Stochastic-HMDs: Adversarial-Resilient Hardware Malware Detectors via Undervolting

Md Shohidul Islam
CSE Dept., DUET, Gazipur, Bangladesh
ECE Dept., George Mason University
Fairfax, VA, USA
mislam20@gmu.edu

Ihsen Alouani
CSIT, Queen's University Belfast, UK
IEMN CNRS-8520
INSA Hauts-de-France
Université Polytechnique Hauts-de-France
i.alouani@qub.ac.uk

Khaled N. Khasawneh
ECE Department
George Mason University
Fairfax, VA, USA
kkhasawn@gmu.edu

***Abstract*—Machine learning-based hardware malware detectors (HMDs) offer a potential game changing advantage in defending systems against malware. However, HMDs suffer from adversarial attacks, can be effectively reverse-engineered and subsequently be evaded, allowing malware to hide from detection. We address this issue by proposing novel HMDs (Stochastic-HMDs), which leverage approximate computing (AC) to harden HMDs against adversarial evasion attacks. Stochastic-HMDs introduce stochastic noise into the computations within the model to build an efficient and low-cost moving-target defense. Specifically, we use controlled undervolting, i.e., scaling the supply voltage below nominal level, to deliberately induce stochastic timing violations in the HMDs' computations during inference (detection). We show that such technique makes HMDs more resilient to adversarial attacks, especially to reverse-engineering and transferability. Our thorough empirical results substantiate that Stochastic-HMDs offer effective defense against adversarial attacks along with by-product power savings, without requiring any changes to the hardware/software nor to the HMDs' model, i.e., no retraining or fine tuning is needed. In particular, Stochastic-HMDs can detect more than 94% of the evasive malware with a negligible (i.e., $< 2\%$) accuracy loss, along with $\sim$15$\%$ power savings.**

***Index Terms*—HMD, Adversarial Attack, Undervolting**

I. Introduction

Although significant effort continues to be directed at making systems more resilient to malware attacks, the number of exploitable vulnerabilities is overwhelming. While preventing compromise is difficult, signature-based static analysis techniques can be easily bypassed using metamorphic/polymorphic malware or zero-day exploits. In contrast, dynamic detection techniques can detect unseen signatures since they monitor the behavior of the program. However, the complexity and difficulty of continuous dynamic monitoring have traditionally limited its use due to constrained resources.

Against this backdrop, several research studies proposed using Hardware Malware Detectors (HMDs) to make the continuous dynamic monitoring resource-efficient [13], [15], [18]. Specifically, these studies showed that malware can be classified as a computational anomaly using low-level hardware features. HMDs offer a significant advantage to defend against malware attacks because they can be 'always on' with small-to-no impact on performance. Moreover, it appears that the industry started to show interest in using HMDs too. In fact, Intel introduced Threat Detection Technology (TDT) [11], as part of their hardware shield suite, which uses hardware performance counters for malware detection. Later, TDT was used by Microsoft Defender for Endpoint to detect Ransomwares [31].

As HMDs showed defense effectiveness and are currently deployed in practice [11], [31], it is natural to expect that attackers attempt to find adaptive techniques to evade detection. As a consequence, it was shown that attackers can adapt malware to continue to operate while avoiding detection by HMDs [12], [14], [19]. Specifically, evasive malware (adversarially generated malware), are generated by carefully modifying the execution behaviour to force the model to output a wrong label (classify it as benign), allowing the evasive malware to evade detection while still being able to execute the intended malicious functionality.

Among the proliferation of defenses against adversarial attacks in the computer vision domain [3], randomization-based defenses are shown to be promising for improving model robustness against adversarial attacks [16], [24], [32]. The main idea is to introduce a non-deterministic component in the model's behavior. These non-deterministic variations of the model lead to: **(i)** a time-variant behavior that makes the reverse-engineering process more difficult, and **(ii)** a stochastic gradient over the input, which makes the estimation of the gradient direction challenging for the adversary. As a result, multiple randomization techniques have been proposed, such as randomly switching between multiple diverse models [32], [33] and introducing noise to the hidden-layers [24]. However, these proposed techniques require substantial changes to the model and/or retraining/fine-tuning procedures, which involve significant additional overhead, making it more challenging to build secure HMDs, especially for resource-constrained devices. Against the additional overhead, recent works [9], [13] demonstrated that hardware can introduce noise that achieves the robustness requirement while avoiding the additional overhead. However, these solutions require new hardware designs and thus cannot be deployed in existing devices. Furthermore, while randomization by switching between multiple diverse models was explored as a defense against evasive malware [13], [19], noise-based randomization was only explored in image classification applications [9], [10], [24].

