Caphammer: Exploiting Capacitor Vulnerability of Energy Harvesting Systems

Jongouk Choi, Jaeseok Choi, Hyunwoo Joe, and Changhee Jung

***Abstract*—An energy harvesting system (EHS) has emerged as an alternative to traditional battery-operated Internet of Things (IoT) devices. An EHS harnesses ambient energy and stores it in a small capacitor, enabling batteryless operation when sufficient energy is available. However, capacitors are susceptible to malicious charging/discharging and over-voltages, which can lead to a loss of capacitance. With the capacitor vulnerability in mind, this paper introduces a capacitor hammering attack, simply Caphammer, that can undermine the security of every EHS. The idea is that Caphammer can degrade the capacitance by using frequent power outages. Once Caphammer degrades the capacitor of the victim EHS, it can suffer from denial of service, data corruption, data encryption failure, and abnormal termination. To defeat Caphammer, this paper presents FanCap, a capacitor bank scheduling scheme that can dynamically transform energy storage organization, taking into account the capacitor vulnerability. The experimental results demonstrate that FanCap can successfully thwart Caphammer with a negligible run-time overhead.**

***Index Terms*—Energy harvesting, Capacitor, Hardware Security, Reliability**

I. Introduction

The number of connected objects in the Internet of Things (IoT) is growing, enabling new application areas such as implantable medical devices, wearables, and smart homes. Powering these IoT devices presents a significant challenge due to their large batteries, limited lifespan, and high replacement costs. This has sparked significant interest in energy harvesting system (EHS) technologies, which capture free ambient energy from their surroundings and offer the intriguing possibility of battery-less computing [19], [21], [23], [26], [50], [57].

Manuscript received March 30, 2024; accepted July 15, 2024. The work of Jongouk Choi and Jaeseok Choi was supported by NSF under Grant 2314680. The work of Hyunwoo Joe was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) Grant funded by the Korea Government (MSIT) through the Development of Core Technology for Autonomous Energy-Driven Computing System SW in Power-Instable Environment under Grant 2021-0-00360 and through the Lightweight Edge Device for Resilient Interaction under Grant 24HR3120. The work of Changhee Jung was supported by NSF under Grant 2001124 (CAREER), Grant 2153749, and Grant 2314681. This article was presented at the International Conference on Embedded Software (EMSOFT) 2024 and appeared as part of the ESWEEK-TCAD special issue (Hyunwoo Joe is the co-first author) (Corresponding author: Jongouk Choi).

Jongouk Choi is with the Department of Computer Science, University of Central Florida, Orlando, FL USA (e-mail: jongouk.choi@ucf.edu).

Jaeseok Choi is with the Department of Computer Science, University of Central Florida, Orlando, FL USA.

Hyunwoo Joe is with the Mobility UX Research Section, Electronics and Telecommunications Research Institute (ETRI) and University of Science and Technology (UST), Daejeon, South Korea.

Changhee Jung is with the Department of Computer Science, Purdue University, West Lafayette, IN USA.

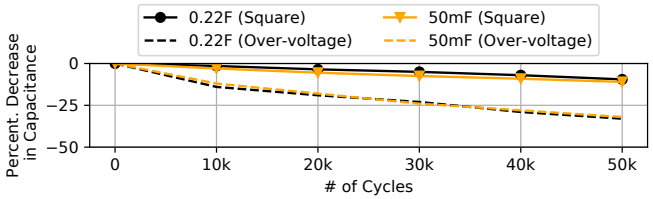


Fig. 1: Capacitor degradation under square wave voltages. A capacitor is considered as dead at 20% degradation [43].

However, since energy harvesting sources are unreliable, the resulting power is inherently unstable, causing frequent and unpredictable power outages. To this end, an EHS leverages a capacitor as energy storage that can buffer the harvested energy; if a sufficient amount of energy is secured in the capacitor, an EHS spends the buffered energy to operate the target device. To achieve crash consistency across power failure, an EHS is equipped with a non-volatile memory (NVM) and a just-in-time (JIT) checkpointing mechanism that keeps monitoring the energy level of the capacitor and checkpoints volatile data (i.e., all registers) when power is about to be cut off [17]. The takeaway is that the capacitor is at the heart of energy harvesting, as it is such an essential component for any EHS to endure frequent power failures.

