Blacklist Core: Machine-Learning Based Dynamic Operating-Performance-Point Blacklisting for Mitigating Power-Management Security Attacks

Sheng Zhang, Adrian Tang, Zhewei Jiang, Simha Sethumadhavan, Mingoo Seok
Columbia University

ABSTRACT

Most modern computing devices make available fine-grained control of operating frequency and voltage for power management. These interfaces, as demonstrated by recent attacks, open up a new class of software fault injection attacks that compromise security on commodity devices. CLKSCREW, a recently-published attack that stretches the frequency of devices beyond their operational limits to induce faults, is one such attack. Statically and permanently limiting frequency and voltage modulation space, i.e., guard-banding, could mitigate such attacks but it incurs large performance degradation and long testing time. Instead, in this paper, we propose a run-time technique which dynamically blacklists unsafe operating performance points using a neural-net model. The model is first trained offline in the design time and then subsequently adjusted at run-time by inspecting a selected set of features such as power management control registers, timing-error signals, and core temperature. We designed the algorithm and hardware, titled a BlackList (BL) core, which is capable of detecting and mitigating such power management-based security attack at high accuracy. The BL core incurs a reasonably small amount of overhead in power, delay, and area.

CCS CONCEPTS

Security and privacy → Security in hardware → Hardware attacks and countermeasures → Side-channel analysis and countermeasures

KEYWORDS

Security, power management, operating performance point, blacklist

1. INTRODUCTION

Power management in modern commodity devices has become ever more important, especially in mobile and embedded devices, due to considerations like battery life and portability. The most widely adopted energy management solution is dynamic voltage and frequency scaling (DVFS) in which supply voltage (V_{DD}) and clock frequency (F_{CLK}) of a processor core are modulated based on computational demand. DVFS has fine cooperative granularities between software drivers and hardware voltage and frequency regulators.

The design of DVFS systems requires a cross-stack effort, but the security concern has been often overlooked. A new class of exploitation vector termed CLKSCREW is uncovered recently exploiting the security-oblivious power management design [1]. Under most widely deployed DVFS devices, processors can be pushed beyond a normal operation limit to produce faulty computation via software access to power management. These faults can be induced from lower privileged software across security boundaries to manipulate sensitive computations. Attacks like CLKSCREW pose a much more serious security threat than the traditional physical hardware-based fault-injection attacks as they can be conducted using no more than a malicious software kernel driver. CLKSCREW has been demonstrated to extract a secret AES (Advanced Encryption Standard) key embedded within ARM Trustzone [2] and load a self-signed code into Trustzone, through low privilege software.

It is theoretically possible to mitigate those attacks by permanently and statically limiting the frequency and voltage modulation range in the DVFS system, i.e., guard-banding. But such static measures can incur large performance degradation due to the variability of modern digital processors. Chip-wise post-fabrication testing may enable us to set a custom limit for each chip, alleviating the performance degradation as compared to the use of a single limit for all chips, but it increases testing time and quickly becomes economically infeasible.

In this paper, to avoid those limitations of static methods, we propose a *dynamic* countermeasure that can *blacklist* fault-inducing operating performance points (OPP) during run-time. The main idea is to detect and mitigate the power management based attack like CLKSCREW using an embedded machine learning core. The core is initially trained in the design time based on the informative features of CLKSCREW, such as voltage and frequency regulation characteristics and temperature. Then, during runtime, the core inspects those along with timing error flags to further update/adjust the blacklist model. Once the core discovers attacks, it can ignore frequency and voltage regulation commands that trigger the attack discovery. It can also notify the operating system (OS) to profile the source processes of the attacks.

Fig. 1 shows a quad-core processor architecture based on the Qualcomm Snapdragon 805 system-on-chip (SoC) in the Nexus 6 smartphone, which we used in the experiments throughout this paper. The original SoC has four per-core phase-locked-loops (PLL) for clock frequency modulation, one unified low-dropout regulator (LDO) for voltage regulation, and a dynamic thermal management (DTM) subsystem for managing temperature-related events. As shown in Fig. 1, we have added the countermeasure hardware comprising (i) a machine-learning core capable of identifying CLKSCREW attack, titled a blacklist core (BL core) and (ii) timing error detection/prediction hardware such as tunable replica circuits (TRC) [14-16] and *in-situ* error detection and correction circuits (EDAC) [3,11-13] for acquiring timing-error flags. The state-of-the-art EDAC techniques incur small silicon area overhead of <5% of a core area. They can also support various microarchitectures and circuits [3,11-13]. TRC can incur less silicon area overhead and design complexity than EDAC at the cost of more guardband to be added.

