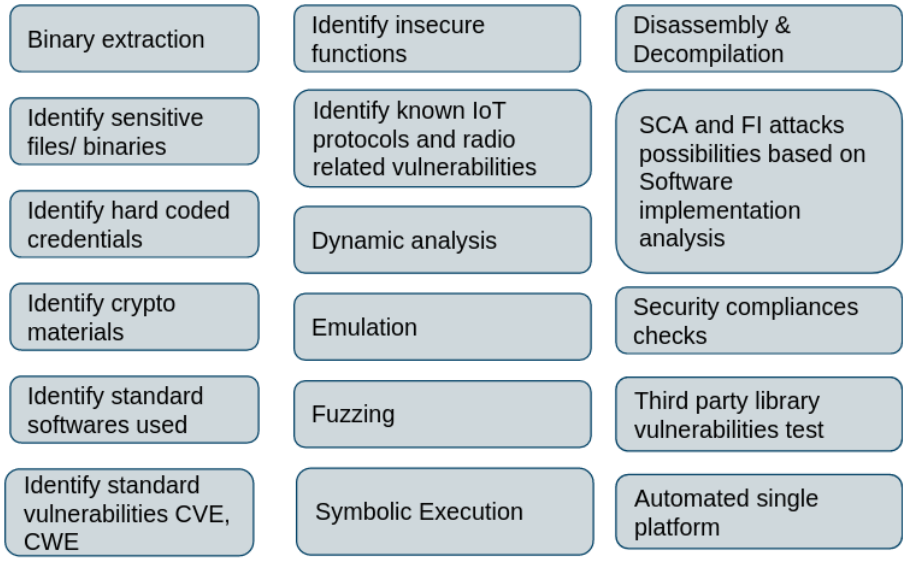

Figure 4: Some feature identification for firmware security testing tool

have been identified as shown in figure 4. As per our study so far there are different tools having these features separately, there is no unified tool or framework that could make it easier for tester to assess both dynamic and static analysis using single framework.

Figure 5 shows the proposed framework future work. This automated framework aim to support all types of firmwares including OS based, non-OS based (bare-metal), and custom OS/RTOS based. Though, the features in all is almost similar but implementation of framework would vary depending on the type of firmware. As discussed before, though there are already existing tools that have some of these features discretely, the proposed method aims to create respective modules with concerned features for each of the three types of firmwares. Later, bring all the modules at one platform, providing testers easy to use and modular platform.

Apart from the unified platform, a lot of contribution and work is required to fix the limitations of existing tools for more accurate, fast and reliable testing. For instance, fixing the current issues of static analysis, emulation, fuzzing, memory vulnerability testing, authentication bypass testing, embedded web/application vulnerability testing, robustness against hardware attack testing, data handling, update and plethora of such individual testings need respective improvement and work in future.

7.2 Hardware

In this section we discuss about our analysis and possible future research directions in the area of embedded system hardware security. Today different research topics are using ML algorithms and methods to solve different problems. Although there are some research work in the hardware security defence mechanisms there are not that much ML based attack mechanisms. So here we discuss some of the ideas in hardware security domain.

- Proposing machine learning based attack methods. By using the leakage or circuit information the model can gives us some prediction about critical information of the design like keys.
- Proposing attack mechanisms that are using reinforcement Learning. In RL based algorithm we train a module called agent to do the a specific task. So, in the case of the hardware attacks in embedded system we can train and agent to do the attack to the circuit instead of a human attacker.
- Proposing attack mechanisms that are using generative models. So, In this approach for example in case of the hardware Trojan we can use generative model to generate the Trojan module for us. As input we give some Trojan inserted circuit to the model and so the model will train and can generate similar circuit.
- Proposing a comprehensive method that can use both the system and gate level vulnerability of the embedded system.