Unfortunately, capacitors are unreliable in frequent power outages, leading to capacitance loss [22] To demonstrate the capacitor reliability issue and its impact, we conducted direct injections of square wave voltages (5V) to two capacitors of typical EHS devices at 1Hz; 0.22F and 50mF are used on typical EHS platforms (0.22F on MSP430 [2] and 50mF on Powercast EVB [64]), respectively. Figure 1 shows how the capacitance degrades over time under the malicious square wave voltage input. It turns out that the two capacitors are both degraded by 10% in terms of capacitance in 30 hours—and they continue to degrade over time. Furthermore, we also injected over-voltage (5V) square wave signals into the capacitors and observed a significant drop in capacitance, as shown in Figure 1; the maximum voltage level of the capacitors is 5V [64]. This is because the capacitor materials can be decomposed at a high voltage, which leads to a resistance increase, i.e., capacitance decrease [16]

With the capacitor vulnerability in mind, this paper presents a capacitor hammering attack, simply Caphammer, that can undermine the security of every EHS. The idea is that Caphammer can degrade the capacitance by exploiting frequent power outages with or without over-voltages. Since they are a norm of EHS due to ambient energy sources' unreliable nature, attackers can stealthily launch Caphammer without a

hassle. Moreover, launching Caphammer is straightforward and thus easy to succeed. If attackers control the energy harvesting source or spoof it with strong input power signals, they can readily stress the capacitor of the victim EHS by repeatedly powering it on/off. This abrupt charging/discharging degrades the capacitance.

After the capacitor is compromised by Caphammer, it can lead the victim EHS to security breaches. If the capacitor starts to malfunction due to the capacitance degradation, the EHS cannot precisely monitor the exact amount of energy available in the storage. The EHS ends up overestimating the amount of energy that the victim capacitor can provide, leading to the failure of the JIT checkpointing. In other words, EHS cannot possibly complete the saving of all registers due to power failure during the checkpointing. That is because the victim capacitor cannot buffer the same amount of energy as intact energy storage. By exploiting the JIT checkpointing failure, Caphammer can launch denial of service, data corruption, data encryption failure, and abnormal termination.

Surprisingly, our findings indicate that existing EHS devices, such as the Powercast Wireless Sensor Node (WSN) [64], WISP [12], and glucose monitor [6], are susceptible to Caphammer. This vulnerability poses a significant threat to the practice and success of EHS-based IoT. We reported this vulnerability to Powercast, a major EHS manufacturer, and it has recently been confirmed by them. Consequently, there exists a compelling need for an effective countermeasure capable of preventing Caphammer without incurring a significant performance overhead or requiring expensive hardware costs.

To this end, this paper presents FanCap, a capacitor bank scheduling scheme that can transform energy storage organization—leveraging programmable stacked parallel-series switched capacitor banks—and mitigate Caphammer attacks proactively. FanCap exploits two unique characteristics of a capacitor and EHS. First, EHS does not boot until its capacitor is fully charged, and it is thus assured that a program can make as much progress as the full capacitor allows in the wake of power failure—unless it is under attack. Second, a capacitor has a resilient nature, i.e., it can be self-healed when it becomes idle [22]. Thus, even if Caphammer degrades a capacitor, it can be recovered by being isolated in quarantine.

With that in mind, FanCap can detect an attack scenario at reboot time by checking whether EHS has made the assured progress since the prior power-on time. If the EHS encountered power failure before making the progress, i.e., they were under attack, FanCap reconfigures the capacitor banks into separate parallel capacitors. They take turns powering the EHS while waiting for their turn in quarantine. In this way, even if one of the capacitors was under attack, it can restore its original capacitance, thanks to its resilient nature, during the quarantine period while the others are used. Once all the parallel capacitors complete their quarantine and Caphammer is defeated, FanCap gets back on track by reconfiguring the capacitor banks to their original organization.

The upshot is that FanCap stands out as a lightweight and practical countermeasure. FanCap effectively mitigates Caphammer without causing a significant performance overhead. The capacitor bank reconfiguration can be quickly done, and it maintains the original capacitance, ensuring the same energy buffering capacity. Our experimental results confirm that FanCap successfully thwarts Caphammer, with an average performance and energy overhead of just 4%.

The contributions of this paper are as follows:

- We discover that current EHS devices are vulnerable to our novel capacitor hammering attack (Caphammer).
- We demonstrate that Caphammer causes critical security implications such as denial of service, data corruption, abnormal termination, and encryption failure in EHS devices. We have also made responsible disclosure to the corresponding EHS manufacturers.
- We propose a capacitor bank scheduler called FanCap that can proactively defeat Caphammer and restore the original capacitance by exploiting unique characteristics of a capacitor and EHS.

II. Background and Motivation

A. Energy Harvesting System Architecture

Due to the unreliable nature of ambient energy sources, an EHS suffers frequent power outages. To address the issue, EHS leverages a low-power microcontroller (MCU) such as TI-MSP430 [2] with a capacitor—as energy storage—to intermittently compute only when sufficient energy is buffered in the capacitor. On the other hand, if the buffered energy is depleted, the EHS dies due to the lack of enough energy to power the MCU, i.e., it is power-interrupted. With that in mind, researchers equip EHS with byte-addressable NVM as the main memory and some form of crash consistency to checkpoint necessary data at run time and restore them in the wake of the power failure.

