SoCurity: A Design Approach for Enhancing SoC Security

Naorin Hossain, Alper Buyuktosunoglu, John-David Wellman, Pradip Bose, and Margaret Martonosi

***Abstract*—We propose SoCurity, the first NoC counter-based hardware monitoring approach for enhancing heterogeneous SoC security. With SoCurity, we develop a fast, lightweight anomalous activity detection system leveraging semi-supervised machine learning models that require no prior attack knowledge for detecting anomalies. We demonstrate our techniques with a case study on a real SoC for a connected autonomous vehicle system and find up to 96% detection accuracy.**

***Index Terms*—Heterogeneous SoC, network-on-chip, denial-of-service, anomaly detection, semi-supervised model.**

I. Introduction

SYSTEMS-ON-A-CHIP (SoCs) offer high-performance processing with low area and power overheads. Heterogeneous SoC designs target a device's computational needs by bringing together specialized processing units tailored to accelerate the device's tasks, all on a single die. State-of-the-art SoCs use network-on-chip (NoC) interconnects for efficient communications between these SoC components [1], [3]. Components can include general-purpose processing cores, hardware accelerators, memory interfaces, and I/O interfaces. Some components may be black-box third-party intellectual property (IP) units.

Despite the performance and power gains achieved by specialized SoC designs, sophisticated attacks can target and take down components, rendering them unavailable. For example, a denial-of service (DoS) attack may be launched by malware (Fig. 1), flooding the NoC with packets targeting specific SoC resources and preventing necessary tasks from reaching them [3]. Prior works used hardware performance counters for malware detection in traditional CPU architectures, but existing SoC-specific solutions are not broadly applicable to real-world heterogeneous SoCs. Our goal is a widely applicable SoC design approach to protect against resource availability attacks. To this end, we propose *SoCurity*, a design method for enhancing security in heterogeneous SoCs. Our contributions are:

Manuscript received 15 June 2023; accepted 9 July 2023. Date of publication 3 August 2023; date of current version 31 August 2023. This work was supported in part by National Science Foundation (NSF) under Grant 1763838, and based on research sponsored in part by the Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under Grant FA8650-18-2-7862. The work of Margaret Martonosi was supported in part by NSF as an IPA rotator. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF, AFRL and DARPA or the U.S. Government. Distribution Statement "A": Approved for Public Release, Distribution Unlimited. *(Corresponding author: Naorin Hossain.)*

Naorin Hossain and Margaret Martonosi are with Princeton University, Princeton, NJ 08544 USA (e-mail: nhossain@princeton.edu; mrm@princeton.edu).

Alper Buyuktosunoglu, John-David Wellman, and Pradip Bose are with the IBM Research, Yorktown Heights, NY 10598 USA (e-mail: alperb@us.ibm.com; wellman@us.ibm.com; pbose@us.ibm.com).

Digital Object Identifier 10.1109/LCA.2023.3301448

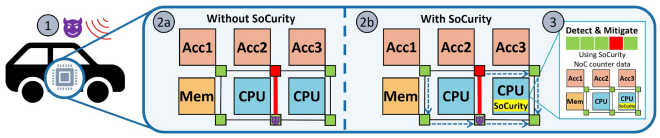


Fig. 1. Example SoC attack: ① Malware is introduced in a network data transfer. ②a Malware launches DoS attack on an accelerator (Acc2) in an SoC without SoCurity (red: increased NoC traffic; green: regular traffic). NoC traffic and task load from the DoS attack make Acc2 inaccessible for device tasks. ②b and ③ show how SoCurity helps. ②b SoCurity anomaly detection system is deployed on a core in the SoC. NoC counter data from each component is collected by the SoCurity unit on an isolated NoC plane (dashed). ③ Anomalous NoC data (red) is found in counters and can be used to find and mitigate threat.

