Detection and Mitigation of Malicious Modifications on the Minnowboard Turbot

Bryan Koch, Bohuai Liu, Alejandro Mera
College of Computer and Information Science
Northeastern University, Boston, USA
{koch.b, liu.boh, mera.a}@husky.neu.edu

Austin Roach
Naval Sea Systems Command
Naval Surface Warfare Center, Crane, USA
austin.roach@navy.mil

***Abstract*—Malicious modifications affecting the hardware or firmware of Commercial off-the-shelf (COTS) devices is far more persistent than traditional software-based malware, and can be hard to detect and remove once the devices are deployed in an organization. This research analyze and propose cost effective procedures to detect and mitigate malicious modifications associated to the supply chain of the Minnowboard Turbot, an open-source COTS device. Findings demonstrate that screening hardware with existing quality assurance techniques and installing clean firmware on the device can successfully mitigate most of supply chain risks. These procedures and risk analysis provide a road map for identifying supply chain risk on other commercial off-the-shelf devices.**

I. Introduction

Large supply chains are cost-effective process of production used by manufacturers to build commercial off-the-shelf (COTS) devices. These cost savings can add security vulnerabilities in the form of malicious modifications to devices and components. By modifying the hardware or firmware of a device, an attacker can gain persistent access to a device by maintaining their control at the lowest levels of functionality. Embedded malware or counterfeit have typically been addressed with non-technical approaches, including vendor selection and strategic product sourcing. However, a comprehensive firmware and hardware screening analysis, at reception of the device, is normally discredited as cost and time prohibitive. In this scenario, optimizing screening procedures is imperative in order to assure the integrity of the devices, while reduced costs and immediate availability are not affected.

The Minnowboard Turbot offers an adequate environment to analyze characteristics of the supply chain, firmware and hardware of an open-source device. This project will use the open-source information of Minnowboard to identify potential malicious modifications and state the difficulty related to execute these changes. The final product of this project will include a risk and vulnerability assessment comparing the probability of an attack with the difficulty to perpetrate/detect a modification. These results will be used to develop recommendations to confidently detect changes to the firmware or hardware of Minnowboard Turbot upon receipt by the customer. This proof-of-concept will provide a cost-effective technical approach to decrease the risk introduced by complex supply chains for COTS products. [1]

[1]This work was supported in part by the National Science Foundation under grant no. 1241668.

II. Literature Review

Supply chain risk analysis of COTS products can be approached in many ways. Some researchers analyzed the risk using the components during the software acquisition life cycle[1]. This approach concentrates on threat modeling and the attack surface at a procedural level. The benefits of this approach are evident because it assures the availability, the implementation, the delivery and use of secure characteristics in the acquisition process of software. However, it does not consider hardware modifications or low-level firmware screening techniques to verify the final product.

Two sets of security professionals and organizations are known to be involved in embedded malware research. The first set consists of organizations looking to exploit hardware, including academic researchers, bug bounty hunters and highly organized groups such as the National Security Agencys Office of Tailored Access Operations (TAO) and other organizations with significant resources and a determined mission[2]. The second set consists of digital forensics specialists, who normally begin their investigations in reaction to malware detection involving ongoing operations. Although there is a major gap in efforts to proactively find embedded malware, any attempt to find malware in the acquisition phase would require both an understanding of historically successful malware deployments and the forensic process used to detect and remove the compromising modification.

In [3], researchers successfully used static analysis, fuzzy hashing and correlation techniques to find malware and vulnerabilities in the firmware of several embedded systems, identifying serious risks with the use of these devices. The main contribution of this research is the signature and extraction algorithms that are available as a web service to analyze firmware. The caveat of this research is the use of a blacklisting schema (signatures, hashes and default configurations); therefore, new malware or vulnerabilities that are not blacklisted cannot be detected by this approach.