B. Crash Consistency for Correct Power Failure Recovery

To achieve correct power failure recovery, EHS often creates a checkpoint on which the volatile registers are saved into the NVM in order to roll back to the most recently checkpointed states when the power comes back after an outage. Nevertheless, this simple checkpointing mechanism alone cannot always achieve correct recovery due to the inconsistency of data in NVM. In other words, NVM data can become corrupted across power failure when a write-after-read (WAR) dependency exists [18], [25], [27]–[30], [39]–[42], [55], [57], [67], [68].

The memory inconsistency stems from the program control rolling back across WAR dependence during power failure recovery, reading values updated by stores left behind the failure. That being said, a solution to this issue is to move forward for recovery instead of rolling back; this is so-called roll-forward recovery [17], [49]. With roll-forward recovery, when power is restored, the mechanism resumes the interrupted program at the same failure point, avoiding the crossing of WAR dependence. Consequently, these hardware roll-forward recovery schemes can naturally achieve crash consistency.

There are two most popular hardware roll-forward recovery schemes called NVP [49] and QuickRecall [17]. To achieve energy-efficient checkpoint/recovery, NVP uses a hybrid register file (HRF) circuitry comprising standard flip-flops and

non-volatile flip-flops (NVFF). Since the volatile and non-volatile flip-flops are laid out right next to each other in the circuit, their data movement is fast, enabling instant register checkpoint/recovery. However, the HRF requires intrusive microarchitecture modification. To lower the hardware cost, QuickRecall dedicates a part of NVM as register checkpoint storage instead of using NVFF. Both schemes exploit the roll-forward recovery for crash consistency based on another mechanism called just-in-time (JIT) checkpointing.

Just-In-Time (JIT) checkpointing: JIT checkpointing saves volatile registers to their checkpoint storage (NVFF as with NVP or NVM as with QuickRecall)—when EHS is about to encounter power failure. To recognize the impending power failure, the EHS is equipped with a voltage monitor to measure the output voltage of a capacitor. If the voltage level is lower than a pre-defined threshold, i.e., V_{backup} for power-failure-free checkpointing of all registers, then the voltage monitor assumes that power is about to be cut off. Thus, the monitor sends a signal to the controller logic to let the processor copy all registers to their checkpoint storage, i.e., NVFF or NVM.

Note that EHS can identify when a sufficient amount of energy is secured in the capacitor to start the MCU. NVP and QuickRecall first buffer harvested energy in the capacitor. The voltage monitor then judges whether the buffered energy is enough to operate the MCU by comparing the capacitor's current-voltage level to another pre-defined threshold V_{on}. If the voltage level became greater than the threshold since the last power failure, the voltage monitor sends a wake-up signal to the controller to restore checkpointed registers and, in turn, resume the power-interrupted) program. The takeaway is that the capacitor is at the heart of energy harvesting—because it is an essential part for EHS to survive across power failure.

C. Capacitor Degradation

Unfortunately, capacitors can be degraded and lose their initial capacitance due to the unreliability issue [22]. When a capacitor is degraded, the JIT checkpointing can be interrupted since the buffered energy in the capacitor is insufficient; this is called a *capacitor error* [22].

Continuous Charge/Discharge. Continuous charging/discharging can degrade the original capacitance due to electrochemical corrosion, which can, in turn, cause capacitor loss or formation of additional dielectric layer [13]. Continuous charging/discharging behaviors are particularly interesting in that they are a norm of EHS, i.e., one can readily exploit the behaviors to launch a hardware security attack.

Over-Voltage. The over-voltages damage a capacitor film and raise the leakage current flow [56] and operating temperature. The dielectric material inside the capacitor can be damaged or stressed, leading to a reduction in the capacitance value. Also, over-voltages can increase the Equivalent Series Resistance (ESR) of the capacitor, resulting in higher power losses and reduced efficiency. Prior works conducted aging tests to see the over-voltage effect [16]. They injected over-voltage square wave signals into a given capacitor (above the nominal voltage) and measured the capacitance under room temperature at about 25°C without cooling or heating. The maximum operating voltage of the capacitor is 2.5V, which is a typical specification for supercapacitors [61]. From the experiments, they found that the capacitance is reduced when the operating voltage is higher than the capacitor's nominal voltage. They also established a relationship between the life expectancy and voltage and defined an exponential function to estimate the lifespan of a capacitor, T_{exp}, as follows: $T_{exp}(U,\theta) = c1 \cdot e^{(\frac{U}{c2}+\frac{\theta}{c3})}$, where $(c1, c2, c3)$ are the constant parameters (negative), and U and θ are the voltage and the temperature, respectively.

