Project Status Report: Analyzing automated security assets identification mechanisms applicable to the SoC architecture (in HW, FW, and SW levels)

Project

GRC Task - 3124.001 - An Evolutionary AI-based Fuzz Testing for Extensive SoC Security Verification.

Performers

Muhammad Monir Hossain, Dr. Farimah Farahmandi, and Dr. Mark Tehranipoor (PI).

1. Introduction

Due to the complexity and size of SoCs, the security assets of SoCs are increasingly exposed to adversaries in a variety of ways, making it difficult to protect the diversified security assets of SoCs [1,2]. In order to ensure that security assets are protected against adversaries, the first step is to identify them in all forms in SoCs. We can classify the assets of an SoC into two types: (i) primary assets and (ii) secondary assets [5]. Primary assets are the definitive targets of SoC, worthy of protection. Secondary assets are those that interact closely with primary assets, requiring protection to ensure the security of primary assets. As part of the design specification, designers and verification engineers are provided with a number of primary security assets. Some examples of primary assets are cryptographic keys, random numbers and seeds, manufacturer keys and IDs, digital rights management (DRM) keys, end-user private data, and configuration bits for operational and privilege modes [3]. There are a variety of infrastructures involved in monitoring, transferring, and using these primary assets, such as wrappers, test infrastructures, etc. The importance of identifying and protecting these infrastructures is due to the fact that they contain information about primary assets. Designers must determine which types of assets should be protected against which types of attacks. One of the most challenging decisions during the design process is determining which asset is most appropriate for a particular set of attacks. Choosing the wrong asset can lead to a waste of time and resources, as well as leaving the design susceptible to unwanted attacks. Modifying a design at the end of the SoC design process is ten times more expensive than doing so at the beginning of the process, according to the rule of ten for product design [4]. Therefore, the assets must be identified and protective measures need to be confirmed at the pre-silicon verification stage.

To develop security countermeasures, it will be less time-consuming and less expensive for design houses to begin with a definite set of security assets. However, it may be difficult to always determine which assets of infrastructure are more critical, what security risks are associated with these assets, and what security schemes should be designed for them. Security assets are also selected based on user applications and threat models. Cryptosystems are designed to protect cryptographic keys against unauthorized access, such as the leaking of the keys. A microcontroller, however, may require a security scheme to avoid integrity violations (for example, unauthorized access to the counter's value can modify program sequences). Due to the different types of assets to protect, coupled with the underlying system implementations, each of these assets is subject to a diverse range of attacks. As a result of the need to recognize the broad spectrum of security assets

spread across different IP blocks and abstraction levels, we require to develop a framework for identifying an SoC's security assets.

2. SoC Assets Identification

Generally, the primary asset is considered to be a part of a trusted system and should only interact with other trusted components. During the design and specification analysis of an SoC design, the designer enlists the primary assets, trusted and untrusted observable points in the SoC. In the SoC design, secondary assets are explored based on the definition of primary assets. In this report, we discuss how to identify secondary assets in the SoC for comprehensive security framework design for verification engineers, as primary assets are obvious and defined by the SoC design and specification.

3. Framework for Secondary Assets Identification

As a means of detecting secondary assets in the SoC, we propose a framework inspired by the SAIF framework [5] developed by our research group. With the help of this proposed framework, semiconductor design houses can develop properties to protect the assets of the SoC by detecting secondary assets. As a part of the application of the proposed framework, we assume verification engineers possess the knowledge of I/O ports of the integrated IPs, SoC functional specifications, primary assets, trusted and untrusted observable points, and potential malicious attacks related to the assets to some extent. The proposed framework has three major stages: (i) primary asset propagation analysis, (ii) candidate components identification, and (iii) detection of secondary assets. We describe these major stages of the framework in this section.