In [4], the author describes a method that automatically maps and explores the firmware/software architecture of COTS devices. This is one of the most advanced approaches that can help to prioritize actions to secure or defend the system according to states and conditions. The spotted drawback of this research is the method of evaluation of the attacks, which only consider each scenario as a Boolean Satisfiability Problem. In this case, the evaluation of each scenario does not consider the impact or feasibility of each attack. For instance, the author of

this research suggests to include other characteristics to model the threats. Our research will complement this work providing levels of difficulty and technical knowledge as factors of evaluation.

The development of Binwalk, an open-source firmware analysis tool, is one of the greatest contributions in support of firmware analysis. Binwalk has been used to extract firmware from COTS devices, such as routers[5] and ARM devices [6]. These research endeavors provide great information on some of the complexities associated with firmware analysis.

In order to support a wider range of applications for its devices, Intel has also released tools specifically for firmware customization such as Intel's Firmware Engine [7]. This tool has enabled many new applications of Intels products, but has also exposed new potentials for vulnerability discovery and malware development. For instance, Wojtczuk and Tereshkin used Intels firmware tools to develop malware embedded directly into an Intel processors firmware [8].

The primary focus of academic research into embedded malware has been into detection and mitigation. The problem of mitigation in the supply chain largely deals with territorial issues throughout the chain. In order to maintain economies of scope, the producer of a finished product relies on a complex network of suppliers down to the raw material level in order to source quality components within cost constraints[9].

When dealing with computers, network devices and electronic components, updates to firmware add a post-acquisition threat vector to the supply chain. A malicious party could modify the firmware to add malicious payloads into programs through the compiler [10]. In the last year, Intel released and corrected a vulnerability to its Driver Update Utility which exposed this exact problem[11]. The Intel Driver Update Utility sent firmware updates to customers using an unencrypted HTTP connection. In this scenario, a man-in-the-middle (MITM) attack would be sufficient to introduce undetectable malware directly into the firmware controlling an Intel processor or network interface. This threat recognizes the need not only to confirm that the firmware is free of malicious modifications, but also that the firmware is hardened against unauthorized modifications once deployed.

III. Problem Statement

This project focuses on detecting malicious modifications on COTS devices. It assumes that modifications can be introduced in any point of the supply chain by different threat actors having basic technical skills or almost unlimited resources and knowledge. Additionally, it considers that screening procedures are expensive and not economically feasible when system designers have to deal with resource constraints. This project will propose an outcome where a less sophisticated technician can adequately detect malicious modifications using open source documentation, a minimal labor, equipment and technical knowledge. The technicians advantage comes from the expectation that the received microcomputer should match the factory open-source specifications. Our proposed solution will take advantage of these specifications to white list the hardware and firmware of the Minnowboard Turbot. This specific analysis of this board will provide a template for determining vulnerabilities to similar COTS microcomputers, and developing similar screening procedures for detecting modifications and mitigating risk on receipt of a product from a supplier.

Device	Description
Minnowboard Turbot	Research target
Power Supply 4A 5V	Provides power to Minnowboard
Micro HDMI cable	Connects Minnowboard to a monitor
HDMI Monitor	Display device
SPI HOOK	USB to SPI and RS232 interface
SD card, Keyboard, mouse, USB hub	PC accessories
Workstation with Linux OS	Kali Linux containing Binwalk and FlashROM

TABLE I. Lab Equipment

This project consists of three major milestones in order to provide a risk analysis of the supply chain of Minnowboard Turbot. These milestones are as follow:

- Develop an understanding of how hardware and firmware can be modified to compromise the integrity of the Minnowboard Turbot through its supply chain.
- Perform a risk and vulnerability assessment of the Minnowboard Turbot, considering the probability of an attack occurring and the difficult of detection by the customer.
- Develop a screening procedure for detecting and mitigating modifications of the hardware and firmware of a Minnowboard Turbot.

IV. Methods and Procedures

This project requires a lab environment to analyze the firmware of the Minnowboard and industry quality assurance (QA) procedures and open source documents for a theoretical analysis of the hardware. These analyses will be used to develop a comprehensive supply chain risk assessment and a detection procedure to mitigate risk on receipt of a Minnowboard from a vendor.