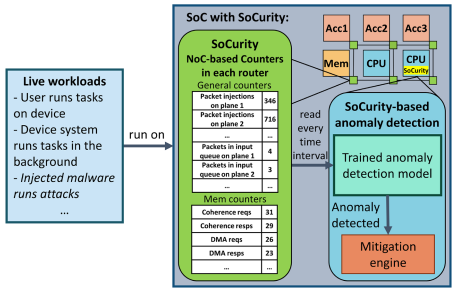


Fig. 2. SoCurity introduces counters to NoC routers for each SoC tile to monitor interactions between them (example counters shown). The counters enable constantly active anomaly detection and mitigation during live workloads.

- SoCurity, a novel SoC monitoring approach to enhance security with easily integrable NoC-based hardware counters.
- A lightweight, automated anomalous activity detection system targeting unwarranted on-chip SoC resource usage.
- Our detection system needs no attack knowledge; developed with an adaptive approach based on semi-supervised machine learning, it enables novel future attack detection.
- We demonstrate these techniques with a study on a real SoC for connected autonomous vehicles (CAVs) in high variability settings. Our results show highly accurate DoS detection (up to 96%) and fast prediction ($\sim$30 μs on ASIC).

In this paper, we focus on the most common availability attack, DoS attacks, though other attack protection is also possible with SoCurity. The disruption from malware and hardware trojans leveraged to flood the NoC with tasks, is distinguishable at the hardware level, inspiring our approach to developing an SoC anomaly detection system.

II. SoCurity: NoC Counter-Based SoC Security

SoCurity enhances security measures on heterogeneous SoCs with NoC-based hardware counters. This paper develops one example of an anomalous activity detection system that can

be implemented with SoCurity. Fig. 2 gives an overview of SoCurity, including NoC counters and the detection system.

A. Threat Model

This work focuses on safeguarding heterogeneous SoCs against attacks that disrupt component operation and accessibility, e.g., DoS attacks or side-channel attacks. Possible attack vectors include hardware trojans embedded by third-party IP tiles and malware injection via connected devices or the cloud (e.g., with system updates). This paper primarily focuses on detecting malware injection-based attacks, though presented techniques can also detect anomalous hardware activity from other described attack vectors. Our threat model assumes the software stack, including the operating system, is untrusted. The attacker is assumed to have knowledge of on-chip accelerators and the ability to send tasks to them. The case study (Section III) highlights detection of DoS attacks on SoC components.

B. NoC-Based SoC Hardware Counters

In a heterogeneous SoC, the presence of black-box third-party IP and various processing units requires that a holistic approach to observing hardware activity throughout the SoC either a) requires a specific interface implemented by each component to monitor and expose its internal activity, or b) brings monitoring outside these components to the NoC-level where usage of and communications between components can be tracked. SoCurity takes the latter approach, as the NoC interconnect provides a reliable medium for monitoring SoC activity without imposing additional requirements on component designs.

NoC-based counters can capture a holistic view of ongoing SoC activity by monitoring: 1) NoC packets injected by each tile, 2) NoC congestion at each tile, 3) memory tile requests, and 4) accelerator usage. Counters can be implemented as protected registers at the router for each tile (examples in Fig. 2). The area overhead of implementing the counters is negligible ($<$0.1%) [11]. NoC packet counters can increment at source and destination routers for each packet. NoC congestion counters can track packet input queue size at each router. Memory tiles counters can track coherence and direct memory access (DMA) requests by incrementing on packet arrival from cores and accelerators, respectively. Accelerator routers can count usage cycles by recording cycles between a packet arrival from a core and a departure of a response packet to that core. Our proposed counters are generic and applicable for any NoC-based SoC.

C. Anomaly Detection Methods

Using the SoCurity NoC monitoring approach, we developed an anomalous activity detection system. The goal for our system is to use a lightweight algorithm to quickly flag uncharacteristic hardware behavior without searching for specific known attack behaviors. Though complex solutions using deep neural networks may provide high accuracy, they require immense processing and memory resources to do so which is not ideal for fast detection in devices with limited resources. Instead, for broader applicability, we explore machine learning (ML) based anomaly detection models that provide accurate and fast detection with lighter computational requirements.