In this paper, we propose Stochastic-HMDs, which unprecedentedly utilize undervolting to defend against adversarial attacks on HMDs. Particularly, we leverage controlled *undervolting*, i.e., supply voltage scaling below the nominal level, to induce stochastic noise in HMD's model computations during inference. From the security perspective, stochastic noise during inference injects a non-deterministic component in the HMDs' behavior. This corresponds to a practical implementation of a *moving-target defense* through inference-time stochastic decision boundaries. This aspect allows HMDs to resist reverse-engineering and makes evasion harder by changing the model behavior at runtime. From the power consumption perspective, controlled undervolting offers a by-product power saving, which is useful specifically for mobile, edge, and IoT devices. We show that a trade-off can be obtained between low accuracy loss and considerable gains in security and power consumption. Thus, our work proves that the security and energy efficiency can be improved at the same time, without adding performance overhead— contrasting the conventional

wisdom that tells us that if you want security, you have to sacrifice performance.

The key contributions of this paper are as follows:

- We study and characterize undervolting-induced noise properties on a real CPU. Our empirical results show that the computational faults induced by undervolting are stochastic, i.e., time-variant, and the undervolting level controls the noise magnitude.
- We propose Stochastic-HMD, which leverages undervolting to make HMDs resilient to malware evasion. Specifically, we show that computational faults via undervolting, i.e., hardware-induced noise, can be used as a defensive technique against evasive malware.
- Our results demonstrate that Stochastic-HMDs can detect more than 94% of the evasive malware with a negligible accuracy loss (i.e., $< 2\%$), along with around 15% power savings. These results substantiate that the security and energy efficiency can be improved at the same time, without performance loss.

II. Characterizing undervolting-induced noise

Setup. For our characterization experiments, we used an Intel Broadwell processor (model number i7-5557U) running an Ubuntu 18.04 LTS with stock Linux v4.15. We scale the voltage using the Model-Specific Register (MSR) `0x150` [28]. Specifically, we set the `plane idx` bits to `0` to scale the core's voltage exclusively, and used the `offset` bits for undervolting. In fact, these bits are used to scale the voltage by the specified offset in mV relatively to the nominal supply voltage. Moreover, the core frequency was kept at 2.2 GHz.

To better understand the nature of faults, we applied undervolting by reducing the voltage in small steps of $1\ mV$ while repeatedly executing the same instruction with the same operands until a fault or system freeze occured. Next, we will show the analysis of undervolting effect on multiplications as well as other instructions.

Undervolting effect. First, we analyse the undervolting impact on multiplications. Thus, we repeatedly run multiply operations on same operands several times for $100k$ sets of operands. We observed that undervolting by $-103\ mV$ to $-145\ mV$, depending on inputs, was sufficient to generate faults. We noticed that the sign bit never flipped. This observation is expected since the output sign bit is the result of a simple XOR between the two input sign bits, which should be far from being a critical path, and hence unlikely to be affected by timing violations. In addition, the 8 least significant bits never flipped, mainly because of their lower propagation delays. More importantly for this work, when using the same operands as inputs for an undervolted multiplication, the locations of faulty bits varied non-deterministically across different runs, demonstrating the stochastic aspect of the undervolting-induced noise. We validated this observation using the approximate entropy test. Please notice that these observations are consistent with [28], which tested on different CPU microarchitectures, as well as [30], which tested on FPGA devices.

To analyze the distribution of the undervolting-induced fault location, we generated faulty results using the above mentioned experiment and computed the bit-wise fault rates. This gives us an overview of the error location distribution within a specific undervolting level. Results are shown in Figure 1 (experiment configurations are listed in the caption).

In addition to undervolting multiplication operations, we tried undervolting addition, subtraction, and bit-wise operations, but no faults were observed. We believe that this is due to their simpler circuits and, by consequence, lower propagation delays compared to the multiplier implementation.

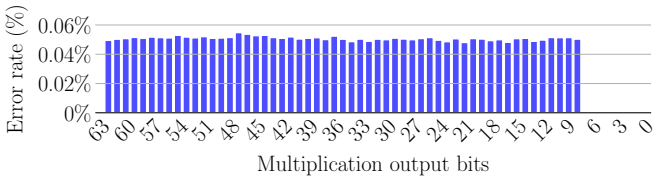

Fig. 1: Probability distribution of faulty bits location for undervolted multiplication results. These results are generated on i7-5557U at 2.2 GHz, with a CPU temperature of 49° C and the CPU was undervolted by $-130\ mV$.

III. Stochastic-HMDs

In this section, we propose a new class of adversarial evasion resilient HMDs, called Stochastic-HMDs. Stochastic-HMDs leverage undervolting to induce a non-deterministic aspect in the inference computations, making the HMD's decision boundaries stochastic over time. Thus, they prevent the adversary from having reliable access to the HMD's output (reverse-engineering attacks) and reduce the transferability of evasive malware built using the victim HMD's exact or proxy model. Specifically, Stochastic-HMDs exploit AC, which is a computing paradigm that trades energy consumption with the accuracy of results.