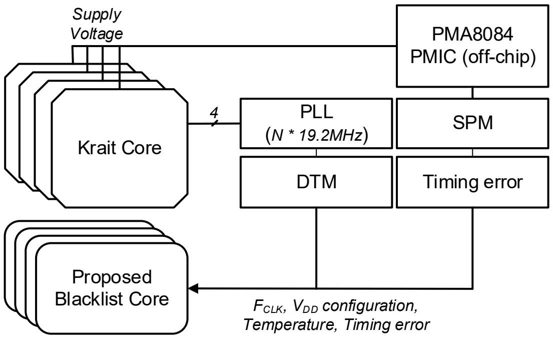

Fig. 1. Processor architecture with the proposed countermeasure

The primary measure for the proposed detection and mitigation of CLKSCREW is the BL core. The core performs feature extraction from various microarchitecture and circuit parameters. It then performs multilayer perceptron (MLP) based feature dimensionality reduction with which the core categorizes the OPP curve as a compressed feature, i.e., project an ensemble of features into a low-dimensional space, using weights trained in design time. During runtime, the core dynamically adjusts the decision boundary between safe and unsafe operating conditions in the low-dimensional feature space by the timing error detection and prediction. This makes the detection and mitigation to be

ISLPED '18, July 23–25, 2018, Seattle, WA, USA © 2018 Association for Computing Machinery. ACM ISBN 978-1-4503-5704-3/18/07 $15.00 https://doi.org/10.1145/3218603.3218624

customized to each specific core and its operating environment during the runtime.

We designed the aforementioned machine-learning algorithm in the fixed-point arithmetic. We then designed the microarchitecture to execute the algorithm with minimal area, delay, and power. Our proposed countermeasure demonstrates 99.69% accuracy in mitigating CLKSCREW, where undiscovered attacks are corrected by timing error prediction/detection circuits and feedback to the decision boundary to further improve the mitigation success rate. The BL core takes 1.11 μs to identify a potential attack, which can be masked within the latency of PLL lock time. It takes the silicon footprint of 0.09 mm^2 in a 65 nm CMOS process and consumes the power of 10.5 mW at V_{DD}=1V.

The remainder of the paper is organized as follows. In Sec. 2, we will discuss the operating mechanism of the CLKSCREW attack. In Sec. 3, we will discuss the limitations of static protection. In Sec. 4, we will detail the mitigation strategy for the proposed dynamic blacklisting. In Sec. 5, we will describe the microarchitecture of the proposed BL core along with the experimental results. The paper will conclude in Sec. 6.

2. CLKSCREW ATTACK

2.1. Background

DVFS regulates V_{DD} and F_{CLK} according to runtime task demands. To track task demands and adjust OPP at acceptable latency and granularity, DVFS requires OS-level power management services and vendor-specific regulator drivers as well as the hardware regulators. For instance, in our experiment in this paper, we used the Krait cores in Qualcomm's SnapDragon 805 SoC on a Nexus 6 device, where frequency and voltage regulators are software exposed to Krait cores.

The voltage regulator is integrated in a separate power management integrated circuit (PMIC) chip, which is not exposed to software interfaces directly. As shown in Fig. 1, the core voltage is indirectly managed by the subsystem power manager (SPM), a hardware block in the Qualcomm SnapDragon 805 SoC that maintains a set of control registers that interface with the PMIC chip. SPM is accessible to privileged software like a kernel driver through these memory-mapped control registers. Likewise, frequency regulators also expose the multiplier and selector control of the PLLs in these registers to software.

Software support for DVFS comprises vendor-specific regulator drivers and OS-level power management services. The drivers provide a convenient means for mechanisms in the upper layers of the stack, such as the Linux CPUfreq power governor [4] to dynamically direct the voltage and frequency scaling. DVFS requires real-time feedback on the system workload profile to guide the optimization of performance with respect to power dissipation. This feedback may rely on layer-specific information that may only be efficiently accessible from certain system layers. For example, instantaneous system utilization levels are readily available to the OS kernel layer. As such, the Linux CPUfreq power governor is well-positioned at that layer to initiate runtime changes to the operating voltage and frequency based on these whole-system measures.

2.2. Fault Injection Mechanism

For a successful CLKSCREW attack, several hardware conditions have to be met. First, hardware regulators have no safeguard in place to prevent unsafe OPP configurations. This is observed in several lines of consumer mobile devices [1].

Second, an unsafe operation needs be contained within a core to attack (called *victim core* hereafter), separate from the code that performs the attack. This stipulates that multi-core processors would operate in different frequency, and attack and victim code can be pinned to different cores from software. This core pinning strategy is possible due to the deployment of increasingly heterogeneous processors like the ARM big.LITTLE [5], and emerging technologies such as Intel PCPS [6] and Qualcomm aSMP [7]. With core pinning, the attack code can manipulate the frequency of the victim core without self-faulting. In addition, interrupts need to be disabled during the duration of victim code execution to ensure that no context switch occurs for that core.

