PCBench: Benchmarking of Board-Level Hardware Attacks and Trojans

Huifeng Zhu
Washington University in St. Louis
zhuhuifeng@wustl.edu

Xiaolong Guo
Kansas State University
guoxiaolong@ksu.edu

Yier Jin
University of Florida
yier.jin@ece.ufl.edu

Xuan Zhang
Washington University in St. Louis
xuan.zhang@wustl.edu

ABSTRACT

Most modern electronic systems are hosted by printed circuit boards (PCBs), making them a ubiquitous system component that can take many different shapes and forms. In order to achieve a high level of economy of scale, the global supply chain of electronic systems has evolved into disparate segments for the design, fabrication, assembly, and testing of PCB boards and their various associated components. As a consequence, the modern PCB supply chain exposes many vulnerabilities along its different stages, allowing adversaries to introduce malicious alterations to facilitate board-level attacks.

As an emerging hardware threat, the attack and defense techniques at the board level have not yet been systemically explored and thus require a thorough and comprehensive investigation. In the absence of standard board-level attack benchmark, current research on perspective countermeasures is likely to be evaluated on proprietary variants of ad-hoc attacks, preventing credible and verifiable comparison among different techniques. Upon this request, in this paper, we will systematically define and categorize a broad range of board-level attacks. For the first time, the attack vectors and construction rules for board-level attacks are developed. A practical and reliable board-level attack benchmark generation scheme is also developed, which can be used to produce references for evaluating countermeasures. Finally, based on the proposed approach, we have created a comprehensive set of board-level attack benchmarks for open-source release.

1 INTRODUCTION

Nowadays, the printed circuit boards (PCBs) are ubiquitous in almost all kinds of electronic systems. The PCBs take in a variety of representations with different sizes, form factors, materials, number of stack layers, and fabrications parameters. Usually the design of a PCB involves hundreds of elements, including both integrated components such as microprocessor chips, field programmable gate arrays (FPGAs), and application-specific integrated circuit (ASICs), and discrete components such as resistors, capacitors, inductors, diodes, transistors. There can also be thousands of signal traces and power rails to enable board-level high-performance communications and power delivery. To address the design complexity, PCB designers need to take various constraints into considerations during the PCB design process. For example, complex hierarchical power delivery system (PDS) is a necessity to fulfill the various power supply requirements of different components. Other constraints such as signal integrity and assemblability should also be carefully considered.

However, among all design constraints, security is often omitted in board-level designs, despite that numerous vulnerabilities are already exploited due to the segmented board-level supply chain [1, 14, 19, 28]. Following the global PCB supply chain, the design, fabrication, test, and the selling of different electronic components are performed by different, often untrusted parties. While researchers start to look into the problem recently and try to develop countermeasures, they meet a main obstacle that targeted security evaluation benchmarks for PCBs are lacking. It is urgently needed to develop benchmarks representing all different types of board-level threats so that different detection techniques can be verified and compared.

In the area of security benchmarking, past research has intensively studied attacks and Trojans at the chip level. In [24] the authors propose standard benchmarks at different levels (RTL, netlist, and layout) to evaluate chip-level hardware Trojans and their detection techniques by leveraging a vulnerability analysis flow. Evaluation metrics of the benchmarks is also introduced such as Trojan detectability. Despite the massive work in developing chip-level Trojan benchmarks [24, 29], those methods, designs, and evaluation results cannot be directly applied to the board level. In the case of inserting chip-level Trojans, the usual assumptions are: both the power supply voltage and the specifications of I/O of on-chip blocks are unified, and malicious circuits can be implemented at any location occupied by filler cells or capacitor cells without routing limitations. However, these assumptions do not hold in benchmarking board-level attacks. For a PCB design, there can be various supply voltage domains and the specifications of the I/O of chips' can be different. Besides, the placing and routing of a malicious circuit are limited by the board spare area and density of the traces.

In this paper, we propose a systematic definition and a comprehensive taxonomy of board-level attacks based on the thorough analysis of existing and emerging board-level threats. Potential attack mechanisms and attack vectors are examined and evaluated. Further, a set of constraints for successfully implementing board-level attacks into the target designs are specified. Based on those constraints, a novel rule-based benchmark generation mechanism is developed to create reliable and practical board-level security benchmarks. The main contributions of this paper are as follows.

- We define and categorize the board-level attacks and Trojans based on the target component and attacking mechanism.
- A practical and reliable board-level hardware attacks and Trojans benchmark generation scheme is proposed. This method can guide researchers in developing and evaluating

ASP-DAC 2021, June 18–21, 2021, Tokyo, Japan
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

countermeasures to emerging board-level vulnerabilities and threats to system security.