Several noise-based methods, such as [5], [24], have been used in the image classification domain to defend against adversarial attacks. However, these techniques add substantial performance and power overhead; they need to query a physical noise source to guarantee the random distribution of noise and require changes to the classification model, e.g, adding noise injection layers to the model structure. In this paper, we use a circuit level approximation, specifically undervolting [4] as a practical source of randomness to harden HMDs against adversarial attacks. Our goal is to intentionally cause controlled random timing violations driven by the supply voltage level [4]. The rationale behind choosing undervolting is as follows:

(i) Injects time-variant stochastic behavior: One of the interesting properties of undervolting is that the timing violations are stochastic (Section II). This offers a significant advantage to defend against adversarial attacks because it offers time-variant decision boundaries, i.e., a moving-target defense [19]. In contrast, other circuit level approximation techniques depend on reducing resource utilization and redesigning the circuit blocks accordingly [1], [9]. Therefore, their behavior is deterministic.

(ii) Easy to deploy: Undervolting is a practical source of randomness that can be easily deployed; doesn't require any changes to the computing stack, e.g., hardware/software, since it only depends on scaling the supply voltage.

(iii) Reduces energy consumption without costing performance overhead: Unlike related randomization based defenses that impose high performance and power overhead, undervolting comes with a by-product reduction in energy consumption due to the super-linear dependence of both dynamic and leakage power on supply voltage. Specifically, for related randomness-based defenses, generating randomness requires a source of entropy, e.g., a true random number generator (TRNG) [27]. As a result, they require n TRNG queries for each of the n MAC operations in a convolution layer, which comes with increasing costs for deeper models. Note that the TRNG is an off-core components, i.e., shared between all CPU cores, thus, require more time and power to query compared to querying on-core resources.

This work is the first to explore if there is an advantage from injecting noise in the malware detection domain, while also providing **a practical way** to inject stochastic noise. Furthermore, the proposed approach: **(1)** does not require additional training or fine-tuning of the protected model, unlike related work [17], [19], **(2)** does not require any changes to the pre-trained model, **(3)** does not require additional input preprocessing, and **(4)** provides energy gains.

Space exploration – The fault rate that a chip is likely to encounter is related to the undervolting level [28]. Therefore, a space exploration of the effect of different undervolting levels on the accuracy and robustness features of a Stochastic-HMD is needed to identify the optimal undeviating level to achieve the best accuracy and robustness tradeoff. In particular, we identify the undervolting level that would result in the minimal to no accuracy loss under no evasion attack, while maximizing the robustness to evasive malware. For this reason, we explore the impact of the undervolting from two perspectives: (i) Baseline accuracy, and (ii) The moving-target impact, i.e., the stochasticity of the decision boundaries that we leverage for the defense. We describe our space exploration methodology and select the best configuration for our experiments in Section VI. Subsequently, we evaluate the security and performance of the selected configurations in Sections VII and VIII, respectively.

Undervolting a CPU – Recent systems/computers have a multi-core CPU. This is not limited to laptops, desktops, and servers, but mobile and low-power devices as well. For example, the Apple Watch Series 7 have a dual-core CPU [2]. In addition, modern processors have several integrated voltage regulators (VRs), which can control the supply voltage of each core independently. Therefore, detection (running the malware detection model) can be offloaded to a specific core to improve the use experience. Specifically, monitored applications can continue running (without interruption) since detection is offloaded to another core.

Trusted control – Trusted control of voltage is an important component of the proposed defense (otherwise the defense can be simply disabled by the adversary). Prior works implement HMDs either as a co-processor [19] or to run in a trusted execution environment [34]. For a co-processor implementation, the processor supports several integrated voltage regulators (VRs), thus, we can simply dedicate the control of one of the VRs to the Stochastic-HMD IP. For a trusted execution environment, the operating system can grant exclusive control of CPU VR to the Stochastic-HMD enclave.

IV. Dataset & feature collection

Database. The dataset consists of 3000 malware and 600 benign programs. The malware programs were downloaded from the Zoo malware database (a.k.a MalwareDB) and include five types of malware, which are *backdoors, rogues, password stealers, trojans, and worms*. The benign programs consisted of a wide variety of applications, including browsers, text editing tools, system programs, and CPU performance benchmarks. The dataset was divided evenly into 3-folds, which are *victim training*, *attacker training*, and *testing*. We use 3-fold cross-validation in our experiments to get accurate results, i.e., eliminate bias. Furthermore, the malware types and the benign application types were distributed evenly and randomly across the folds to ensure that the datasets are not biased.

Features Collection. To collect hardware features dynamically, we used Intel's Pin tool [26], running on an isolated machine that is running Windows 7. All security and firewall services were disabled so malware runs freely. The extracted features are based on the frequency of executed instruction categories; based on Intel's sub-grouping of instructions, e.g., binary arithmetic, control transfer, and system instructions sub-groups. The trace collection and features extraction are modeled after the RHMD study [19] to establish the feasibility and provide trustworthy experimental results.

