International Journal of Engineering Sciences Paradigms and Researches [Volume 47, Issue: Special), March 2018]
www.ijesonline.com **ISSN (Online): 2319-6564**

An analysis of embedded systems hardware security

Venkata Ramana Kothakota
Raajdhani Engineering College, Bhubaneswar
venkata Ramana@rec.ac.in

Abstract

Throughout the product development lifecycle, safe hardware design is frequently disregarded, leaving many devices open to hacker attacks that could cause service theft, financial loss, or reputational damage. After a negative event, items frequently need to be changed, which drives up overall development costs and lengthens time-to-market. The broad principles of hardware security are the main topic of this study. Necessary.This document covers attack and threat models, enclosure, circuit board, and firmware layer design solutions, and suggestions for integrating security into the product development cycle.

Keywords: *Firmware, PCB, Security, Attacks, Tamper Mechanisms.*

1. Introduction

Embedded systems are electronic computer systems designed for dedicated operating functions, often while respecting several constraints like real-time computing, power consumption, size and cost, etc. Embedded systems control many devices in common use today such as Smartphone's, GPS, codec GSM, decoders, MP3, MPEG62, MPEG4,PDAs,RFIDs, smart card sand networked sensors etc. Generally, they are controlled by one or more main processing cores that are typically either Microcontrollers, Digital Signal Processors (DSPs) or Field Programmable Gate Arrays (FPGAs). These systems are embedded as part of a complete electronic system, often including software, hardware, and communication and sensor parts. By contrast, a general-purpose computer - such as a Personal Computer (PC) - is designed to be flexible and to meet a wide range of end-user needs. The key characteristic of an embedded system is that it is dedicated to the handling of a particular task. They may require very powerful processors And extensive communications. Ideally, these embedded systems are completely self-contained and will typically run off a battery source for many years before the batteries need to be changed or charged. Since such systems are embedded and dedicated to specific tasks, design engineers search to optimize them by reducing their size (miniaturization made possible by advanced IC design in order to couple full communication subsystems to sophisticated sensors) and cost in terms of energy consumption, memory and logic resources, while increasing their reliability and performance. Consequently, embedded systems are especially suited for use in transportation, medical applications, safety and security. Indeed, in dealing with security, embedded systems can be self-sufficient and should be able to deal with communication systems. Considering these specific Conditions, in the fields of information and communication technology, embedded systems designers are faced with many challenges in terms of both the tradeoff between cost/performance/power and architecture design. This is especially true for embedded systems designs, which often operate in non-secure environments, while at the same time being constrained by such factors as computational capacity, memory size and - in particular – power consumption. One challenge is in the design of hardware architecture able to meet the appropriate level of security and consequently the best trade-off between hardware resources and the best throughput rates for real-time embedded applications. The primary goal of this paper is to introduce the reader to the concepts of designing secure hardware in embedded systems. Understanding the major classes of attack and the mindset of potential attackers will go a long way in helping one to decide on the best and proper secure hardware design methods for a particular application. Examples of previous hardware attacks are discussed throughout the paper. By learning from prior attacks, we can understand the weaknesses and mistakes of such designs and improve upon them in our own products. We also provide numerous references and resources for the reader to explore in more detail[1].

2. Security in the Product Development Cycle

As designers, the best we can do is understand the potential attacks against our system, design methods to prevent such attacks, with the understanding that nothing is ever 100While many design methodologies exist, the primary concern is to incorporate risk analysis and security considerations into each step of the cycle. Having high-level processes in place will help to ensure that the low-level design details are properly implemented. In NIST's Engineering Principles for Information Technology Security (A Baseline for Achieving Security), a number of security principles are provided that can be applied to any design process: 1. Establish a sound security policy as the "foundation" for design. The security policy identifies security goals the product should support .The goals guide the procedures, standards, and controls of the development cycle. It may be necessary to modify or adjust security goals due to other operational requirements. 2. Treat security as an integral part of the overall system design. Security must be considered during product design. It is very difficult to implement security measures properly and successfully after a system has been developed.
3. Reduce risk to an acceptable level. Risk is defined as the combination of the probability that a particular threat source will exploit vulnerability and the resulting impact should this occur. Elimination of all risk is not cost-effective and likely not possible. A cost-benefit analysis should be conducted for each proposed secure hardware mechanism. 4. Implement layered security (Ensure no single point of failure). Security designs should consider a layered approach of multiple security mechanisms to protect against a specific threat or to reduce a vulnerability. 5. Strive for simplicity. The more complex the mechanism, the more likely it may possess exploitable flaws. Simple mechanisms tend to have fewer exploitable flaws and require less maintenance. 6. Minimize the system elements to be trusted. Security measures include people, operations, and technology. Where technology is used, hardware, firmware, and software should be designed so that a minimum number of elements need to be trusted in order to maintain protection. In Kocher's Hacking[2]

Cryptosystems presentation, it is recommended to"put all your eggs in one basket" by isolating all critical content into one secure area instead of having multiple secure areas throughout the design. This way, you can focus on properly securing and testing a single critical area of the product instead of many disparate areas. 7. Do not implement unnecessary security mechanisms. Every security mechanism should support one or more defined goals. Extra measures should not be implemented if they do not support a goal, as they could add unneeded complexity to the system and are potential sources of additional vulnerabilities. All likely classes of attack should be protected against. Many times, an engineering change will be made to the product circuitry or firmware without re-evaluating the effect such a change may have on system security. Without a process in place to analyze changes throughout the design cycle, security that was properly implemented at the beginning of the design may become irrelevant by the time the product goes into production. Requiring trusted third party design reviews of the product during the prototype and pre-production phases allow a "fresh set of eyes" to examine the product for any critical design flaws that are non-obvious or have been simply overlooked by the product designers.