- Sample attacks on selected PCB designs are created following the proposed method to validate the proposed benchmarking method.

2 BACKGROUND

2.1 PCB Design Process

Figure 1 (right) illustrates a flight controller board that represents a typical PCB design commonly found in today's electronic systems. In general, the PCB design process is composed of following steps: top-level design, schematic capture, PCB layout, PCB fabrication, and electrical components assembly. During the top-level design, system architecture and block diagram are defined according to the functional requirements and physical constraints of the system. The detailed specifications of the PCB (e.g., physical dimensions, power routing strategy, cost budget, etc.) are thus determined in this process. Based on the block diagram, the schematic of the PCB is captured. In modern designs, designers will first select the chips that can fulfill the planned functions of each block. Peripheral circuits are then designed around each chip, which are used to supporting the functionality of these chips. At the end of the schematic step, the schematic file (i.e., netlist) and the bill of materials (BOM) file can be generated. The former contains the information of the electrical net connections and the latter defines the exact model including value, vendor, package, etc. of all components in this design.

The PCB layout is similar to the Place&Route (P&R) process of the digital IC design and is designed after the schematic capture step. Different from chip-level P&R process which is often performed automatically relying on design automation tools, the PCB layout is often performed manually due to the sophisticated considerations of power integrity, signal integrity, interference, and assemblability. The layout file (i.e., Gerber file) is prepared in this step. The file contains all components and the routing graphical information and can be used for PCB fabrication. After the PCBs are fabricated, the components are assembled and soldered either manually or automatically guided by the BOM file. Tests and validations can be performed between each step of the process.

2.2 Board-level Attacks

Based on the PCB design process described above, we define board-level attacks as *intentional malicious modifications of a PCB during any stages of design, fabrication, assembly, and in-field usage of the PCB.* In this definition, we assume any attack that targets the board and its associated component as the victim could constitute a board-level attack. As illustrated in Figure 1, the board-level attacks include the malicious modifications applied to processing units (microcontroller, FPGA, etc.), components (passive components and ASICs), trace/via, fabrication parameters, as well as the deliberate violations of normal usage constraints set by the designers or distributors, i.e. malicious probing and accessing. The modifications can be injected at any phase of the entire procedure of the PCB, including PCB design, fabrication, assembly, and in-field usage.

According to the scope specified in our definition, board-level Trojan is a major subset of board-level attacks. A board-level Trojan generally contains two parts: the trigger and the payload [24]. The trigger monitors variations of the signals or a series of events on

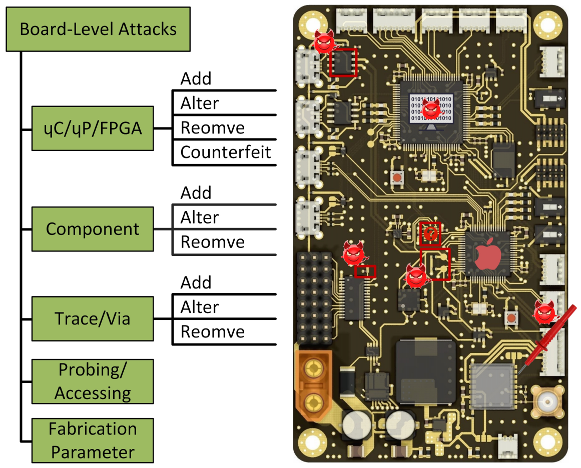

Figure 1: The category of board-level attacks and a flight controller board design with example malicious modifications inserted.

the board. Once the preset conditions of the trigger are satisfied, the payload will be activated and perform malicious behaviors [29]. Note that the adversaries do not necessarily introduce modifications to the original circuits and may perform the attacks without using any trigger. For board-level attacks, we assume that attackers will not maliciously modify the internal structures of the integrated circuits, i.e. chip-level hardware Trojans. We believe that any modifications inside ICs are part of the chip-level hardware security, rather than board-level hardware security.

2.3 Existing Board-Level Attacks Benchmark

Compared to the abundant literature on chip-level attacks [9, 29], security research on board-level threats is rather sparse. In [17] authors develop a benchmarking solution to facilitate an unbiased and comparable evaluation of countermeasures applicable to PCB trust assurance. However, only KiCAD *.NET* format netlists are provided by the benchmark to generate the layout of PCBs with and without Trojans. This abstraction level limits the usage of the benchmark and prevents users from evaluating schematic capture and circuit simulation using the same design. Further, Trojans introduced in these benchmarks comprise mainly of discrete components, i.e. BJTs, resistors, and capacitors, to mimic the behaviors of the logic gate. Such simplified strategy does not consider the fact that attackers can take full advantage of off-the-shelf small package chips, e.g. 74-series circuits, to implement more advanced attacks. In this paper, we develop several rules constraining the generation of functional and sneaky attacks. Following these rules, we curate a set of state-of-the-art PCB reference designs projects and generate the benchmark at different abstraction levels based on the rules.