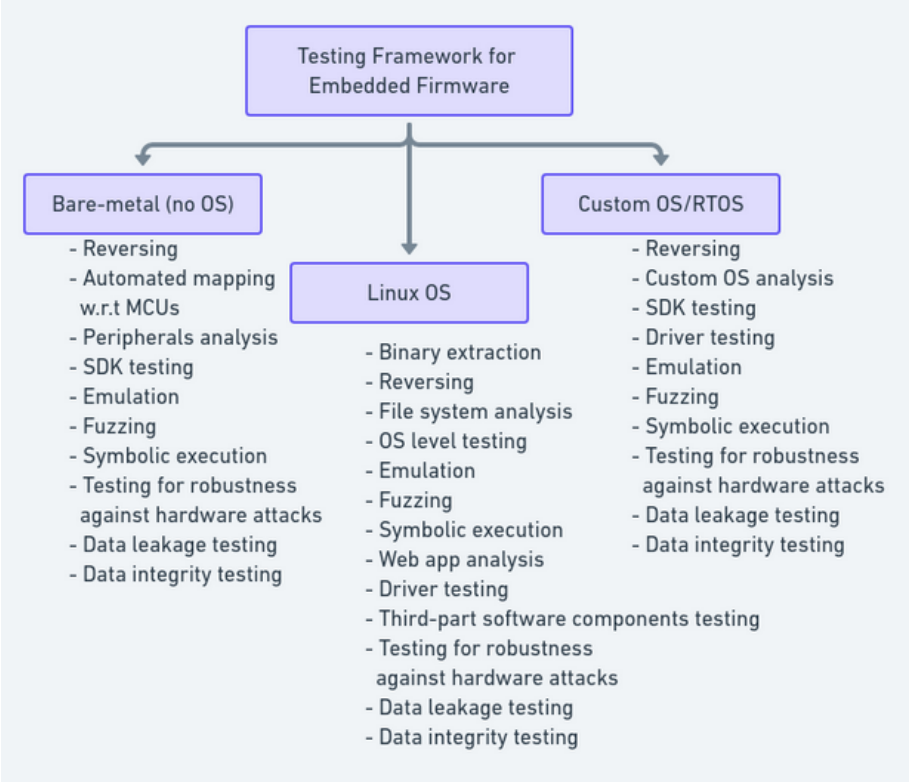

Figure 5: Proposed unified framework features for firmware testing

8 Conclusion

In this work we did a survey on the different security threats on the embedded systems targeting hardware and firmware. We studied both firmware and hardware attacks on embedded devices. We find the existing tools that can be used in these scopes. Some of the attacks scenarios and ongoing research are discussed. Finally challenges, ideas and future work are proposed. The main goal of the paper is to make reader aware of significance of embedded system testing and the need of the hour regarding the requirements for contribution in this research domain.

References

[1] A. P. Fournaris, L. Pocero Fraile, and O. Koufopavlou, "Exploiting hardware vulnerabilities to attack embedded system devices: A survey of potent microarchitectural attacks," *Electronics*, vol. 6, no. 3, p. 52, 2017.

[2] M. D. L. W. B. L. Abdullah Qasem, Paria Shirani and B. L. Agba, "Automatic vulnerability detection in embedded devices and firmware: Survey and layered taxonomies," *ACM Computing Surveys*, vol. 54, 2021.

[3] a. M. A. T. Meriem Bettayeb, Qassim Nasir, "Firmware update attacks and security for iot devices: Survey," *ACM, ArabWIC 2019: Proceedings of the ArabWIC 6th Annual International Conference Research Track*, 2019.

[4] Y. T. N. D. Wei Xie, Yikun Jiang and Y. Gao, "Vulnerability detection in iot firmware: A survey," *IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS)*, 2017.

[5] J. L. F. Maialen Eceiza and M. Iturbe, "Fuzzing the internet of things: A review on the techniques and challenges for efficient vulnerability discovery in embedded systems," *IEEE INTERNET OF THINGS JOURNAL*, vol. 8, no. 13, 2021.

[6] Weilin Zhong, "Command injection," accessed Oct 23, 2021. [Online]. Available: https://owasp.org/www-community/attacks/Command_Injection#

[7] "Arbitrary code execution," accessed Oct 23, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Arbitrary_code_execution

[8] "Buffer overflow," accessed Oct 23, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Buffer_overflow

[9] "Return-oriented programming," accessed Oct 23, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Return-oriented_programming

[10] W. Hu, C.-H. Chang, A. Sengupta, S. Bhunia, R. Kastner, and H. Li, "An overview of hardware security and trust: Threats, countermeasures and design tools," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020.

[11] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "Flipping bits in memory without accessing them: An experimental study of dram disturbance errors," *ACM SIGARCH Computer Architecture News*, vol. 42, no. 3, pp. 361–372, 2014.