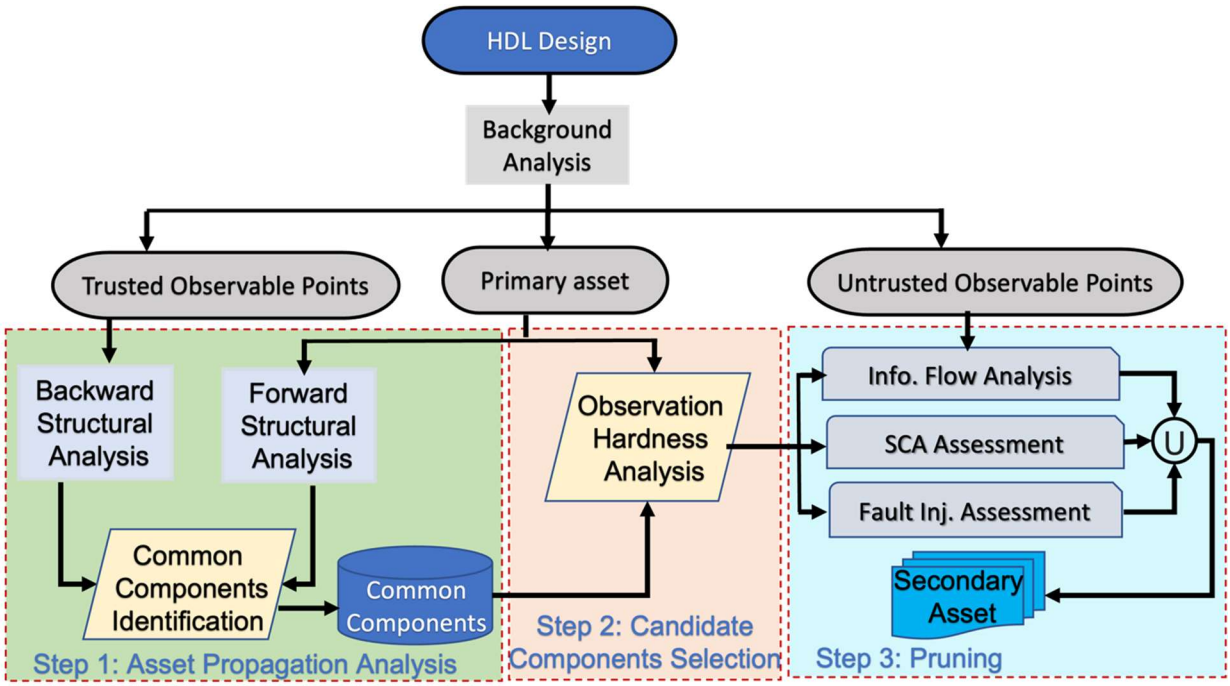

Figure 1: Framework for secondary assets identification.

3.1 Primary Asset Propagation Analysis

At this stage, we analyze the propagation of primary assets to all internal nodes of the SoC. We consider any potential leakage of the primary asset, even a bit. For example, leaking one bit of the AES key can reduce the attack space significantly. Our framework takes the RTL design of the SoC, a list of primary assets of the SoC, and the trusted observable points. We do a structural analysis of the RTL of the hardware design/SoC to find all common intermediate components between each primary asset and trusted observable point. First, we do a forward structural analysis to find the components in the primary asset's fan-out cone. We calculate the sequential depth for each component to all primary outputs. Next, we perform backward structural analysis to find all components in the fan-in of trusted observable points. We also calculate the sequential depth from all trusted observable points to all primary inputs. Finally, we extract the common components that closely associate with the primary assets and may possess any information from the primary assets to any trusted observable points.

3.2 Candidate Components Identification

At this stage, we identify potential secondary components, i.e., candidate components. Components and blocks that are far from the primary assets cannot be exploited as easily as those that are near the primary assets. Therefore, in the proposed framework, the hardness of observing a primary asset from a component or block is estimated for a particular hardware design using fault modeling [6]. The following is a description of the observation hardness metric H(P, R) that we propose to quantify the fault impact of a primary asset on a component.

$$H(P,R) = \frac{DF_R}{IF_P} \quad \text{..... (1)}$$

In Equation 1, DF_R: the number of faults detected at component R, IF_P: the total number of faults injected at the primary asset P.

