Accepted for publication and to appear in the proceedings of the
IEEE International Symposium on Technologies for Homeland Security, October 2018

Enabling Hardware Trojan Detection and Prevention through Emulation

Alfred Crouch
Director of Hardware Engineering
Amida Technology Solutions, Inc
Washington, DC 20036, USA
Alfred@amida.com

Eve Hunter
Cybersecurity Engineer
Amida Technology Solutions, Inc.
Washington, DC 20036, USA
Eve@Amida.com

Peter L. Levin
Co-Founder and CEO
Amida Technology Solutions, Inc.
Washington, DC 20036, USA
Peter@Amida.com

***Abstract*— Hardware Trojans, implantable at a myriad of points within the supply chain, are difficult to detect and identify. By emulating systems on programmable hardware, the authors have created a tool from which to create and evaluate Trojan attack signatures and therefore enable better Trojan detection (for in-service systems) and prevention (for in-design systems).**

Keywords—Trojan, FPGA, hardware assurance, HWA

I. INTRODUCTION

Hardware vulnerabilities, whether intentional or inadvertent, can be exploited to: 1) leak data or structure to expose critical data or allow reverse-engineering; 2) modify device or system behavior to prevent the device or system from accomplishing its intended mission, substitute an alternate mission, or allow a non-authorized party to take over and operate the device or system; or 3) degrade, break, or destroy the device or system's hardware, operation, or reputation. These are the fundamental classes of Trojan payloads as illuminated by our systematic review of hundreds of Trojans; we have named them: Data Leakers, Behavior Modifiers, and Reliability Impacts.

From a black-hat attacker's perspective, it is crucial that a Trojan placed within a system remains hidden, disguised, or sequestered until the ideal or scheduled opportunity arises, and payload activation is triggered. This results in a so-called zero-day vulnerability. Successful attacks meticulously preserve the "stealth factor" of the Trojan to prevent the Trojan from being discovered or accidentally activated too early — such as during device testing or characterization. Some Trojans that have been discussed in the public literature use the ScanEnable signal's assertion (an indication that testing is underway) to power down the domain that includes the Trojan circuit [1]. Some Trojans are simply over-doped or under-doped transistors that make certain gates sensitive to voltage or temperature — but these Trojans leave no previously detectable trace in any functional model (behavioral, gate, or physical layout). Trojan payloads are often implemented with minimal impact to design overhead, and can be as small as a single transistor or route segment, or as complicated as a group of gates or a multiplexor.

Triggers (the event that activates the Trojan payload), on the other hand, are often very sophisticated. Triggers are generally designed as rare-events (also known as rare-states), or complex and contrived sequences, that prevent the Trojan from being discovered or prematurely activated. The triggers are often conditionals such as comparators, timers, state-machines, or hidden items such as unused or intentionally mislabeled pins. Whenever possible, a Trojan designer will use existing features and co-opt them to provide the trigger event.

The most sophisticated and difficult-to-find triggers and Trojan payloads are the side-channel or parametric effects that use temperature, voltage, power, and other non-digital effects to either trigger a Trojan payload or as a Trojan payload (for example, leaking data by modulating a temperature hot spot). Through this type of insidious behavior, a Trojan may surreptitiously exfiltrate or obfuscate critical data without impacting any of the normal circuit operation. For example, an RFID (radio frequency identification) circuit can be hidden within a semiconductor's package, and can radiate stored, stolen data whenever a reader signals the RFID element — the RFID element uses no power and remains passive. It only actively emits when a reader is in proximity and activates the circuit (it does not interfere or otherwise alter the normal operation of the device).

Consequently, hardware Trojans are extremely difficult to detect, especially before they are triggered, and as such, they are often implemented with the aim of exploiting high value targets critical to national security.

In this paper, the authors present a novel tool that combines the findings of other researchers with recent observations to create a new model that identifies behavioral signatures of hardware Trojans through the emulation of pristine systems and the gamut of hardware Trojans on a programmable device (FPGA). This tool, named the Trojan Emulation and Exploration Environment (TE3©), reproduces and baselines a Golden Model system on an FPGA device, introduces numerous Trojans from the Data Leaker, Behavior Modifier, and Reliability Impact categories, collects subsequent data, and compares the two models (infected and

pristine) through automated analysis of real-time data collection and instruments. TE3© will enable identification, localization, and ideally eradication (but at the very least, management) of stealthy hardware Trojans. We will describe how hardware Trojans are: 1) able to be deconstructed into distinct and repeatable units, and 2) are therefore both detectable and identifiable.

