Applied Cryptography and Network Security Report

Defense against hardware Trojans

Diya Ilinani

December 16, 2021

1 Introduction

In recent years, the complexity and frequency of the implementation of hardware Trojans has grown as they are being used to access or tamper with important applications used by the government such as military cyber-infrastructure. A hardware Trojan is implemented by introducing malicious circuitry into a hardware component and disguising the application as a useful device. The Trojan could be used to leak sensitive information from the IC (e.g. by radiating electromagnetic waves that are then intercepted and interpreted by the attacker), to perform a Denial-of-service attack which would stop the device from functioning entirely, or to degrade the performance or change the functionality of the IC (by giving inconsistent outputs).

In this report, various methods to defend Integrated Circuits against Trojans and to detect them have been discussed.

2 Design of a hardware Trojan

Typically, a hardware Trojan consists of two distinct parts: the trigger and the payload. The trigger is the part that will read every input received by the circuit to discern each signal individually or the pattern of signals that are inputted. The payload is the part of the Trojan that executes malicious activities by activating additional circuitry. It is is usually activated after the trigger compares the signals it receives to a specific value and sends a signal to the payload triggering it.

Often, the trigger event is an uncommon event or a combination of events that is uncommon resulting in the IC behaving as a benign component most of the time, which would make the Trojan hard to detect. The method in which a Trojan operates has been demonstrated below.

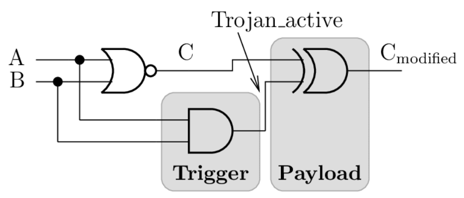

Figure 1: This figure has been taken from [1] , C modified is the inverted value of C which is what the Trojan outputs when it is activated.

3 Defense against Trojans

There are two ways in which we can defend ourselves from the existence of a Trojan, the first is to detect its presence and discontinue the use of that hardware component and secondly, we can hinder its malicious properties by ensuring that the payload part of the Trojan is never activated.

3.1 Trojan Detection

Due to the addition of multiple redundant paths in the circuitry, Trojans will often change the behavior of the IC. We can take advantage of this change by comparing the characterisitcs of an unknown IC with that of a known Trojan-free IC. In [2], various techniques to detect the activity of a Trojan are described as specified below.

3.1.1 Functional Tests

Functional tests are performed by giving a large number of input signals to the IC to activate the payload of the Trojan and recording the corresponding outputs. These outputs are compared to the actual output that was supposed to occur to detect possible change in functionality. The problem with this method is that the large number of permutations of input signals makes it highly inefficient and oftentimes impossible.In [3], the researchers attempted to detect patterns that would be rare input signals to the IC in order to trigger the Trojan, however, this method also suffers from the same inhibitions as above. It should be noted that this method would be completely useless if the Trojan was only attempting to transmit confidential information and not affect the functionality.

3.1.2 Side-Channel Analysis

When a Trojan is present, regardless of the effect that it conspires to have, the path taken by an input signal will be changed to execute malicious activity or to go through the operation of the trigger itself , this implies that the characteristics of the IC would inevitably change as opposed to a Trojan-free circuit. These characteristics can be in the form of heat, electromagnetic waves produced, the amount of time it takes the IC to process the input, or the amount of power consumed during a fixed amount of time. Side-channel analysis is the method of comparing these statistics to a known "golden reference" which is the statistics of the same IC without the Trojan. Although this method doesn't depend on the activation of the Trojan like functional testing, it is highly unreliable in the scenario that the changes caused by the Trojan are too small to offset deviations caused due to manufacturing inconsistencies. It also depends on the fact that the same IC has been manufactured before, and we have an accurate metric of its statistics to compare an unknown version of the IC to.