A threshold value of observation hardness (H_{TH}) is requested during the verification process for identifying the assets. The proposed framework identifies candidate components with the observation hardness greater than H_{TH}. The threshold value is determined based on the expected robustness (objective) against the vulnerabilities, i.e., compromised level of security assets. For more robustness, the verification engineers must keep the threshold value low. Therefore, even if there is less correlation of the infrastructure(s) with the primary assets should be identified as the potential candidate of secondary asset. An intermediate component with a higher value of H indicates that an attacker can gain access to the asset value from that location without much difficulty. Consequently, when a component's observation hardness is greater than the threshold value (H_{TH}), it becomes a potential candidate for secondary asset identification.

3.3 Detection of Secondary Assets

The purpose of this stage is to analyze the data obtained from the previous stage to identify the most vulnerable candidate components. The most vulnerable candidates are referred to as secondary assets, and they should be protected against adversaries. We analyze each of the candidate components (from the second step) to see whether it has any potential vulnerability characteristics or behavior by looking at the vulnerability database. A security path verification approach based on information flow analysis (IFA) is employed in order to identify the most

vulnerable design paths to security attacks [7]. A tainted source signal is used to determine if the taint can be recovered at the destination node using path sensitization techniques. Considering the candidate components as a tainted source and untrusted observable points as destinations, we propose developing security properties. Using formal verification tools, we evaluate whether vulnerable paths exist between candidate components and untrusted observable points. It appears that information flow leakage has occurred for those candidates for which such a security path exists. The identified candidates with information leakage vulnerabilities are therefore subjected to further pruning analysis in order to quantify other inherent vulnerabilities such as fault injection attacks assessment and side-channel leakage assessments.

4. Future work

The framework presented in this report identifies secondary security assets of the SoC. The purpose of detecting security assets is to develop security properties to be checked by a formal or dynamic tool for security verification at later stages as a part of SoC security verification. We intend to develop methodologies to protect both primary and secondary assets considering attack scenarios for known vulnerabilities and also unknown vulnerabilities utilizing evolutionary techniques. Automating verification is one of the biggest bottlenecks in current practices. We object to expediting the verification flow by automating the framework upon detection of assets and thus enhancing the security assurance of the SoC. Again, we intend to do some rigorous experiments with the framework to identify assets in some real SoCs in the future.

5. Conclusion

Using the proposed technique, the verification or the design engineers can identify the secondary assets who already possess the primary assets. Identifying the assets of the SoC helps the verification engineer to develop the security properties based on the asset type and the corresponding adversarial attacks to exploit the vulnerability. Thus, we can increase the security coverage of the SoC and protect against malicious attacks on the developed SoC applications.

6. References

1. Farzana, Nusrat, Fahim Rahman, Mark Tehranipoor, and Farimah Farahmandi. "Soc security verification using property checking." In *2019 IEEE International Test Conference (ITC)*, pp. 1-10. IEEE, 2019.
2. Azar, Kimia Zamiri, Muhammad Monir Hossain, Arash Vafaei, Hasan Al Shaikh, Nurun N. Mondol, Fahim Rahman, Mark Tehranipoor, and Farimah Farahmandi. "Fuzz, Penetration, and AI Testing for SoC Security Verification: Challenges and Solutions." *Cryptology ePrint Archive* (2022).
3. Basak, Abhishek, Swarup Bhunia, and Sandip Ray. "A flexible architecture for systematic implementation of SoC security policies." *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2015.
4. Anderson, David M. *Design for manufacturability: How to use concurrent engineering to rapidly develop low-cost, high-quality products for lean production*. Productivity Press, 2020.
5. Farzana, Nusrat, Avinash Ayalasomayajula, Fahim Rahman, Farimah Farahmandi, and Mark Tehranipoor. "Saif: Automated asset identification for security verification at the register transfer level." In *2021 IEEE 39th VLSI Test Symposium (VTS)*, pp. 1-7. IEEE, 2021.
6. Karunaratne, M., A. Sagahayroon, and S. Prodhuturi. "RTL fault modeling." In *48th Midwest Symposium on Circuits and Systems, 2005.*, pp. 1717-1720. IEEE, 2005.
7. Guo, Xiaolong, Raj Gautam Dutta, Jiaji He, Mark M. Tehranipoor, and Yier Jin. "Qif-verilog: Quantitative information-flow based hardware description languages for pre-silicon security assessment." In *2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 91-100. IEEE, 2019.