A. Firmware Analysis

The lab environment configured for this project is basically the "hardware setup" described on the official documentation of the Minnowboard [12]. Our specific implementation adds a redundant configuration where two independent Minnowboards (MB1 and MB2) are tested and compared between each other. This double configuration allows us to verify if findings are constant across different boards provided by the same vendor. TABLE I describes the elements of our lab environment.

1) Read/Write with SPI Hook: The SPI Hook is a USB dual function utility board that includes a SPI flashing tool that reads, writes, or erases the SPI Flash located on the Minnowboard Turbot, and a virtual RS232 serial port for communicating with the Minnowboard MAX/Turbot [13]. The SPI Hook uses the open source software FlashROM, a utility application for identifying, reading, writing, and erasing flash chips. The SPI Hook connects the Minnowboard Turbot to our workstation, enabling FlashROM to read and write the firmware. This procedure involved first reading the firmware from each board for further analysis. Figure 1 depicts a schema of the elements and connections of our lab environment.

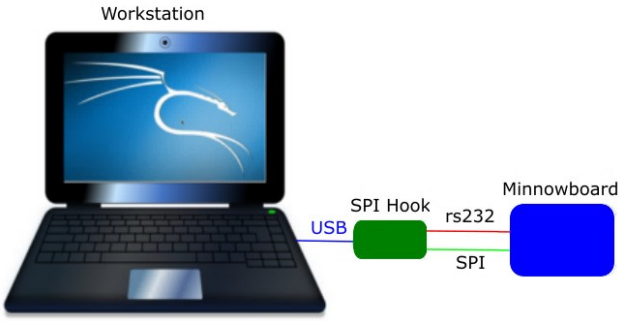

Fig. 1. Connections between Laptop, SPI Hook and Minnowboard Turbot

2) Write Firmware Using UEFI Commands: This procedure uses the command line interface provided by the standard firmware of the Minnowboard and a USB flash drive which contains the firmware to be flashed. To write the firmware:

- Connect the Minnowboard to a monitor using a Micro HDMI cable and boot the board.
- Check the version of firmware (FW1) in Minnowboard.
- Perform a write operation with a different version of firmware (FW2) from a USB flash drive using UEFI commands as described on documentation [12].
- Reboot and verify the new version of the firmware (FW2) is installed.

3) Read/Write Firmware From Operating System: This technique was not included in the Minnowboard Turbot documentation, but is mentioned as an option in the FlashROM documentation. This procedure requires booting the Minnowboard Turbot using a supported operating system, such as Ubuntu 16.04 [14], installed on a bootable USB flash drive. After installing and booting Ubuntu on the Minnowboard, FlashROM can access the firmware of the Minnowboard and write a different version of the firmware. During the next boot operation, the new version will be installed if properly written to the firmware memory.

4) Firmware Screening Methods using Hashes: Hash functions are often used to map data to a unique fixed-length string and are commonly used for integrity checks on binary files. Using hash functions to screen the entire firmware is an effective method to detect any modification, but it is not useful to determine if that modification corresponds to data (dynamic) or code (static) sections of the firmware. In this scenario, finding critical segments of the firmware is a must to avoid false positives. Using the firmware extracted with the SPI-Hook (FW1 and FW2) and the "clean" version from Intel's website (FW0), critical segments on the images with BinWalk are located. To determine the critical parts of a firmware, the analysis will be performed as follows:

- Derive the firmware of MB1, store as FW1;
- Boot the MB1 into BIOS and change a configuration parameter (for instance, boot sequence was changed).
- Turn off MB1, use SPI-Hook to read the firmware, store as FW1.1
- Use Binwalk to extract the firmware and compare the hash value of each segment in FW1.1 with the corresponding segments in FW1 to determine which segments include data of configurations and which segments correspond to code.