Third, hardware regulators can operate across security boundaries. Two leading industry security-oriented technologies, ARM Trustzone and Intel SGX [8] can execute both trusted and untrusted code on the same physical core while relying on architectural features such as specialized instructions to support isolated execution. On such architectures, the voltage and frequency regulators typically operate on domains that apply to the entire core. Per-core voltage domain has been experimented yet limited to the use of linear regulators and thereby demonstrated for the applications having a limited range of dynamic voltage scaling. To enable a wide range of dynamic voltage scaling, a *shared* DCDC converter is typically employed as in the 805 SoC.

With the aforementioned prerequisites satisfied, we now detail the steps of CLKSCREW in Fig. 2. In this particular attack, we consider two cores: victim, and attack cores which respectively execute victim and attack threads. The optional third core is a measure core, which observes various parameters for our experiment purpose. The first task is to clear microarchitectural residual states from prior executions since cache-based profiling techniques will be used in later steps. To do so, both the victim and attack threads are run multiple times in quick succession.

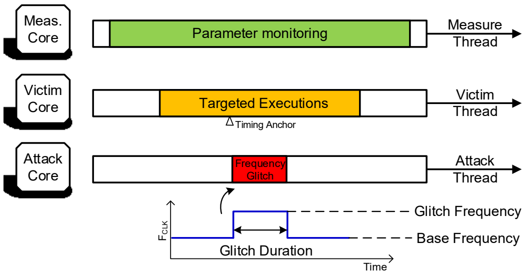

Fig. 2. CLKSCREW fault injection timing

Next, victim thread is profiled to identify a consistent point of execution just before the target code to be faulted. The profiling step yields a timing anchor to guide when to deliver the fault injection. In some attack scenarios, fine-tuning the exact delivery timing of the fault is required. In such cases, attack thread is set to spin-loop with a predetermined number of loops before inducing the actual fault. The use of these loops consisting of no-op operations can induce timing delays with high precision.

Finally, the attack thread raises the clock frequency of the victim core from the base to the glitch frequency level and keeps that frequency for a precise number of loops (defined as glitch duration), and then restore the frequency to the base level. After this, a typical fault-analysis attack algorithm is applied to the fault injection result. In some of the experiments, the measure core performs various administrative tasks including SPM register monitoring for our experiment purpose. The measure core is not essential to the attack. The fourth available core in the SnapDragon 805 SoC is either shut down or asked to perform regular workloads.

In the CLKSCREW attack process, several attributes on the hardware stack can help identify the attack attempt. For faulting injection, the attacker issued OPP must reflect disparate execution demands, namely, glitch frequency indicates demanding computation and a low voltage indicates low workload. Hence, target voltage and frequency and its change from baseline can serve as CLKSCREW attack warning. And for the injected fault to be of use to the attacker, the controlled timing is another important indicator.

The duration of the frequency spike and voltage dip can be another important feature for the countermeasure. For example, if the glitch duration is too long, it can induce too many faults across various parts of the target execution and makes the fault analysis infeasible. Too many faults can also reboot and freeze the device, which makes not only the attacker to fail to acquire faulty results but also users to think suspiciously on the malfunction. On the other hand, if the glitch duration is too short, it can cause no fault and the attack is failed.

3. STATIC PROTECTION

Countermeasures imposed during design and manufacturing test time are considered static protection strategies while runtime protection with the adaptive decision is dynamic protection strategies. In this section, we will discuss the advantage and disadvantage of the static countermeasures, followed by the description of the proposed dynamic countermeasure in Sec. 4.

Static protection strategies against CLKSCREW attack can be approached from mainly two angles. One approach is to set a hard limit to the voltage and frequency pairings in DVFS systems in each chip so as to prevent cores from operating at the condition that may allow attacks like CLKSCREW. Unfortunately, in the nanometer CMOS, processor cores' true operational limits exhibit significant variability due to process, voltage, and temperature variations, and need to be obtained through individual electrical chip testing after manufacturing. However, such chip-wise testing is not economically viable, the testing would be extremely costly as any additional tests in large production volume lead to large manufacturing overhead.

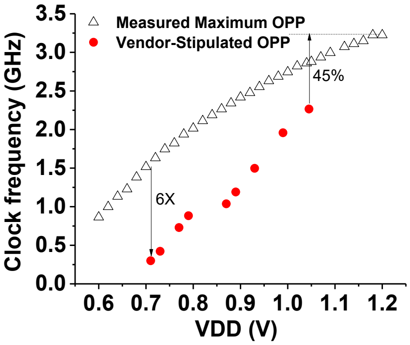

Fig. 3. Measured OPPs and vendor-stipulated OPPs across V_{DDS}

The more economical approach is to apply a conservative hard limit under which all chips meeting the production specification are guaranteed fault-free operation in any reachable frequency and voltage conditions. In fact, the manufacturer has provided the OPPs that guarantee error-free operation in all of their production chips (called vendor-stipulated OPP). Applying such highly conservative hard limits to all processors, however, can stifle processor performance and energy-efficiency. As shown in Fig. 3, we characterized the true operating F_{CLK} of the tested processors, and indeed imposing a hard limit based on vendor-stipulated OPPs would severely limit the performance capability of the processor from 45% to 6X.