Based on the findings, we explored the impact of over-voltages on EHS. We observed that over-voltages can significantly degrade capacitors, resulting in checkpoint failures in both NVP [49] and QuickRecall [17]. The capacitors showed considerable aging under higher voltages, particularly above 3.0V, leading to an increase in ESR. When the capacitor's degradation reaches 10%, the EHS experiences failures in JIT checkpointing. This is primarily due to the degraded capacitor's inability to store the same amount of energy as an intact energy storage system. Consequently, the system cannot complete JIT checkpointing at the same checkpoint voltage threshold, e.g., V_{backup}.

The continuous charging and discharging, along with over-voltages, are particularly noteworthy because they are common in EHS. More importantly, attackers can readily exploit them to launch a stealthy hardware security attack. Even if attackers control the frequency of power outages, it can be challenging to distinguish between an ordinary harvesting situation and an attack scenario.

D. Capacitor Resilience

Despite the capacitor vulnerability, i.e., capacitance degradation, a (super)capacitor can be recovered during idle time due to its resilient nature [22], [43]. A previous study illustrates the capacitor recovery phenomenon by constantly injecting square wave voltage into a capacitor and leaving it idle [22], [43]. With the resilient nature of capacitors in mind, we propose an OS-driven solution (FanCap), the countermeasure of Caphammer which dynamically quarantines the victim capacitor, leaving it idle, so that it can restore the original capacitance; the detailed discussion is deferred to §V.

III. Threat Model

A. Energy Harvesting System Features

Applications The current EHS applications are mostly wireless sensor node, implantable medical devices, highway toll card, and wearables [65]. They harvest ambient energy and run an infinite loop that repeatedly senses and alarms when something turns out to be wrong based on the sensing result; users can also read the sensor data by scanning the device (e.g., RFID). For example, a continuous glucose monitor is a type of wearable medical devices that can be used for diabetic patients [6], [51]. The device can harvest energy from blood pressure and monitor/log the user's health status, such as temperature and blood sugar/glucose. When the glucose level is too low or too high, the device sends an alarm to end-user. The beauty of this solution is that the monitoring

Applications	Platform / PMU	V. Monitor/ Regulator	Capacity Monitor
Sensors [17], [22], [31], [53], [54], [59], [60]	TI-MSP430 [2]	Yes	No
Healthcare/Robot [9], [33], [47]	TI-BQ25570 [3]	Yes	No

TABLE I: Properties of EHS Applications: PMU stands for power management unit.

is possible without blood sampling, even when the inpatient sleeps or walks around. For such applications, Microchip and Powercast sensor devices are widely used [63], [64].

Weak Input Power. The actual harvested power for EHS devices is very weak (0∼2000μW [32]). In this harsh environment, the EHS devices can operate for a short amount of time (e.g., 15ms [36]) quickly depleting the buffered energy, but hibernate for a long time (e.g., more than 1s).

OS/Runtime support for power failure recovery. EHS devices use either OS or run-time system that controls the underlying crash consistency support—that leverages the JIT checkpointing. This paper will discuss about their vulnerability and possible security implications in §IV-C.

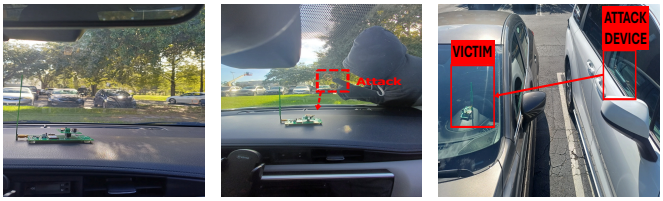
Voltage Margin. For the JIT checkpointing, current EHS devices have a checkpoint voltage margin because the voltage monitor can suffer a signal delay that can lead to checkpoint failure. However, since the margin can only be used for checkpointing, not for computation/progress, it is not practical to have a large margin; if they had a large voltage margin, the device would quickly stop the program execution and checkpoint data, leading to more power failure than having a small margin. That is why none of the prior works uses such a large voltage margin; instead, their voltage margin is only 0.01% to 7% [17], [49]. This paper assumes that the voltage margin can be enlarged at most by 10%, as suggested in a recent work [22].

Lack of Maintenance. EHS devices are mostly unmanaged once deployed, primarily due to their battery-free design. The lifetime of EHS is either infinite or determined by the manufacturers, which eliminates the need for users to actively manage and monitor the systems regularly. Users can rely on EHS to function without the need for continuous maintenance or status checks until they receive any notice or specific indications that require their attention.

Lack of Capacity Monitor. Most EHS schemes omit a capacity monitor in their platform or power management unit (PMU), assuming capacitors are reliable (Table I). Moreover, a capacity monitor is overkill for power-hungry EHS devices, as it requires charging the capacitor with a known current and measuring the voltage to calculate capacitance. This process prevents EHS devices from performing tasks, as they must avoid using the buffered energy to ensure accurate monitoring.