5) Preventing modifications of firmware: To perform a modification using SPI Hook, the attacker must have physical access to the Minnowboard Turbot. Therefore, restricting the physical access can significantly mitigate this risk. Since this technique bypasses the processor and writes directly to the flash memory chip, it cannot be otherwise disabled. However, modification from using UEFI commands and a USB flash drive or using FlashROM from a supported operating system is preventable. By changing the SPI Flash Descriptor in the firmware, the system will deny any software-based modification. SPI Flash Descriptor is a parameter in firmware that can enable or disable firmware modification from UEFI commands and OS tools [7]. The following procedure modify the default configuration of the Minnowboard Turbot to protect the firmware:

- Configure the development environment and download the source tree to build the firmware of the Minnowboard Turbot according to Intel's documentation [7].
- Compile the firmware disabling CPU/BIOS access to SPI Flash Descriptor Region, store it as FW3.
- Use SPI Hook to write FW3 to the Minnowboard turbot.
- Reboot the system and attempt to read/write the firmware using the UEFI commands and OS tools (FlashROM), previously described.
- Verify that reading the firmware is possible, but writing a new one is not allowed.

B. Hardware Analysis

Detection and screening of malicious hardware requires different levels of effort and accuracy. The most effective method to conduct a hardware risk analysis requires specialized training with hardware inspection equipment. Targeted exploitation requires intimate analysis of the proprietary functionality of each hardware component to completely reverse-engineer this board. Instead of focusing on identifying actual vulnerabilities, generic hardware risk analysis for the Minnowboard Turbot hardware will focus on critical components/vendors, and the cost and accuracy to detect any modification of the board. This include,

1) Bill of materials: The bill of materials, included in the open source documentation, is a rich source of information related to the supply chain of the Minnowboard Turbot. Passive components such as capacitors and resistors are not a means of advanced threats, because the attacker has no control on triggering an attack using these components. On the other hand, active components such as Integrated Circuits represent

Procedure	Cost	Effectiveness
Automated X-Ray Inspection (AXI)	$ 2.021	82.0%
Automated Optical Inspection (AOI)	$ 1.516	58.0%
Integrated Circuit Test (ICT)	$ 36.226	65.0%
Functional Test (FT)	$ 5.383	31.0%
System Test	$ 25.573	98.3%

TABLE II. INDUSTRY QA PROCEDURES [15]

a real threat, where highly technical and expensive attacks can be perpetrated.

2) Quality assurance procedures: Common industry QA procedures for printed circuit boards (PCBs) are the proven method for detecting changes to hardware [15]. The procedures shown in TABLE II will be used for comparison in order to determine the most cost effective procedure for detecting and mitigating any hardware modifications to the Minnowboard Turbot.

Each of these procedures will be assigned a quantitative cost and likelihood of detection for each method based on industry data. Estimates from actual companies and previous research to determine the cost to perform each procedure on an individual board. This past research will approximate the effectiveness of each procedure at detecting modifications based on specifications similar to the Minnowboard Turbot.

C. Risk Assessment

The firmware and hardware analysis results provide detailed information on potential risks introduced throughout the supply chain. These findings and results from past work combine to create a comprehensive list of each attack vector throughout the supply chain. This data will be segmented using using Schneier's Attack Trees methodology [16] segmented by supply chain phase[17] tailored to the Minnowboard Turbot.

D. Procedure Development

The firmware and hardware analysis results comprise the several successful and unsuccessful approaches to detection of modifications added throughout the supply chain. Assembling these procedures in the correct order results in a highly effective procedure for detecting malicious modifications within the Minnowboard Turbot's firmware or hardware. This procedure will assume the end-user has a known good version of the hardware and firmware to "white-list" Minnowboard's received from a supplier (open-source documentation). This procedure will provide a repeatable, adaptable method for mitigating supply chain risks of on similar COTS devices.

V. RESULTS

Through our study, we have used open-source documentation of the Minnowboard Turbot. This fact, warranties that our methods are repeatable. However, multiple contributions of the open-source community could slightly modify some of the results, specially those that depend on source code and tool chains (firmware). This section will break down our findings maintaining an adequate level of detail, where our previous statement could be applicable.

