2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS) | 978-1-6654-1913-0/20/$31.00 ©2021 IEEE | DOI: 10.1109/AICAS51828.2021.9458488

Evaluation of Machine Learning-based Detection against Side-Channel Attacks on Autonomous Vehicle

Han Wang[1], Soheil Salehi[1], Hossein Sayadi[2], Avesta Sasan[3], Tinoosh Mohsenin[4],
Sai Manoj P D[3], Setareh Rafatirad[1], Houman Homayoun[1]
[1]University of California at Davis, [2]California State University at Long Beach
[3]George Mason University, [4]University of Maryland at Baltimore County
[1]{hjlwang,ssalehi,hhomayoun,srafatirad}@ucdavis.edu,
[2]{hossein.sayadi}@csulb.edu,[3]{asasan,spudukot}@gmu.edu,[4]{tinoosh}@umbc.edu

***Abstract*—Autonomous vehicles are becoming increasingly popular, but their reliance on computer systems to sense and operate in the physical world has introduced new security risks. Recent studies have shown that using Cache-based Side-Channel Attacks (SCAs) could infer sensitive users' information (e.g., which route the user is taking) highlighting significant vulnerability posed to today's computer systems. As a result, it is crucial to propose effective detection mechanisms against emerging microarchitectural SCAs on autonomous driving systems. In response, we first identify the threat model and victim applications of autonomous driving systems in this work. Next, we explore the suitability of various machine learning-based classifiers trained by information collected from built-in hardware performance counter registers available in modern autonomous vehicle systems. To this end, various supervised machine learning models are implemented for cache-based SCAs detection and precisely compared and characterized in terms of detection accuracy, robustness, and latency of the detection. Our experiments conducted on an Intel Xeon, which Waymo autonomous driving vendor uses, demonstrate that J48 achieves 99.5% accuracy with the highest efficiency compared with other investigated models.**

***Index Terms*—Autonomous vehicles, side-channel attacks, machine learning, hardware performance counters.**

I. Introduction

Autonomous vehicles have achieved great success in academia and industry due to the progress made in cheaper sensors, higher computation capability of processors, and effective object detection. Many passenger vehicles are also equipped with autonomous driving capabilities, like Tesla [1], Uber [2], Waymo [3], Baidu [4], etc. As shown in Figure 1, there are three main components in autonomous vehicles including sensors, electronic control component, and compute unit. Sensors include camera, radar, GPS, etc. used for collecting information from the real world environment. The control component is in charge of operations on vehicles, like putting the brake on a car. For the computing unit, there are two parts: navigation system and utility. The navigation system is equipped with path planning, localization, and perception ability based on the information collected from sensors and reinforcement learning and deep learning techniques. The computation result from the navigation system is a command like turning right, stopping, etc. which are sent to the electronic control component. Since it can directly influence the operations of a car, they are security-critical and provided by vendors. The other part running on the computing unit is utility including cloud services and entertainment apps, which could be provided by a third party. Hence, they share compute unit with the navigation system.

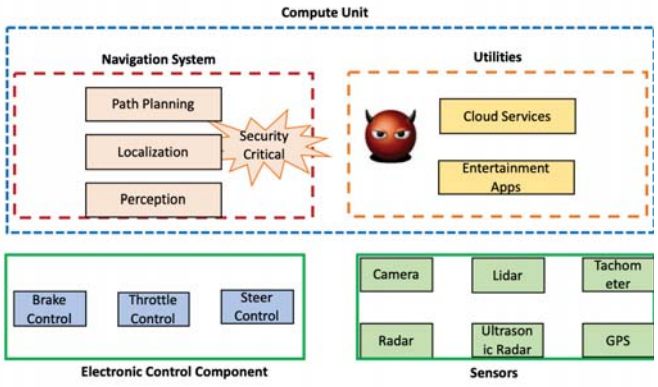

Fig. 1: General hardware and software architecture of an autonomous vehicle

Though great success obtained,recent advancements have shown that the autonomous driving architecture introduces new vulnerabilities and attacking surface for cache-based side-channel attacks (SCAs). Recent work [5] demonstrates that using Prime+Probe [6], [7] can infer the location and route that the user is taking by inferring the data access pattern of the Adaptive Monte-Carlo localization (AMCL) algorithm. It could infer the route/location of drivers with up to 81% accuracy for route prediction and 75% accuracy for location prediction. In recent years, there are a number of new types of SCAs developed, like Flush+Reload [8], Flush+Flush [9], Prime+Probe [7], etc. that attempt to infer users' sensitive information.