3 BENCHMARKING METHODOLOGY

3.1 Board-Level Attack Vectors

In this section, we summarize and classify the known board-level attacks vectors according to the life cycle of PCB design, the in-field use and the types of modifications. This comprehensive classification also allows us to identify new attack vectors that has not been exploited in the past. The possible attack vectors are listed in Table 1, where board-level attacks can be injected at any phase of the PCB design and fabrication by adversaries, as well as during the in-field usage by users. We divide the entire procedure of the

Table 1: Summary of Example Board-Level Attack Vectors

	S1: Circuit Design	S2: Fabrication/Assembly	S3: In Field
M1: μC/μP/FPGA	Insert an extra chip that can be used as hardware backdoor; Replace the model of the chip to the one with lower specifications.	Insert maliciously programmed chips[22]; Use unqualified chips [13, 26]; Use counterfeit chips.	Insert extra malicious chips [16, 18, 25]; Replace chips to the ones with malicious firmware [1] or maliciously programmed [11, 15]; Replace the System-in-Package (SIP) to the one with malicious circuit inserted.
M2: Component	Change models or modify values of components to enable malfunction under special operating conditions [19]; Insert transistor as the backdoor to interrupt boot; Trojans based on logic gates [17]; Use unqualified components; Remove ESD protection diodes; Insert sampling resistor to enable power side-channel analysis.	Remove capacitors to degrade PDS and make it vulnerable to power viruses; Use counterfeit components; Alter specifications of capacitors to make them ineffective under specific condition [19]; Modify configuration resistor of power management IC to disable under- or over-voltage protection [19].	Implement handmade antenna to enable remote side-channel information leakage; Insert power rail sampling resistor to enable power side-channel analysis; Implement transistors to power rail and enable fault injection attacks.
M3: Trace/Via	Expose sensitive inner-layer signal; Maliciously access enable/shut-down pin of the chip; Bypass the protection chips; Insert backdoor by connecting unused pins to sensitive signal.	Remove fault flag wires; Expose traces with sensitive signals; Increase wire coupling effects to leak information [14]; Alter the width, thickness, and space of traces to cause parametric failures in the field or to induce early failure [19].	Add shorting wires to Flash to interrupt boot and gain privilege [1]; Create the backdoor by connecting sensitive signal with fly-air wire.
M4: Fabrication Parameters	N/A	Change the thickness of stack layers to cause communication failure at specific frequency [14]; Use unqualified solder material to create premature failure triggered by specific thermal profiles [19]; Changing the characteristics of the dielectric to cause early failure [19].	N/A
M5: Probing/Accessing	N/A	N/A	Glitch supply voltage for DoS [21]; Glitch supply voltage for fault injection [23]; Maliciously access debug ports/buses [1, 8, 10, 20, 27, 28]; Power side-channel analysis [12].

PCB life into three phases: circuit design, fabrication and assembly, and in-field operation, which leads to an extension of attack models compared to ones of chip-level Trojans [29]. For *S1: Circuit Design* phase, we assume the design house is untrusted and the attackers can directly introduce modifications to the schematic. For *S2: Fabrication/Assembly*, we assume the foundry of PCB fabrication, the entity for PCB assembly, or the vendors of electrical components are untrusted. At *S3: In-Field* stage, we assume that some adversaries can assume the role of normal users to gain access to the product and perform attacks or apply malicious changes.

In terms of modifications, compared to the chip-level threats, board-level threats have broader representations of attacks. We categorize board-level malicious modifications into five types according to the main source of the malicious function. The *M1: μC/μP/FPGA* is referring to the threats caused by adding, altering, removing, or using counterfeit microcontrollers, microprocessors, or FPGAs. The *M2: Component* refers to adding, altering, or removing the passive components or the chips. If the attackers only apply malicious modifications to the signal wire or PCB traces/vias, it will be categorized as *M3: Trace/Via*. For example, the foundry may intentionally expose the signal lines containing sensitive information by adding vias, where the lines originally are hidden between PCB stack layers by the designer. The *M4: Fabrication Parameters* refers to the malicious alterations to the fabrication process, such as the stack layer thickness, the material of the board or solder, etc. And the *M5: Probing/Accessing* is the type of attack that the adversaries maliciously access the board port or probe the board-level signal. Examples of such attacks are using a modchip on Xbox to avoid game authority check [28] or power side-channel analysis on encryption chips. Both *M4* and *M5* can only be implemented at *S2* and *S3* stage respectively. Also note that some board-level attacks overlaps with chip-level attacks (e.g., side-channel analysis) at *M5*.