II. MOTIVATION

A. Overview of the Problem

The absence of a reliably secure supply chain can permit an adversary to weaponize hardware in two ways: infected systems with hardwired backdoors and vulnerabilities that allow an adversary to change the device's behavior, or counterfeit parts and systems, which may simply fail at unacceptable rates.

The use of commercial off-the-shelf (COTS) hardware components in critical systems, including those used by the United States Department of Defense (DoD), creates serious security challenges. In addition to inherent vulnerabilities of the embedded code, there is a risk that the components themselves may be exploited because of accidental design flaws, or by the insertion of deliberate Trojans or vestigial logic into the device during the fabrication and manufacturing process. Component-level threats compromise the complete system, either by disabling or degrading key functions or by creating backdoors that can be manipulated by malware.

Developing systematic methods to anticipate, block, and potentially remediate both compromised software and contaminated hardware is essential and extremely difficult. Ordinary software threats are vectors into presumably pristine hardware; we can build in "valid function" surveillance to defend against attack. However, if the silicon platform itself is compromise-able, or compromised, developing electronic defense becomes a much more difficult proposition. Vulnerable or exposed hardware itself must now be included in the design and development process.

The problem is therefore multi-layered: 1) Trojans may be implanted into hardware components at any time within the design and manufacturing processes; 2) manufacturer verifications of security are less meaningful due to attacker sophistication and often insider knowledge; and 3) end-users are unable to decipher whether a change in device behavior is an ordinary design error, a hardware defect or a malicious Trojan.

B. History/Literature Review

The study of Trojan detection is currently driven primarily by academia and is a loose collection of various techniques that can be applied at various stages of design, development, and deployment. The main focus of pre-manufacturing efforts involves evaluating files and either 1) searching for anomalously written code or unnecessary code elements that would generate hardware features and functions in either a very specific manner (such as generating a state machine in such a manner where the normally un-specified states would actually be mapped into real functions that could be used as a trigger); or 2) identifying code that would generate extra hardware circuit features not required by the device specification, but that could be used to allow external attacks, such as viruses, to have a place to store their malware code (such as adding a shadow register in real memory-mapped code storage space). Other methods require that the "model to be assessed" be written in a particular format or formal language so unnecessary code or added code can easily be identified, which impacts the cost and schedule of the design and development cycle.

For example, a team from New York University advocates physical Trojan insertion into FPGA designs; they created an Electronic Design Automation (EDA) tool to assist with that process [2] but they limit their effort to evaluation of designs meant to become FPGA-based systems. Many of the post-manufacturing analyses require complex test, verification, and characterization methods to be applied in an attempt to activate (trigger) a Trojan by producing the rare event or generating the test vector that allows the exploration of the redundant fault space (the assumption that the trigger or the Trojan is hidden in the non-detected or redundant fault space for which vectors cannot easily be generated by EDA automation [ATPG] tools). Increasing the amount of testing can directly impact the cost of produced devices and electronics and can directly impact the reliability lifetime (e.g. 24-hour burn-in type testing within an environmental chamber and constant vector application to attempt to activate any hidden Trojans).

No documented methodology has considered a comprehensive physical environment that includes the use of the design model emulated as a physical device, and to then undergo Trojan exploration through emulation. In our TE3© environment, we include Trojan detection instruments that are selected, matched, and placed based on the Trojan selected for insertion. Once the Trojan has been inserted, the Trojan detection instruments can be used to generate data that, once collected, can generate a signature and allow a data analysis to assess the "threat impact." An overall threat impact resolution can be used to recommend which of the Trojan detection instruments should be deployed to enhance detection and identification of critical, catastrophic or destructive threats — and the TE3© environment allows exploration of the countermeasures that can be applied to manage or obviate those Trojans.