B. Attacker's Goal and Scenario

The adversary's objective is to degrade a capacitor in the victim's EHS device covertly, causing a checkpoint failure and silent data corruption. If Caphammer is executed successfully, the victim device cannot save sensor data in NVM or send alarms when necessary. Consequently, users can only access outdated data from the victim device. It is essential to highlight that naive physical access attacks and attempts to damage the victim device, such as using strong microwaves, would be readily detected. For instance, physically tampering with the victim devices or energy harvesting sources would draw the attention of users or administrators, resulting in detection in most cases. In contrast, the Caphammer does not assume physical access to the device or source within its threat model.



(a) User Case: An energy harvesting device is placed underneath a vehicle's windshield

(b) Attack Case 1: Adversaries degrade a victim capacitor remotely outside of the vehicle.

(c) Attack Case 2: Adversaries degrade a victim capacitor remotely from another vehicle.

Fig. 2: Attack Scenario

Equipment. We assume that attackers can employ an radio frequency (RF) signal jammer or an electromagnetic fault injection (EMFI) device [38] to initiate a remote attack (Caphammer); we found that adversaries can also employ the EMFI device to inject the over-voltages to a victim capacitor. Such malicious signals can penetrate common physical barriers like walls and windows. The attack devices will be described more detail in §IV-D.

Remote Attack. Caphammer can be launched remotely by using the attack devices. The attack distance depends on the capabilities of the attacker; attackers can go farther away from the target as long as they can exploit power outages with their devices.We argue that such a remote attack is practical due to the maintenance-free nature of EHS (§III-A), yet it would be also easy to come close to the victim devices without attracting users' concern just like placing a Trojan horse in guard-free systems. It's important to note that we are not assuming that attackers intend to physically damage or toast the victim device with the attack devices.

Attack Scenario An user wears an energy-harvesting glucose sensor device on his/her arm or places a RF toll transponder under a vehicle's windshield, as shown in Fig. 2(a). In this case, adversaries cannot physically access and damage the targeted EHS device. We assume that adversaries can move around and pass by the victim device without getting caught due to the maintenance-free nature of EHS. When attackers are near to the victim as shown in Fig. 2(b), they launch Caphammer using attack devices concealed. Also, as shown in Fig. 2(c), to attack a toll transponder in a victim's car parked in a public lot, adversaries can park their vehicles within 5 meters of the victim's car and launch Caphammer. In our experiment, the attack lasted about 5 hours.

IV. Caphammer

A. Caphammer Design at a High Level

To launch Caphammer, attackers exploit one of two strategies, i.e., flooding or spoofing, by using the attack devices. Attackers can intermittently flood the victim EHS with strong synthetic power that can be combined with the original power

signal being harvested or directly feed malicious power inputs to the EHS spoofing the original energy harvesting source.

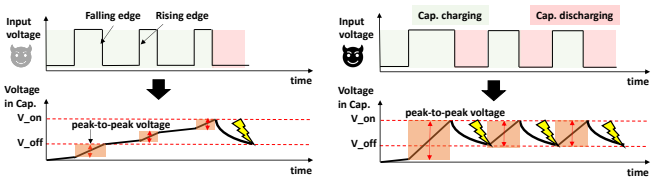
First, for the flooding attack, attackers use the attack device to intermittently add a significant amount of RF energy to the original ambient energy—harvested from the RF source—and quickly drop it, so that the combined input power fed to the victim device is akin to the square wave voltages. Since the strong input power is what EHS devices desire due to the power-hungry nature of EHS devices, the power flooding attack would be considered as a good harvesting condition in the user's perspective. In other words, the users would not even expect that their capacitor can be damaged with a strong input power, which makes Caphammer a practical/unexpected hardware attack.

Second, for the spoofing attack, attackers first sense when the original harvesting source cannot provide a power signal, which is detected by a harvester of the attack device. Then, the attackers supply the malicious power using the attack device—spoofing the original energy harvesting source—while it is idle. In this case, since the attack device is the only available power source, attackers can more easily generate the high-voltage square wave signals. Both flooding and spoofing attack strategies can launch Caphammer by inducing a target capacitor to be repeatedly charged and discharged as long as their square wave voltages are meticulously prepared.

B. Synthesizing Malicious Power Inputs

Unfortunately, it is insufficient to just provide the target device with malicious voltages (square wave high voltages) unless they are carefully synthesized considering the capacitor's behavior. If attackers end up using naive square wave voltages, it might be impossible, or at least take a while, to launch Caphammer (i.e., damaging the capacitor) due to the resilient nature of a capacitor—especially when the charging/discharging rate is low. For example, as shown in Fig. 3(a), the naive power signal causes only one power outage in three power cycles; the device reboots after the third rising edge of the signal and hibernates after the third falling edge. The malicious/efficient voltages, on the other hand, cause three power outages (at each falling edge of the input signal), as shown in Fig. 3(b). With this in mind, this paper proposes a two-step heuristic approach for generating efficient square wave power signals that can launch Caphammer: (1) estimating its charging and discharging times, and (2) synthesizing efficient square wave voltages by considering the resulting charging and discharging times.