4. DYNAMIC DETECTION & PROTECTION

4.1. Informative Features of CLKSCREW

Dynamically blocking of security-wise unsafe OPPs is preferable to static blocking. For dynamic protection to function, a measure of self-tuning is introduced in the design. As the basic trend of unsafe voltage and frequency is the same for each processor, a small number of parameters can suffice to tailor the regulator limits to each processor. These parameters are batch-trained in design time and then adjusted during runtime based on the feedback from the processor cores and their subsystems.

In Sec. 2.2, we described the CLKSCREW attack carried out in several precise steps. Some of these steps such as core pinning and timing profiling are not easily observed on the hardware stack and requires software level behavior monitoring and are thus forgone in this paper. The basic hardware-accessible features indicative of a CLKSCREW attack are (i) low voltage and (ii) sharp increase of clock frequency for a short duration, which together induces faults in the targeted part of a victim thread execution. Factors influencing the timing fault injection are also necessary to consider in the protection scheme, some of the most important of which are temperature and activities of other cores. SoCs contain a DTM subsystem and thus core temperature information is available in both software and hardware.

We conducted a number of CLKSCREW attacks on the Nexus 6 device across a range of parameters, namely base frequency, glitch frequency, glitch duration, V_{DD}, the number of active cores and their F_{CLK}s, and temperature. Fig. 4(a) shows the trends that the glitch frequency of successful CLKSCREW attack is proportional to the V_{DD} of the victim core. This is expected since the transistors and thus cores can operate faster at higher V_{DD}.

Fig. 4(a) also shows the effect of temperature on the unsafe OPPs. Across temperatures, we observe the same trends that the glitch frequency increases with V_{DD}. However, the slope is found to be temperature-dependent. This is due to so-called inverse temperature coefficient effect (ITC) [9], where high temperature makes modern IC faster at low V_{DD} but slower at high V_{DD}. This requires non-linear classification on detecting CLKSCREW across temperatures.

Fig. 4(b) shows the impact of the glitch duration. If the attack uses a wide glitch duration, it achieves lower success rate. It also makes the device more likely to reboot or freeze. This is because the amount of injected fault is too much, and affects the operation outside the targeted one. On the other hand, shorter glitch durations tend to make it fail to inject a sufficient amount of fault. During the glitch duration, the processor may not process the data that exercise the critical paths.

Fig. 4(c) shows the impact of the activity of the core unrelated to the CLKSCREW attack. We perform the CLKSCREW attack while activate and deactivate the fourth core, titled a spare core, hereafter. Interestingly the activation of the spare core and its working clock frequency has a non-linear impact on the glitch frequency of the victim core. We suspect that electrical coupling of power grids of the processor cores may cause this.

Fig. 4(d) shows the detailed impact of F_{CLK} of the spare core at 0.79V. The glitch frequency of 2 GHz is sufficient for successful CLKSCREW attacks if the spare core runs at the clock frequencies below 0.883 GHz, which is interestingly vendor-stipulated OPP at the V_{DD} of 0.79V. However, the glitch frequency increases to 2.5 GHz with the higher clock frequency of the spare core. Considering all the complex relationships between features denoting CLKSCREW attack, dynamic response to regulator commands is invaluable to its mitigation.

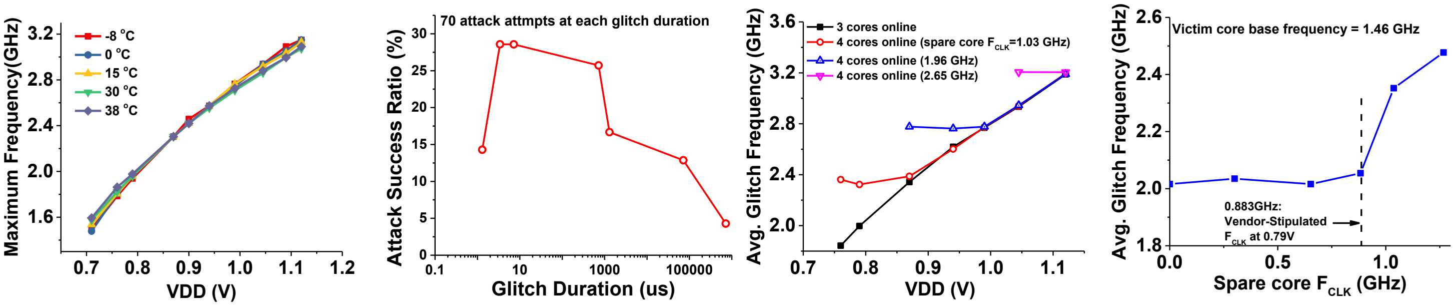

Fig. 4. (a) V_{DD} and temperature effects; (b) glitch duration impact; (c,d) spare core activity effects on successful CLKSCREW attacks.