[12] L. P. Fraile, A. P. Fournaris, and O. Koufopavlou, "Revisiting rowhammer attacks in embedded systems," in *2019 14th International Conference on Design & Technology of Integrated Systems In Nanoscale Era (DTIS)*. IEEE, 2019, pp. 1–6.

[13] J. Szefer, "Survey of microarchitectural side and covert channels, attacks, and defenses," *Journal of Hardware and Systems Security*, vol. 3, no. 3, pp. 219–234, 2019.

[14] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Annual international cryptology conference*. Springer, 1999, pp. 388–397.

[15] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2004, pp. 16–29.

[16] A. Sayakkara, N.-A. Le-Khac, and M. Scanlon, "A survey of electromagnetic side-channel attacks and discussion on their case-progressing potential for digital forensics," *Digital Investigation*, vol. 29, pp. 43–54, 2019.

[17] R. Callan, A. Zajic, and M. Prvulovic, "A practical methodology for measuring the side-channel signal available to the attacker for instruction-level events," in *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE, 2014, pp. 242–254.

[18] R. Callan, A. Zajić, and M. Prvulovic, "Fase: Finding amplitude-modulated side-channel emanations," in *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2015, pp. 592–603.

[19] S. Patranabis, A. Chakraborty, P. H. Nguyen, and D. Mukhopadhyay, "A biased fault attack on the time redundancy countermeasure for aes," in *International workshop on constructive side-channel analysis and secure design*. Springer, 2015, pp. 189–203.

[20] A. Tang, S. Sethumadhavan, and S. Stolfo, "{CLKSCREW}: exposing the perils of security-oblivious energy management," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 1057–1074.

[21] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE design & test of computers*, vol. 27, no. 1, pp. 10–25, 2010.

[22] "National vulnerability database," accessed Oct 25, 2021. [Online]. Available: https://nvd.nist.gov/vuln/search/results?form_type=Basic&results_type=overview&query=firmware&search_type=last3months

[23] M. C. Ang Cui and S. J. Stolfo, "When firmware modifications attack:a case study of embedded exploitation," *NDSS*, 2013.

[24] "Eavesdropping bugs in mediatek chips affect 37% of all smartphones and iot globally," accessed Nov 25, 2021. [Online]. Available: https://thehackernews.com/2021/11/eavesdropping-bugs-in-mediatek-chips.html

[25] "Cwe-787: Out-of-bounds write," accessed Nov 25, 2021. [Online]. Available: https://cwe.mitre.org/data/definitions/787.html

[26] "Mama always told me not to trust strangers without certificates," accessed Nov 25, 2021. [Online]. Available: https://blog.grimm-co.com/2021/09/mama-always-told-me-not-to-trust.html

[27] "Nvd cve-2021-40847 detail," accessed Nov 25, 2021. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-40847

[28] "Nvd vulnerability metrics," accessed Nov 25, 2021. [Online]. Available: https://nvd.nist.gov/vuln-metrics/cvss

[29] "Remote code execution in cctv-dvr affecting over 70 different vendors," accessed Nov 25, 2021. [Online]. Available: http://www.kerneronsec.com/2016/02/remote-code-execution-in-cctv-dvrs-of.html

[30] R.-P. Weinmann and B. Schmotzle, "Tbone – a zero-click exploit for tesla mcus," 2020.

[31] M. Kol and S. Oberman, "Ripple20," June 2020.

[32] S. C. A Matheus E. Garbelini and C. Wang, "Sweyntooth: Unleashing mayhem over bluetooth low energy."

[33] E. Itkin, "Don't be silly – it's only a lightbulb," 2020. [Online]. Available: https://research.checkpoint.com/2020/dont-be-silly-its-only-a-lightbulb/

[34] J. Fruhlinger, "The mirai botnet explained: How teen scammers and cctv cameras almost brought down the internet," 2018. [Online]. Available: https://www.csoonline.com/article/3258748/the-mirai-botnet-explained-how-teen-scammers-and-cctv-cameras-almost-brought-down-the-internet.html

[35] A. Greenberg, "This bluetooth attack can steal a tesla model x in minutes," 2020. [Online]. Available: https://www.wired.com/story/tesla-model-x-hack-bluetooth/

[36] D. K. Oka, "Analysis of an attack on automotive keyless entry systems," 2021. [Online]. Available: https://www.synopsys.com/blogs/software-security/key-fob-hack-analysis/

[37] L. Tung, "Microsoft: Firmware attacks are on the rise and you aren't worrying about them enough," 2021. [Online]. Available: https://www.zdnet.com/article/microsoft-firmware-attacks-are-on-the-rise-and-you-arent-worrying-about-them-enough/

[38] S. Adee, "The hunt for the kill switch," *IEEE Spectrum*, vol. 45, no. 5, pp. 34–39, 2008.