For our system, we studied a subset of semi-supervised ML algorithms specific to anomaly detection. They are binary classification models that are trained solely with benign data; anomalous data need not be included in the training set. Training data is used to define bounded regions of possible benign activity. Samples falling outside of these regions get labeled *anomalous*. With no anomalous training data, these models are capable of capturing both existing and novel future attacks. We considered four semi-supervised anomaly detection models: one-class nearest neighbors (OCNN), one-class support vector machines (OCSVM), isolation forest (iForest), and local outlier factor (LOF). These models are well-studied anomaly detection algorithms and their simplicity, speed, accuracy, and general applicability best fit our goals. For our case study (Section III), we used available model implementations from the Python scikit-learn library and wrote our own OCNN implementation based on the k-Nearest Neighbors (kNN) module in scikit-learn.

D. End-to-End Anomaly Detection System

Model Training: Effective anomaly detection with a semi-supervised approach relies on strong training data that accurately portrays regular system behavior. Collecting representative data sets is feasible since SoCs are designed for specific uses. The detection model is statically trained with counter data collected during representative workloads in a secure, offline setting. Once deployed, the trained model can be updated periodically with data that is collected during the device's use.

Secure Implementation: When deployed on an SoC, the base configuration for our proposed anomaly detection system is executed on an isolated core at the highest privilege level, independent of the software stack, protecting it against attackers (Fig. 2). Alternatively, the detection system can be implemented as a standalone hardware unit, enhancing its security from intruders. For both cases, the NoC counters should only be accessible to the detection system through a dedicated NoC plane that is unavailable to other SoC units. This NoC plane introduces minimal area overhead at $<$2% [7]. Memory for the detection model and counter data should reside in a trusted, secure space that only the detection system can access without interrupting ongoing program executions. After detection, the anomalous activity source must be verified to mitigate threats.

Full defense process for malware-injected attack:

1) NoC counter data is collected while running representative workloads on the SoC in a secure, offline setting.
2) Anomaly detection model is trained with collected data.
3) Trained model is deployed on the SoC for live detection.
4) NoC counter data is regularly read and input to the model.
5) Detection model predicts if the counter data is anomalous.
6) If the data is anomalous, it is sent to a mitigation engine.
7) The mitigation engine finds the anomalous process and mitigates it, or determines the data was mispredicted.
8) Counter data labeled benign by detection or mitigation engine is used to update detection model over time.

III. Case Study: Connected Autonomous Vehicle

We studied SoCurity detection on a real SoC designed for a CAV. CAVs share data on surroundings through "swarm" communications with nearby vehicles [10]. Despite complexity and high input variability, we show our techniques are effective.

A. CAV SoC Implementation

We implemented an SoC prototype for the swarm communication component of a full CAV SoC, emulated on a Xilinx Virtex UltraScale+ VCU118 FPGA running at 78 MHz using the ESP platform [7], [11]. Our SoC has a 64-bit RISC-V Ariane

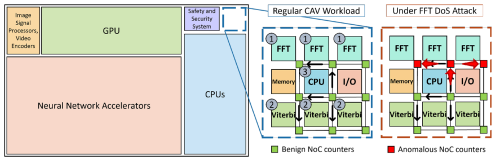


Fig. 3. Overview of implemented SoC in a larger CAV SoC (e.g., Tesla SoC [1]). Tiles are labeled 1-3 with CAV workload tasks in Section III-A. Black arrows depict benign network traffic. Red arrows (right) show FFT DoS attack traffic.

core, a memory tile, an I/O tile, 3 FFT accelerators, and 3 Viterbi decoder accelerators (Fig. 3). The Ariane core has a 16 KB L1 I-cache and a 32 KB L1 D-cache. The memory tile has a 256 KB LLC. Tiles are connected by a 2D mesh NoC with 6 planes.

Workload: Our SoC ran a representative CAV workload focused on swarm communication for navigational decision-making (Fig. 3) using a generated trace of upcoming road obstacle positions. It performed the following at each time step:

1) Calculate distance to surrounding objects with FFT calculations on radar readings done by an FFT accelerator.
2) Messages from swarm communications (i.e., message traffic) are processed by Viterbi decoders. Messages vary in size. One to three messages are received per time step. This reflects real-world variety in swarm communication.
3) Combine data from 1 and 2 to plan and execute actions.