III. APPROACH

A. Hardware Emulation

Not only can the payload of a hardware (HW) Trojan affect the physical device, in some cases Trojans can only be triggered by physical manipulation of the device or the device's environment. The purpose of emulating this system, as opposed to simulating it in software, is to enable use of

parametric triggers and to allow detection of parametric effects (also referred to as side-channel effects, such as radio frequency [RF], temperature, voltage, etc.) that may damage or alter the functionality of the target system. The emulation environment provides the ability to include parametric HW attacks in the testbed, and therefore may be able to reduce the likelihood of such an attack going unnoticed in an operational target system.

The TE3's© physical emulation environment supports a pristine Golden Model, data collection instruments, and the insertion of a range of Trojans. The hardware emulation process entails: selecting an FPGA emulation platform, establishing a connection between the platform and a computer to begin to program and to provide communication with the FPGA, designing a Golden Model that represents the target system, synthesizing the Verilog or VHDL of the Golden Model to an FPGA programmable form (a bit file), and ultimately operating the resulting device while actively collecting data – or, in other words, emulating the Golden Model.

B. Trojan Insertion

Our primary method of assessing a given design model, implementation model, or silicon device for vulnerabilities is to emulate the device within a programmable framework and insert Trojans into it. Then, in simplified terms, one can generate characteristic signatures for operational anomalies based on the type of hardware Trojan: Data Leakers, Behavior Modifiers, or Reliability Impacts.

Our methodology is as follows: first, we emulated a pristine Golden Model on the FPGA board (as described in Section II.A.). We inserted emulated Trojans into the Golden Model, drawing upon existing academic literature, and our knowledge of our Golden Model to determine which Trojan types would be most representative of the threat landscape, and where to place them in the FPGA platform [3]. We then methodically and individually inserted each Trojan into the Golden Model to observe the physical, electrical, logical, and behavioral anomalies they introduced.

In practice, Trojans may be inserted at several places in the design – development – fabrication – packaging – distribution flow. The pre-fabrication steps, our focus for this work, have several potential entry points – the modeling process, Intellectual Property (IP) acquisition, IP integration, synthesis, test insertion, layout, and routing. When a system is attacked at these points, the adversary leaves evidence of the Trojan in the design files, but that does not make the Trojans easy to find. The impact of pre-fabrication attacks is that they can impact the current design and reuse of the files, IP, or design generation tools can result in Trojans existing in future designs or derivations of the current design.

Ideally, Trojan insertion should model all potential alternatives of a Trojan existing within a real device or system — placing the Trojan in the design model, the gate implementation model, the physical implementation model, and even within the packaging step. However, the goal of this work is not to provide an exact and specific match to a real Trojan that could exist within an FPGA-based design, but to emulate the effect of the Trojan against a design that is targeted to become a real device or system such as an ASIC, SoC, ASSP, or an electronic board. We note that programming a model into an FPGA will not create the same exact physical implementation as that model being processed to become a silicon device such as an ASIC or SoC — the behavior will be the same, even the cycle-by-cycle response will be the same, but the gates, routes, and other elements may be resolved and implemented differently.

Understanding this concept, then, almost all Trojans can be emulated by adding trigger elements and Trojan payload elements to the behavior or register-transfer (RTL) model that must pass through the FPGA synthesis process. Triggers can be represented by simple conditional elements such as comparators, timers and state machines, or by direct connections to package pins, and Trojans can be represented by simple gates (AND, OR, NAND, NOR, XOR, XNOR, or MUX).

An analysis of the model can identify the best locations for Trojan placement using two main considerations: the type of Trojan to be inserted and the consideration of stealth (where can the goals be met with the least amount of added model impact?). For example, in order to implement a Data Leaker, there must be something inside the model worth leaking — critical data, a novel design structure, or codes. This Trojan selection process establishes the target and the stealth consideration establishes the best locations — "where can I leak the target data by using the least amount of logic, gates, power consumption, etc.?" A similar analysis can be done for the behavior modifier Trojan payload — "where can I take over a function or stop a function and with the least amount of logic, gates, power consumption, thermal impact, etc.?" And the answer is generally where a function resolves down to a single assertion signal or pinch point (single point of failure).

C. Data Collection

Data collection is usually performed on a hardware device for one of the following reasons: bug hunting for design errors, determining location or root cause of a defect, or monitoring the health of the system. Regardless of the purpose, collecting data enables the system owner to have a holistic, comprehensive picture of the target system. The purpose of the TE3© is to continuously collect data and monitor for any changes in behavior that may be indicative of the presence of a Trojan – and so a fourth category can be added – to ascertain hardware assurance in the context of trust and security.

