future internet

MDPI

Review

Exploiting JTAG and Its Mitigation in IOT: A Survey

Gopal Vishwakarma * and Wonjun Lee

Department of Electrical and Computer Engineering, University of Texas at San Antonio, San Antonio, TX 78249, USA; wonjun.lee@utsa.edu

* Correspondence: gopalvishwakarma19@yahoo.com; Tel.: +1-210-663-6442

Received: 30 October 2018; Accepted: 30 November 2018; Published: 3 December 2018

Abstract: Nowadays, companies are heavily investing in the development of "Internet of Things(IoT)" products. These companies usually and obviously hunt for lucrative business models. Currently, each person owns at least 3–4 devices (such as mobiles, personal computers, Google Assistant, Alexa, etc.) that are connected to the Internet 24/7. However, in the future, there might be hundreds of devices that will be constantly online behind each person, keeping track of body health, banking transactions, status of personal devices, etc. to make one's life more efficient and streamlined. Thus, it is very crucial that each device should be highly secure since one's life will become dependent on these devices. However, the current security of IoT devices is mainly focused on resiliency of device. In addition, less complex node devices are easily accessible to the public resulting in higher vulnerability. JTAG is an IEEE standard that has been defined to test proper mounting of components on PCBs (printed circuit boards) and has been extensively used by PCB manufacturers to date. This JTAG interface can be used as a backdoor entry to access and exploit devices, also defined as a physical attack. This attack can be used to make products malfunction, modify data, or, in the worst case, stop working. This paper reviews previous successful JTAG exploitations of well-known devices operating online and also reviews some proposed possible solutions to see how they can affect IoT products in a broader sense.

Keywords: Internet of Things; JTAG; exploitation; mitigation; IOT; a survey

1. Introduction

Nowadays, there is a lot of development happening in "Internet of Things (IoT)" and connected smart devices in the market. The Internet of Things (IoT) has been marked as "the next Industrial Revolution" by The Business Insider [1] due to the way it will change the way individuals live, work, engage, and travel, as well as how governments and organizations associate with the world. Moreover, Gartner, Inc. (Stanford, CT, USA) [2] predicts that nearly 20.4 billion IoT connected devices will be online by 2020. IoT framework mainly comprises sensing, communication, data collection in the cloud and processing, and delivery of data back to the user. Here, sensing world parameters such as temperature, pressure, heart rate, blood pressure, GPS locations, etc. is usually performed by IoT node devices (sensor devices) which inject information into the worldwide web directly or indirectly. Besides, node devices (sensor devices) are unintelligent devices with less hardware complexity, high power efficiency (i.e., battery operated), and often lower cost that only sense world parameters and pass them to the cloud for further processing. Importantly, the vast majority of these devices are physically available to the public, thus illegitimate parties could influence data, firmware binary file and memory footprints. This attack can be classified into a physical attack which can influence the behavior of a product by playing with the actual piece of hardware. Thus, this introduces a noteworthy threat to embedded sensor nodes. Skorobogatov [3] already evaluated possible physical attack scenarios, as explained below for embedded devices.

Theft of Service could be significant as malicious users could get access to sensitive data within the chip. For example, Satellite TV is a kind of IoT node that is directly connected to the service provider. An attacker could modify security to gain free services, which could cause huge losses to the service provider. In addition, *cloning and IP Piracy* is a notorious challenge faced by the electronic industry today. Various gadgets get cloned in the industry through physical attacks by getting internal memory contents such as device ID, unique keys, etc. Due to this, not only development cost is avoided but also the sellers of these cloned device receive a significant amount of profit. This practice is unlawful since they use the intellectual property of a company who already invested time and money.

Denial of Service can be used as an attack by devices to satisfy selfish motives. For example, IoT nodes can be easily reverse-engineered to send fake data to deceive the destination server.

These physical attacks can introduce huge threats not only to customers but also to the companies that are bound to protect people's personal data. Now, these physical attacks can happen through standard hardware interfaces such as USB, JTAG, LAN, WLAN, Serial interface, etc.

JTAG [4] is an IEEE standard developed in the 1990s for debugging, updating and storing firmware on the chip and still this standard is widely used in the industry. Usually, IoT nodes are small embedded devices that have a program that runs continuously in a loop called firmware. These application programs are directly written into the chip through either serial or JTAG debugging interface. Manufacturers keep JTAG port on PCB board to test/validate if every single electronic part is appropriately mounted or not, as well as for upgrading the firmware in the future. There are many JTAG debuggers available in the market which cost from ten to a few hundred dollars, which makes it even easier for attackers to get access to the internals of a chip and firmware. If we consider this scenario with thousands of sensor IoT devices, then product manufacturing companies need to provide not only secure data transmission between multiple devices but also should be more resilient against physical theft and the intellectual property of the device. Moreover, the IoT business model is not only about one time product selling but also includes monthly/yearly service subscription model such as TV channel subscriptions, Google assistant services, etc. This paper studies how JTAG port access to hardware could be severe and huge in terms of IoT endpoints nodes that are less complex, battery operated, accessible to the public and importantly those will be in millions of numbers. This paper covers both various exploitation of products through JTAG port and previous proposed solutions that make JTAG access secured.

This paper is further organized as follows. In Section 2, we describe the basic background of JTAG IEEE 1149.1 standard. Section 3 includes various JTAG exploitations that have been demonstrated before from commercial products to industrial products. Section 4 contains different methods that have been proposed to make JTAG interface secure. Section 5 provides the discussion of the topic and Section 6 gives the conclusion and future insight in terms of JTAG interface and IOT infrastructure.

2. JTAG Background

JTAG [4], commonly referred to as boundary-scan and defined by the Institute of Electrical and Electronic Engineers (IEEE) 1149.1, was developed for validating proper assembly of components on PCB boards in the 1990s. JTAG is mainly used by IC manufacturer at the developing and testing stage as well as giving access to the internal subsystem of IC, or for breakdown examination and debugging. When this standard became popular for the first time, it became necessary to differentiate between JTAG compatible devices and normal devices. Thus, the committee came up with a standard hardware description language called "Boundary Scan Description Language (BSDL)", which unveils the internal resources of an IC such as logical port description, scan port identification, package pin mapping, entity declaration, boundary register description, etc. which can be unique for each IC manufacturer. JTAG consists of

- *Test Access Port (TAP)*: This port gives easy access to test functions built into a component. It has four input ports, i.e., Test Clock (TCK), Test Mode Select (TMS), Test Data Input (TDI), Test Reset (TRST) (optional), and one output port, i.e., Test Data Output (TDO). Here, TCK, TMS and optional

TRST are broadcast signals, whereas TDI builds serial chain, in which test signals are injected and get test output signals at TDO port. Only these four pins are required to inject test signals on PCB regardless of the number of components on PCB.

- *TAP Controller*: The TAP controller is a finite state machine which is responsible for the behavior of JTAG boundary scan logic in the chip. The TAP controller generates signals to control the operation of the test data registers, instruction registers, and associated circuitry. The TAP controller finite machine has two main paths for setting and retrieving information from either the instruction register or the data register.
- *Instruction Register*: The "Instruction register" decides on the operation mode of the Boundary Scan IC. The IEEE standard has three main instructions (modes): BYPASS, SAMPLE/PRELOAD, and EXTEST.
- *One or more data register(s)*: These registers are used to read-out information in the component. There are a minimum of two mandatory registers described in this standard, i.e., Boundary Scan and bypass registers. There are data registers which contain "device ID" and are known as "idcode" register.

JTAG specification enables testing of all components which are not time critical such as resistors, crystals, driverICs, logic gates, reset ICs and even RAM ICs or Flash ICs (parallel as well as serial). This specification offers a low cost, less time requirement for the testing operation, a simplified connection architecture and hence widely accepted and followed in the industry.

3. JTAG Exploitations

Already many JTAG exploitations have been demonstrated by various researchers before. In this section, we discuss some of them.