Implementing NoC-Based Counters: In this study, we deployed NoC counters in our SoC (Section II-B) using Cohmeleon's lightweight hardware monitoring system [11]. We monitor 355 total NoC counters spanning the following events:

- *General:* NoC packets, NoC backpressure cycles
- *Memory:* off-chip memory accesses, coherence/DMA requests/responses, LLC hits/misses
- *Accelerator:* total, memory, and TLB-loading cycles

B. DoS Attack Targets

We ran four DoS variants on the SoC to slow down CAV tasks:

1) *FFT:* 1,000 FFT accelerator tasks are injected.
2) *Viterbi:* 1,000 randomly sized Viterbi tasks are sent.
3) *FFT + Viterbi:* 1,000 tasks are sent to FFT, Viterbi tiles.
4) *Memory:* 500,000 memory requests are made for random addresses in a space that is twice the size of the LLC.

C. Evaluation Methods

We conducted offline and online experiments to evaluate SoCurity detection on each attack. Based on offline studies, we deployed the best ML model on the SoC for online tests.

Training: Training data was collected by reading counters every 100 ms over three workload (Section III-A) runs on the SoC, each for 300 time steps (∼155 ms per step). This was the most frequent data collection possible on our single-core SoC. The training set had 1094 samples with 355 counter features.

Testing: Test sets were collected by reading counters every 100 ms over five runs of each DoS attack on the CAV workload and five runs of the workload alone. The workload ran for 300 time steps. Test sets had 3,684 to 4,932 samples. Ground truth labels for test data were automated with k-means clusters.

Preprocessing: To maintain model performance, we reduced our feature set from 355 NoC counters by: 1) selecting the k best features based on variance, 2) using correlation matrices to select unique counters, and 3) combining counters into informative features, e.g., summing usage cycles for all FFT accelerators. Table I gives our final six representative features.

TABLE I
SIX FEATURES USED FOR ANOMALOUS ACTIVITY DETECTION

Counter Type	Feature Descriptions
General	• CPU NoC packets for memory-mapped register access, I/O reads/writes, interrupt requests
Memory	• Off-chip memory accesses • DMA requests • Coherence responses
Accelerator	• FFT total cycles • Viterbi total cycles

TABLE II
AVG. ATTACK DURATION, TOTAL WORKLOAD DURATION WITH ATTACKS, AND ATTACK IMPACT (HIGHER VALUE, HIGHER IMPACT)

Attack	Attack Time (s)	Total Time (s)	Impact
No attack	–	46.56	–
FFT	11.55	55.59	5.75
Viterbi	94.13	146.46	4.89
FFT + Viterbi	104.13	155.74	5.00
Memory	40.89	61.71	2.00

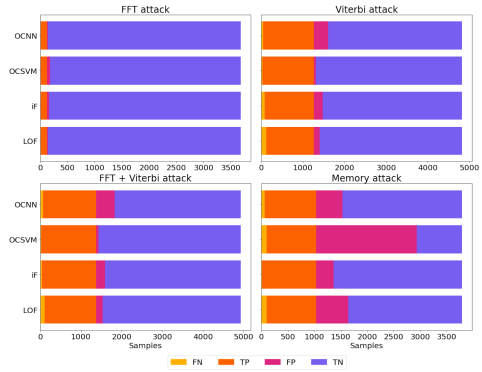


Fig. 4. FN, TP, FP, TN for each attack/model. Low FN, FP are better. All did well on FFT attacks. OCSVM was best on Viterbi and FFT + Viterbi attacks. All had high FPs on memory attacks.

IV. RESULTS

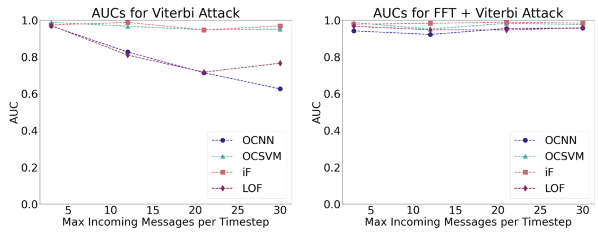
Table II gives each DoS attack duration, time to run the CAV workload for 300 time steps during attacks (on 78 MHz FPGA), and an impact metric for overall effect of attacks on the workload. Impact is quantified as the ratio of workload slowdown ($\frac{t_{total}}{t_{no\_attack}}$) to the proportion of the run in which the attack occurred ($\frac{t_{attack}}{t_{total}}$).