4.2. Detection and Protection Algorithm

Fig. 5 shows our algorithm to detect and mitigate CLKSCREW. During runtime, the algorithm is triggered at each issued clock frequency regulation command. We designed the voltage regulator control not trigger the algorithm execution since it is of a rougher granularity and less agile in its switch. In CLKSCREW, voltage is controlled during the preparation stage to lower the glitch frequency required to induce fault, not as a nob in precision fault injection.

Once triggered, the algorithm first performs feature extraction, which detects glitch dynamics by inspecting the current and past clock frequency commands through a sliding window. Once it detects glitch dynamics, it extracts three key parameters discussed earlier in Fig. 2, namely base frequency, glitch frequency, and glitch duration.

These glitch parameters along with other parameters, namely clock frequencies of other cores, V_{DD}, and temperatures, are the inputs to a multi-layer perceptron (MLP) neural network. The role of this network is feature dimensionality reduction, i.e., to project the multi-dimensional features onto a simple 1-dimensional space. This enables to represent the decision boundary in a single scalar and thus make it easy to adjust during runtime based on the timing error flag.

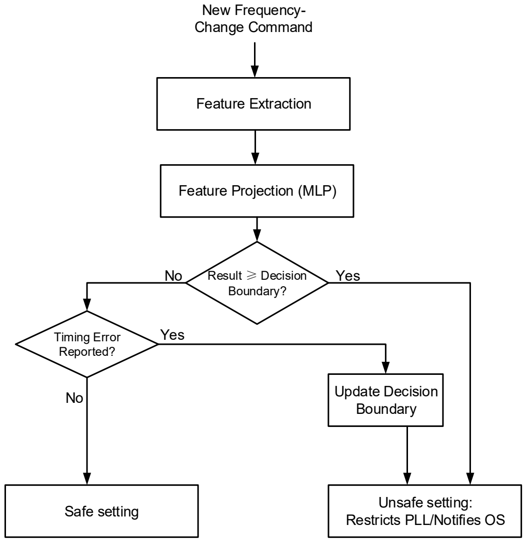

Fig. 5. Proposed mitigation strategy

Specifically, the MLP is composed of two fully connected layers. We swept the hidden neuron count and found 50 units to be optimal for the accuracy and complexity trade-off. There are eight inputs, one bias, and one output neuron in this implementation. We chose the widely-deployed ReLU for the activation function of the hidden neurons, which leaves the positive input unchanged and sets negative inputs to zero, for its low hardware cost. The output of output neuron, on the other hand, is not rectified, i.e., simply producing the weighted sum of the last hidden layer. This output is then compared directly to a *decision boundary*, whose initial value is set to 0.5. Exceeding the boundary would mean the current OPP is unsafe and thus classified to the blacklist, at which point the PLL is flagged to abandon the current frequency change command.

The weights of the MLP are trained during design time in a batch, aimed to model the approximate behavior of CLKSCREW. We use the classic back-propagation to train the network [10]. The weights are not updated dynamically due to the high cost of implementing dedicated training algorithm in hardware although online training of weights would be interesting for future research. On the other hand, we designed the decision boundary to be updated during runtime, aimed to account for chip-wise subtle variation impact on unsafe OPPs. Specifically, in the case that the MLP output indicates the glitch dynamics as a safe OPP but a timing error is reported, our algorithm lowers the decision boundary. Note that this online update of the decision boundary can be viewed as training of the activation function of the output layer in the MLP.

In this work, we assume to use hardware, i.e., TRC and EDAC, to acquire the timing error flags and adjust the decision boundary. This mechanism, however, can be further extended so that such hardware flags the OS to investigate the source of the fault, e.g., whether it is a malicious kernel. Core-pinning, rapid PLL commands, and other potential CLKSCREW characteristics can be used in the software-level investigation to determine whether MLP output decision boundary should be updated. In the case considered in this paper, the decision boundary is always updated regardless whether the fault is from CLKSCREW injection or natural causes like circuit aging. In this implementation, the circuit gradually becomes a general fault mitigation circuit in typical use (where CLKSCREW attack is rare if at all). In the event of the mitigation circuit failure, the processor requires leak-free correction. Even if a processor has timing error prediction/detection circuits, it is still desirable to have a CLKSCREW detection mechanism like the proposed one because timing-error circuits cannot differentiate between CLKSCREW and other various fault-inducing scenarios such as voltage droop and device aging.

4.3. Algorithm Experiment

We designed the proposed algorithm in a fixed point number to map it efficiently onto hardware. Each input feature to the MLP uses 16 bits. The weight of the MLP is 12 bits. We use 16 bits for most of the other parts of the algorithm.