3.1. Exploitation of JTAG Using Physical Pin Modification

Karri et al. [5] (2010) explained different kinds of attacks such as sniffing the TDI or TDO signals, modifying the TDI or TDO signals, controlling the TMS and TCK signals, or accessing the keys used by testers. JTAG has a mode of operation called boundary scan, which tests all different modules of a PCB in serial fashion. Here, the TDI pin is used to inject test signals or secret data into "n" number of devices, which are connected in series such as memories, sequential or combinational circuits, etc. If the tester or user is injecting some secret data into the TDI pin, then that data can be sniffed by a device such as memory, controller, etc. that is already present in that series of devices. This is called a "sniffing attack". These attacks are most likely possible in IOT nodes since these nodes may contain memory to store sensor data or encrypted global unique sensor ID (i.e., unique device ID). Three methods are used in this paper, namely hash function, message authentication code (MAC), and stream cipher, in the proposed defense against JTAG attacks. In this paper, an experiment is conducted to ensure the validity of various components on PCB and the confidentiality of communication between multiple devices and JTAG debugger.

3.2. Exploitation Using JTAGulator Tool

According to Drake et al. [6], it is one of the important tasks to find open pinouts first that are kept by manufacturers on PCB board for future debugging and improvement because PCB manufacturers usually hide JTAG interface points from user to provide basic security rather than exposing it directly. Here, the authors explained the JTAGulator, an open source hardware tool developed by developer Grand [7]. The authors explored the open pins that are indirectly available on PCB to normal user through brute force attack. Once the hacker gets to know the pinout from PCB board through manual inspection, then the attacker can easily get access to the firmware or bootloader. JTAGulator board assists all users in identifying OCD (On Chip Debug) interfaces from test points or components pads which are available on PCB boards. The JTAGulator tool can be connected to a PC via USB interface as

well as gives interactive CLI for the user. Figure 1 shows JTAGulator image which has USB interface for PC connection.

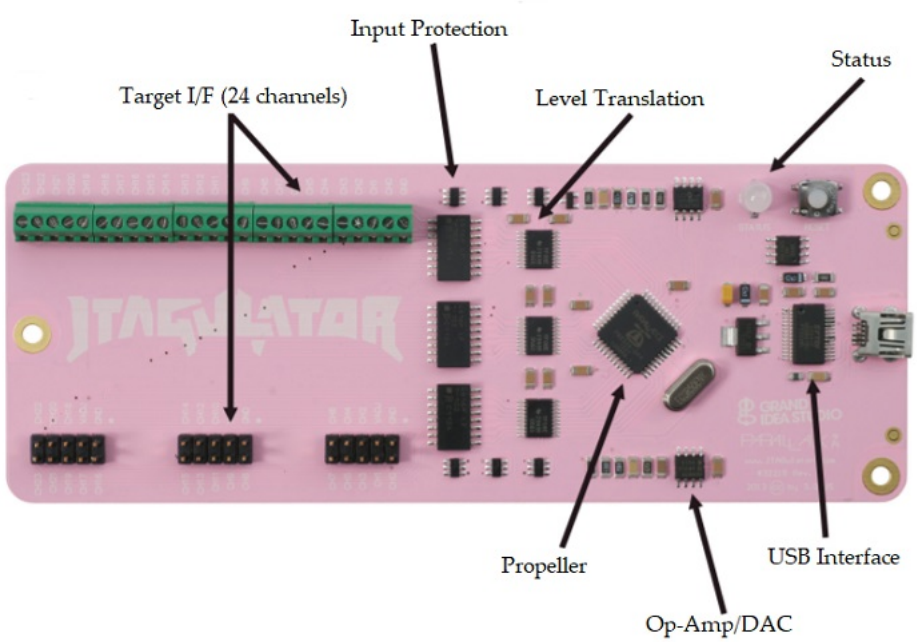

Figure 1. JTAGulator device [8].

Here, Android-based HTC Dream cellphone and a Luminary Micro LM3S8962 ARM Evaluation Board are used in the experiment for exploitation. Users can set off target device operating voltage levels, which facilitate connecting any device to JTAGulator. the authors configured it to 3.3 V for HTC dream mobile. JTAGulator is able to perform both boundary scan and IDCODE scan. the IDCODE scans are used for identifying JTAG slaves. In the experiment, JTAGulator continuously injected signals at various pins and recorded responses. The authors used these responses to identify pins on PCB boards through CLI interface. Based on pin information, the authors used J-Link debugger tool from "segger" to get access through JTAG interface. This method could be the starting point for any illegitimate user to get JTAG access into IOT node with open source registered toolsets.

3.3. Exploitation of Tiny OS and Mantis OS Using Open Source Software Available on Internet

Hartung et al. [9] conducted experiments to show how physical attacks including JTAG attacks can be used to compromise sensor nodes. In this paper, researchers exploited TinyOS (an event-based operating system) and Mantis OS (a multi-threaded operating system). Tiny OS is an open source, BSD-licensed operating system developed specifically for low-power wireless devices and according to the official website it has averaged 35,000 downloads in a year. Mantis OS is popular due to its feature such as multi-threading capability with a small memory footprint (500 bytes). The author experimented on sensor wireless node "Mica2 mote" manufactured by XBow. the Mica2 internally uses an 8-bit processor (Atmega128) that runs on 4 MHz, contains 128 KB of instruction flash, 4 KB EEPROM, and 4 KB SRAM. The authors used AVR ICE JTAG programmer to dump and retrieve HEX code.The open source programs such as linux based standard GDB debugging tool, UISP, avr-objcopy and avr-tool are used to conduct this experiment. The authors got object file through serial interface connected to the PC. The Hex converter tool is used for reverse conversion of the assembly code. Generally, SRAM is considered as the safest among all memories to store keys and sensitive information due to its volatile nature. This experiment showed that the researchers not only collected sensitive data from the flash memory and EEPROM but they also copied all data from the SRAM. The authors showed

that, by using simple hardware and free available software tools, one can get a memory footprint less than a minute from lightweight operating systems.

Moreover, there has been rapid growth in development of operating systems specifically designed for small-sized IoT embedded platforms such as Windows 10 IOT OS, ArmMBED OS, Yocto project, etc. Devices such as routers, cameras, wireless sensor nodes, etc. now contains minimal sized OS (as small as in Kbytes) for managing hardware/software resources. Since these tiny OSs are used in many devices; such computing-feasible devices can be used as the bot to execute malicious code, or to scan to find other vulnerable devices. In comparison to this, recently, a self propagating botnet malware known as "Mirai" became very popular in industry and researchers community. The Mirai malware attacked website of "Krebbs; a software security consultant company", with 620 Gbps of traffic in September 2016, which was one of the huge DDoS attacks (Distributed denial-of-service). This Mirai botnet code infects devices such as routers and IP cameras that are still using their factory default username and password. Usually, manufacturing companies of these products expects that the user will change default credentials but in practice it has been observed that users are quite ignorant about its severity. Kolias et al. [10] discussed the severity of this DDoS attack. In short, there are three main modules in this malware: Bot module, ScanListen module, and Load module. The Bot module scans other online devices in the network, ScanListen module sends collected information to load module and Load module performs actual attack. Due to this attack, Hangzhou Xiongmai Technology recalled 4.3 million Internet-connected camera products from the U.S market [11].