Several recent works have proposed to modify cache hierarchy or cache memory architecture to mitigate SCAs. Cache partitioning techniques [10], [11] are proposed to mitigate cache-based side-channel attacks by statically or dynami-

978-1-6654-1913-0/21/$31.00 ©2021 IEEE

cally partitioning cache memory for each application process. Thereby SCAs are not able to observe "side-channel information" of victim applications. Another approach employs access randomization [12]–[14] which primarily randomizes cache interference, remaps cache indices, or replaces demand fetch with random cache fill to eliminate security vulnerabilities in the hardware architecture. However, such works require new cache and cache memory translation architectures that pose extra design costs and can not be applied to legacy systems. Hence, it is urgent to develop an efficient SCAs detector with low performance and computation cost.

To address the challenge of SCAs, there are some machine learning (ML) classifiers based detectors proposed [15]–[18] to monitoring victim applications under no attack (normal) and under attacks scenarios and then applying ML classifiers to conduct binary prediction for capturing SCAs. While all the works are done in desktops, the underlying architectures and demand for detectors are different, making prior detectors not applicable for autonomous driving platforms. Hence, in this work first the threat model of SCAs and corresponding victim applications in autonomous driving systems are introduced. Next, we propose a highly accurate machine learning-based countermeasure for autonomous vehicles against emerging cache-based side-channel attacks. The proposed ML-based countermeasure is trained with microarchitectural features collected from Hardware Performance Counter (HPC) registers available in modern microprocessors. We explored a comprehensive range of ML classifiers to determine the most accurate and efficient model across various performance metrics including detection accuracy, robustness, and efficiency.

II. Machine Learning based Detector

In this section, we first introduce the hardware architectures used in commercial autonomous vehicles and then present the details of our proposed ML-based detector with run-time microarchitectural behaviors against SCAs.

A. Threat Model

This work mainly focuses on the software attacks that exploit hardware vulnerability (shared memory and cache) and infer location, route, and other private information without permission to sensor data and access to the physical measurement. As introduced in Section I, there are two types of applications residing in compute unit: navigation system and utilities. Since the navigation system is developed and maintained by vehicle vendors, applications from the utility are more likely to be inserted with SCAs' codes. Hence, this work considers SCAs installed in the utility part and resided in the same compute unit with the navigation system, indicating the shared cache hierarchy between attacks and programs in the navigation system.

TABLE I: Common Compute Unit for Autonomous Vehicles

Vendors	Architecture	Compute Unit
Baidu [4]	ARM	Arm Cortex-A53, FPGA, GPU
Telsa [1]	ARM	Cortex-A72 CPU, GPU, Accelerator
Waymo [3]	x86	Intel Xeon CPU, GPU
Uber [2]	x86	Intel CPU, GPU

B. Hardware Performance Counters

Modern microprocessors are equipped with a set of special-purpose registers for measuring hardware-related events, i.e. hardware performance counters (HPCs). Both the ARM and Intel x86 architecture provide the performance monitoring unit (PMU) interface to access HPCs. Since the Pentium, Intel processors enable PMU feature while the ARMv6 is the first PMU-enabled ARM processor and the afterward ARM11, Cortex-R, and Cortex-A cores also provide the functionality. As listed in Table I, we report the most prevalent processors used by top car vendors and research. It shows that ARM and x86 Intel architectures are the two most popular ones, and GPU, FPGA are included for the acceleration of complex computation. Intel processors are also commonly used for research [19], [20]. It is also noticeable that all processors listed in the Table I have access to hardware performance counters, indicating that using hardware performance counters to detect side-channel attacks is a viable approach.

C. Building ML-based Detector

As shown in Figure 2, there are three main steps to constructing a ML-based SCAs detector: a) data collection and feature evaluation; b) Training classifier ; c) online testing the optimal predictive model.