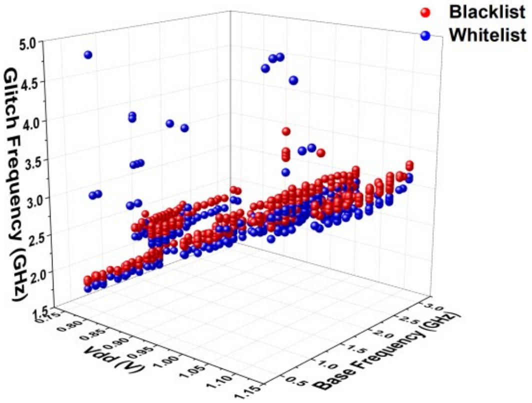

Fig. 6. A snapshot of the dataset of the settings that yield successful and unsuccessful CLKSCREW attacks.

We trained and tested the proposed algorithms using the test data we collected by performing CLKSCREW attacks on the Nexus 6 device. Fig. 6 shows a subset of the settings that succeed CLKSCREW attacks. Note that only three features are shown in the figure whereas each setting includes other features (e.g., temperature). The dataset contains 578 unsafe and 384 safe settings. Some of the safe settings exhibit high glitch frequency if the glitch duration is very short. Interestingly those do not incur either of faults, reboots, and freezes. We remove safe settings that are too close to the unsafe settings to ease the task of dynamic blacklisting.

We trained the MLP using about 640 settings randomly picked from the dataset. The label is set to 1 for unsafe settings and 0 to safe settings. Fig. 7(a) shows the output of the MLP before training. The distributions of safe and unsafe settings are largely overlapped. After training, as shown in Fig. 7(b), however, the overlap is significantly reduced. The overlap would contribute false rejection and false alarm.

The initial decision boundary is set to 0.5 since the MLP is trained to produce 1 for unsafe and 0 for safe settings.

We then use about 320 randomly-selected settings for testing the online adjustment of the decision boundary. The settings used for training are not used for this testing. Fig. 8 shows the output of the MLP. Most of the test settings are projected into the correct part of the feature space, i.e., unsafe settings make the MLP produce values greater than the decision boundary while safe settings smaller than the boundary. In this particular test, one setting (setting index = 40) has the MLP to produce the wrong value; thus the algorithm decrements the decision boundary.

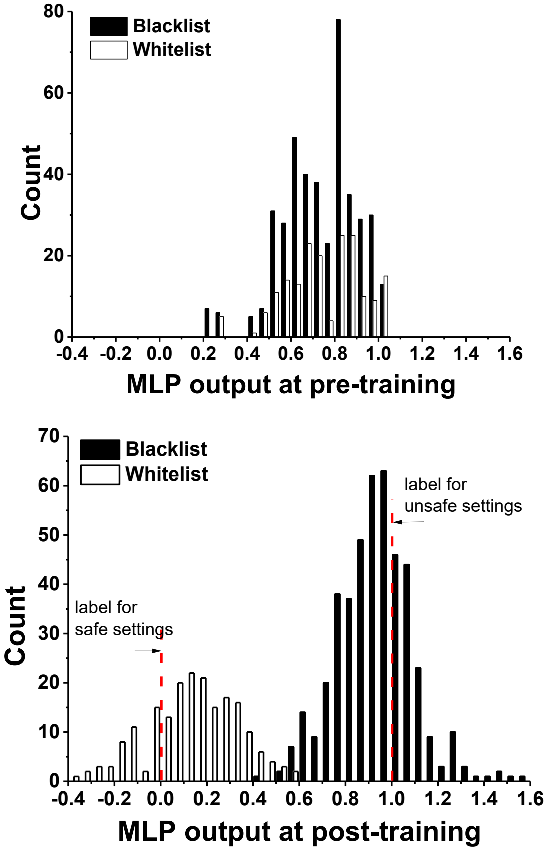

Fig. 7. Feature projection results before training (top) and after training (bottom)

Fig. 9 shows the accuracy of the algorithm in the fixed-point number. The algorithm can detect and mitigate CLKSCREW with online adjustment at the probability of 99.69%. Without online adjustment, i.e., the static boundary, it achieves 98.38%. The false detection ratio (i.e., flagging a safe setting as unsafe) is 0.92% with or without online adjustment. The benefit of the online adjustment is expected greater for the experiment over multiple devices.

5. BLACKLIST CORE DESIGN

5.1. System Architecture

We designed the BL core that implements the proposed algorithm. We will present the microarchitecture and circuits of the BL core in detail, but before that, we first want to describe the system architecture, i.e., the necessary system requirements to accommodate the BL core.

Several security considerations are necessary for the BL core to be effective. First, the countermeasure must not be accessible from the malicious kernel. The boundary setting which is modulated by timing error feedback is not software exposed in normal operation, however, OS should retain the privilege to control the countermeasure to limit overly aggressive adjustment of decision boundary that severely limits processor performance. Hence, the OS cannot have full freedom in boundary setting register, least the mitigation strategy be sidestepped completely.