3.4. Exploitation Using Interrupt-Oriented Bugdoor Programming Method

Tan et al. [12] exploited embedded firmware through JTAG interface. They explained the concept called "bugdoor". "Backdoor" is a popular term used in software industry for a special code snippet left in the program to secretly control future debugging. However, if attacker gives proper inputs to the target program in the device, then the attacker can inject malicious code in the firmware causing unintentional behavior of device or stealing sensitive data. Generally, server exploitation considers getting shell (root) access, whereas microcontroller firmware access could be more devastating, since attacker can change the device behavior, steal protection keys and more. It can go unnoticed by the owner of the device so that attacker can continue stealing data and can keep a watch on the device. Authors targeted Texas Instrument's MSP430, a 16-bit ultra-low power microcontroller for exploitation through JTAG interface. The author preferred "interrupt oriented programming tricks", e.g. microcontroller-based architecture frequently uses interrupt input either from internal modules such as timer/counter, status registers, etc. or from external devices. Normally, external interrupt can easily be triggered by attacker by changing the pin states. After this, microcontroller generally executes interrupt handler code snippet. Similarly if the user carefully injects a specific code snippet at the start of execution of interrupt handler, then target microcontroller can also have programmed or program flow can be changed. Usually desktop, servers use high clock speed processors however, microcontroller runs on significantly less clock speed and usually downclocked in power down mode. This results in it being feasible to inject well timed code at execution of interrupt handler called "Interrupt-oriented Programming (IOP)". When interruption occurs, there is a state of change between interrupt handler code and some accumulation of state in interruption called "IOP primitives". The authors explained various IOP primitives such as state accumulation, memory write, arithmetic, stack growing, and stack alignment. To exploit TI's MSP430F2618, the authors created a setup in which two microcontrollers are connected to the target device. One microcontroller is connected through JTAG interface and other is connected to send signals to target device. The author performed three different exploits by writing a simple firmware on a microcontroller to generate timely nested interrupts. First, the state accumulation exploit acquires state changes information when interrupt occurs on target device, through which program behavior change has been done. Second, the loop execution exploit changes memory contents by nesting interrupt loop. Third, the stack growing exploit is done when the target device receives continuous interruptions, in which stack contents grow by

2 bytes each time. The authors exploited open source lightweight operating system, TinyOS. The team has added few lines of code on USART1 to receive an interrupt request, which makes interrupt handler re-entrant. As a result, the author showed stack growing primitive. The team also performed static analysis of the firmware of five MSP430-powered devices, FET430UIF, EZ430U, MSP430FET, EZ430URF and GoodFET, using IDApro [13] debugger and wrote automation using IDAPython [14] scripts in which they did not find any significant results. However, in the hardware analysis, they observed the CPU state information that they received, which was modified when JTAG reset occurred, was not an "actual" reset interruption triggered by a signal edge on the RST/NMI pin. This is because JTAG debugger resets after every gain/loss action of JTAG control on target device. This is a contradictory observation with official document provided by MSP manual [15]. To bypass this JTAG reset problem, the authors wrote firmware that reads register values just before resetting into some fixed memory location and retrieves it afterwards to compare it with current CPU state. They observed that MSP430F2618 loads fixed values into CPU registers after each reset except PC and SR registers, so most of the time it is predictable. Overall, these firmware code snippets and experiments demonstrated concept of interrupt-oriented programming as well as changing behavior of program execution, even though these changes are not significant but cannot be ignored in the future.

3.5. Exploitation of Modern Game Console

DeBusschere et al. [16] listed past successful attacks on the game consoles such as XBOX 360 and Sony PS3. The authors reviewed three significant XBOX 360 exploits so far, i.e., the King Kong glitch/JTAG exploit, Timing Attack, and the Glitch Attack in this report. The King Kong glitch, also called JTAG exploit, was exposed in late November 2006. A Kernal update (4532) was exposed to a privilege escalation vulnerability [17] in which an unchecked memory exports in some games has been done by unsigned shader memory. Generally, Xbox 360's security system is built around a hypervisor concept. The hypervisor is responsible not only for memory access but also offers encryption and decryption services. It manages policies for code that will be executed in either privileged or unprivileged mode. Normally application executable code interacts with hypervisor via "sc" (syscall) instruction which switches machine into hypervisor mode. The authors explained "cmpwi" instruction which caused problem for memory management. XBOX contains basically two types of memory, i.e., encrypted and unencrypted. Here, "syscall jump table" is kept in encrypted format in the memory. When "cmplwi" is executed, it is responsible for memory jump which compares only the lower 32 bits of the system call number. The upper 32 bits are being ignored where other instructions operate on 64 bit values. As a result, hackers intentionally injected hypervisor syscall with preloaded register values via unprivileged application code. Moreover, these memory operations are used to write the result of pixel shader (graphic card memory valus) into main memory while execution of graphics related operation. However, the hacker used this shader to hold some register values and instructs kernel to jump on certain location in memory. Due to this, the hacker could get full control access to hypervisor and run unsigned code. This exploit is also called DMA (Direct media access) attack since hacking can be done using the authorized memory access tool such as JTAG interface. In the computer architecture industry, it is well known that the SMC (System Management Controller) could trigger a DMA because SMC cannot write into NAND registers which are required to enable correct DMA. A group of hackers reverse engineered GPU JTAG to see if DMA can be triggered by hardware or not. JTAG could not be used to trigger the DMA since generally interface gets disabled in early boot process. The smart hackers tried to get JTAG access with DMA address and SMC is used to initiate DMA. This attack was well spread in hacker community till Microsoft has patched in all new Xboxes and kernel update. Due to this, Microsoft as well as game design companies have faced significant loss.

3.6. Baseband Attacks: Remote Exploitation of Memory Corruptions in Cellular Protocol Stacks

The GPRS/GSM 1st generation system was launched and commercialized in the 1990s. At that time, companies were charging significant money even for incoming calls. However, the consistent and advanced development of cellular network (e.g., 2G, 3G, and 4G) not only made it available to people throughout the world but also companies now mostly only charge money for Internet service on mobile phones. Currently, IOT power efficient protocols are emerging, such as LoRaWAN, NBIoT, InGenu, and SigFox, in the market. These protocols are currently in the first phase of the development and industry sees this as one of the emerging business model. Some of the protocols use open radio band for the communication such as LoRaWAN protocol which uses 433 MHz/868 MHz/915 MHz open frequencies but the newest IoT radio protocols such as NBIoT have been developed around existing cellular technology since cellular network has been widely deployed already around the world.

Weinmann [18] experimented and exploited memory corruption in 2G protocol stack. In his experiment, the author set up an illegitimate BTS (Base Transceiver Station) to target mobile station (MS) in the vicinity. This BTS broadcasts fake network and surprisingly mobile station (MS) connects to this BTS not only due to strongest signal received by it but also GSM does not provide any protection against fake BTSs. While experimenting with GSM and UMTS stack, the author found some of exploitable bugs: (1) When the cell phone accesses network service for first time, a 32-bit long TMSI number is assigned to handset which is variable by nature. By sending longer TMSI such as 128 bytes, it caused iPhones/mobile phones to crash. (2) For authentication between BTS and MS, BTS sends challenge to the handset. This challenge is a 16-byte value called RAND value for GSM and AUTN value for UMTS. The author forcibly transmitted three messages, each 16-byte long. These messages are stored in the buffer and apparently cause stack overflow. To show these exploitation, The JTAG access is required to get debug messages from firmware. Most chipsets in mobile phones have configurable JTAG access nowadays. However, access to PCB board through JTAG interface is provided by OEM manufacturer not chipset manufacturer. The author experimented on mobile phones such as HTC dream and Apple iPhones to get JTAG access. Before, HTC dream has provided limited JTAG access but author changed the register flag by setting breakpoint at secondary boot process whereas iPhone JTAG was completely locked down from the company itself giving more security, but debug messages and baseband crash report logs can be found at `/Library/Logs/CrashReporter/Baseband` location. The author exploited HTC mobile through JTAG interface by turning ON auto-answer facility provided in the 2G protocol stack. The TMSI and AUTN bug exploitation method is used in this experiment. When AUTN payload is forwarded to testcall UDP port after establishing channel using testcall command, this bug turns ON auto-answer facility without understanding of user. The TMSI overflow bug concept which is described earlier is used to configure auto-answer facility in iPhone. A LOCATION UPDATING REQUEST is sent by mobile to the BTS as soon as it connects to fake network. Here, the BTS sends intentional LOCATION UPDATING ACCEPT reply with some payload. This enabled auto-answer call in iPhone. These experiments showed successful exploitation of memory corruption in GSM baseband software stack through JTAG interface. In case of IOT protocols and its stack development, these considerations should be considered to make IOT products secure.