Step 1. Attackers first provide strong power for charging the capacitor until the EHS device is booted; the higher the input power, the more stressed the capacitor is because a sudden increase in voltage generates a higher inrush current to the victim capacitor, resulting in high peak-to-peak voltage, as shown in the first charge cycle as shown in Fig. 3(b). Then, they measure the capacitor charging time, i.e., how long the power has been provided to the device to wake it up (i.e., reboot) since the last power-off point. To identify when the target EHS device is powered off and on without physically accessing it, we assume attackers can measure a change in



(a) Naive voltage inputs (b) Efficient voltage inputs

Fig. 3: Comparison of two square wave voltages; capacitor's voltage fluctuation (bottom) is controlled by attacker's input voltages (top).

the electromagnetic (EM) field of the victim. A jamming device can remotely measure such EM field changes when the victim is powered on and off. Similarly, to estimate the discharging time of the capacitor, the attackers stop providing stable power when the target EHS device is awake by turning off the disguised energy source (i.e., the jamming device for either spoofing or flooding attacks). Then they measure the power-on period (capacitor discharging time) of the device, i.e., how long it can sustain without harvested power input.

Step 2. This step leverages the capacitor charging/discharging times determined in Step 1. Taking this into consideration in the preparation of efficient square voltage synthesis, the attackers generate a power signal whose falling and rising edges correspond to capacitor discharging and charging points, respectively, as shown in Fig. 6(b). They feed the resulting voltages to the target device. As a result, such synthesized square wave voltages can achieve high peak-to-peak voltage, accelerating the attack process.

Discussion. Attackers can also employ a heuristic approach to synthesize power signals by utilizing square-wave voltages without the EM measurement. Specifically, they can provide strong power to charge the capacitor and then halt the power supply during program execution, aligning the falling edges of their power inputs with capacitor discharging points. Conversely, to synchronize the rising edges of their power inputs with capacitor charging points, attackers initially extend a half of the square wave to ensure a victim power outage within a single cycle. Using these initial power inputs, they monitor the launch of Caphammer over time. If Caphammer is not initiated, they progressively shorten the power-off duration. This iterative process allows attackers to ultimately synthesize their power signal. Moreover, attackers can regulate the degradation rate of the capacitor by controlling the electromagnetic (EM) field strength—though this paper defers a detailed, fine-grained analysis of this aspect to our future work.

C. Security Vulnerabilities

Data Corruption and Abnormal Termination. Caphammer easily causes a data corruption problem. Due to the malfunctioning capacitor damaged by Caphammer, the victim processor is likely to fail to make a checkpoint that saves necessary data, e.g., registers, in NVM for recovery purposes. Then, the victim would have only partial or corrupted data in NVM, thereby causing the wrong recovery across power failure. In particular, when special purpose registers such as a program counter (PC) or stack pointer are not checkpointed

correctly (in a power failure atomic manner), the victim will jump into an invalid PC or access outside of the NVM region beyond the limit of the designated stack area, in the wake of power failure, thereby causing an abnormal termination.

Denial of Service. Caphammer leads to a denial of service (DoS) issue, specifically, a lack of forward execution progress. This paper reveals that the DoS problem can manifest in atomic tasks and I/O operations. Since these atomic functions must be executed without any power failure interruptions, their length (execution time) is constrained by the capacitor's size. In other words, these functions are designed to complete within a single capacitor charge cycle [60], [67]. Consequently, if the capacitor is degraded by Caphammer, the atomic functions can never be completed across power failures, no matter how many times the victim device is rebooted. This situation results in the DoS problem.

Encryption Failure. To achieve confidentiality for EHS devices, prior works [10] employ an AES-CTR (counter mode) encryption algorithm. This mechanism utilizes a counter to generate a one-time pad (OTP) string, which is always unique by updating the counter once it is used. Then, it XORs a plaintext (PT) with the OTP for encryption. It is important to note that this algorithm is implemented using custom hardware support in the memory controller. It persists both PT and OTP in a power-failure-atomic way with the JIT checkpointing. It can flush all the write pending queue contents to NVM when power is about to be cut off.

Unfortunately, prior works are vulnerable to Caphammer, leading to an encryption failure problem. Caphammer can cause the victim to fail in storing encrypted data and its associated security metadata as a pair [10] with a power failure occurring between the data store and the metadata store. In this scenario, JIT flushing cannot be completed due to insufficient energy provided by the damaged capacitor, i.e., the victim EHS cannot decrypt the encrypted data across power failures since its associated metadata was lost.