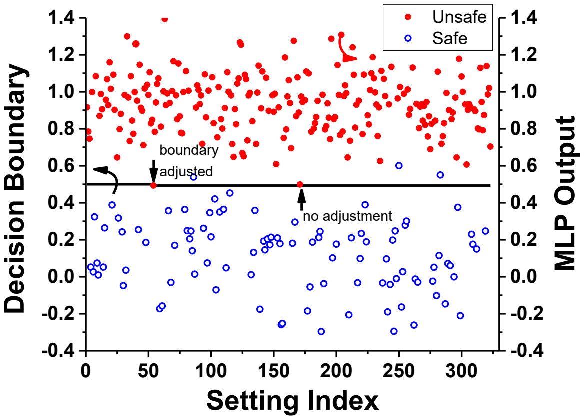

Fig. 8. Online adjustment of the decision boundary

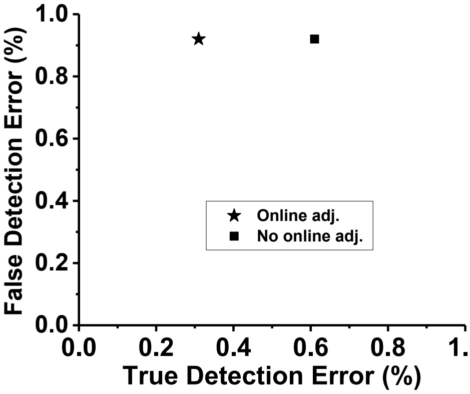

Fig. 9. The accuracy of our algorithm. The same results are achieved for the fixed- and float-point versions of the algorithm

Second, the BL core must not be conducive for fault injection via forced timing violation. The BL core should operate on the base frequency such that the operating voltage cannot be lowered enough to cause timing violation before the system freezes or reboots. Alternatively, a separate higher voltage domain can be used for the design, however, this is not the preferred choice as a separate voltage domain is not as readily available as the base frequency.

Third, there is a timing consideration for the BL core since the CLKSCREW attack is identified after a PLL configuration is already set in the control registers. Hence, the identification process must be completed before the frequency change is applied to the victim core. Frequency regulation is a relatively long procedure (in the time constant of µs). Thus the goal is to hide the latency of BL core computation within this phase-locking period of PLL circuits. We optimized the microarchitecture of the BL core such that it can finish the computation less than ~1 µs. Alternatively, although we do not pursue in this paper, the PLL could be designed so as to be required to be flagged safe by the mitigation circuit before regulating the frequency. This adds some cost in PLL control logic and adds some latency to frequency regulation.

5.2. Microarchitecture

Fig. 10 shows the microarchitecture of the proposed BL core. We designed each BL core to support one Krait core but multiple BL cores can be consolidated to save hardware redundancies. It performs the algorithm that we outlined in Sec. 4. The microarchitecture consists of feature extraction, MLP, and decision. In the feature extraction, the desired features are collected from regulator configuration registers and also computed from simple feature extraction circuits such as a counter for the PLL command interval. The features are modified through shift and truncation operations to reach its appropriate value scale and bit precision.

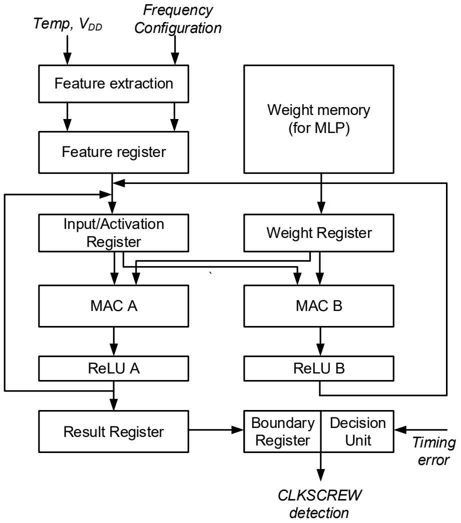

Fig. 10. BL core microarchitecture

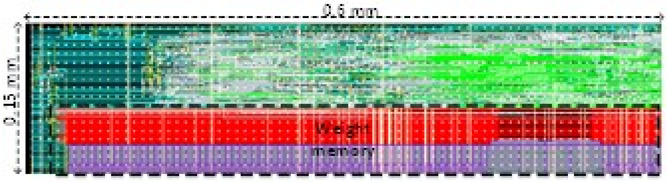

Fig. 11. BL core layout in a 65 nm CMOS

In the MLP, the main computational primitive is the multiply-and-accumulate (MAC). The BL core needs to finish the calculation on the classification task before the PLL operation. We, therefore, include two MAC computing units (MAC A and MAC B in Fig. 10) such that the computation delay of the BL core is shorter than the PLL operation and allow the blacklist event to stop the PLL from applying the fault-inducing frequency setting to the victim core.

The decision unit takes the output of MLP and compares it against the decision boundary. It also takes the timing error flag from TRC or EDAC and adjusts the boundary if it finds a discrepancy between the MLP output and the timing error flag.