[39] "Fbi fears chinese hackers have back door into us government and military." [Online]. Available: https://abovetopsecret.com/forum/thread350381/pg1

[40] S. Mangard, E. Oswald, and T. Popp, "Power analysis attacks - revealing the secrets of smart cards," 2007.

[41] M. Luo, A. C. Myers, and G. E. Suh, "Stealthy tracking of autonomous vehicles with cache side channels," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020, pp. 859–876.

[42] A. Nilsson, P. N. Bideh, and J. Brorsson, "A survey of published attacks on intel sgx," *arXiv preprint arXiv:2006.13598*, 2020.

[43] G. Haas, S. Potluri, and A. Aysu, "itimed: Cache attacks on the apple a10 fusion soc." *IACR Cryptol. ePrint Arch.*, vol. 2021, p. 464, 2021.

[44] M. Sabbagh, Y. Fei, and D. Kaeli, "A novel gpu overdrive fault attack," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[45] C. Shepherd, K. Markantonakis, N. van Heijningen, D. Aboulkassimi, C. Gaine, T. Heckmann, and D. Naccache, "Physical fault injection and side-channel attacks on mobile devices: A comprehensive survey," *arXiv preprint arXiv:2105.04454*, 2021.

[46] NIST, "Source code security analyzers." [Online]. Available: https://owasp.org/www-community/Source_Code_Analysis_Tools

[47] "Source code analysis tools," 2021. [Online]. Available: https://www.nist.gov/itl/ssd/software-quality-group/source-code-security-analyzers

[48] R. Labs, "Binwalk." [Online]. Available: https://github.com/ReFirmLabs/binwalk

[49] C. Smith, "Firmwalker." [Online]. Available: https://github.com/craigz28/firmwalker

[50] N. S. A. (NSA), "Ghidra." [Online]. Available: https://github.com/NationalSecurityAgency/ghidra

[51] Hex-rays, "Ida pro." [Online]. Available: https://hex-rays.com/ida-pro/

[52] "Radare2." [Online]. Available: https://github.com/radareorg/radare2

[53] Intel, "Cve binary tool." [Online]. Available: https://github.com/intel/cve-bin-tool

[54] "Cve-search." [Online]. Available: https://github.com/cve-search/cve-search

[55] Fkie-cad, "Cwe checker." [Online]. Available: https://github.com/fkie-cad/cwe_checker

[56] "Binare." [Online]. Available: https://binare.io/

[57] "Expliot firmware auditor." [Online]. Available: https://expliot.io/pages/firmware-auditor

[58] "The firmware analysis and comparison tool (fact)." [Online]. Available: https://github.com/fkie-cad/FACT_core

[59] "Emba- the security analyzer for embedded device firmware." [Online]. Available: https://github.com/e-m-b-a/emba

[60] "Qemu." [Online]. Available: https://qemu-project.gitlab.io/qemu/

[61] "Unicorn engine." [Online]. Available: https://github.com/unicorn-engine/unicorn

[62] "Qiling framework." [Online]. Available: https://qiling.io/2021/09/25/intro/

[63] M. Ludwig, A. Hepp, M. Brunner, and J. Baehr, "Cress: Framework for vulnerability assessment of attack scenarios in hardware reverse engineering," 2021.

[64] M. Gay, T. Paxian, D. Upadhyaya, B. Becker, and I. Polian, "Hardware-oriented algebraic fault attack framework with multiple fault injection support," in *2019 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*. IEEE, 2019, pp. 25–32.

[65] J. Cruz, Y. Huang, P. Mishra, and S. Bhunia, "An automated configurable trojan insertion framework for dynamic trust benchmarks," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018, pp. 1598–1603.