TABLE II: List of HPC events collected for SCAs detection

L1 HIT	L1 MISSES
L2 HIT	L2 MISSES
L3 HIT	L3 MISSES
All BRANCHES RETIRED	BRANCHES MISPREDICTED
BR_NONTAKEN_CONDITIONAL	BR_TAKEN_CONDITIONAL
TAKEN_INDIRECT_NEAR_CALL	UOPS_RETIRED.ALL
INST_RETIRED.ANY	DTLB_LOAD_MISSES
DTLB_STORE_MISSES	ITLB_MISSES

1) Data Collection and Feature Evaluation: The tested attacks are Flush+Reload, Flush+Flush, and Prime+Probe while victim applications use Fast R-CNN [21] for Perception , the adaptive Monte-Carlo Localization (AMCL) [22] for localization, optimization method [23] for path planning . Since their computation patterns might reveal users' environment, location, or other privacy as demonstrated in prior work [5]. Run-time microarchitectural behaviors of victim applications are collected by Perf [24] tool to form a database with a known label ("under no attack" or "under attack"). Based on the behavior and functionality of studied SCAs, 16 HPC features are considered in this work for further analysis as listed in Table II. Since the HPCs can be collected simultaneously, it is important to identify the most prominent HPCs. These hardware performance counters data are collected using the four available HPC registers in the experimented Xeon processor at every sampling interval (10 ms). Next, both "under attack" and "under no attack" HPC data from each same sampling intervals are merged to create the final dataset for the corresponding sampling interval.

2) Training Classifier Selection: The ML classifiers evaluated in this work that are selected from five different categories, including NaiveBayes, Multi-Layer Perceptron (MLP),

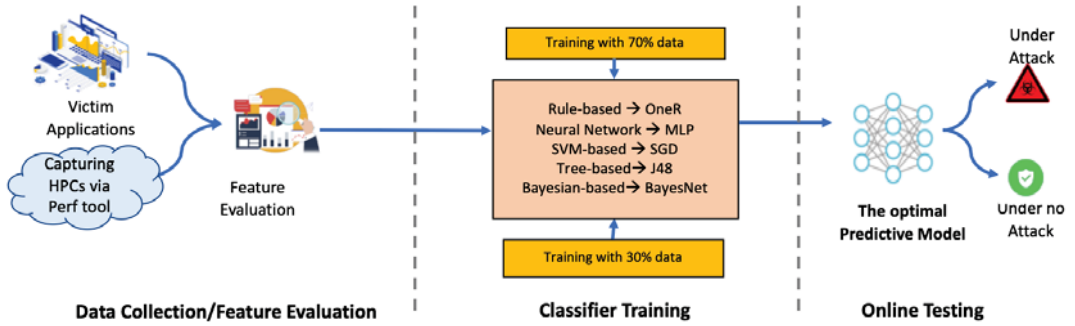

Fig. 2: Overview of the ML-based Detector

SGD, OneR, and J48. The rationale for choosing these machine learning models is that they are from different branches of ML including Bayesian network-based, neural network, support vector machine, rule-based, and tree-based techniques covering a diverse range of learning algorithms that are inclusive of modeling both linear and nonlinear problems. The prediction model produced by these learning algorithms can be a binary classification model that is compatible with the SCAs detection problem in our work. Furthermore, Weka data mining tool is leveraged for implementing the ML classifiers. A standard 70%-30% dataset split for training and testing is conducted to validate each of the utilized ML classifiers. Next, for the percentage split testing 70% of the randomized data is used for training the classifiers, and the rest of 30% is used for testing evaluation. In addition, a k-fold (k=10) cross-validation is also conducted on the training dataset.

3) Online Testing: Once classifiers are trained with the 70% dataset and tested with the rest 30%, the optimal classifier is chosen as the predictive model, which will be deployed online against SCAs based on accuracy, robustness, and computation latency.

III. Results Evaluation

In this part, all experiments are conducted on an Intel E5-2650 with 8 cores, 16GB DRAM. To demonstrate the effectiveness of ML-based detectors against SCAs on autonomous computing platforms, we build our ML-based detectors with microarchitectural events captured with Perf tool [24] and evaluate its detection accuracy, robustness, and computation efficiency.

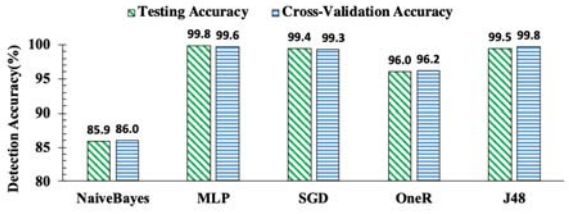

Fig. 3: Testing accuracy and cross-validation accuracy

A. Detection Results

1) Detection Accuracy: As shown in Figure 3, the percentage split testing and cross-validation accuracy of the five