3. Security issues

Before deciding on acceptable secure hardware methods to design into your product, risk assessment of three key areas must take place: What needs to be protected? The critical components in your circuit that need to be protected should be identified before the product is actually constructed, as it is extremely difficult (if not impossible) to implement proper security mechanisms after-the-fact. Such components may include specific algorithms, device identifiers, digital media, biometrics, cryptographic keys, complete product firmware, or other product-specific data. In addition to protecting discrete data contents, you may be interested in implementing a secure product boot sequence, field programmability, or remote management interface. Be aware that in some cases, even on-critical portions can unknowingly compromise the security of the system. Why it is being protected. In some countries, protecting certain content may be a legislative requirement (for example,
a medical device containing confidential patient information must be secured in order to meet the U.S. HIPAA requirements). In most situations, critical data is being protected to prevent a specific security threat .It is important to acknowledge that an attack threat may exist and to implement the proper mechanisms to prevent them. Ignoring or overlooking the possibility of attack can lead to a vulnerable product. Who you are protecting against. The types of attackers vary greatly, from a curious hardware hacker to an entire group backed by organized crime, government, or a competitor. As such, it is important to attempt to properly identify the skill level and theoretical goals of the primary attackers[1].

4. Attacks on Embedded System

Attacks on embedded systems can be categorized in three classes i.e. software attacks, physical attacks and side channel attacks. Software attacks have largest share in total number of attacks on embedded systems and it is most difficult to protect against such attacks. In this article we will focus on software attacks and countermeasure against these attacks. An overview of physical and side channel attacks will be provided.

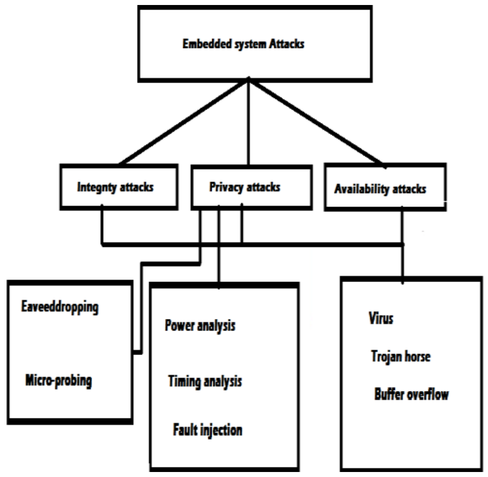

Figure 1: Embedded System Attacks

- **Side Channel Attacks:**

Side channel attacks are based on observing system properties e.g. time, power consumption while system is performing computations e.g. cryptographic operations. In certain systems timing information can lead to entire secret key, though it seems that timing information can give very little information but it has found that with proper study of timing sequence entire secret key can be found. Along with this power consumption can also lead to the entire secret key, well equipped labs have the equipment that can measure the changes in the power consumption with about 1 accuracy and are very in expensive. To overcome timing attacks one may add random timing delays to various operations, in similar manner we can overcome power consumption attacks by adding random noise or by proper shielding of the equipment but it leads to increased cost of the equipment[1].

- **Software Attacks**

Software attacks are very common in embedded systems capable of downloading application from internet or have some means of communication to interact with external world. As compare to physical and side channel attacks, software attacks are very cheap and does not requires any big infrastructures thus making it an immediate challenge for embedded system design[3].

- **Practical Design Solutions**

This section examines three levels of the product: enclosure, circuit board, and firmware. Design solutions and protection methods are proposed, and attack examples are provided for historical reference.

- **Product Enclosure**

The design of a secure product enclosure is critical to prevent attackers from gaining access to the internal circuitry. Once the circuit board is accessible to the attacker, they will typically reverse-engineer the design to create a schematic and then identify possible attack vectors. Opening some products is as simple as loosening a few screws or prying open the side with a hobby knife or screwdriver.

- **External Interfaces**

External interfaces are typically a product's lifeline to the outside world. Such interfaces may be used for a number of purposes, including connecting to peripherals, field programming, or testing during product manufacturing. Typical interfaces include Firmware, USB, RS232, Ethernet, or JTAG IEEE 1149.1. Products often implement development or programming interfaces that are not meant for everyday consumer use, but can benefit an attacker immensely. Simply obfuscating these interfaces with proprietary connector types or hidden access doors or holes is not suitable as they will easily be discovered[9].