We designed the BL core in Verilog HDL and implement it in a 65nm CMOS process through the standard cell design flow. Fig. 11 shows the layout whose area is 0.09 (=0.6 × 0.15) mm^2. As marked, close to 50% of the area is used by the weight memory for the MLP operation. It is implemented in an industrial 6-T SRAM array. A single Krait core takes ~2.6 mm^2 in a 28 nm [17]. The normalized area of the BL core is thus ~0.64% of that of the single Krait core. We further characterized the delay and power consumption of the BL core via the static timing and power analysis tools. We use the post-layout netlist with annotated parasitics and actual switching vectors for nodes from the Verilog simulation of the netlist. The accuracy of the flow is calibrated to match the SPICE simulation for a benchmark circuit. The BL core consumes 10.5mW at the clock frequency of 250 MHz and the supply voltage of 1V. It takes about 280 cycles to complete one detection and boundary adjustment if needed. The total latency is simulated to be 1.11 µs.

6. CONCLUSION

In this paper, we propose a dynamic blacklisting technique to detect and mitigate the power-management based fault injection attack titled CLKSCREW. We devised an algorithm which performs feature extraction, dimensionality reduction, and decision. The algorithm is trained in the design time and also supports online adjustment. We designed the BL core hardware that performs the algorithm. The BL core is optimized to make a minimal impact on silicon area and power dissipation. The decision latency of the core is also optimized to the level that can be hidden in a typical PLL locking latency.

ACKNOWLEDGEMENT

The project is in part supported by NSF (CCF-1453142), Catalyst Foundation, and DARPA (HR0011-18-C-0017)

REFERENCE

[1] A. Tang, S. Sethumadhavan, and S. Stolfo, "CLKSCREW: exposing the perils of security-oblivious energy management". *USENIX Security Symposium*, 2017.

[2] ARM. Security Technology - Building a Secure System using TrustZone Technology. ARM Technical White Paper, 2009.

[3] S. Kim, and M. Seok, "Analysis and Optimization of In-Situ Error Detection Techniques in Ultra-Low-Voltage Pipeline," *IEEE/ACM International Symposium on Low Power Electronics and Design*, 2014, pp. 291-294.

[4] V. Pallipadi, and A. Starikovskiy, "The on-demand governor." *Linux Symposium*, 2006, vol. 2, sn, pp. 215–230.

[5] B. Jeff, "big.LITTLE system architecture from arm: Saving power through heterogeneous multiprocessing and task context migration," *ACM/IEEE Design Automation Conference (DAC)*, 2012

[6] Intel The Engine for Digital Transformation in the Data Center. http://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/xeon-e 5-brief.pdf. Intel Product Brief. .

[7] QUALCOMM. Snapdragon S4 Processors: System on Chip Solutions for a New Mobile Age. https://www.qualcomm.com/documents/snapdragon-s4-processors-system-chip-solutions-new-mobile-age, Jul 2013.

[8] I. Anati, et al., "Innovative technology for CPU based attestation and sealing," *International workshop on hardware and architectural support for security and privacy (HASP)*, 2013, vol. 13.

[9] R. Kumar, V. Kursun, "Reversed temperature-dependent propagation delay characteristics in nanometer CMOS circuits," *IEEE Transactions on Circuits and Systems II: Express Briefs*. 2006 Oct;53(10):1078-82.

[10] D. E. Rumelhart, et al., "Learning representations by back-propagating errors," *Nature*. 1986 Oct; 323(6088):533.

[11] S. Kim, M. Seok, "Variation-Tolerant Near-threshold Microprocessor Design with Low-Overhead, Within-a-Cycle In-situ Error Detection and Correction Technique," *IEEE Journal of Solid-State Circuits*, 2015

[12] S. Kim, et al., "A 450mV Timing-Margin-Free Waveform Sorter based on Body Swapping Error Correction," *IEEE Symposium on VLSI Circuits (VLSI)*, 2016

[13] S. Kim, et al., "Near-Vt Adaptive Microprocessor and Power-Management-Unit System based on Direct Error Regulation," *European Solid-State Circuits Conference (ESSCIRC)*, 2017

[14] J. Tschanz, et al., "Tunable replica circuits and adaptive voltage-frequency techniques for dynamic voltage, temperature, and aging variation tolerance," *IEEE Symposium on VLSI Circuits (VLSI)*, 2009

[15] B. Zimmer, et al., "A RISC-V vector processor with tightly-integrated switched-capacitor DC-DC converters in 28nm FDSOI," IEEE Symposium on VLSI Circuits, 2015

[16] X. Sun, et al., "A Combined All-Digital PLL-Buck Slack Regulation System with Autonomous CCM/DCM Transition Control and 82% Average VoltageMargin Reduction in a 0.6-to-1.0V Cortex-M0 Processor," *IEEE International Solid-State Circuits Conference (ISSCC)*, 2018

[17] Introducing NVIDA Tegra 4i. URL: http://phx.corporate-ir.net/External.File?item=UGFyZW50SUQ9MTcyNDc2fENoaWxkSUQ9LTF8VHlwZT0z&t=1