3.7. Attacks on Programmable Logic Controllers (PLC)

In a broader perspective, IoT infrastructure will be wide enough covering not only commercial, medical, military and automobile industry but down the line that will also be used in industrial field to make an improvement into industrial process management. Schuett [19] explained a JTAG exploitation in the Supervisory control and data acquisition (SCADA) systems which is dominantly used to monitor and control huge system infrastructure such as electric power transmission lines, water distribution systems, gas and oil pipelines, etc. The programmable logic controller (PLCs) gathers data and interacts with sensors and valves for streamlining huge industrial process. For example, nowadays, the vehicle manufacturing factory assembles 20 cars a minute on an average, so if any single PLC hardware is being attacked and shut off for even for an hour, it could cause a huge

loss for company. According to the author, attacks on SCADA systems have predominantly focused on network protocols (e.g., MODBUS, Ethernet, and CAN protocol) and high-level systems where human–machine interface occurs, however it is equally important to consider security of PLCs and monitoring field devices in SCADA systems.

The author experimented on Allen-Bradley's Controllogix 1756-L61 PLC controller for firmware exploitation. This experiment used commercial tools such as ControlFLASH for uploading firmware to PLC, "IDA-Pro Hex Rays" for disassembling the firmware, RealView ICE as JTAG hardware debugger and ARM Development Studio v5 in this experiment. The author followed three steps for exploitation: firmware acquisition, analysis of hardware and its components, and analysis of disassembled firmware. Here, the RealView ICE debugger is used in auto configuration mode to get all ARM related information. According to author, the hardware analysis is mandatory to identify the instruction set used in firmware. The author followed static and dynamic analysis for firmware by using IDA Pro disassembler. Previously developed IDA Pro scripts by Santamarta [20] and Basnight [21] are used to identify ARM functions. During the static analysis, the location of firmware images in volatile memory is identified where initial jump to the entry point of the firmware is occurred. During the dynamic analysis, firmware execution paths are identified by using JTAG interface through which memory breakpoints can be set and test read/write access. This JTAG interface can also dump memory and register values before PLC goes into fault state which helped understanding conditional flags of hardware status. While working on the device, author identified vulnerable firmware sections through various diagnostic tests such as processor mode diagnostics, processor register r0 diagnostics, CIP (Common Industrial Protocol) communication protocol diagnosis, etc. In processor mode diagnosis method, the author was able to get current working mode of system. Usually, PLC has four mode settings: RUN, PROGRAM, REMOTE RUN, and REMOTE PROGRAM. Registers r0, r1, r2, r3 and r4 are kept observing to see the values while uploading and running program using RSLogix. The author observed that "cpmode" is being called continuously. Cpmode from JTAG architecture acts as an event handler which is responsible for above four mode settings in PLC. Thus, the author suggested that continuous execution of cpmode function could at least hang the system. Next, the author worked on Allen-Bradley PLC's CIP communication protocol. This protocol is object-oriented protocol which is distributed over serial or ethernet link across all parts of system. The author found that the breakpoint can be triggered if identity object request is sent through CIP messages. All these diagnosis methods are combinatorially used to influence PLC system. In addition, the author initiated DoS attack, which is triggered by CIP commands, cpmode check and writing a sentinel value to an unused area of flash memory. Thus, It keeps executing cpmode function due to presence of sentinel value in memory which forces PLC to go in fault state. As per author insight, recovery can be done through updating device firmware or sending device back to manufacturer for repair.

3.8. Attacks on Android OS

Fault injection is a well known term in the software industry for when the robustness of software is tested by injecting various code paths and analyzing behavior of the code. In software, there are mainly two types of methods, namely compile time injection and run time injections, and various tools that utilize these methods are available to date. These techniques are predominantly used in the software field to test the flow of codes/algorithms. However, Majeric et al. [22,23] combined this software field with a hardware interface (JTAG) to exploit the recent Linux Android OS 6.0.1 (Marshmallow). As we know, OS contains two major spaces for organizing basic OS functionalities, i.e., user space and kernel space. Usually, kernel space access is given to authorized administrators, developers or certified applications to use/manage system resources. To do this, the OS hides all details of processes/applications to normal users to keep security intact. Here, the authors used the JTAG tool to get access of internal memory and injecting the faults. This experiment have successfully changed modes from normal access (restricted) to root access (unrestricted) which is known as a

privilege escalation attack. This kind of attack can put systems at a huge risk of sharing sensitive data and, moreover, can corrupt the whole system.

In Android OS, `/proc/sys/kernel/kptr_restrict` provides a security feature which hides the addresses of kernel functions. The kernel function `sys_setresuid` provides the attribute which denotes whether the application contains root privilege or not. The Author modified this bit to get root privilege to an unauthorized application. In this experiment, the authors used a 32-bit ARM Cortex A9 processor which had Android OS version 6.0.1 Marshmallow along with Android debug tool (ADB) (command line tool), and Lauterbach Trace32 JTAG tool. This experiment is performed with three main steps: (1) find the address of `sys_setresuid`; (2) modify this system call; and (3) change root privilege from user space. The first two steps are carried out with the JTAG probe. %pK is kernel specifier which hides kernel pointer. If an attacker changes this %pK specifier with %p, then the system starts to display its symbol pointer regardless of its privileges. A non-privileged user can now ask for addresses with the command:

```
$cat /proc/kallsyms | grep setresuid
```

This system call address is easily traceable in the memory footprint. Now the attacker can modify assembly code as per his wish. In the third step, root privilege is assigned from user space through ADB command line by executing simple C program. This is achievable because ADB command line provides a new terminal window with root identity. In this whole experiment, JTAG plays a crucial role since it can see internals of memory directly. If the attacker is skilled and knows about system internals, then even the latest smartphone's security can be breached.

4. JTAG Security Solutions

Many solutions have been proposed to make JTAG secured. In this section, we discuss JTAG attacks solutions and applicability of solutions on IOT nodes.

4.1. Based on Physical Unclonable Functions (PUF)

Suh et al. [24] (2007) provided unique lightweight "Physical Unclonable Functions (PUFs)" scheme to establish a secure communication between master and all slaves. The PUF is an innovative primitive that prefers looking key secrets in complex physical characteristics rather than storing them in digital memory. Usually, the cryptographic secret keys are stored in the non-volatile digital memory and those keys are susceptible to the physical attack [25–27]. Moreover, continuous battery powered, resource hungry add-on tamper sensing circuitry is often required for higher memory security. On the other hand, each IC has a different microstructure that depends on random physical factors introduced during manufacturing even if same IC manufacturing with same process is being followed. Here, the PUF generates volatile secrets when chip is powered on and running since it considers random delay characteristics of wires and transistors. This experiment proposed challenge response-based solution also known as Ring Oscillator (RO) PUF to provide JTAG security. The challenge sequence fed to the IC circuit for pseudo random generator then PUF delay circuit is evaluated k times. A ring oscillator is simple circuit which oscillates at a certain frequency. However, this ring oscillator in each IC gives slightly different frequency due to the manufacturing variation. Thus, even if the same input is challenged to different ICs, it has different response output. In addition, this ring oscillator is highly sensitive to temperature and voltage changes. Hence, the PUF output could be distorted while generating keys. Thus, the authors proposed the solution of creating challenge response pairs to minimize the error rate. These pairs are created by comparing two ring oscillators that have a huge frequency difference called masking. This ring oscillator solution gives unique key generation but limited number of challenge pairs. To overcome this, the authors proposed two ways to create many challenge response pairs. First, extensions of oscillators need to have configurable delay paths. Here, a challenge can select the configuration path that is within a delay loop so that distinct challenges gives distinct oscillation frequency. Second, a the facility of programmable logic such as FPGA, which has

inverters, look-up tables and wires, can be used to determine oscillator configuration. To authenticate and authorize trusted party with this solution, the manufacturer of IC has to keep record of different challenge response at the time manufacturing process. If response of IC while getting JTAG access is matched with record, then only JTAG access is given to the party considering it as trusted party. To prove this, the authors experimented on ASIC implementation of the 64-stage arbiter PUF on 15Xilinx Virtex4 LX25 FPGAs (90 nm) which shows PUF can be used secret key generation and authentication. The authors successfully conducted "inter-chip variation" and "intra-chip variation" experiments to test uniqueness and reliability of PUF output. The PUF is not only lightweight and unique but also ideal for extremely resource constraint platform such as RFID and IOT nodes. Another advantage of PUF is that it does not require any special manufacturing process or programming and testing steps.