D. Validating and Evaluating Caphammer

Experimental Setting. To demonstrate Caphammer, we conducted experiments targeting a real EHS sensor device, Powercast WSN [63], [64]. Furthermore, as a proof of concept, we conducted additional experiments with a TI-MSP430FR5994 [2] evaluation board and Powercast P2110-EVB [64]. 0.22F and 50mF are used on typical EHS platforms (0.22F on MSP430 and 50mF on Powercast EVB; both capacitors have an absolute maximum voltage rating of 5.5V, but the recommended operating voltage rating is 2.5V or less. To power the EHS, we used a Powercast TX91501-3W RF transmitter with a 915 MHz frequency and a 6.1 dBi patch antenna; the transmitter was placed 50cm from the victim.

On the evaluation board, we ported the prior works called QuickRecall [17] and Samoyed [59]. Both QuickRecall and Samoyed enable JIT checkpointing for power failure recovery; however, Samoyed disables the JIT checkpointing during a peripheral operation to avoid possible memory inconsistency problem, i.e., the JIT checkpointing does not work for peripheral operations. With the two prior works, we repeatedly ran benchmark applications [17], [35] on the board. For JIT checkpointing, we set the checkpoint voltage (V_ckpt), power-off voltage (V_off), and power-on voltage (V_on) thresholds as 2.00003V, 1.8V, and 3.3V, respectively (as a prior work states [17]). Moreover, we allocated a checkpoint storage starting at 0x33016 in NVM space. To launch Caphammer, we used a RF jamming device and an EMFI device. For the RF jamming device, we used an Arduino Nano and a 915 MHz EBYTE LoRa Module supplying a 3.1 dBi antenna. For the EMFI device, we developed a small device that is capable of generating approximately 0-4kV [38].

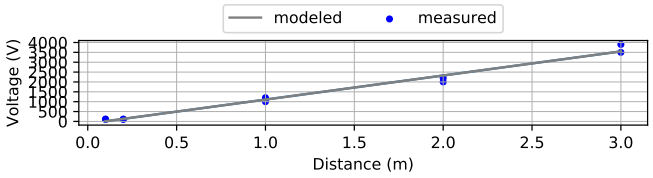
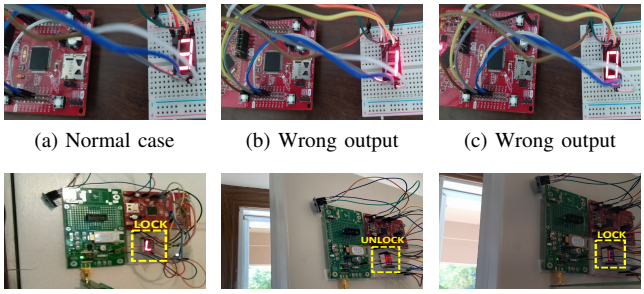


Fig. 4: Caphammer attack model and measurement. We measured the required voltage levels at different distances to launch Caphammer.

We discovered that high voltage square wave signals induce capacitor degradation within 20k capacitor charge cycles. This degradation subsequently leads to the checkpoint failure issue, as discussed in §II-C. In particular, attackers should consider the input voltage level to ensure effective execution of the attack as: $P = \frac{E^2}{377}$, where P is the power density (W/m^2), E is the electric field strength (V/m), and 377 represents the impedance of free space. With the model, they also need to consider possible energy loss along the space path modeled as: free space path loss = $(4 * \pi * r/\lambda)^2$ where r and λ are the distance and the wavelength, respectively. The implication is that if attackers move farther away from the target, they should increase the voltage level to make a stronger EMFI device; we successfully launched Caphammer, intentionally generating the checkpoint signal, 10cm$\sim$3m away from the target device with a small EMFI device as described in Fig. 4.

EHS Sensor with Silent Data Corruption We conducted experiments with EHS applications that collect data from sensors and process the data to generate outputs [60], [62]. The applications make a checkpoint to persist the sensing data in a designated NVM storage when power failure occurs by using the same JIT checkpointing—along with other volatile data—for crash consistency (§II-B). As discussed, since the JIT checkpointing can be failed by Caphammer, the sensor data storage can also be corrupted. In particular, such sensor data corruption is not easily noticeable due to the non-transparent nature of sensor operation; this is being called *silent data corruption* (SDC) [14].