Recently, it has been shown that hardware features collected through hardware performance counters (HPCs) are not reliable to be used in security application due to their non-determinism [6]. In this work, we do **not** use HPCs, and we make sure that our feature collection framework is deterministic; we get the exact same trace in every run when we supply the same input. We manually verified this by running samples of programs multiple times, using different machines and using a virtual machine.

V. Threat Model

We assume a realistic common setting which is a black-box adversarial attack scenario. This setting is consistent with related work on HMDs [13], [19]–[22]. In addition, the adversary has no access to specific thermal or process variability, or internal functions of the model. We follow the same attack methodology described in [13], [19], which starts by reverse-engineering the victim HMD to generate a proxy model and then uses the proxy model to generate evasive malware examples that can bypass HMD detection.

VI. Stochastic-HMDs space exploration

We are interested in two criteria upon which we explore the optimal undervolting level that maximizes the decision boundaries stochasticity, *under the constraint of minimal accuracy loss*. Thus, we will explore the effect of different undervolting levels on the Stochastic-HMD accuracy and robustness features, i.e., the stochasticity of decision boundaries.

A. Experimental setup

For our experimental setup, we apply undervolting to the baseline HMD to construct a Stochastic-HMD. In particular, to model the computational stochasticity caused by the undervolting, we built a stochastic fault injection tool[1] that emulates timing violations at the output of arithmetic operations, based on the error distribution model detailed earlier in Section II. Practically, the tool injects timing violation errors that follow the distribution that matches the undervolting level. Note that we did not implement any measures to detect or correct potential timing violation-induced computational faults since they are the foundation of our proposed defense. Finally, we integrated our tool to the Fast Artificial Neural Network Library (FANN) [29] to simulate the behavior of our neural network model under undervolting (the Stochastic-HMD).

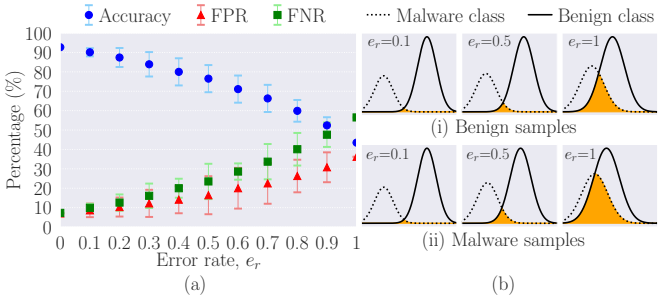

Fig. 2: Space exploration: (a) Accuracy, False Positive Rate (FPR), and False Negative Rate (FNR); (b) Confidence distribution of malware class and benign class for (i) benign samples and (ii) malware samples with a different error rate, e_r.

[1] https://github.com/CAMLsec/Stochastic-HMD

B. Detection accuracy

We study the effect of undervolting on HMDs detection accuracy, *i.e.*, detection performance of Stochastic-HMDs under no attacks. We used the *testing* set for this experiment. We performed 3-folds cross-validation, repeated each experiment 50 times, and computed the mean and standard deviation to obtain representative results. Figure 2(a) shows the Stochastic-HMD detection accuracy, false positive rate, and false negative rate while increasing the error rate (scaling the voltage). An interesting observation is that the standard deviation increases while increasing the undervolting until a 0.5 error rate, and then it starts decreasing. Notice that the standard deviation represents the stochasticity that undervolting adds to the output due to the non-deterministic decision boundaries. More interestingly, the accuracy degradation diverges logarithmically as the error rate approaches 1; the relationship is not linear. The same observation also applies to the false positive rate and false negative rate. This is a strong advantage from the defender's perspective since adding more errors (specifically, until 0.5 error rate) would not significantly impact detection accuracy loss. For example, at 10% error rate, the detection accuracy of Stochastic-HMDs drops by around 2% only. Furthermore, increasing the error rate from 0.1 to 0.4 would result in 0.1× detection accuracy loss. A discussion of the potential trade-offs is detailed in Section IX.

C. Stochasticity of decision boundaries

We explore the impact of undervolting on the HMD confidence distribution. We infer the model for different error rates comparatively with the baseline HMD and we present the distribution of the output scores. Figure 2(b) shows the confidence distribution for the two classes: Benign and Malware. As depicted in the figure, the higher the error rate, the higher the uncertainty is observed through a higher stand deviation of the scores distribution. While these results are correlated with the results in Figure 2(a), we can see that an uncertainty level could be injected while the accuracy remains almost equal to the baseline.

VII. Security Evaluation

We explore whether Stochastic-HMDs can increase the robustness of HMDs against evasive malware, *i.e.*, both steps of the attack—reverse-engineering and transferability. Our results show that undervolting helps making HMDs resistant to evasive malware. In particular, undervolting introduces noise to the reverse-engineering component that is bounded by a function of the amount of introduced noise, *i.e*, undervolting level.