A. Firmware

1) Analysis of Source Code and Library Dependencies: The official firmware of the Minnowboard is based on the EDK II (Enhanced Development Kit) project of the Tianocore.org community. This community supports a firmware development environment that uses the open source components of Intels implementation of UEFI and some proprietary precompiled libraries that initializes the Intel Atom processor. Principally, the EDK II environment includes a set of libraries (OpenSSL and ACPI specifications) and building tools that process the content of firmware, which includes configuration files, compiling scripts, compilers, binary layout definitions and intel precompiled objects. The EDK II project comprises complementary projects to support a UEFI 2.0 shell, FAT12/16/32 file system drivers, Coreboot payload (a replacement of proprietary BIOS), driver development for independent hardware vendors (IHV), and a security package which implements Trusted Platform Module (TPM), User Identification (UID), Secure Boot and authenticated variable.

2) Vanilla Firmware Extraction and Analysis: The size of the firmware images FW1 and FW2 extracted from MB1 and MB2, is 8388608 bytes. This size is the same of the firmware downloaded from Intel (FW0) and matches the capacity of the serial flash memory chip W25Q64V implemented in the Minnowboard Turbot (64 Mbit). FW1 and FW2 are x64 architecture compatible, which corresponds to the standard retailer specification for this board. The hash sums of the complete firmware images are different for all three versions (FW0, FW1 and FW2).

3) Binwalk Signature Analysis and File Extraction: The signature scanning of Binwalk tool determined that Minnowboards firmware blocks are LZMA compressed data, Microsoft executable portable (PE) and encrypted data that uses blowfish-448 in cipher block chaining (CBC) mode. The offsets and size of each block are constant between the firmware extracted from the Minnowboards (FW1 and FW2), but these characteristics differ from Intels firmware (FW0). For instance, the size and offset of the LZMA block are 4456464 bytes and 0x600078 on FW1 and FW2; whereas the size and offset are 5120016 bytes and 0x510078 for the same block on Intels firmware. A complete and detailed report of the signature scanning is included on Appendix I. The analysis of the LZMA block determined that it contains a private RSA key, six x.509 certificates and many PE files. According to EDK II documentation, these certificates are included as part of the Attribute Certificate Table, which contains contiguous certificate entries that are used for signing verification and secure boot [18]. The content and internal offsets of this block is the same on FW1 and FW2, but it differs from FW0, which contains alike keys and certificates on different offsets. Consequently, the hash sum of the LZMA block is the same for FW1 and FW2.

4) Binwalk Entropy Analysis: The entropy analysis of the firmware demonstrated that some blocks are empty, therefore they have no information or they correspond to consecutive and predictable unused memory space, i.e. 0x00 and 0xFF values. Consequently, the block with the highest entropy (information content) is the compressed LZMA, whose entropy and offset is depicted in Figure 2 and 3 for FW0 and FW1 respectively.

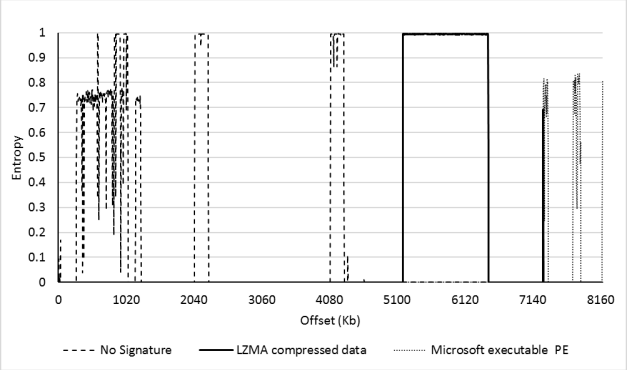

Fig. 2. Entropy Analysis of Intel Firmware (FW0)