3.2 Rules for Board-Level Benchmarking

Due to the different properties exhibited at the board- and chip-level circuits, the methodology of building a board-level attack benchmark is distinct from building chip-level Trojan benchmark. To facilitate the development of board-level benchmarking, various design rules are created.

Rule 1: *The power supply requirements of inserted malicious circuits, unless self-powered, should be compatible with at least one of the voltage domains on the PCB.*

At board-level, different chips/components have different power supply requirements such as supply voltage level, maximum current, supply noise tolerance, etc. Thus the PCB developers need to design a hierarchical PDS creating multiple voltage domains to fulfill the requirements of board-level circuits. Similarly, the attackers need to make sure these requirements of the board-level malicious circuit, unless it is self-powered, are compatible with at least one of the voltage domains provided by the board PDS. Even if the malicious circuit is self-powered, the reference voltages, i.e. the zero potential, between the malicious circuit and the victim board need be aligned.

Rule 2: *The specifications, e.g. signal type, voltage level, frequency, etc., of the malicious circuits' I/Os should match the ones of the victim PCB circuit's I/Os.*

At board level, different signals can have different I/O specifications in a PCB design. For instance, the full-speed USB2.0 protocol defines that the signals are differential-ended running at $12Mbps$ and the voltage level of the signal line is in $0 \sim 3.3V$. While for full-speed I2C protocol, the signals are single-ended running at $400Kbps$ and the voltage level can be $1.8V$, $3.3V$, or $5V$. Therefore, when inserting a board-level malicious circuit, the attackers need to make sure the I/O specifications match the requirements of the victim circuit.

Rule 3: *The wire line accessibility and signal integrity of the victim circuit should not be violated when inserting malicious circuits, except that the violation itself is the mechanism of the attack.*

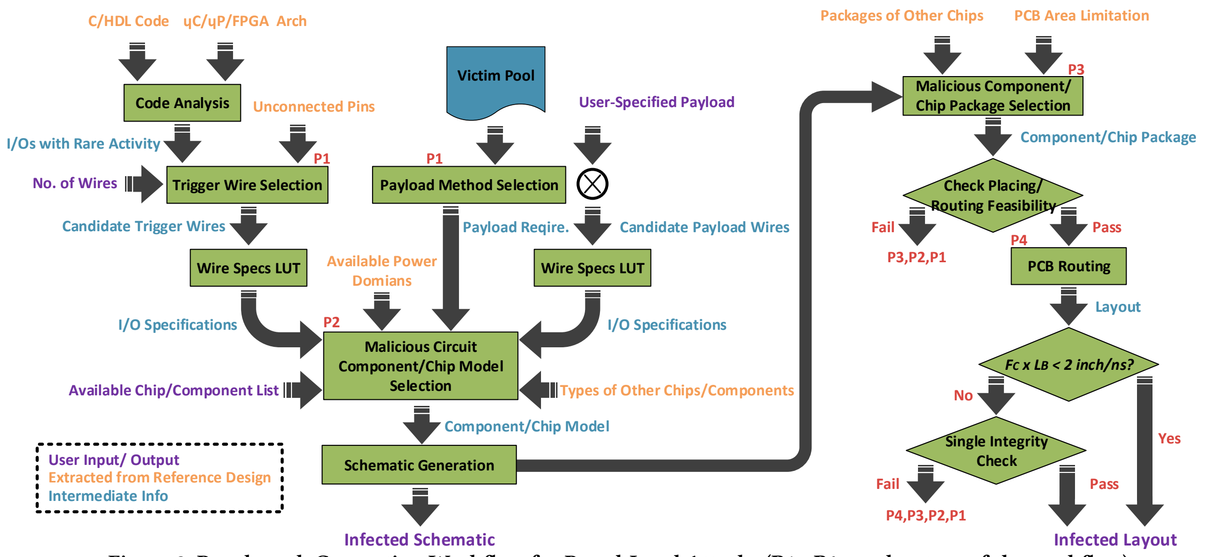

Figure 2: Benchmark Generation Workflow for Board-Level Attacks (P1 - P4 are the steps of the workflow).

For PCBs with high density or with few stack layers, the routing resources are highly limited and one signal cannot be accessed from every part of the board. Some signals can also be hidden by inner-layer traces and buried vias. Although one signal is accessible, the adversaries need to take care of the signal integrity. At high frequencies the wire lines are regarded as transmission lines and the impedance match needs to be satisfied. Modifications to the signal line may directly disable all the communications on this wire. Therefore, when benchmarking board-level attacks, wire line accessibility and signal integrity must be checked to ensure practical malicious circuits.