[66] C. O'Flynn, "A framework for embedded hardware security analysis," 2017.

[67] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2015, pp. 137–143.

[68] S. Roshanisefat, H. Mardani Kamali, H. Homayoun, and A. Sasan, "Rane: An open-source formal de-obfuscation attack for reverse engineering of logic encrypted circuits," in *Proceedings of the 2021 on Great Lakes Symposium on VLSI*, 2021, pp. 221–228.

[69] C. S. N. R. A. M. Y. F. A. F. D. B. Y. R. C. C. K. Eric Gustafson, Marius Muench and G. Vigna, "Toward the analysis of embedded firmware through automated re-hosting," *USENIX Association, 22nd International Symposium on Research in Attacks, Intrusions and Defenses*, 2019.

[70] E. K. S. K. Y. J. Mingeun Kim, Dongkwan Kim and Y. Kim, "Firmae: Towards large-scale emulation of iot firmware for dynamic analysis," *ACSAC '20: Annual Computer Security Applications Conference - ACM*, p. 733–745, 2020.

[71] Y. J. Z. L. W. C. X. J. Jian Gao, Yiwen Xu and J. Sun, "Em-fuzz: Augmented firmware fuzzing via memory checking," *IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS*, vol. 39, no. 11, 2020.

[72] F. K. Zhijie Gui, Hui Shu and X. Xiong, "Firmcorn: Vulnerability-oriented fuzzing of iot firmware via optimized virtual execution," *IEEE Access*, 2020.

[73] R. J. Seoksu Lee, Joon-Young Paik and E.-S. Cho, "Toward machine learning based analyses on compressed firmware," *IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, 2019.

[74] A. Z. Andrei Costin and A. Francilon, "Towards automated classification of firmware images and identification of embedded devices," *Eurecom*.

[75] P. L. Wei Zhou, Le Guan and Y. Zhang, "Automatic firmware emulation through invalidity-guided knowledge inference," *Usenix*, 2021.

[76] F. S. Max Hoffmann and C. Paar, "Armory: Fully automated and exhaustive fault simulation on arm-m binaries," *arXiv:2105.13769v1 [cs.CR]*, 2021.

[77] Z. Chen, G. Vasilakis, K. Murdock, E. Dean, D. Oswald, and F. D. Garcia, "Voltpillager: Hardware-based fault injection attacks against intel {SGX} enclaves using the {SVID} voltage scaling interface," in *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021.

[78] H. Naghibijouybari, A. Neupane, Z. Qian, and N. Abu-Ghazaleh, "Beyond the cpu: Side–channel attacks on gpus," *IEEE Design & Test*, vol. 38, no. 3, pp. 15–21, 2021.

[79] A. R. Javed, S. U. Rehman, M. U. Khan, M. Alazab, and H. U. Khan, "Betalogger: Smartphone sensor-based side-channel attack detection and text inference using language modeling and dense multilayer neural network," *Transactions on Asian and Low-Resource Language Information Processing*, vol. 20, no. 5, pp. 1–17, 2021.

[80] K. Sahu, R. Kshirsagar, S. Vasudeva, T. Alzahrani, and N. Karimian, "Leveraging timing side-channel information and machine learning for iot security," in *2021 IEEE International Conference on Consumer Electronics (ICCE)*, 2021, pp. 1–6.

[81] X. Wang, T. Hoque, A. Basak, R. Karam, W. Hu, M. Qin, D. Mu, and S. Bhunia, "Hardware trojan attack in embedded memory," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 17, no. 1, pp. 1–28, 2021.

Authors

Asmita- She recevied her bachelor degree in Electronics and telecommunication engineering from International Institute of Information Technology, Pune, India. She has 3 years industrial experience in embedded system design and security assessment. She is currently a PhD student in ASEEC lab at Electrical and Computer Engineering Department in University of California, Davis. Her research interest includes Embedded System Security, Firmware Security, IoT Security.

Banafsheh Saber Latibari- She received her bachelor and master degrees in Computer Engineering from K.N.Toosi University of Technology and Sharif University of Technology, Tehran, Iran respectively. Currently she is pursuing her PhD degree in GATE lab at Electrical and Computer Engineering Department in University of California, Davis. Her research interest includes Hardware Security and Machine Learning.