4.2. Based on Public Key Cryptography

Das et al. [28] (2013) implemented a secured JTAG solution with asymmetric key based approach by using Public-key Cryptography (PKC) scheme. Specifically, an Elliptic Curve Cryptosystem approach is used over other public PKC schemes (such as RSA) due to small key size, reduced storage requirement, and transmission requirement. According to the authors, ECC (An elliptic curve cryptosystem) based Schnorr protocol provides "no information leak" property while running protocol. The researchers gave full proof implementation in hardware along with detailed timing results to prove this. In the draft, the researchers discussed about possible application scenario of attacker model. In the Manufacturing Test/Firmware or Code Update at Manufacturer's end scenario needs to have one way authentication since this environment is controlled whereas In-the-field Update, Debug and Test is considered as uncontrolled environment, hence needs to have a two-way authentication, i.e., device side as well as server needs to be authenticated. The authors preferred ECC over RSA to solve key management problems because ECC offers same security as RSA but with small footprint. The experiment also proposes offline and online mode with server as two modes of operation. The online mode uses Authentication Server (AS) to validate certification with the public key. The disadvantage of this mode is that it needs server access while getting JTAG access at device, which is not a practical approach for most of the cases. In context of these modes, the offline mode seems more promising in the IOT infrastructure since nodes might not always connect to server directly. This offline mode uses clock from the device to validate expiration date of certification. In this paper, man in middle attack is also prevented by a Elliptic Curve Digital Signature Algorithm (ECDSA). Two different implementations are provided in this paper: projection of coordinates and affine coordinates. Projection coordinates method uses few inversion modules, while affine coordinates method uses dedicated inversion module based on extended Euclidian algorithm. This helps efficient execution of ECDSA. In addition, integration of the ECC based Schnorr controller and ECC point multiplier with the JTAG interface along with the other modules is also another important contribution, as shown in Figure 2. This contribution does not interfere with the timing aspects of the IEEE 1149.1 JTAG standard. The system supposed to be in LOCKED state in the initial stage. Basically, Schnorr controller interacts with instruction decoder from the JTAG architecture. This Schnorr controller gets input from ECC module and a 192-bit random number generator module and generates "release_unlock" signal. After this step, all the JTAG functionality such as boundary scan, etc. can be used. The impact of this solution consists of initial delay for execution of Schnorr protocol/ECDSA but once it is unlocked the remaining functionality is the same as normal JTAG. This kind of protocol is useful in IOT nodes when nodes contain sensitive identification numbers and need of secured firmware upgrade through JTAG programmer.

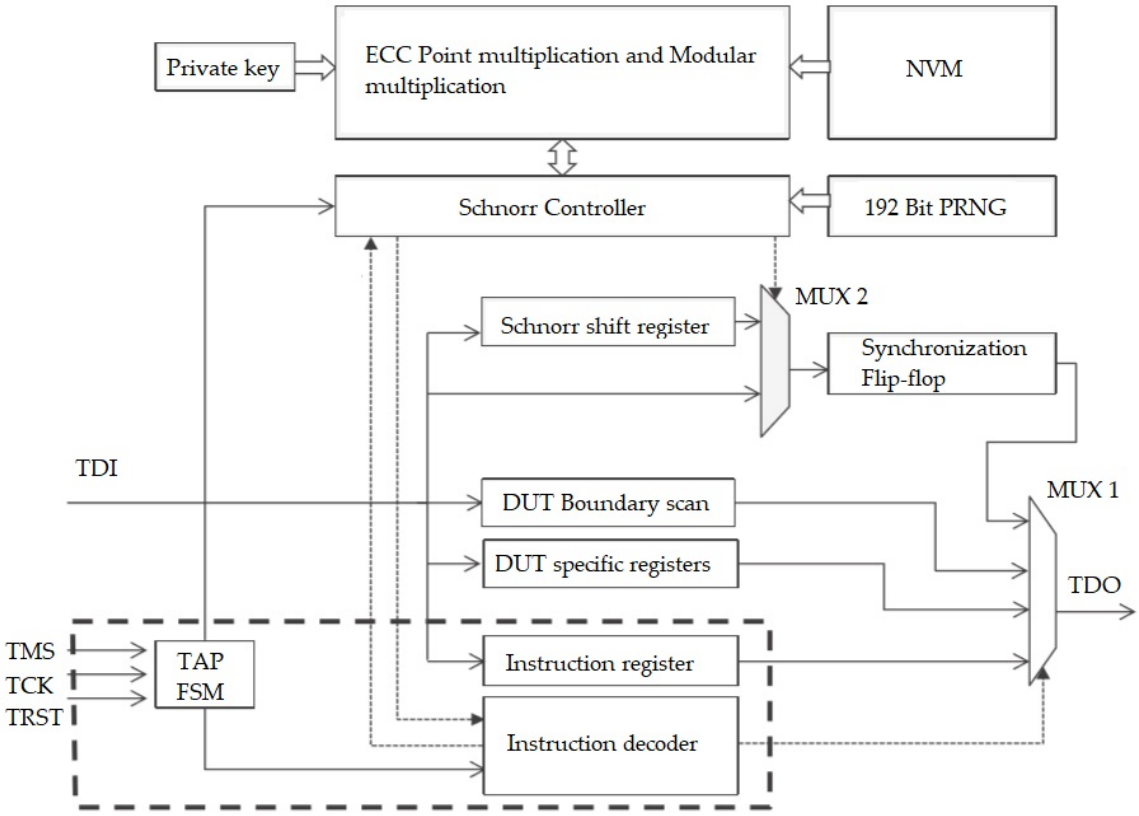

Figure 2. JTAG ECC controller integration block diagram [28].

4.3. Based on Challenge Response

Clark [29] proposed solution by using SHA256 secure hash and a true random number generator (TRNG). This solution creates a low gate overhead challenge/response-based access system for IC test and on-chip internals. The challenge has to be generated at IC and response has to be by JTAG software on PC, which computes hash value. Figure 3 shows basic block diagram of anti-tamper JTAG. This method increases hardware complexities since challenge is generated at IC for which a random generator is required on chip.

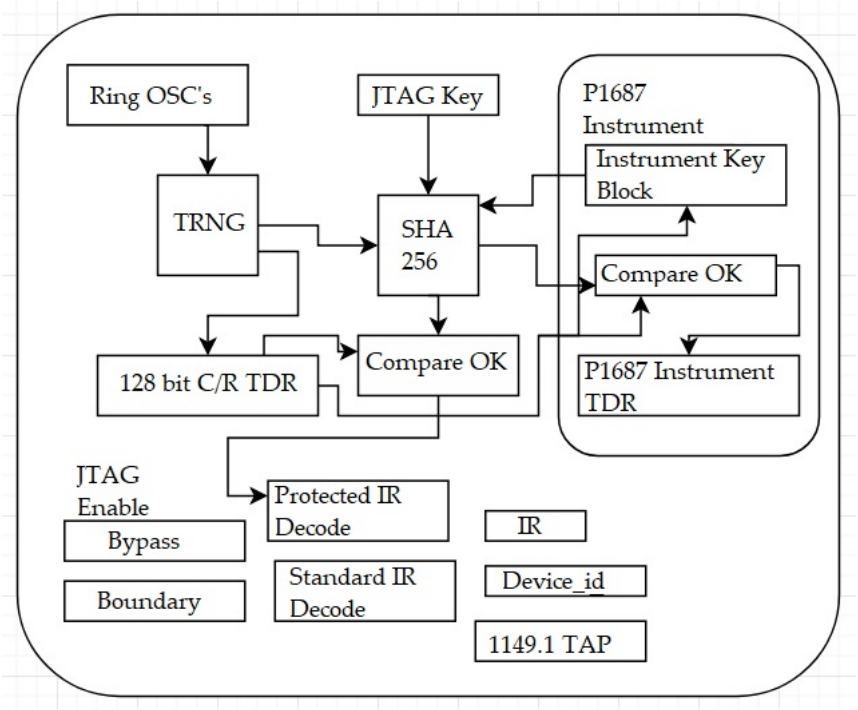

Figure 3. IC with anti-tamper JTAG [29].