A. Comparing Models for DoS Attack Detection

To evaluate the detection models, we counted true/false positive (TP/FP, anomalous) and true/false negative (TN/FN, benign) predictions. We tuned model parameters to prioritize low FNs (<10%), minimizing missed anomalies, and low FPs, avoiding system resource costs. Fig. 4 shows FNs, TPs, FPs, and TNs for each model and attack. Higher *impact* attacks (Table II) were detected more easily. Due to its lower impact, memory attack data was similar to benign data. Minimizing FNs resulted in models that were more susceptible to predicting FPs for the memory attack test set. Storage-wise, iForest, LOF, and OCNN required 807.3 KB, 316.0 KB, and 187.0 KB, respectively. OCSVM required just 2.9 KB, making it lightweight and ideal for fast online detection. It also best distinguished variable benign message traffic from Viterbi-based attacks.

B. SoCurity Robustness

We stress tested our detection system to see how Viterbi-based attack detection is impacted under high incoming message traffic conditions, e.g., extreme urban settings with large CAV



(a) Viterbi Attack (b) FFT + Viterbi Attack

Fig. 5. AUC of each model when incoming message traffic variability for CAV workload increases. Higher is better. OCSVM and iForest maintain high AUCs despite increased variability.

TABLE III
AVERAGE DETECTION CYCLES TO FLAG ATTACK (LOWER IS BETTER), ACCURACY (HIGHER IS BETTER), AND FPR (LOWER IS BETTER) PER ATTACK

Attack	Detection Cycles	Accuracy	FPR
FFT	0.96	0.96	0.04
Viterbi	2.82	0.87	0.05
FFT + Viterbi	1.06	0.87	0.06
Memory	2.22	0.60	0.05

presence. We randomized message traffic with up to 3, 12, 21, and 30 incoming messages of random size per time step. Beyond three messages put a higher task load on Viterbi units. Fig. 5 shows areas under receiver operating characteristic curves (AUCs, higher is better) for detecting Viterbi and FFT + Viterbi attacks under these workload conditions with each detection model. Distance-based OCNN and LOF could not separate high Viterbi activity from benign message traffic and the Viterbi attack, but fared better with added FFT noise in the FFT + Viterbi attack. OCSVM and iForest incorporate feature relations for a stronger model and maintained high AUCs for both attacks with increasing benign message traffic.

C. SoCurity in Action on the CAV SoC

We selected OCSVM for our online study due to its low storage needs and robust performance across offline experiments. We statically trained and deployed the model on the FPGA using m2cgen. When run on the 78 MHz FPGA, the model had a prediction time of 320μs (32μs on an ASIC [7]). For each *detection cycle*, NoC counters were read every 100 ms (like training data) and input to the detection model for prediction. Table III presents the average number of detection cycles required to flag attacks after they are launched, accuracy ($\frac{TP+TN}{TP+FP+TN+FN}$), and false positivity rates (FPRs: $\frac{FP}{TN+FP}$) for each DoS attack on the FPGA. The model had low FPRs for all attacks and high accuracy for accelerator-based attacks. The memory attack's lower impact (Table II) made counter data harder to recognize as anomalous during some detection cycles, so overall accuracy was lower. Nonetheless, it was flagged within a few detection cycles, so a mitigation engine would quickly investigate it.

V. DISCUSSION

Scalability: Larger SoCs with SoCurity require more NoC counters throughout. To reduce load on a detection unit, detection can be parallelized across multiple units, each monitoring a subset of tiles and maintaining regional NoC traffic knowledge.

Future Work: SoCurity is simple and generalizable, so it can be used to detect other SoC threats (e.g., hardware trojans or side channel attacks) or hardware resource failures. SoCurity detection systems can also be built with evasion attacks in mind by randomly selecting from a set of detection models [6].