3.1.3 Functional validation and Code coverage in the design stage

The design of a hardware component is decided by writing its functionality in the Hardware Description Language. If an attacker was to tamper with the code in the design process itself, it would affect the IC without the attacker ever being involved in the manufacturing process. To detect such instances, we can perform tests on the code itself by simulating a large number of inputs and comparing the output with a benign circuit as we have seen above in functional testing, it is important to acknowledge that this would also suffer from the same cons. Another way to use the code to detect Trojans is to check for redundancy in the form of redundant sections of code or redundant paths taken by the IC. This is done by examining the code coverage (which is the percentage of number of lines of code used) while running inputs , a smaller amount of code coverage would indicate a higher chance of the existence of a Trojan because the paths which are not covered when the input is run could be used for malicious behavior.

3.2 Bypass Trojans

In the above section, we have to depend on detecting changes that occur in the behavior of a circuit, however, it is possible to dodge the execution of the logic of the Trojan itself by preventing it from ever triggering. In [4], the researchers tailored this technique to different types of triggers as described below.

3.2.1 Ticking time-bomb trigger

In a ticking time-bomb trigger, The Trojan is automatically activated after a fixed amount of time passes.
This trigger is bypassed by frequently turning off the power of an IC and then switching it back on, this will ensure that the clock of the Trojan is not aware of the amount of time that has passed, and the payload will never be activated.

3.2.2 Input signal trigger

The Trojan is activated when it receives a specific input value.
This trigger is combated by performing data obfuscation of the input received. The type of obfuscation that is allowed depends on the operations performed by a unit.

Non-computational Units: If a unit does not have to perform any logical operations and only has to move data around, then input signal triggers can be avoided by applying a small encryption function on any input to the unit and then decrypting the output. This encryption function can be in the form of a simple XOR of the input with a random value, the output for which is decrypted by XORing it with the same chosen value.

Computational Units: Here, the input could be changed by adding another parameter so it doesn't activate the trigger.
- For example , lets take a unit that computes the square of a value ($f(x) = x^2$).The input can be changed by multiplying it with another number.
- To get x^2 we choose a number y and pass it as the input to get $(xy)^2$ ($f(xy) = (xy)^2$) square and then we get the actual output we need by dividing $(xy)^2$ (the output) by y^2 square (known value as y is a chosen parameter) ($f(x) = f(xy) / f(y)$). This ensures that the value x is never actually passed to the IC which would prevent it from getting triggered if the input signal trigger was x.

3.2.3 Sequence of Input signals

The Trojan is activated by a sequence of multiple inputs.
Sequence Breaking: The input to a unit is obfuscated by reordering the input sequence and ensuring that the reorder does not impact functionality. If it is not possible to reorder without preserving correctness then dummy events are added such that the sequence changes but the result of the unit is unaffected. However, this method could be ineffective if the codes need not be consecutive to activate the trigger.

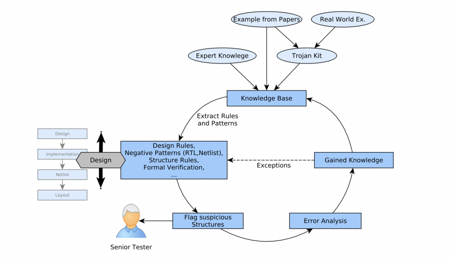

Figure 2: The above figure taken from [5] shows the rules that were used to find the list of indicators in [5].

4 Proposed Method

The following list is a collection of potential malicious indicators in hardware that were compiled by the properties of existing malicious structures in hardware. **This list has been taken from an existing paper [5] and my proposal is to use this existing list to analyze hardware ICs detect trojans.** In malicious software signatures of existing malware are often used in combination with machine learning to identify new malicious software. In the method I have proposed, this idea is used on the list. The procedure is to record the frequency of occurrence of potential malicious indicators for a data set of known malicious hardware Trojans and benign ICs.These frequencies are then used to train machine learning classifier algorithms and run on unknown ICs. List of common components used to design malware in hardware:

- Asynchronous latch: A latch not clocked by one of the typically low number of global clocks.
- Gated wire or output: A latch not clocked by one of the typically low number of global clocks.
- Ring oscillator: A latch not clocked by one of the typically low number of global clocks.
- Unused pin or bond wire: Used for dissemination.
- Hidden finite state machine (FSM) state(s)
- Latch or Flipflop independent from global reset
- Local or gated clock