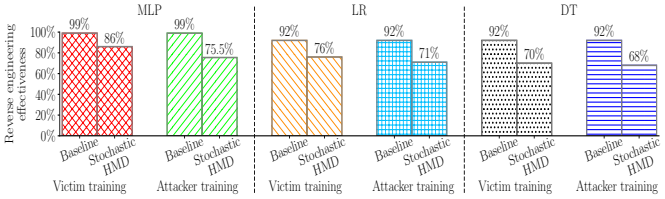

Fig. 3: Reverse-engineering effectiveness.

A. Resilience against reverse engineering

We assume two attack scenarios: (1) the attacker knows the Stochastic-HMD (victim) training data; we use the *victim training* set to perform the reverse-engineering, and (2) the attacker does not know the Stochastic-HMD (victim) training data; we use the *attacker training* set to perform the reverse-engineering. Note that the first scenario assumes a stronger attacker than the second scenario since the attacker will use the same data distribution that the victim is trained on. In addition, in both scenarios, we use the *testing* set to evaluate the proxy model performance (*i.e.*, the reverse-engineering effectiveness). Moreover, we perform reverse engineering using Multi-Layer Perceptron (MLP) neural network, Logistic Regression (LR), and Decision Tree (DT). We selected MLP for its state-of-the-art performance, LR for its simplicity, and DT for its non-differentiability.

Figure 3 shows the reverse-engineering effectiveness of Stochastic-HMDs using the two attack scenarios while setting the error rate of Stochastic-HMD to 0.1. The results show that using a Stochastic-HMD makes the reverse-engineering substantially more difficult; the reverse-engineering effectiveness using MLP drops from 99% to 75.5% (around 24% drop) when using the *attacker training* set and from 99% to 86.0% (around 13.3% drop) when using the *victim training* set. Furthermore, we noticed that the Stochastic-HMDs resilience to reverse-engineering increases by increasing the error rate, irrespective of the machine learning algorithm used to perform the attack. As seen from the results, reverse-engineering attacks become harder with undervolting.

B. Resilience against evasive malware (Transferability)

Having a reverse-engineered model of the victim HMD, transferability is defined by the percentage of evasive malware designed to evade the reverse-engineered model that can also evade the victim HMD's detection, *i.e.*, transfer to the defender model. Therefore, in this experiment, we generated evasive malware samples based on each of the reverse-engineered models of the Stochastic-HMD with error rate 0.1 (shown in Figure 3), and tested their success rate in evading the Stochastic-HMD. Figure 4 shows the transferability results when attacking Stochastic-HMDs. The results show that Stochastic-HMDs are robust against transferability attacks regardless of the machine learning algorthim used for the reverse-engineering. In addition to detection accuracy and transferability robustness, a comprehensive trade-off investigation should also include the robustness against reverse-engineering, which we discuss in Section IX.

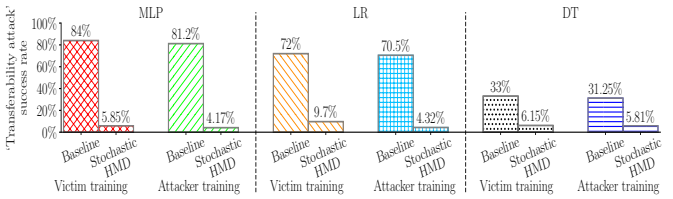

Fig. 4: Evasive malware success rate.

C. Stochastic-HMDs vs RHMDs resilience

After performing the security analysis of Stochastic-HMDs, we compare Stochastic-HMDs resilience to state-of-the-art randomization based solution for HMDs, RHMDs [19]. Basically, RHMDs is a randomization-based defense that switches between multiple diverse HMDs, *i.e.*, decision boundaries, at run-time to perform malware detection. The RHMDs' resilience increases while increasing the number of base-HMDs (decision boundaries). Therefore, similar to [19], we constructed four RHMDs such as randomizing two features vectors (RHMD-2F), three features vectors (RHMD-3F), two features vectors with two detection periods (RHMD-2F2P), and three features vectors with two detection periods (RHMD-3F2P). We reverse-engineer each RHMD construction using all the feature vectors used in the construction to get the most accurate proxy model and most efficient reverse-engineering. Furthermore, based on the obtained proxy models for each of the RHMD constructions, we use our

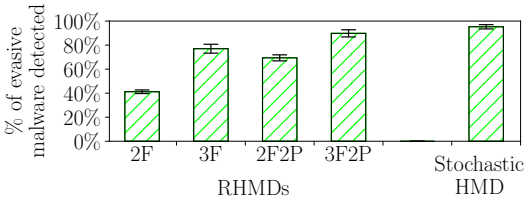

Fig. 5: Percentage of evasive malware detected

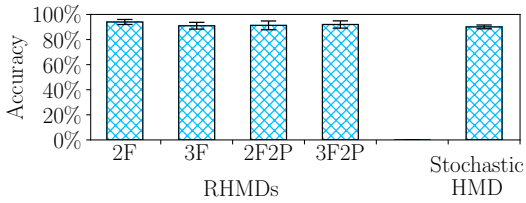

Fig. 6: Accuracy of RHMDs and Stochastic-HMDs.

evasion framework to inject instructions to evade it. The accuracy results of detecting evasive malware with all RHMD constructions as well as the most resilient Stochastic-HMD (with 10% error rate) are shown in Figure 5. The results show that Stochastic-HMDs can detect more than 53% of the evasive malware missed by the most resilient RHMD construction (RHMD-3F2P).