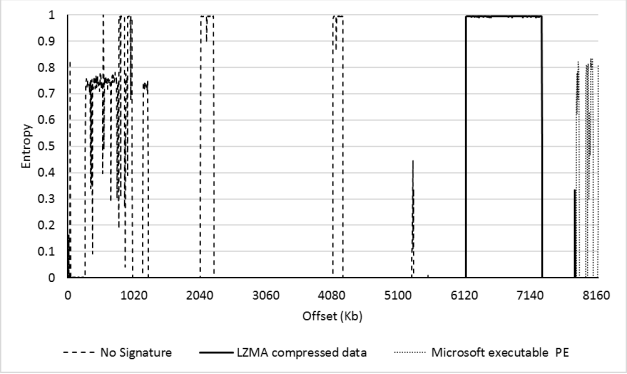

Fig. 3. Entropy Analysis of Extracted Firmware (FW1)

5) Toolchain Analysis: EDK II is a framework that aims to be a cross platform development environment. Therefore, it is compatible with different compilers, assemblers and operating systems, such as Windows and Linux. The Windows based environment can use the C compiler included in Visual Studio .NET 2008 to 2013 and the ACPICA ASL compiler Version 20141107. For the Linux environment it requires GCC 4.6.X, the latest ACPICA UNIX ASL compiler, flex 2.5.4 or greater and bison version 2.4.1 or greater. The extensive compatibility of EDK II offers flexibility for developers, however different tool-chains produces heterogeneous binaries, whose functionality is similar but cannot be verified using hashes. These differences are due to compiler optimization, versions and architecture. These results used the current source code and a Windows 10 X64 development environment with VS2013 to build and analyze a compiled version of Minnowboard firmware (FW3). Analysis of FW3 showed the same offset from FW0, but a different LZMA block size. Factory images FW1 and FW2 each had unique LZMA block sizes and offsets.

6) Toolchain UEFI/BIOS Configuration and Memory Layout: The Minnowboard UEFI, built with EDK II, is a configurable and rich featured interface, which is similar to commercial and closed source products shipped on modern personal computers. This environment includes a shell and tools to configure security characteristics, boot sequence, language, connected devices, etc. Our analysis found that parameters and configurations of the Minnowboard are stored in some sections of the flash chip reserved as variables or NVRAM (non-volatile RAM). For instance, the activation of the FTPM (firmware-based TPM) in the Minnowboard changed the firmware. This proved extracting the firmware from MB1 again and verifying that the hash is not the same as the original version (FW1), however the LZMA area is not altered and the hash of this specific block is the same. According to EDKII documentation, the flash layout is defined in the *.fdf files, which are specific for each platform, i.e. Minnowboard, Beagleboard, etc. The EDK II includes specific and separated configuration files for Windows and Linux tool-chains. This specific configuration is due to different compiler optimizations, which produce binaries of different sizes that requires different amounts of memory to be accommodated [18]. The analysis of these files showed that, during the building process of the firmware, EDK II uses those files to define base address, size, block size, number of blocks and offsets for microcode, recovery and non-volatile regions of the firmware.

B. Hardware Analysis Results

1) Bill of Materials: The Minnowboard Turbot, version x205, is a MinnowBoard MAX-compatible derivative board designed and manufactured by ADI Engineering[19].The Minnowboard Turbot's bill of materials has 811 components, including 687 resistors and capacitors. Although changes to the lower level components could also indicate malicious modifications, the most concerning changes would be within one of integrated circuits on the board. The critical components include the Atom processor, SDRAM, flash SPI memory, system power control, switching regulator, HDMI companion, Ethernet interface and thermistor. Some of these components have internal processing capabilities and a high level of integration; therefore, they can be target for tampering or high technical attacks.

2) Quality assurance procedures: In order to calculated cost for each procedure, similar results used in other scholarly works provide the closest estimation without manually performing each of these tests. Teradyne, an automated test equipment supplier, provided costs and effectiveness for several test which assumed costs of $150,000 for AOI and $550,000 for AXI with annualized costs over 5 years of $30,000 and $110,000 respectively[15]. The board used in this with 400 components and 4,000 solder joints for a total of 4,400 fault opportunities. For the purposes of estimated costs and effectiveness, this research assumes that the current costs and effectiveness of these tests on a Minnowboard Turbot are similar to those of the subject board from the sourced previous work.