VI. RELATED WORK

Several works explored CPU hardware performance counters for anomaly and attack detection using ML models [4], [5], [6], [9]. As discussed in [3], others have explored security measures for NoC-based SoCs in which CPU performance counters are not a sufficient solution [2], [8], [9]. [8] detected bandwidth denial attacks by a malicious NoC. [9] evaluated malware detection with CPU performance counters from SoC cores and found high FPRs averaging 11.3%. [9] also extracted coherence NoC traffic information from packets to detect DoS attacks. [2] relied on predictability of NoC traffic latencies. SoCurity NoC counters are simpler and provide more coverage for heterogeneous SoCs by not requiring information extraction from packets and monitoring more than just coherence packets. SoCurity counters are also inaccessible through software tools like `perf` which can be used to reverse-engineer CPU counter-based detectors [5]. Further, we found the semi-supervised ML models used in our study to outperform supervised counterparts from related work [4], [9], but omit these results for space.

VII. CONCLUSION

This letter presents SoCurity, a design approach for enhancing SoC security by enabling NoC-level hardware activity monitoring with NoC-based counters. Using SoCurity, we developed a fast and lightweight anomaly detection system for detecting impacts on SoC component availability. We demonstrated our detection system on a real SoC implementation for a CAV and showed highly accurate (up to 96%) DoS attack detection.

REFERENCES

[1] P. Bannon, G. Venkataramanan, D. D. Sarma, and E. Talpes, "Computer and redundancy solution for the full self-driving computer," in *Proc. IEEE Hot Chips 31 Symp.*, 2019, pp. 1–22.
[2] S. Charles et al., "Real-time detection and localization of distributed DoS attacks in NoC-based SoCs," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 39, no. 12, pp. 4510–4523, Dec. 2020.
[3] S. Charles, Y. Lyu, and P. Mishra, "A survey of network-on-chip security attacks and countermeasures," *ACM Comput. Surv.*, vol. 54, 2021, Art. no. 101.
[4] J. Demme et al., "On the feasibility of online malware detection with performance counters," in *Proc. 40th Annu. Int. Symp. Comput. Architecture*, 2013, pp. 559–570.
[5] S. M. P. Dinakarrao et al., "Adversarial attack on microarchitectural events based malware detectors," in *Proc. IEEE/ACM 56th Des. Autom. Conf.*, 2019, pp. 1–6.
[6] M. S. Islam, K. N. Khasawneh, N. Abu-Ghazaleh, D. Ponomarev, and L. Yu, "Efficient hardware malware detectors that are resilient to adversarial evasion," *IEEE Trans. Comput.*, vol. 71, no. 11, pp. 2872–2887, Nov. 2022.
[7] T. Jia et al., "A 12nm agile-designed SoC for swarm-based perception with heterogeneous IP blocks, a reconfigurable memory hierarchy, and an 800 MHz multi-plane NoC," in *Proc. IEEE 48th Eur. Solid State Circuits Conf.*, 2022, pp. 269–272.
[8] R. JS et al., "Runtime detection of a bandwidth denial attack from a rogue network-on-chip," in *Proc. 9th Int. Symp. Netw.-on-Chip*, 2015, Art. no. 8.
[9] Z. Pan, J. Sheldon, C. Sudusinghe, S. Charles, and P. Mishra, "Hardware-assisted malware detection using machine learning," in *Proc. Des. Autom. Test Europe Conf. Exhib.*, 2021, pp. 1775–1780.
[10] A. Vega, A. Buyuktosunoglu, and P. Bose, "Towards "smarter" vehicles through cloud-backed swarm cognition," in *Proc. Intell. Veh. Symp.*, 2018, pp. 1079–1086.
[11] J. Zuckerman, D. Giri, J. Kwon, P. Mantovani, and L. P. Carloni, "Cohmeleon: Learning-based orchestration of accelerator coherence in heterogeneous SoCs," in *Proc. IEEE/ACM 54th Annu. Int. Symp. Microarchitecture*, 2021, pp. 350–365.