VIII. Performance Evaluation of Stochastic-HMDs

In this section, we compare Stochastic-HMDs detection accuracy and implementation overhead to state-of-the-art randomization based solution for HMDs, RHMDs [19].

We used the *testing* set to evaluate the detection accuracy and implementation overhead of RHMDs and Stochastic-HMDs.

Detection accuracy: We used the same RHMDs constructions described in Section VII-C and the most resilient Stochastic-HMD (with 10% error rate). Note that here we are comparing the baseline accuracy, *i.e.*, percentage of correctly classified benign and non-evasive/non-adaptive malware programs. The detection accuracy results are shown in Figure 6. The results show that Stochastic-HMDs have comparable results (less than 2% accuracy loss) to the most resilient RHMD (RHMD-3F2P). Note that the accuracy drop is negligible and acceptable, which comes from the fact that Stochastic-HMDs uses only one detector (rather than multiple detectors in RHMDs). We interpret the minor accuracy loss as the cost of improved security (*cf.* Figure 5).

Memory space: RHMDs store multiple HMDs models, *i.e.*, base-detectors, to be able to switch between them. In contrast, only one model needs to be stored in Stochastic-HMD. Therefore, storage savings of Stochastic-HMDs over RHMDs can be measured using the following equation:

$$Storage\ savings = \frac{\#\ of\ base\ detectors\ in\ RHMD-1}{\#\ of\ base\ detectors\ in\ RHMD} \quad (1)$$

For example, Stochastic-HMD storage saving over an RHMD-2F, *i.e.*, smallest RHMD implementation in terms of size in memory, is 50%. This saving not only reduces the storage space in memory, but also the pressure on the CPU resources, such as the caches and the bus; thus, reducing the overhead on the system overall performance. In our experiments, every HMD takes 71 KB of memory, while the L1 cache size in Intel's Tiger Lake CPU is 32 KB.

Inference time: We ran 100k detection for Stochastic-HMD , RHMD-2F (with 2 base-detectors), and RHMD-2F2P (with 4 base-detectors) on an Intel i7-5557U CPU. The average inference time is $7\mu s$, $7.7\mu s$, and $7.8\mu s$ for Stochastic-HMD , RHMD-2F, and RHMD-2F2P, respectively. The results show an average of at least 10% performance overhead of the simplest RHMD (RHMD-2F) over Stochastic-HMD. The inference time of RHMDs is higher owing to its additional task of randomly selecting a model from its set of base models; such random model selection also has impact on L1 cache eviction. We noticed that scaling the voltage has no effect on the inference time because undervolting has no effect on the cycle time since we are only scaling the CPU voltage but not frequency.

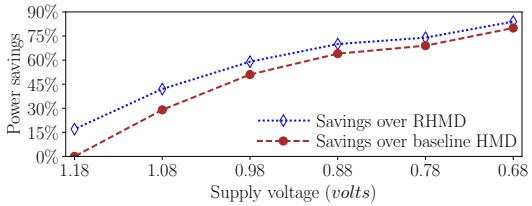

Fig. 7: Power savings of Stochastic-HMD.

Power consumption: We ran 100k detection for both Stochastic-HMD and RHMD-2F (with 2 base-detectors) on an Intel i7-5557U CPU. We use Intel's Power Gadget [23] to measure the power consumption and report the average consumed power per inference. Figure 7 shows the power savings for voltages ranging from $1.18\ V$ (the nominal voltage) to $0.68\ V$ with a voltage step size of $0.1V$. Results show over 75% in power saving compared to RHMD achieved by Stochastic-HMD under 40% voltage scaling.

Comparison with TRNG: We evaluate the overhead of modifying non-secure baseline HMD to inject noise after each MAC operation based on queering a TRNG. Our results shows that the TRNG based implementation adds $\approx 62\times$ performance and $\approx 112\times$ energy consumption overheads. We repeated the experiment while using a pseudo random number generator (PRNG) [25] instead of a TRNG. Our result shows that the PRNG based implementation adds $\approx 4\times$ performance and $\approx 5.7\times$ energy consumption overheads.

IX. Discussion

This work has unprecedentedly shown that a promising robustness enhancement technique of HMDs can be implemented in a practical and efficient manner using undervolting.