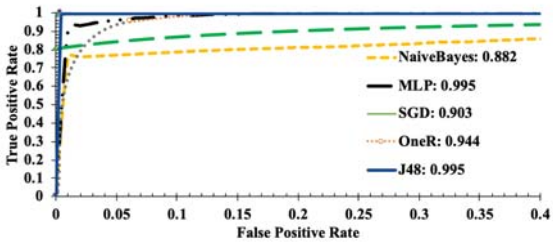

Fig. 4: ROC Curve and AUC value of various classifiers

implemented ML classifiers are presented. It is observed that testing accuracy and cross-validation accuracy have shown similar trends across all five classifiers. Moreover, NaiveBayes achieves lowest accuracy with less than 90% for both testing and cross-validation accuracy. The three classification classifiers, namely MLP, SGD, and J48, achieve high detection accuracy (>99%), while OneR has around 3% less accuracy compared to the three classifiers.

2) Robustness: Receiver Operating Characteristics (ROC) Curve is produced by plotting the fraction of true positives rate versus the fraction of false positives for a binary classifier. The best possible classifier would yield a point in the upper left corner or coordinate (0,1) of the ROC space, representing 0% false positives and 100% true positives. Area under the ROC Curve (AUC) metric. The AUC corresponds to the probability of correctly identifying ”under attack” and ”under no attack” and robustness is referred to how well the classifier distinguishes between the two classes for all possible threshold values. Higher AUC indicates better robustness for ML classifiers. Figure 4 depicts the ROC Curve of various ML classifiers with corresponding AUC values. It can be observed in Figure 4 that the NaiveBayes algorithm performs the worst in terms of ROC Curve, having the largest distance to the point (0,1). The J48 classifier's AUC value is closer to the coordinate (0,1), indicating a higher true positive rate and less false positive rate than the other four classifiers evaluated in this work. The ROC curve and AUC value of MLP are similar to those in J48, indicating that the mispredicted instances are evenly distributed between ”under attack” and ”under no attack” classes.

TABLE III: F-measure of various classifiers

Classifiers	NavieBayes	MLP	SGD	OneR	J48
F-measure	0.862	0.934	0.894	0.945	0.993

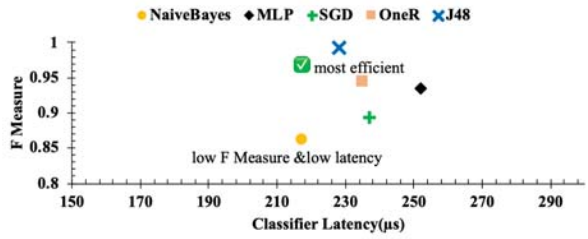

Fig. 5: Efficiency comparison among various classifiers

3) Efficiency: Lastly, to accordingly account for both performance rate and cost of ML classifiers, in Figure 5 we compare detection rate over a computational latency (F-measure/Latency) for various ML classifiers. F-measure is interpreted as a weighted average of the precision (p) and recall (r) which is formulated as $\frac{2\times(p\times r)}{p+r}$. F-measure is a more comprehensive evaluation metric over accuracy (percentage of correctly classified samples) since it takes both the precision and the recall into consideration. The F-measure of investigated classifiers are presented in Table III. We use F-measure over latency to identify the SCA detectors that require small cost can detect the program's maliciousness with high accuracy and performance. A higher ratio classifier is considered more efficient than a classifier with a lower ratio. As shown in Figure 5, a clear trade-off is seen between F measure and latency achievable for real-time hardware-assisted SCAs detection. The NaiveBayes show the smallest timing costs with the lowest SCAs F-measure. For highly resource-constrained embedded systems, techniques such as J48 provide low computational overhead (only 10 us higher than NaiveBayes), while achieving an F-measure of close to 0.993 on average.

IV. Conclusion

Connected and autonomous vehicles have emerged with the rise of recent technology in electronics and artificial intelligence. As we demonstrate in this work, the reliance of autonomous driving systems on computer systems to sense and operate in the physical world has introduced novel security challenges at the hardware level that need to be explored in a systematic way. In this work, we demonstrate the vulnerability of autonomous vehicles to emerging side-channel attacks and potential privacy leakage. It is further shown that the widely used processors in autonomous vehicles industry are equipped with hardware performance counters registers. As a result, we propose a highly accurate machine learning-based detector trained with microarchitectural features against emerging side-channel attacks. To this end, we explored a comprehensive range of ML classifiers to find the most effective model across various evaluation metrics. The results indicate that J48 classifier obtains 99.5% detection accuracy and 0.995 AUC with the highest efficiency compared to other tested ML classifiers.