Rule 4: *The design of the malicious circuits, including the selection of the model, package of the corresponding chips/components, should blend in with the design style of the victim PCB circuits.*

A board-level malicious circuit can be built by various types of chips/components and accomplished by various methods. For example, there are multiple ways to perform a 4-input XOR logic operation: 1) build the digital circuit using discrete BJTs or MOSFETs; 2) utilize the 74 series logic ICs which contain XOR gates; 3) use a software program in a microcontroller, e.g. ATtiny5. For each method, there are many options for the models and the packages, which forms a large design space. However, to keep the malicious circuits reasonably sneaky, the design should be similar to the victim circuit from the perspective of chip/component types, packages, etc. For instance, if the PCB design is composed of microcontroller chips and passive components in surface mount packages (SMD), it is not reasonable to have the malicious circuit built by BJTs in the TO-92 package with three mounting holes, as the latter would be too conspicuous for detection. It should be noted that this rule does not apply to in-field attacks.

3.3 Workflow of Benchmark Generation

Based on these rules, we develop a methodology for generating benchmarks of board-level attacks and a workflow using the method (see Figure 2). The first step is to collect the signals with rare activities for trigger selection. The candidate trigger wires come from two parts: I/Os with rare activities and unconnected pins of the chips. Since the I/O activities of the chips are highly related to the programs of the chips, we perform code analysis based on the code and the architecture of the chips. Wires are further selected as candidate trigger wires (note that the number of trigger wires are decided by attackers). Then the specifications of the wires are looked up from the reference design. Meanwhile, the payload can be specified by attackers or picked from the victim pool, which is built based on the attack vectors mentioned in Section 3.1. In this step, both the traditional digital Trojans and the attacks based on analog properties are considered for the payload. Once the candidate payload wires are selected, corresponding specifications are also looked up from design files and datasheets. The malicious circuit is designed based on the information of trigger/payload wires specifications, available PDS voltage domains, types of other chips/components, and the available chips/components. Rule 1, 2, and 4 are incorporated in this step. If there exists a malicious circuit design that meets all these rules, an infected schematic can be generated.

For layout level benchmark, packages of the malicious circuit are selected according to the packages of other chips (PCB spare area is a constraint here). Rule 4 is considered in this step. Then based on Rule 3, the placing and routing feasibility are checked, following the signal integrity check. The signal integrity check is not necessary for the board with low speed. Therefore we set up a threshold to determine whether to perform the check, as:

$$k\frac{L_b}{6\ inch/ns} < \frac{1}{f_c} \qquad (1)$$

where f_c is the clock frequency of the signal, L_b is the board circumstance, and k is the safety factor. In this case we set $k = 3$. The constant 6 is the propagation speed of the signal at board-level $v_p = c_0/\sqrt{\epsilon_r} = 6\ inch/ns$, where $c_0 = 3 \times 10^8\ m/s$ is the speed of light, and $\epsilon_r = 4$ is the relative dielectric constant for the FR4 material that widely used in PCBs. If the infected layout passes the checks, the benchmark can then be generated. Otherwise, we will return to previous steps (marked as *P1*, *P2*, *P3*, and *P4*) and re-design the malicious circuit.

It should be noted that this process is finished manually for two reasons: The peripheral circuits of the chips consist of mostly analog-style circuits. Besides, at the board level, the parasitic effects

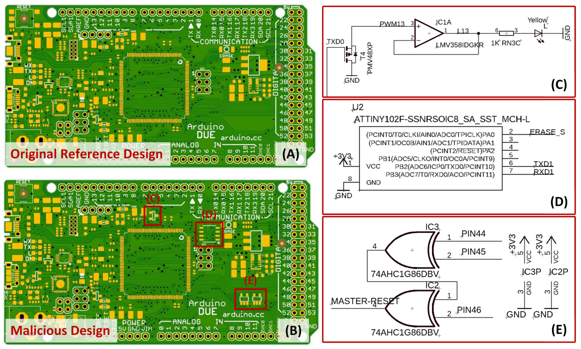

Figure 3: The PCB layout of (a) original reference design and (b) malicious design of Arduino Due; (c) A malicious MOSFET is connected to the UART transmitter to leak information through LED; (d) The maliciously programmed ATtiny microcontroller monitors the UART and generates Flash erase signal when triggered; (e) Two XOR logic chips output a signal to reset the board once triggered.

of the components and routing are much more obvious. Therefore, the automatic board-level schematic generation is not yet broadly adopted in practice. Secondly, as mentioned in Section 2.1, almost all PCB layouts are generated manually due to massive considerations. But the development of a tool to facilitate auto-generation will be the future work to help automate this workflow.