4.4. Security Level Based Approach

Buskey et al. [30] (2006) proposed new protected JTAG architecture solution for microcontrollers and processors. The main motive of this paper is to protect the program residing inside controller. Thus, to prove this, they proposed a separate hardware protecting mechanism that guarantees reliable authorization. This solution gives not only facility of three protection levels to user but four different access modes as per their requirement. AM0 access mode discussed in this paper might be useful if it becomes implemented since this mode restricts all kinds of data through JTAG interface, giving highest security to sensitive data. Moreover, Kumar et al. [31] also proposed and simulated multi-level JTAG architecture. They offered VHDL implementation of JTAG architecture with four levels of privilege access modes: unprotected level (p1), low level protection level (p2), medium level protection level (p3), and maximum protection level (p4). AES encryption or decryption method is adopted for private key generation and authorization. These privilege level accesses are based on different stages involved in product life cycle development. Unprotected level (p1) can used by developers in initial development stage since they need full access to internal registers and flash memory. Low protection level (p2) can be used for just circuitry functions and uploading firmware. Medium protection level (p3) gives limited debugging facility such as board level interconnection testing for certified product distributor or service center. Maximum protection level (p4) is a locked state in which JTAG port is completely disabled hence giving maximum security to owner or user of product. Figure 4 shows block diagram of Secure JTAG Architecture.

Conventional JTAG architecture gets added by an Authentication and Authorization Module (AAM) and access provider. AAM register is implemented for communication protocol and user level access settings whereas Access Provider is a memory module that holds different access levels to actual JTAG architecture. The authors modeled this using VHDL and simulated using MODEL SIM 10.0d. In this paper, VHDL programming details are limited but the concept of level-based security seems promising for IoT devices.

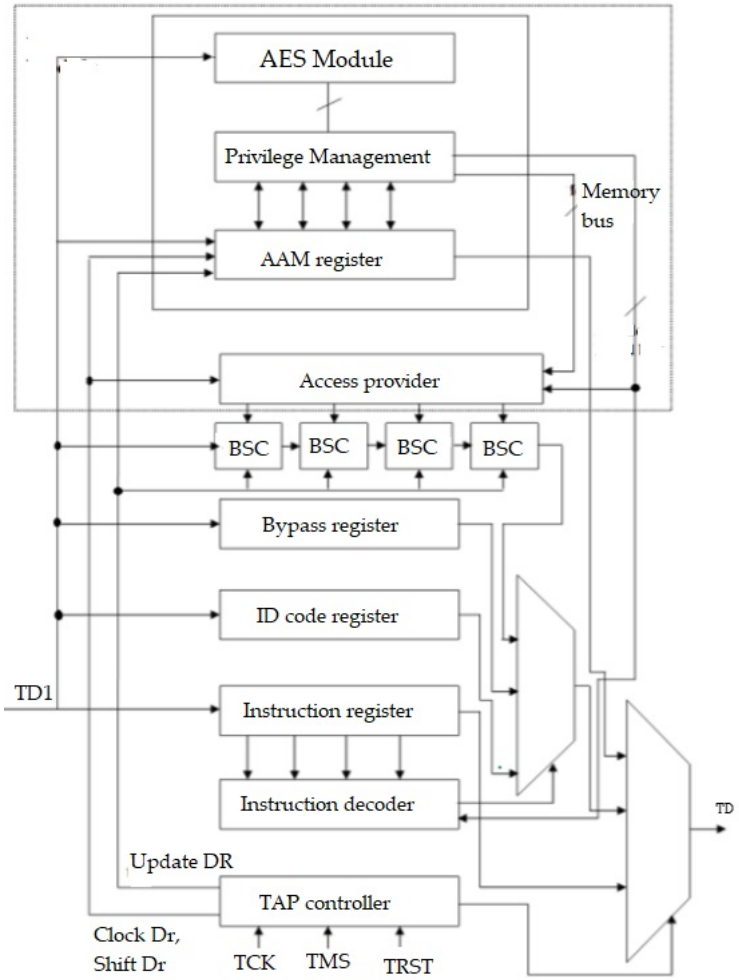

Figure 4. Secure JTAG architecture [31].

4.5. Credential Based Approach

Park et al. [32] proposed and implemented new end-to-end authentication infrastructure in which server issues credential and password to user through Internet connection for secure JTAG interface. This solution has two main phase: credential issue phase and the user authentication phase. In credential issue phase, a remote server verifies user identity and gives access privileged permissions, while the user authentication phase validates credentials and password provided by remote server and allows access to JTAG device as per defined in permissions. According to the authors, this method uses Hash and XOR calculation for interaction between host (computer) and JTAG device, which makes this solution less expensive.

4.6. Exploitation Using Statistical Machine Learning

Ren et al. [33] (2015) conducted an experiment to detect unauthorized access through JTAG port. The authors proposed statistical learning in chip (SLIC-J) scheme for JTAG protection. They proposed this scheme by assuming illegitimate person will act differently compared with authorized person. In this, the authors simulated results using openSPARC T2 benchmark. Basically, JTAG functions such as direct memory observe, clock control, internal scan, shadow scan, L2 cache scan, etc. are stored inside instruction register, which can be vulnerable with respect to attackers. Authors described JTAG attacking strategies in three main steps: (1) identifying the JTAG port on chip/PCB board; (2) finding basic profile of data register (DR) that can be used to extract or write the data; and (3) investigating the undocumented functions. Table 1 explains the main steps and their sub-steps.

Table 1. Attacking strategies.

Steps for JTAG Attacking Strategies	Sub-Steps
Identification of JTAG ports on PCB	**1.** Identification of JTAG port is performed either manually or third-party tools such as JTAGulator [7], ftjrev [34], etc.
Exploration of Data Register(DR)	**2.** Recollect details about length of Instruction register (IR). **3.** Recollect details about length of data register (DR). **4.** Get details about opcodes and their association with DR register. **5.** Check configurability status of DR register. **6.** Identify internal scan chains.
Get Details of undocumented functions	**7.** Differentiate functions and opcodes which we extracted during Sub-Step 4. **8.** Understanding of functioning of each DR register. **9.** Get all details of opcodes and their functioning.

To perform these steps, the authors used a two-phase scheme, i.e., offline learning and online prediction. Offline learning is used to train decision tree classifier, which is stored in nonvolatile memory, whereas feature extraction and supplying those features to the decision tree is done in online prediction. There are basically eight type of features that are explained in the paper: higher and lower 4 bits of JTAG instruction, number of clock cycles in shift-DR, run-test/idle, test-logic-reset, number of test-mode-select (TMS) transitions, type of transition, and legal opcode. These features are kept in a look-up table (LUT) and fed to decision-tree classifier. The classifier must have enough datasets for illegitimate user detection such that SLIC-J is the preferred *"normal_transition"*.

$$0 <= Tl <= Th <= k \tag{1}$$

where k is instructions, m is number of illegitimate predictions within a period, and Tl and Th are two thresholds.

1. If $m = k$, then the user is considered as an attacker.
2. If$Th \leq m \leq k$, then the user is most likely be an attacker.
3. If $Tl \leq m \leq Th$, then the user may or may not be an attacker.
4. If $M < +Tl$, then the user is considered as legitimate.