We have further defined three categories of data collection for security: Basic, Enhanced, and Advanced. The Basic level of data collection is the operator's manual or visual observation of indicators or display elements that an internal function is or is not working properly (a Trojan signature may include the fact that the visible display is not corrupted and all things look and act normal). Logged errors for Basic data collection include, for example, if a delivered message is corrupted and does not match the sent message, or if a message exceeds the apportioned time for completion. Operators will be notified via

custom created alerts and alarms to indicate that the operation is beyond its normal bounds.

To implement Enhanced Data Collection, passive data collection units are inserted (data collection instruments) into the TE3© and run continuously to collect or generate data in the background. In general, passive resources can be considered part of the pristine, unaffected emulated system. For example, a system's normal features include a timer that will produce a Time-Out alert if the sending and receiving of communications take too long. Another example would be built-in error-correction associated with data communication (where a data error at a receiver would assert a Bit-Error signal and ask for a re-transmit of the corrupted packet). These items increase the number of tracked variables for a given system; they allow, for example, logging the increase in Bit-Error Rate over a specified time period or applying Time-Framing (tracking the time used to complete operations successfully) to the operations that may be targeted for Trojan attacks.

Data Leaker Trojans present a special case for enhanced data collection. These Trojans often attempt to access hidden data, structure or information within the device without observably impacting behavior. Stealth is the primary attack objective and therefore hidden resources (hidden/unused package pins or seldom-accessed register or memory locations) are often co-opted for this purpose. To detect Data Leaker Trojans, instruments must be configured to identify and monitor unused resources that would normally be disconnected, un-powered or quiescent for any type of anomalous activity (e.g. logic level changes, signal or clock edges).

The Advanced Data Collection effort involves the introduction of autonomous, non-digital instruments that convert the non-functional or non-digital parameters (e.g. temperature-monitoring ring-oscillator) into a digital value (that could be interpreted by data analysis software). The goals of this data collection are: 1) to detect a non-digital trigger that would activate the Trojan payload (e.g. using an RF pulse; modulating the hardware power supply in a particular sequence; tampering with the hardware device's temperature using cooling spray, etc.); and 2) to detect the difference in the side-channel statistics after a Trojan has been operated or activated or, if possible, after a Trojan has been included in the design, but before activation.

For initial development of the TE3©, we created a library of data collection instruments aimed at Trojan detection such as pin monitors, JTAG Instruction Monitors, and side-channel monitors. The goal of these instruments is to: 1) capture intermediate data or states along the normal pathway to help discriminate Trojan vs. non-Trojan when doing any data collection activities; 2) capture the data or activity associated with non-pathway ports or pins to see if portions of the circuit that are not part of the normal operational pathway are being used to exfiltrate data or information; and 3) capture the non-digital signature of either the normal pathway or the non-pathway portions of the circuit to see if a side-channel is being used either as a trigger or as a Trojan payload. The identified instruments reside within an IJTAG (IEEE 1687) network. Each instrument is sequestered behind a SIB (segment insertion bit) within an IJTAG scan path to allow the instruments to be turned on and off and removed from the active serial scan chain as needed (to manage power and operational noise).

The aggregate of this data collected by the various monitors provides a recognizable pattern with which to determine the instruments that are best suited to detect each Trojan. The association of data collection monitors to Trojans enables more accurate and reliable detection of embedded threats (note that some Trojans may require multiple different instruments).

To extract data from the FPGA board and embedded monitors, we used an IJTAG architecture paired with the ASSET ScanWorks® software program. The 1687-2014 IJTAG Standard [4] describes the use of embedded instruments in conjunction with an 1149.1 Test Access Port (TAP) and JTAG TAP Controller [5]. The two standards, when coupled together, create an efficient embedded instrument access architecture.

D. Data Analysis

The TE3's© automated data analysis function can detect, classify, locate, and assess Trojans, as well as trigger countermeasures to mitigate their effects. The near-term objective is to decrease the time needed to identify whether a Trojan is present and, if so, its characteristics, location and threat impact (i.e. degraded, critical, or catastrophic). This in turn will allow for quick prescription of countermeasures and proper application of tactics, techniques, and procedures (TTPs) for the specific Trojan.