Based on these calculations, the use of AXI and AOI has the highest combined effectiveness (92.44%) with the least cost ($ 3.54). Other previous research supports the use of AXI and AOI together for maximum cost effectiveness. An early study of PCB detection techniques concluded that as the complexity of board increases, x-ray inspection should be used instead of optical or visual inspection techniques. However, the same study also concluded that AXI and AOI should be used together for test batches with high complexity and high volume of boards to test. This research validates our above conclusion of AXI and AOI being the most cost effective combination of procedures to detect modifications.

Other industry studies also recommend the use of both AOI and AXI for detecting modifications and defects on printed

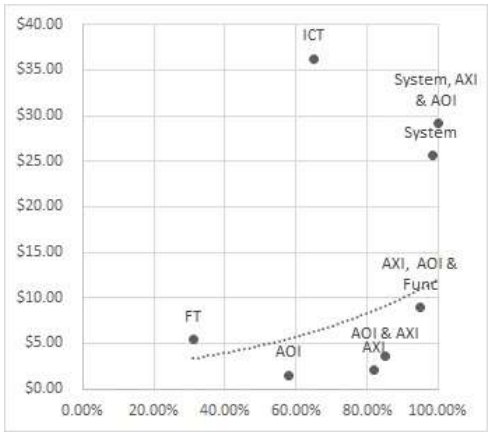

Fig. 4. Cost & Effectiveness of QA Procedure Combinations

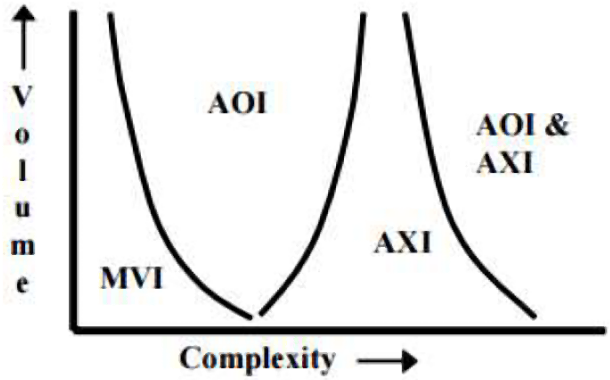

Fig. 5. AOI vs. AXI[20]

circuit boards [20]. These results validate QA measures as an effective method for detecting modifications to hardware. For the purposes of reducing supply chain risk, manufacturing defects and embedded malware are both undesirable states for received hardware and should not be cleared for use. This hardware should be returned to the supplier or sent for further forensic analysis.

VI. Discussion

This research has analyzed the open source documents of the Minnowboard, the characteristics of the EDK II framework, methods to read/write the firmware from/to the board, and QA procedures which can be used to detect modifications to hardware. This investigation has lead to a detail procedure for detecting modifications to the Minnowboard Turbot and to a full risk assessment on the Minnowboard Turbot supply chain. Because ADI Engineering and Intel incorporated security into the supply chain considerations when developing the Minnowboard Turbot, this project was able to make assumptions and risk estimates based on those decisions. These assumptions assume that the purchasing party will be able to possess a "known good" version of both the hardware and firmware prior to receiving bulk shipments. Part of this assumption is due to the intrinsic characteristics of an open-source project, which is susceptible to audit and deep analysis.

Potential future work should consider how to determine modified hardware or firmware when the exact characteristics based on the manufacturer's specifications can not be trusted or are not disclosed. Dynamic and static analysis of firmware for malicious signatures is one potential direction to pursue in order to find malware embedded into firmware. Tools, such as VirusTotal, can find common signature patterns for specific exploits used embedded within a device's firmware [21], a reverse engineering project into known compromised hardware would provide missing data on how to detect malicious hardware modifications.

VII. Conclusion

This project proves modifications to the firmware and hardware of Minnowboard Turbot can be detected on receipt from a vendor. Due to the complexities and costs associated with verifying that no unauthorized modifications have been made to the firmware, using an SPI hook to write a clean firmware image to the device is the recommended method avoiding the risk. The risk associated with hardware modifications cannot be so easily avoided. However, using both AOI and AXI with a known "clean" version of the hardware will maximize the likelihood of detection while minimizing the overall cost. These techniques can be used to detect firmware and hardware modifications on receipt from a supplier. Furthermore, the complete supply chain risk analysis of the Minnowboard Turbot shows that all risks can be detected with these efforts, providing a template for further studies on other COTS device risk assessments.