4 BOARD-LEVEL ATTACK BENCHMARKS

4.1 Reference Designs

To facilitate the development of board-level attack benchmarks, representative circuit boards are chosen as the victim designs. Open-source hardware projects are mainly selected due to their modern design styles, the availability of PCB designs as well as detailed documents, firmware, and/or application software. As a result, researchers can easily reconstruct the board-level security evaluation benchmarks. The complexity of the reference designs ranges from 2-layer boards with about 20 components to 12-layer boards with several hundred components, representing small to large electronic systems. The type of reference designs also covers a broad range of application space, including the digital wallet for bitcoin [7], microcontroller/FGPA development boards [2, 4], controllers for unmanned aerial vehicles [5], single-board computer [6], embedded AI development board [3], etc. Based on the workflow, we can generate a variety of different board-level attacks to compile a full set of benchmark suite. Due to the page limit, in the following sections, we only present two sample benchmarks. More benchmarks can be found at https://github.com/xz-group/PCBench.

4.2 Benchmark based on Arduino Due

Figure 3(a) shows the original PCB layout of the Arduino Due board [2]. It is a microcontroller development board based on the ATsam3x8e microcontroller. Arduino Due board is a 2-layer design with 141 components while its clock frequency is $84MHz$. The dimension of the board is $102\ mm \times 53\ mm$. In this benchmark, we insert three malicious circuits to perform information leakage, memory corruption, and deny-of-service (DoS) attacks, respectively. Each malicious circuit is designed by running through the workflow proposed in Section 3.3 for several iterations. All three malicious circuits can be inserted in *S1* and *S2* stages.

Figure 3(c) illustrates the schematic of the *M2* type attack where the information is leaked through the LED by inserting an extra MOSFET. The LED is used for indicating the output pulse-width modulation (PWM) signal of the microcontroller. The information from micro-controller passing through the UART0 port is leaked to the LED. Its clock frequency is $3.68MHz$ and the voltage level is $0 \sim 3.3V$. We select the P-channel MOSFET PMV48XP as the malicious circuit. It has $1V$ threshold and turn-on/off delay no more than $70ns$, fitting the specifications of the victim. When the UART0 TX is in idle mode, the voltage is $3.3V$ and the P-channel MOSFET is turned off so that the original function of the LED indicator circuit will not be affected. In addition, during the *P2* and *P3* steps (see Figure 2), we ensure the model and package of the MOSFET are the same as the ones used in the original design to keep the malicious function less suspicious. In schematic generation, the gate of the MOSFET is connected to the TX pin of the UART0 port. The source of the MOSFET is connected to the input of the LED driving amplifier. Attackers can thus monitor the flashing LED to snoopy the transmitted information.

Figure 3(d) shows the malicious modification of an *M1* type attack. A maliciously programmed microcontroller ATtiny102F is inserted into the design. This microcontroller supports UART communication with no peripheral circuits. We select the UART1 port as the trigger and the *ERASE* pin of the ATsam3x8e as the payload. ATtiny102F can operate under $3.3V$ supply voltage which is the supply voltage of the Arduino Due board. This also ensures the specifications of ATiny102F's I/Os are consistent with the victim's. Attacker can send a specific message through UART1 port to Arduino Due and trigger ATtiny102F. Upon receiving the triggering signal, a pulse longer than $200ms$ is generated to *ERASE* pin, forcing the ATsam3x8e to erase the embedded Flash memory, causing data integrity violation. The trigger message can be complex enough to avoid unintentionally triggering.

Figure 3(e) shows another *M2* type attack, which is similar to a chip-level Trojan. Through code analysis (see Figure 2), we find that three GPIOs are not used in the program. These GPIOs are then used as triggers with the payload the *RESET* pin of ATsam3x8e. Two 74AHC1G86 XOR logic chips with small packages are inserted to the design in the *P2* and *P3* steps of the workflow. By running the program that triggers the assigned GPIOs, the malicious circuit can enable *RESET* to induce hardware reset and crash the system. For all three malicious circuits, the layouts are generated after the P&R feasibility and the integrity check (determined by Equation 1). The malicious circuit included PCB layout is illustrated in Figure 3(b).