Tradeoff: For a practical deployment, the security aspects need to be considered under the baseline accuracy constraint, *i.e.*, the accuracy without evasive malware. An overview of the security and accuracy trends and the possible tradeoff is shown in Figure 8. Specifically, Figure 8 shows the detection accuracy and the transferability robustness, *i.e.*, percentage of evasive malware created using the proxy model that fails to evade victim HMD's detection. From the figure, it is clear that a fault rate that exceeds 0.2 (area (2)) would no longer be practical. However, as shown in area (1), trade-offs between security and performance could be reached with low error rates, which can considerably increase the defender HMD reverse-engineering robustness, transferability robustness (achieves more than 94%, even with effectively reverse-engineered victim HMD), with negligible accuracy loss, along with around 15% power saving.

Calibration: Undervolting-induced faults vary across devices. Therefore, a separate calibration needs to be done for each device to determine the undervolting level that leads to the best accuracy/robustness tradeoff. Furthermore, the temperature needs to be considered, since it affects the faults [8]. Therefore, the voltage regulator that controls the Stochastic-HMD needs to dynamically adjust the undervolting level based on the current temperature to achieve the best accuracy/robustness tradeoff.

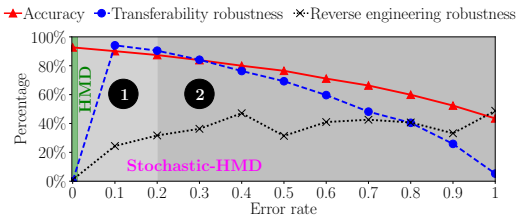

Fig. 8: Stochastic-HMDs trade-off

Implication of undervolting on the rest of the system: Undervolting may cause faults in the rest of the system making the whole system unstable/unreliable. To avoid this issue, the undervolting should be applied only when executing the HMDs detection component (*i.e.*, detection model inference). If the detection component is running in a trusted execution environment (TEE), the voltage needs to be undervolted directly after entering the TEE and scaled back to the nominal voltage just before exiting the TEE. Note that voltage scaling can be controlled from software through the MSRs [7]. Another approach is to dedicate one core for the detection component only in a multi-core system and implement the detection component on bare-metal to guarantee that there are no system components running on the undervolted core.

Limitations: Since least significant bits cannot flip, models that operate on numbers that are very close to zero are not protected.

X. Conclusion

We propose Stochastic-HMDs that leverage undervolting to secure HMDs. We show that such detectors prevent the adversary from having reliable access to their model (*i.e.*, resist reverse-engineering) and reduce the success rate of evasive malware attacks (over 94% detection). In addition, Stochastic-HMDs resilience to the attacks can be achieved with low accuracy cost (*i.e.*, $< 2\%$ accuracy loss). Compared to related approaches, we offer additional benefits such as power savings ($\sim 15\%$) and easier deployment.

Acknowledgment

The work in this paper is partially supported by National Science Foundation grants CCF-2212427 and CNS-2155002, and by EdgeAI KDT-JU European project (101097300).

References

[1] I. Alouani, H. Ahangari, O. Ozturk, and S. Niar, "A novel heterogeneous approximate multiplier for low power and high performance," *in Embedded Systems Letters*.

[2] Apple, "Apple watch series 7 - technical specifications," https://support.apple.com/kb/SP860?locale=en_US, 2021.

[3] F. Behnia, A. Mirzaeian, M. Sabokrou, S. Manoj, T. Mohsenin, K. N. Khasawneh, L. Zhao, H. Homayoun, and A. Sasan, "Code-bridged classifier (cbc): A low or negative overhead defense for making a cnn classifier robust against adversarial attacks," in *ISQED*, 2020.

[4] V. K. Chippa, D. Mohapatra, K. Roy, S. T. Chakradhar, and A. Raghunathan, "Scalable effort hardware design," *Trans. on VLSI Systems*.

[5] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *ICML*, 2019.

[6] S. Das, J. Werner, M. Antonakakis, M. Polychronakis, and F. Monrose, "Sok: The challenges, pitfalls, and perils of using hardware performance counters for security," in *S&P*, 2019.

[7] M. Elerščič, "Guide to linux undervolting for haswell and never intel cpus," 2019, at https://github.com/mihic/linux-intel-undervolt.

[8] I. Filanovsky and A. Allam, "Mutual compensation of mobility and threshold voltage temperature effects with applications in cmos circuits," *IEEE Transactions on Circuits and Systems I*, 2001.

[9] A. Guesmi, I. Alouani, K. N. Khasawneh, M. Baklouti, T. Frikha, M. Abid, and N. Abu-Ghazaleh, "Defensive approximation: Securing cnns using approximate computing," in *ASPLOS*, 2021.

[10] ——, "Defending with errors: Approximate computing for robustness of deep neural networks," *arXiv preprint arXiv:2211.01182*, 2022.