The goal of this paper is to discover undocumented JTAG functions that can be used to hack chip.

In addition, the same authors implemented another two-layered approach to secure JTAG attacks based on machine learning methods [35]. In this experiment, results are compared with other previously published protection schemes such as encryption, signature detection, anomaly detection, and SLIC-J statistical learning [33] and showed overall 94% accuracy. They proposed protection scheme to secure the system from unknown attacks. As we know, if one wants to add security measures in the existing IEEE 1149.1, then there are primarily two measures considered: (1) the level of security that is enhanced; and (2) the overall overhead that is added on the existing system. Thus, this experiment ensures high level security with a medium level of extra overhead. The study proposes a two-layered approach, where the violation of basic rules of JTAG operation are in the first layer. The second layer allows those who were successfully examined by first layer, in which Support Vector Machine (SVM) based classification is performed. The main motivation behind the implementation of the first layer is to label attackers immediately if one uses illegal opcodes and/or invokes incorrect length of read/write data. This behavior can only be observed in unauthorized users who have little or no knowledge of IC internals and its JTAG operation. In the second layer, behavioral pattern of the user is validated by considering the sequential order of JTAG instructions. Usually, there are multiple different operations with different JTAG instructions, which mostly have different opcode lengths. Thus, this shows nonlinear boundaries and overlapping regions while classifying between legitimate users and illegitimate users. The study tries to solve this classification problem by using a supervised

learning method known as SVM. Input sample behavior data (training data) is fed into SVM for classification. Here, the SVM decides the boundary between attacker behavior and normal behavior. In addition, delayed labeling method is used to avoid false positive and/or false negative results.

The authors used the OpenSPARC T2 platform for this experiment. The dataset included 767 authentic programs which contained 125,017 opcode sequences and 1092 attacking programs which contained 154,980 opcode sequences. The study also demonstrates two different sets of results following two different sets of strategies. In the first set, Ren et al. made divisions based on the component under target. They came up with eight different IC components targeted by attacks. They used SVM with neural network (hidden layer = 1, hidden neuron = 10), *k*-nearest-neighbor (*k*-NN, $k = 3$) and decision tree models. In the second set, the authors made nine divisions based on attacking strategies used for exploitation. The whole experiment showed them 94% overall accuracy while detecting unseen attacks. This article shows the results based on a two-layered learning technique, which is significant, but this scheme has more area overhead as well as more latency. This scheme is well suited where hardware complexity is higher and is not battery powered. The accuracy is dependent on the datasets of previous attacks to a great extent. The results of this experiment are significant, e.g., physical security is enhanced up to 94% with machine learning techniques, and improved by 50% on average compared to previous work.

4.7. JoKER, Trusted Detection of Kernel Rootkits in Android Devices via JTAG Interface

In recent years, there has been tremendous development in OS for handheld devices. These smartphones and tablets have become very popular and consist of full-fledged operating systems such as Android, iOS, etc. As we know, there are many antivirus programs available in the market that promise security against malicious apps. In practice, most of them provide application level security not low level (kernel) security, which is more prone to attacks. However, it is always best to prevent kernel level system attacks. Moreover, the JTAG hardware debugging standard is the most trusted debugging technique at hardware level in the industry.

Guri et al. [36] demonstrated detection of kernel level of rootkits in recent Android devices through the JTAG interface. Kernel level attacks can grant permissions to hide virus/malware from antivirus programs so that authors used well known JTAG hardware debugging techniques to get low level memory screenshots. This experiment is performed on Android versions 2.6.35 and 3.4.0. The authors downloaded vanilla OS version from its website and compared its memory contents with the same version OS, which already had hidden malware installed in it. The authors followed three main steps in the detection process: (1) halting the target processor; (2) extracting kernel data structures from memory; and (3) running analysis programs to hunt for rootkit footprints. They performed tests on Samsung Galaxy (S2 and S4) mobile phones with a RIFF Box JTAG controller to execute the first two steps. The third step consists of two important analysis, i.e., integrity checks and detection of stealthy hidden processes. Integrity checks comprises testing system call tables, the exception vector table, and software interrupt handler, since these objects are not usually modified on regular Android system. However, malicious processes can remove their entry from a process list, hiding their details. Thus, the authors compared objects from normal process list objects with the protected cache pool objects to acquire their presence. Although JTAG is designed for system testing and verification, in this experiment, it is also used for low level threat analysis of Android OS.

5. Discussion

In this article, it is assumed that the IoT node devices have less resources such as battery power, circuit complexity, memory size and most importantly lower cost for customer. There has always been a trade-off between product security and all other resources, such as hardware complexity, cost of product, etc. The linear relationship is always considered between product security and its cost. Moreover, as we have seen in this article, multiple solutions provided by researchers add security against JTAG attacks by adding extra hardware circuitry in it. Industry believes changes in

software is always less expensive than that of hardware changes and the JTAG works on most core layer of the system. Thus, there should be a very fine balance when providing highest security with limited resources.

Besides, nowadays, various IoT wireless technologies are coming to the market such as LoRaWAN, NBIoT, LTE, etc. that provides reliable end-to-end communication network link from endpoint nodes to the centralized server. In this case, most IoT endpoint nodes contain small-sized OS. As discussed in Section 3.3, we cannot escape from the fact that these operating systems are prone to malware botnets such as Mirai. These kinds of botnets usually reside in DRAM memory of the device, which is volatile. Thus, updating the firmware/OS eliminates such malware from the system. Thus, JTAG access port must be available on PCB board in worst case to debug and update software.

Let us evaluate the JTAG solutions that we discussed one by one. First, it seems unrealistic to add extra JTAG security circuitry into the resource-constrained IoT nodes. This extra circuitry could be added to products that are not battery operated, such as game consoles, PLC machines, set top boxes, etc. Second, a credential-based solution looked promising couple of years back but exponential growth rate in the number of IoT devices will make it difficult. Usually, JTAG debugging is used by technicians in the field either to debug circuit or upgrade firmware. Thus, in practical case, technicians use common password on the field and employees sometimes move to other companies (Job change). This makes products vulnerable over time, since it would be expensive to configure each device with new password. Third, If we take secured JTAG key methods, then the database of unique keys has to be well protected and maintained. Moreover, extra protected access has to be granted to technicians, which increases system dependency. However, note that, currently, SSL or TLS secure transport is affordable and cheap in server client protocol. The main advantage of this architecture is offloading processing on server side. Thus, global unique ID for each node can strengthen overall security. Fourth, recently machine learning techniques are coming forth to provide prevention against physical threats. The solutions that are provided with learning methods are mostly based on sample lab datasets. Hence, realistic datasets with all corner cases can only improve the accuracy of threat detection.

6. Conclusions

In this paper, we review articles of JTAG debug standard, which mainly demonstrated past PCB exploitation of some well-known products (such as GSM mobile, XBOX, PLC, etc.) as well as some proposed design solutions to make JTAG interface as a secure debug interface. Secure JTAG solutions that are reviewed in this paper are considered for making JTAG interface secure at any cost. Moreover, this might not help in the case of IOT technology since IOT sensor nodes are considered battery operated, less complex, low computational power/dumb devices, which only sense and forward data towards gateway or cloud.

If we consider the evolving nature of IOT infrastructure, there will be billions of devices that can communicate each other independently. Moreover, the Internet network (development of 4G, 5G, LPWAN, and Bluetooth) is becoming more resilient each day and offers high speed communication due to healthy competition between network service provider companies. Well known chip manufacturing companies such as ATMEL, NXP, Intel, Samsung, Texas Instruments, Qualcomm, Toshiba, etc. have already adapted JTAG standard in the last 18 years. In addition, some of the companies such as NXP [37] have already come up with their own security enhancement in JTAG standard to make their own chip secure. Thus, there should be a combined effort from all communities either to revise the existing standard or to frame-up IOT based secured JTAG by forming a global committee which will ensure security against physical attacks.

Author Contributions: G.V. conceived and wrote the manuscript under the supervision and guidance W.L.