Acknowledgment

This research was supported in part by NSF program under the award number 1936836.

References

[1] "Tesla autopilot." *https://www.tesla.com/ autopilot.*
[2] "Inside a self-driving uber." *https://www.infoq.com/presentations/uber-self-driving-software.*
[3] "Waymo's autonomous fleet has intel inside." *https://www.electronicdesign.com/automotive/waymo-s-autonomous-fleet-has-intel-inside.*
[4] "Baidu apollo," *https://www.electronicdesign.com/markets/automotive/article/21119589/xilinx-soc-fpga-powers-baidus-apollo-driverless-platform.*
[5] M. Luo, A. C. Myers, and G. E. Suh, "Stealthy tracking of autonomous vehicles with cache side channels," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020, pp. 859–876.
[6] D. A. Osvik, A. Shamir, and E. Tromer, "Cache attacks and countermeasures: the case of aes," in *Cryptographers' Track at the RSA Conference*. Springer, 2006, pp. 1–20.
[7] F. Liu, Y. Yarom, Q. Ge, G. Heiser, and R. B. Lee, "Last-level cache side-channel attacks are practical," in *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE, 2015, pp. 605–622.
[8] Y. Yarom and et.all, "Flush+ reload: A high resolution, low noise, l3 cache side-channel attack." in *USENIX Security Symposium*, 2014.
[9] D. Gruss, C. Maurice, K. Wagner, and S. Mangard, "Flush+ flush: a fast and stealthy cache attack," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2016, pp. 279–299.
[10] D. Page, "Partitioned cache architecture as a ėide-channel defence mechanism," 2005.
[11] F. Liu, Q. Ge, Y. Yarom, F. Mckeen, C. Rozas, G. Heiser, and R. B. Lee, "Catalyst: Defeating last-level cache side channel attacks in cloud computing," in *2016 IEEE international symposium on high performance computer architecture (HPCA)*. IEEE, 2016, pp. 406–418.
[12] Z. Wang and R. B. Lee, "New cache designs for thwarting software cache-based side channel attacks," in *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2. ACM, 2007, pp. 494–505.
[13] ——, "A novel cache architecture with enhanced performance and security," in *Proceedings of the 41st annual IEEE/ACM International Symposium on Microarchitecture*.
[14] F. Liu and R. B. Lee, "Random fill cache architecture," in *47th Annual IEEE/ACM International Symposium on Microarchitecture*, 2014.
[15] H. Wang, H. Sayadi, S. Rafatirad, A. Sasan, and H. Homayoun, "Scarf: Detecting side-channel attacks at real-time using low-level hardware features," in *IOLTS*. IEEE, 2020.
[16] H. Wang, H. Sayadi, T. Mohsenin, L. Zhao, A. Sasan, S. Rafatirad, and H. Homayoun, "Hybrid-shield: Accurate and efficient cross-layer countermeasure for run-time detection and mitigation of cache-based side-channel attacks." IEEE/ACM International Conference on Computer-Aided Design, 2020.
[17] H. Wang, H. Sayadi, G. Kolhe, A. Sasan, S. Rafatirad, and H. Homayoun, "Phased-guard: Multi-phase machine learning framework for detection and identification of zero-day microarchitectural side-channel attacks," in *2020 IEEE 38th International Conference on Computer Design (ICCD)*. IEEE, 2020, pp. 648–655.
[18] H. Wang, H. Sayadi, A. Sasan, S. Rafatirad, and H. Homayoun, "Hybridg: Hybrid dynamic time warping and gaussian distribution model for detecting emerging zero-day microarchitectural side-channel attacks," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2020, pp. 604–611.
[19] A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholtz, D. Hong, A. Wicks, T. Alberi, D. Anderson *et al.*, "Odin: Team victortango's entry in the darpa urban challenge," *Journal of field Robotics*, vol. 25, no. 8, pp. 467–492, 2008.
[20] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA urban challenge: autonomous vehicles in city traffic*. springer, 2009, vol. 56.
[21] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *arXiv preprint arXiv:1506.01497*, 2015.
[22] D. Fox, "Adapting the sample size in particle filters through kld-sampling," *The international Journal of robotics research*, vol. 22, no. 12, pp. 985–1003, 2003.
[23] "Path planning," in *https://github.com/ser94mor/path-planning*.
[24] "Perf," in *https://perf.wiki.kernel.org/index.php/Main Page*.