4.3 Benchmark based on A13-OLinuXino

As shown in Figure 4(a), the Olimex A13-OLinuXino [6] is a single-board Linux computer, which contains an Allwinner A13 Cortex-A8 processor. The board has 4 layers and includes 370 components. The dimension of the board is $12cm \times 12cm$ and the clock frequency of the processor is $1GHz$. We apply three malicious modifications to the design, covering *M1*, *M2*, and *M3* types, respectively. All of these malicious modifications are inserted in *S1* and *S2* stages. The modified PCB layout is illustrated in Figure 4(b).

Figure 4(c) shows the schematic of the *M2* type attack violating the availability of the system. The payload is the *N_OE* and *APS*

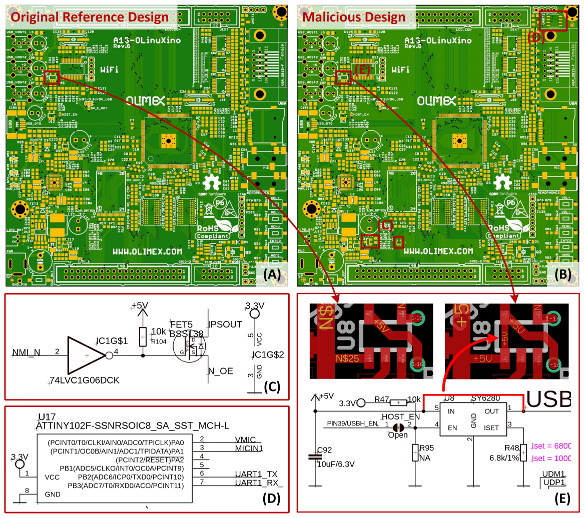

Figure 4: The PCB layout of (a) original reference design and (b) malicious design of Olimex A13-OLinuXino; (c) The malicious circuit receives the interrupt signal from a real-time clock chip and reset system; (d) The maliciously programmed ATtiny microcontroller snoops the microphone and leak information through UART; (e) A malicious trace is inserted to bypass power rail isolation.

pins of power management IC AXP209. Once the two pins are connected, the system is forced to perform a hardware reset. The *INT* pin of the on-board real-time clock (RTC) chip PCF8563 is selected as the trigger since this pin is rarely used. The malicious circuit is composed of an inverter logic chip 74LVC1G06, an N-channel MOSFET BSS138, and the pull-up resistor. During the *P2* step (see Figure 2), we ensure the I/O specifications of the malicious circuit are compatible with the victim's. For the *P3* step, The model and the package of the MOSFET is also the same as the ones used in the origin design. Attackers can take advantage of the alarm function of PCF8563 to enable a time-controlled attack. Once triggered, PCF8563 outputs a low pulse at the *INT* pin. The MOSFET will then be turned on, connecting *N_OE* and *APS* pins to reset the system. Signal integrity is validated in the generated PCB layout.

Figure 4(d) demonstrates the *M1* type attack based on maliciously programmed ATtiny102F for snooping the microphone of the system. ATtiny102F has embedded ADCs which can monitor the analog signals of the microphone input and the information can be transmitted through a board-level UART port. Similar to the previously presented malicious circuit based on ATtiny102F, its power supply is available from the board-level PDS. The P&R feasibility are validated while the signal integrity check is passed according to Equation 1. Figure 4(e) represents an *M3* type attack, which causes a DoS attack. In the USB-hub block of A13-OLinuXino, a power distribution switch chip SY6280 is used to electrically isolate the core power supply and the power for USB in one direction. The attackers can short the input and output of SY6280 through a short PCB trace, which disables the protection for over-voltage attacks through the USB port.

5 CONCLUSION

In this paper, we provide a systematic framework to classify and identify possible board-level attacks and develop a practical and comprehensive benchmarking method for such attacks. Practical benchmarking rules are developed to ensure the success of implementing attacks into the victim devices based on the detailed the attack vectors we have summarized at the board level. Following these rules, a workflow of generating board-level attack benchmarks is developed. Two sample board-level security benchmark are demonstrated with 6 types of attacks.

ACKNOWLEDGMENTS

This work was partially supported by DARPA.

AVAILABILITY

A suite of board-level attack benchmarks using the proposed method is released at https://github.com/xz-group/PCBench.