Funding: This research received no external funding and the APC was funded by The University of Texas at San Antonio, USA.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. BI Intelligence. Here's How the Internet of Things Will Explode by 2020. 2016. Available online: http://www.businessinsider.com/iot-ecosystem-Internet-of-things-forecasts-and-business-opportunities-2016-2 (accessed on 2 December 2018).
2. Egham, U.K. Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent from 2016. 7 February 2017. Available online: https://www.gartner.com/newsroom/id/3598917 (accessed on 2 December 2018).
3. Tehranipoor, M.; Sergei, S.; Wang, C. *Introduction to Hardware Security and Trust*; Springer Publishing Company, Incorporated: Berlin, Germany, 2011.
4. IEEE. *1149.1-2001 IEEE Standard Test Access Port and Boundary-Scan Architecture 2001*; Tech. Rep.; IEEE Computer Society: Washington, DC, USA, 2001.
5. Rosenfeld, K.; Karri, R. Attacks and Defenses for JTAG. *IEEE Des. Test Comput.* **2010**, *27*, 36–47. [CrossRef]
6. Drake, J.J.; Lanier, Z.; Mulliner, C.; Fora, P.O.; Ridley, S.A.; Wicherski, G. *Android Hacker's Handbook*, 1st ed.; Wiley Publishing: New York, NY, USA, 2014.
7. JTAGulator by Grand Idea Studio. Available online: http://www.grandideastudio.com/jtagulator/ (accessed on 2 December 2018).
8. JTAGulator Official Documentation. Available online: https://www.parallax.com/sites/default/files/downloads/32115-JTAGulator-Product-Brief-1.1.pdf (accessed on 2 December 2018).
9. Hartung, C.; Balasalle, J.; Han, R. *Node Compromise in Sensor Networks: The Need for Secure Systems*; University of Colorado at Boulder: Boulder, CO, USA, 2005.
10. Kolias, C.; Kambourakis, G.; Stavrou, A.; Voas, J. DDoS in the IoT: Mirai and other botnets. *Computer* **2017**, *50*, 80–84. [CrossRef]
11. Chinese Firm Recalls Camera Products Linked to Massive DDOS Attack. Available online: https://www.pcworld.com/article/3133962/chinese-firm-recalls-camera-products-linked-to-massive-ddos-attack.html (accessed on 2 December 2018).
12. Tan, S.J.; Bratus, S.; Goodspeed, T. Interrupt-oriented bugdoor programming: A minimalist approach to bugdooring embedded systems firmware. In Proceedings of the 30th Annual Computer Security Applications Conference (ACSAC "14), New Orleans, LA, USA, 8–12 December 2014; pp. 116–125.
13. IDA Pro TechnicAl Documentation. Available online: https://www.hex-rays.com/products/ida/debugger/index.shtml (accessed on 2 December 2018).
14. IDA Python Docs. Available online: https://www.hex-rays.com/products/ida/support/idapython_docs/ (accessed on 2 December 2018).
15. Texas Instruments. *SLAU144I: MSP430x2xx Family: User's Guide*; Texas Instruments Inc.: Dallas, TX, USA, January 2012.
16. DeBusschere E.; McCambridge, M. *Modern Game Console Exploitation*; Technical Report; Department of Computer Science University of Arizona: Tucson, AZ, USA, 2012.
17. Anonymous Hacker. Xbox 360 Hypervisor Privilege Escalation Vulnerability. February 2007. Available online: http://securityvulns.com/Qdocument211.html (accessed on 2 December 2018).
18. Weinmann, R.-P. Baseband attacks: Remote exploitation of memory corruptions in cellular protocol stacks. In Proceedings of the 6th USENIX Conference on Offensive Technologies, Berkeley, CA, USA, 7–8 June 2012.
19. Schuett, C.; Butts, J.; Dunlap, S. An evaluation of modification attacks on programmable logic controllers. *Int. J. Crit. Infrastruct. Prot.* **2014**, *7*, 61–68. [CrossRef]
20. Santamarta, R. *Project Basecamp—Attacking Controllogix, Project Basecamp Report*; Digital Bond: Sunrise, FL, USA, 2012.
21. Basnight, Z.; Butts, J.; Lopez, J., Jr.; Dube, T. Firmware modification attacks on programmable logic controllers. *Int. J. Crit. Infrastruct. Prot.* **2013**, *6*, 76–84. [CrossRef]
22. Gonzalvo, B.; Bourbao, E.; Majéric, F.; Bossue, L. JTAG combined attacks. In Proceedings of the 2016 8th IEEE IFIP International Conference on New Technologies, Mobility and Security (NTMS),Larnaca, Cyprus, 21–23 November 2016; pp. 26–35.
23. Majeric, F.; Gonzalvo, B.; Bossuet, L. JTAG Fault Injection Attack. *IEEE Embed. Syst. Lett.* **2018**, *10*, 65–68. [CrossRef]

24. Suh, G.E.; Devadas, S. Physical unclonable functions for device authentication and secret key generation. In Proceedings of the 44th ACM/IEEE Design Automation Conference (DAC'07), San Diego, CA, USA, 4–8 June 2007; pp. 9–14.
25. Anderson, R.; Kuhn, M. Tamper resistance—A cautionary note. In Proceedings of the 2nd USENIX Workshop on Electronic Commerce, Oakland, CA, USA, 18–21 November 1996.
26. Anderson, R.; Kuhn, M. Low cost attacks on tamper resistant devices. In *Lecture Notes in Computer Science, Proceedings of the IWSP: International Workshop on Security Protocols, Paris, France, 7–9 April 1997*; Springer: Berlin/Heidelberg, Germany, 1997.
27. Skorobogatov, S.P. Semi-invasive attacks—A new approach to hardware security analysis. In *Technical Report UCAM-CL-TR-630*; University of Cambridge Computer Laboratory: Cambridge, UK, April 2005.
28. Da Rolt, D.J.; Ghosh, S.; Seys, S. Secure JTAG implementation using Schnorr protocol. *Electron. Test.* **2013**, *29*, 193–209.
29. Clark, C.J. Anti-tamper JTAG TAP design enables DRM to JTAG registers and P1687 on-chip instruments. In Proceedings of the 2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), Anaheim, CA, USA, 13–14 June 2010; pp. 19–24.
30. Buskey, R.F.; Frosik, B.B. Protected JTAG. In Proceedings of the 2006 International Conference on Parallel Processing Workshops IEEE, Columbus, OH, USA, 14–18 August 2006.
31. Kumar, P.A.; Kumar, P.S.; Patwa, A. JTAG Architecture with Multi Level Security. *IOSR J. Comput. Eng.* **2012**, *1*, 54–59. [CrossRef]
32. Park, K.; Yoo, S.G.; Kim, T.; Kim, J. JTAG security system based on credentials. *J. Electron. Test.* **2010**, *26*, 549–557. [CrossRef]
33. Ren, X.; Tavares, V.G.; Blanton, R.D. Detection of Illegitimate Access to JTAG via Statistical Learning in Chip. In Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 9–13 March 2015.
34. ftjrev. JTAG Reverse Engineering Tool. Available online: http://www.alexforencich.com/wiki/en/projects/ftjrev/start/ (accessed on 2 December 2018).
35. Ren, X.; Blanton, R.D.; Tavares, V.G. A Learning-based Approach to Secure JTAG against Unseen Scan-based Attacks. In Proceedings of the 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Pittsburgh, PA, USA, 11–13 July 2016.
36. Guri, M.; Poliak, Y.; Shapira, B.; Elovici, Y. JoKER: Trusted detection of kernel rootkits in Android devices via JTAG interface. In Proceedings of the Trustcom/BigDataSE/ISPA, Helsinki, Finland, 20–25 August 2015; Volume 1, pp. 65–73.
37. Configuring Secure Jtag for the i.mx 6 Series Family of Applications Processors-Application Note. NXP. March 2015. Available online: http://www.nxp.com/files/32bit/doc/app_note/AN4686.pdf (accessed on 2 December 2018).