References

[1] R. J. Ellison and C. Woody, "Supply-chain risk management: Incorporating security into software development," in *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*. IEEE, Conference Proceedings, pp. 1–10.

[2] G. Greenwald, *No place to hide : Edward Snowden, the NSA, and the U.S. surveillance state*, first edition. ed. New York : Metropolitan Books/Henry Holt, 2014.

[3] A. Costin, J. Zaddach, A. Francillon, and D. Balzarotti, "A large-scale analysis of the security of embedded firmwares," in *23rd USENIX Security Symposium (USENIX Security 14)*, Conference Proceedings, pp. 95–110.

[4] S. Jilcott, "Securing the supply chain commodity it devices by automated scenario generation," in *Technologies for Homeland Security (HST), 2015 IEEE International Symposium on*. IEEE, Conference Proceedings, pp. 1–6.

[5] F. Eric, M. Schulte, W. Weimer, and Stephanie, "Repairing cots router firmware without access to source code or test suites: A case study in evolutionary software repair." in *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*. Association of Computing Machinery, Conference Proceedings. [Online]. Available: http://www.cs.virginia.edu/ weimer/p/weimer-netgear-repair-preprint.pdf

[6] Z. Ruijin, Y. an Tan, Q. Zhang, Y. Li, and J. Zheng, "Determining image base of firmware for arm devices by matching literal pools," *Digital Investigation*, vol. 16, pp. 19–28, 2016.

[7] "Minnowboard maxturbot 0.94 uefi firmware open source release notes," "https://firmware.intel.com/sites/default/files/MinnowBoard_MAX-Rel_0_94-ReleaseNotes.txt", 2016.

[8] "Attacking intel bios," "https://www.blackhat.com/presentations/bh-usa-09/WOJTCZUK/BHUSA09-Wojtczuk-AtkIntelBios-SLIDES.pdf", 2009.

[9] S. Chopra and P. Meindl, *Supply Chain Management: Strategy, Planning, and Operation*. Prentice Hall, 2001. [Online]. Available: https://books.google.com/books?id=gd22AAAAIAAJ

[10] O. Lysne, K. J. Hole, C. Otterstad, . Ytrehus, R. Aarseth, and J. Tellnes, "Vendor malware: Detection limits and mitigation," *Computer*, vol. 49, no. 8, pp. 62–69, 2016.

[11] Core Security. (2016) Intel driver update utility mitm. [Online]. Available: https://www.coresecurity.com/advisories/intel-driver-update-utility-mitm

[12] "Hardware setup of minnowboard," "http://wiki.minnowboard.org/", 2016.

[13] "Spi hook," "http://www.tincantools.com/", 2016.

[14] "Download ubuntu desktop," "https://www.ubuntu.com/download/desktop", 2016.

[15] P. Edelstein, "Comparing costs and roi of aoi and axi," *Electronics Production and Test Europe*, vol. 1, no. 2, 2007. [Online]. Available: https://www.smtnet.com/library/files/upload/EPPEuropeArticle.pdf

[16] B. Schneier, "Attack trees," *Dr. Dobbs journal*, vol. 24, no. 12, pp. 21–29, 1999.

[17] D. Shackleford, "Combatting cyber risks in the supply chain," *SANS. org*, 2015.

[18] "Edk ii faq," "https://github.com/tianocore/tianocore.github.io/wiki/EDK2015.

[19] "Minnowboard turbot," "http://wiki.minnowboard.org/MinnowBoard_Turbot", 2016.

[20] S. Oresjo, "When to use aoi, when to use axi, and when to use both," *Nepcon West, December*, pp. 4–6, 2002.

[21] y. Virus Total, "Virustotal-free online virus, malware and url scanner."