[11] "Detect ransomware and other advanced threats with intel threat detection technology," 2022, last Accessed February 1st, 2022 from https://www.intel.com/content/dam/www/central-libraries/us/en/documents/intel-threat-detection-technology-solution-brief.pdf.

[12] M. S. Islam, I. Alouani, and K. N. Khasawneh, "Enhancing hardware malware detectors' security through voltage over-scaling," in *2021 5th ACM SIGARCH Workshop on Cognitive Architectures*.

[13] 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 Transactions on Computers*, 2021.

[14] M. S. Islam, A. P. Kuruvila, K. Basu, and K. N. Khasawneh, "Nd-hmds: Non-differentiable hardware malware detectors against evasive transient execution attacks," in *ICCD*, 2020.

[15] M. S. Islam, B. Omidi, and K. N. Khasawneh, "Monotonic-hmds: Exploiting monotonic features to defend against evasive malware," in *ISQED*, 2021.

[16] S. Islam, I. Alouani, and K. N. Khasawneh, "Lower voltage for higher security: Using voltage overscaling to secure deep neural networks," in *ICCAD*, 2021.

[17] J. Jia and N. Z. Gong, "Attriguard: A practical defense against attribute inference attacks via adversarial machine learning," in {*USENIX*}, 2018.

[18] M. Kazdagli, L. Huang, V. Reddi, and M. Tiwari, "EMMA: A new platform to evaluate hardware-based mobile malware analyses," 2016.

[19] K. N. Khasawneh, N. Abu-Ghazaleh, D. Ponomarev, and L. Yu, "Rhmd: evasion-resilient hardware malware detectors," in *MICRO*, 2017.

[20] K. N. Khasawneh, N. B. Abu-Ghazaleh, D. Ponomarev, and L. Yu, "Adversarial evasion-resilient hardware malware detectors," in *ICCAD*, 2018.

[21] K. N. Khasawneh, M. Ozsoy, C. Donovick, N. Abu-Ghazaleh, and D. Ponomarev, "Ensemble learning for low-level hardware-supported malware detection," in *RAID*, 2015.

[22] K. N. Khasawneh, M. Ozsoy, C. Donovick, N. A. Ghazaleh, and D. V. Ponomarev, "Ensemblehmd: Accurate hardware malware detectors with specialized ensemble classifiers," *IEEE TDSC*, 2018.

[23] S.-W. Kim, J. J.-S. Lee, V. Dugar, and J. De Vega, "Intel® power gadget," *Intel Corporation*, vol. 7, 2014.

[24] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, "Certified robustness to adversarial examples with differential privacy," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 656–672.

[25] P. A. W. Lewis, A. S. Goodman, and J. M. Miller, "A pseudo-random number generator for the system/360," *IBM Systems Journal*, 1969.

[26] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "Pin: building customized program analysis tools with dynamic instrumentation," in *Acm sigplan notices*, vol. 40, no. 6. ACM, 2005, pp. 190–200.

[27] J. P. Mechalas, "Intel digital random number generator (drng) software implementation guide," https://software.intel.com/content/www/us/en/develop/articles/intel-digital-random-number-generator-drng-software-implementation-guide.html, 2014.

[28] K. Murdock, D. Oswald, F. D. Garcia, J. Van Bulck, D. Gruss, and F. Piessens, "Plundervolt: Software-based fault injection attacks against intel sgx," in *S&P*, 2020.

[29] S. Nissen, "Fast artificial neural network library (fann)," 2013, available online (last access November 2019) at http://leenissen.dk/fann/wp/.

[30] B. Salami, E. B. Onural, I. E. Yuksel, F. Koc, O. Ergin, A. C. Kestelman, O. Unsal, H. Sarbazi-Azad, and O. Mutlu, "An experimental study of reduced-voltage operation in modern fpgas for neural network acceleration," in *DSN*, 2020.

[31] K. Selvaraj, "Defending against cryptojacking with microsoft defender for endpoint and intel tdt," 2021, accessed online July 2021 at https://www.microsoft.com/security/blog/2021/04/26/defending-against-cryptojacking-with-microsoft-defender-for-endpoint-and-intel-tdt/.

[32] S. Sengupta, T. Chakraborti, and S. Kambhampati, "Mtdeep: Moving target defense to boost the security of deep neural nets against adversarial attacks," *Proc. GameSec*, 2019.

[33] X. Wang, S. Wang, P.-Y. Chen, Y. Wang, B. Kulis, X. Lin, and P. Chin, "Protecting neural networks with hierarchical random switching: Towards better robustness-accuracy trade-off for stochastic defenses," *arXiv preprint arXiv:1908.07116*, 2019.

[34] B. Zhou, A. Gupta, R. Jahanshahi, M. Egele, and A. Joshi, "Hardware performance counters can detect malware: Myth or fact?" in *AsiaCCS*, 2018.