- **Tamper Mechanisms**

The goal of tamper mechanisms is to prevent any attempt by an attacker to perform an unauthorized physical or electronic action against the device. Tamper mechanisms are divided into four groups: resistance, evidence, detection, and response. Tamper mechanisms are most effectively used in layers to prevent access to any critical components. They are the primary facet of physical security for embedded systems and must be properly implemented in order to be successful. From the designer's perspective, the costs of a successful attack should outweigh the potential rewards. Often, existing tamper mechanisms can only be discovered by attempted or complete disassembly of the target product. This may require an attacker to obtain more than one device in order to sacrifice one for the sole purpose of discovering such mechanisms. Once the mechanisms are noted, an adversary can form hypotheses about how to attack and bypass them[9].

- **Physical Access to Components**

Sensitive components that are most likely to be targeted for an attack (such as the microprocessor, ROM, RAM, and programmable logic) should be made difficult to access. Reverse engineering the target product usually requires one to determine the part numbers and device functionality of the major components on the board. Understanding what the components do may provide details for particular signal lines that may be useful for active probing during operation. Components are easily identified by their part numbers and manufacturing markings on the device packaging , and by following their traces to see how they interconnect with other components on the board. Nearly all IC manufacturers post their component data sheets on the Web for public viewing, and online services such as IC Master, Data Sheet Locator, and Part Miner provide part number searches and pin out and package data for hundreds of thousands of components. To increase the difficulty of reverse engineering and device identification, it is recommended that all markings be scratched off the tops of the chips[9].

- **PCB Design and Routing**

Proper engineering practices should always be exercised. Traces should remain as short as possible. Differential signal lines should be aligned parallel even if located on separate layers. Noisy power supply lines should be kept away from sensitive digital and analog lines. Properly designed power and ground planes should be employed to reduce EMI emissions. Additionally, any unnecessary test points should be removed from the design, as they allow unwanted noise and interference to pass through the PCB. If test points are required, consider using a copper-filled pad as opposed to a through-hole pad. Critical traces should be hidden on inner board layers and trace paths should be obfuscated to prevent easy reverse engineering of circuitry. Use buried vias, which connect two or more inner layers but no outer layer and cannot be seen from either side of the board, to reduce potential probing points for the attacker. Be aware of electrical noise issues that these modified traces may introduce[8].

- **Bus Protection**

Device operation and information can be gleaned by analyzing the internal address, data, and control bus lines with a logic analyzer, digital oscilloscope, or custom circuitry. Targeted bus lines could be probed by simply removing the solder mask on the circuit board. Be aware of the data being stored in memory at all times and what is transferred across exposed and accessible buses[9].

5. Conclusion

The goal of this paper was to introduce the reader to secure hardware design concepts, attacks, and potential solutions. It is by no means complete as such mechanisms are constantly evolving. The beginning sections of the paper provided information on security policies and a baseline classification of attackers, attack types, and threat vectors. The majority of the paper focused on the many aspects of the secure hardware design process, divided into enclosure, circuit board, and firmware layers. Wherever possible, we referenced previous hardware attacks for educational and historical purposes. When designing a product, it is essential to first establish a security policy that defines the security goals of the product, as you must first understand what you are protecting and why you are protecting it before security can be successfully implemented. Staying aware of the latest attack methodologies and trends will enable you to choose the proper means of protection for your particular product.

References

[1] D.G. Abraham, G.M. Dolan, G.P. Double, and J.V. Stevens, ”Transaction Security System,” IBM Systems Journal, vol. 30, no. 2, 1991: http://www.research.ibm.com/journal/sj/302/ibmsj3002G.pdf.

[2] Aleph One,”Smashing the Stack for Fun and Profit:http://www.securityfocus.com/library/14.

[3] R.J. Anderson and M. Kuhn,”Low Cost Attacks on Tamper Resistant Devices,” Security Protocols, 5th International Workshop, 1997, http://www.cl.cam.ac.uk/mgk25/tamper2.pdf.

[4] R.J. Anderson,”Security Engineering-A Guide to Building Dependable Distributed Systems,” John Wiley and Sons, 2001 Techniques,” John Wiley and Sons, 1998.

[5] B. Dipert,”Cunning Circuits Confound Crooks,” EDN Magazine, October 12, 2000.

[6] J. Dyer, M. Lindemann, R. Perez, R. Sailer, S.W. Smith, L. van Doorn, and S. Weingart, ”Building the IBM 4758 Secure Coprocessor,” IEEE Computer, October 2001.

[7] M. Fisher,”Protecting Binary Executables,” Embedded Systems Programming, February 2000. 13. M.G. Graff and K.R. Van Wyk, ”Secure Coding: Principles and Practices,” O’Reilly and Associates, 2003. 14. J. Grand (Editor), et al.,”Hardware Hacking: Have Fun While Voiding Your Warranty,” Syngress Publishing, 2004.

[8] P. Gutmann,"Secure Deletion from Magnetic and Solid-State Memory Devices," Sixth USENIX Security Symposium, 1996.
[9] Practical Secure Hardware Design for Embedded Systems Joe GrandGrand Idea Studio, Inc. http : www.grandideastudio.com