REFERENCES

[1] 2017. All Your Things Are Belong To Us. https://www.exploitee.rs/. DEF CON.
[2] 2020. Arduino DUE. https://store.arduino.cc/usa/due.
[3] 2020. BeagleBone AI. https://github.com/beagleboard/beaglebone-ai.
[4] 2020. CIAA-ACC. https://github.com/ciaa/Hardware/tree/master/PCB/ACC.
[5] 2020. HADES Flight Controller. https://github.com/pms67/HadesFCS/.
[6] 2020. Olimex OLinuXino. https://github.com/OLIMEX/OLINUXINO.
[7] 2020. Trezor Digital Wallet. https://github.com/comit/TREZOR-pcb.
[8] Ben Blaxill and et.al. 2019. PicoDMA: MDA Attacks at Your Fingertips. https://i.blackhat.com/USA-19/Wednesday/us-19-Sandin-PicoDMA-DMA-Attacks-At-Your-Fingertips.pdf. Black Hat USA 2019.
[9] Jonathan Cruz, Yuanwen Huang, Prabhat Mishra, and Swarup Bhunia. 2018. An automated configurable Trojan insertion framework for dynamic trust benchmarks. In *Design, Automation&Test in Europe Conference&Exhibition(DATE)*. IEEE.
[10] Andy Davis. 2013. Using laptop docking stations as hardware-based attack platforms. https://media.blackhat.com/eu-13/briefings/Davis/bh-eu-13-Docking-Stations-Davis-Slides.pdf. Black Hat USA 2013.
[11] Thomas Jose Mazon De Oliveira and et.al. 2017. Detecting Modifications in Printed Circuit Boards from Fuel Pump Controllers. In *30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. IEEE.
[12] Colin Flynn. 2018. I, for One, Welcome Our New Power Analysis Overlords. https://i.blackhat.com/us-18/Wed-August-8/us-18-OFlynn-I-For-One-Welcome-Our-New-Power-Analysis-Overloards.pdf. Black Hat USA 2018.
[13] Louis Friedman. 2012. Phobos-Grunt Failure Report Released. https://www.planetary.org/blogs/guest-blogs/lou-friedman/3361.html.
[14] Swaroop Ghosh, Abhishek Basak, and Swarup Bhunia. 2014. How secure are printed circuit boards against trojan attacks? *IEEE Design & Test* 32, 2 (2014).
[15] Rop Gonggrijp and Willem-Jan Hengeveld. 2007. Studying the Nedap/Groenendaal ES3B voting computer: A computer security perspective. In *Proceedings of the USENIX workshop on accurate electronic voting technology*. 1–1.
[16] Andy Greenberg. 2019. A new proof-of-concept hardware implant shows how easy it may be to hide malicious chips inside IT equipment. https://www.wired.com/story/plant-spy-chips-hardware-supermicro-cheap-proof-of-concept/.
[17] Tamzidul Hoque and et.al. 2020. An Automated Framework for Board-level Trojan Benchmarking. *arXiv preprint arXiv:2003.12632* (2020).
[18] T Hudson. 2019. Modchips of the State. https://fahrplan.events.ccc.de/congress/2018/Fahrplan/events/9597.html. The 35th Chaos Communication Congress.
[19] Matthew McGuire, Umit Ogras, and Sule Ozev. 2019. PCB Hardware Trojans: Attack Modes and Detection Strategies. In *37th VLSI Test Symposium (VTS)*. IEEE.
[20] Dmitry Nedospasov and et.al. 2018. Wallet.Fail. https://wallet.fail/wallets/nanos/physical-design/.
[21] Dark Purple. 2015. USB killer. http://kukuruku.co/hub/diy/usb-killer.
[22] Jordan Robertson and Michael Riley. 2018. The big hack: How china used a tiny chip to infiltrate us companies. *Bloomberg Businessweek* 4 (2018).
[23] Thomas Roth and et.al. 2019. Chip.Fail. https://chip.fail/chipfail.pdf. Black Hat.
[24] Bicky Shakya, Tony He, Hassan Salmani, Domenic Forte, Swarup Bhunia, and Mark Tehranipoor. 2017. Benchmarking of hardware trojans and maliciously affected circuits. *Journal of Hardware and Systems Security* 1, 1 (2017), 85–102.
[25] Omer Shwartz, Amir Cohen, Asaf Shabtai, and Yossi Oren. 2020. Inner conflict: How smart device components can cause harm. *Computers & Security* 89 (2020).
[26] Mark Tehranipoor and et.al. 2017. Invasion of the Hardware Snatchers: Cloned Electronics Pollute the Market Fake hardware could open the door to malicious malware and critical failures. https://spectrum.ieee.org/computing/hardware/invasion-of-the-hardware-snatchers-cloned-electronics-pollute-the-market.
[27] Wikipedia. 2008. NSA ANT catalog. https://en.wikipedia.org/wiki/NSA_ANT_catalog#cite_note-28.
[28] Wikipedia. 2019. Xbox modding. https://en.wikipedia.org/wiki/Xbox_modding.
[29] Kan Xiao and et.al. 2016. Hardware trojans: Lessons learned after one decade of research. *ACM Transactions on Design Automation of Electronic Systems* (2016).