An example of how this list could be utilized in malicious hardware is, an unused pin could be used to transmit leaked information by radiating electromagnetic waves. A ring oscillator is a sequence of an odd number of NOT gates whose output changes frequency when the voltage changes. A Trojan could use this behavior by making a ring oscillator a part of the trigger which is activated by measuring outside parameters like voltage fluctuations. An asynchronous latch could be used as the trigger for malicious behavior by having the input be two rare events. Although, these usage of these components in malware is justified, recording only the existence of these components would not be very revealing because they are often used in perfectly harmless devices to execute tasks that don't actually participate in malicious activity, resulting in a lot of false positives. Therefore, my attempt is to derive features by recording patterns of malicious logic using this list. We know that comparing the frequency of indicators of malicious behavior generally works for the detection of malicious software so its not far-fetched to think that its possible to detect malicious behavior in hardware in the same way.
One way of detecting patterns of malicious circuitry is to see what elements the above components are connected to, for example, lets take the first element in the list, if the IC has an asynchronous latch and a large amount of malware has the combination (asynchronous latch ————> XOR gate) or (asynchronous latch ————> OR gate) then the presence of this combination would be a good feature to use while attempting to detect a Trojan. My solution is to identify all of the above components and the elements they are connect to in the IC, the frequency of (this) occurrence of some element in the above list connected to that particular input is taken as a feature. For example, the frequency of every latch independent from the global reset is recorded as a feature, as is the frequency of an independent latch connected to an XOR gate (independent latch ————> XOR gate) , NOR gate, asynchronous latch etc.

These frequencies will then be used as features to train classifier algorithms using ICs which are known to be either malicious or benign.This algorithm is run on an unknown IC which will

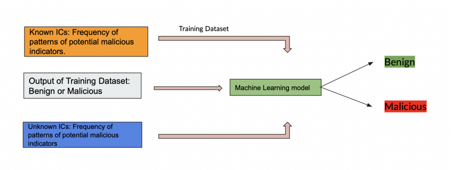
Known ICs: Frequency of patterns of potential malicious indicators.
Training Dataset
Output of Training Dataset: Benign or Malicious
Machine Learning model
Benign
Malicious
Unknown ICs: Frequency of patterns of potential malicious indicators

Figure 3: Proposed Method

be classify it as benign or malicious. I believe that my reasoning is convincing because software malware detectors often record sequences or frequencies of system calls that are generated as feature vectors, which has been shown to be effective. In the context of hardware, the above described patterns are the malicious indicators as opposed to API calls in software.

5 Conclusion

Although, hardware Trojans are a formidable threat to security of applications and have been on a significant rise, the existing methods to detect and bypass Trojans are outdated. They have a number of flaws and a lot of dependencies. New techniques can be developed by examining the underlying logic of a malicious circuit and discerning its patterns. The method that has been proposed attempts to carry this out analogous to how software Trojans are detected.

References

[1] Bhasin, Shivam Danger, Jean-Luc Guilley, Sylvain Ngo, Xuan Thuy Sauvage, Laurent. (2013). Hardware Trojan Horses in Cryptographic IP Cores. Proceedings - 10th Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2013. 15-29. 10.1109/FDTC.2013.15.

[2] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor. 2016. Hardware Trojans: Lessons Learned after One Decade of Research. ACM Trans. Des. Autom. Electron. Syst. 22, 1, Article 6 (December 2016), 23 pages. DOI:https://doi.org/10.1145/2906147

[3] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan. 2014. Hardware Trojan attacks: Threat analysis and countermeasures. Proceedings of the IEEE 102, 8 (Aug. 2014), 1229–1247. DOI:http://dx.doi.org/10.1109/JPROC.2014.2334493

[4] A. Waksman and S. Sethumadhavan, "Silencing Hardware Backdoors," 2011 IEEE Symposium on Security and Privacy, 2011, pp. 49-63, doi: 10.1109/SP.2011.27.

[5] A. Dabrowski, H. Hobel, J. Ullrich, K. Krombholz and E. Weippl, "Towards a Hardware Trojan Detection Cycle," 2014 Ninth International Conference on Availability, Reliability and Security, 2014, pp. 287-294, doi: 10.1109/ARES.2014.45.