Typically, human testers use a script to discriminate Trojan-infected outputs from pristine baselines. Our ultimate objective, however, is to use machine learning to broaden the aperture of detectable intrusion. The anomaly detection algorithms will be trained on a wide, and growing, range of available variables.

To guarantee detection without overloading on-board resources, we must be able to select precisely which instruments to collect data from; a subset of all the instruments may be enough to detect a particular Trojan. In the future, our team will integrate machine learning algorithms into the data acquisition process to streamline Trojan detection. We will also identify a Trojan's physical location, as well as a logical attachment. When an instrument captures anomalous behavior, we will see which logic is impacted and/or which physical location of the chip becomes active.

From there, we can trace the Trojan's behavior because, generally, when multiple instruments are activated in a system, one of them will log the first change. This corrupted state propagates through the system and causes other instruments to capture results. Using timestamps and instrument location tags, we can build a picture of how and where the corruption started and then how it propagated as time went on – these are techniques that have been proven using debug-diagnosis data collection architectures and methods. The data analyzed for each Trojan will be fed back into the analysis engine in order to better detect and identify future Trojans tested on the TE3©.

IV. Results

The authors have developed a proof-of-concept model of the TE3© to demonstrate the feasibility of this approach. In the process of development, we achieved our four main objectives:

1. Develop a representative pristine "Golden Model" and show that it can be emulated on an FPGA platform;
2. Insert seven fundamental canonical Trojans that represent the gamut of possible Trojans (Data Leaker, Behavior Modifier, Reliability Impact and their dominant sub-categories);
3. Collect physical data from the FPGA platform;
4. Analyze collected data and definitively detect the presence and categorize each type of embedded Trojan.

Each of these four objectives served to validate the TE3's© main functions or features: hardware emulation, Trojan insertion, data collection and data analysis. Based on this successful proof-of-concept, we are now actively working to increase automation of the TE3© methodology.

V. Conclusions

Influential organizations implement COTS hardware on a daily basis; there is an implied trust present in that purchase and installation. We believe that such trust is too easily betrayed, and that the TE3© can act as a validating and even diagnosing secondary check on critical systems. The TE3© has two primary use cases:

1. To make a model of a COTS-based system and to explore its weaknesses with Trojan insertion – responses to Trojans can be captured as signatures and used as "look up table results" for real system behaviors.
2. To evaluate a system-under-design for "design-for-security" issues by using Trojan insertion. Exploration can lead to a more robustly secure 'hardened' design that includes detection and countermeasure instruments.

Whether the TE3© is used pre-manufacturing or post-manufacturing (in service), it will provide an essential security mechanism to ensure that (until now) stealthy, devastating embedded attacks are caught before adverse effects occur – and if they occur, they can be understood quickly and the threat assessed, understood, and managed. Further development will result in a model that may be scaled to emulate any number of complex environments, and this can be continuously tested and refined to allow detection of and response to subsequent new and dangerous threats.

Acknowledgment

The authors would like to thank their colleagues – Jackie Medina – for her invaluable assistance; Sergio Orellana and Matthew Ritonia for their help in developing the TE3 and reviewing this paper; as well as ASSET InterTech for their work in helping to develop the Golden Model and the JTAG and IJTAG portions of the design and operation infrastructure.

References

[1] V. Jyothi, P. Krishnamurthy, F. Khorrami, and R. Karri, "TAINT: Tool for Automated INsertion of Trojans," IEEE International Conference on Computer Design (ICCD), November 2017.

[2] W. Hu, B. Mao, J. Oberg, and R. Kastner, "Detecting Hardware Trojans with Gate-Level Information-Flow Tracking," Computer, 49(8), pp. 44-52, August 2016.

[3] X. Zhang and M. Tehranipoor, "Case study: Detecting hardware Trojans in third-party digital IP cores," *2011 IEEE International Symposium on Hardware-Oriented Security and Trust*, pp. 67–70, June 2011.

[4] "IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device," *IEEE Std 1687-2014*, December 2014.

[5] "IEEE Standard for Test Access Port and Boundary-Scan Architecture, " *IEEE Std 1149.1-2013 Revis. IEEE Std 1149.1-2001*, May 2013.