To analyze the SDC problem, we implemented an RF-based sensor node, emulating an EHS glucose monitor. For the sensor node, we attached a temperature sensor to an MSP430FR5994 evaluation board with a one-digit, seven-segment LCD, in a similar way of prior works [62]. In this design, the node initially buffers temperature sensor data. Then, it computes the data and determines whether the temperature exceeds a predetermined threshold. If this is the case, the sensor node should increase the output number and display it on the LCD, alerting users. While the node is operating in this



(a) Normal case (b) Wrong output (c) Wrong output

(d) Door lock (L) (e) Door unlock (U) (f) Wrong output

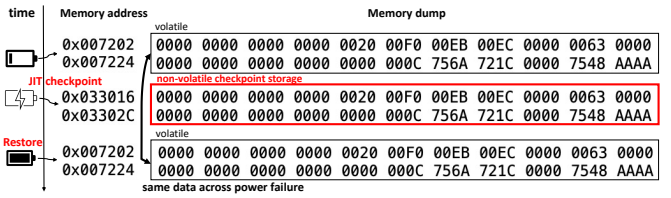
Fig. 5: Silent data corruption in two EHS devices. The displayed character in (b) and (c) should be "3" as in (a). Also, the displayed character in (f) should be "U" when a door is unlocked as in (e).

manner, we launched Caphammer. According to experiment results, the sensor node was susceptible to Caphammer, which caused the checkpoint failure issue. The output number in the LCD sensor should have been increased as shown in Fig. 5(a); however, the data were corrupted, preventing the number from being updated as shown in Fig. 5(b). Even worse, the data was not updated at all when the attack lasted for a long time as shown in Fig. 5(c).

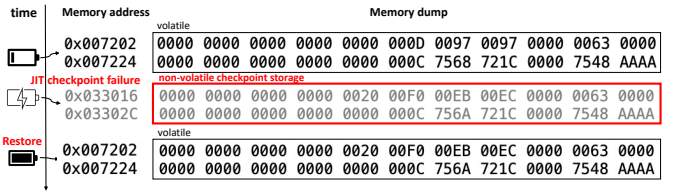
We also implemented an RF-based door sensor node—which is attractive thanks to easy installation without electrical wiring work—by leveraging the specifications of prior work [62]. As such, we mounted an EHS device that was equipped with an one-digit seven-segment LCD and a motion sensor (GY-521/MPU-6050) to a door as shown in Fig. 5(d). In this design, the EHS device repeatedly senses/buffers a position of the sensor module. Then, it processes the buffered data and finally checks whether the door is opened or not (locked or unlocked). If opened, the sensor node is supposed to show "U" (unlocked) on the LCD—as shown in Fig. 5(e)—and in turn alarm the users (if configured). While the door sensor node is operating in this way, attackers attempt to launch Caphammer remotely outside the door.

Our experiments demonstrate that the sensor node is indeed vulnerable to Caphammer, and attackers can launch it successfully. We found that attackers can cause a power outage as soon as the victim EHS detects the door opening; however, the EHS is not able to checkpoint the sensed data in NVM before the impending power failure. When the EHS tries to access the data in the wake of the failure, it ends up reading wrong data, i.e., old status telling the door is locked. As shown in Fig. 5(f), although the door is unlocked, the output character on the LCD remains "L" (locked) under Caphammer, failing to alarm the users. Thus, by leveraging the problem, attackers can finally break into the victim's home.

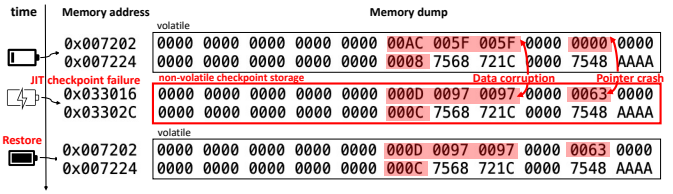
Breaking Over-Voltage Protection Protecting capacitors against malicious charging has been explored at the logic level such as over-voltage protection (OVP) and transient-voltage suppression (TVS). However, it is challenging to thwart Caphammer with conventional OVP/TVS techniques since they only protect MCU, not the capacitor banks; we managed to launch Caphammer in the presence of MSP430's built-in OVP. That being said, to defeat the attack, fusing extra OVP/TVS to each capacitor bank is required, which is not only costly but also power-consuming, thus being overkill for EHS.



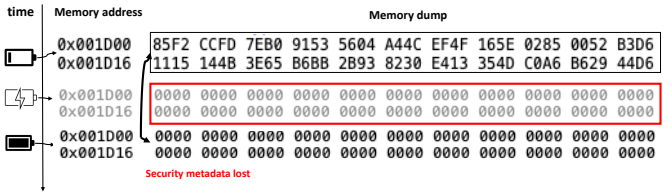
(a) Normal case: All volatile data remain the same across power failure.



(b) DoS problem: Due to checkpoint failure for all volatile states, a processor restarts from a previously checkpointed point across power failure.



(c) Abnormal termination: A processor restores a corrupted pointer value across power failure, then it accesses an invalid memory space out of NVM causing abnormal termination.



(d) Encryption failure: A processor loses security metadata across power failure.

Fig. 6: Security Implications. Shaded boxes represent the data corruption across power failure. In all examples, PC counter is remained the same, which is stored at 0x033038.

E. Proof-of-Concept Study

Prior Works with DoS. To investigate the DoS problem, we tested benchmark applications [22], [35] with our evaluation boards where the JIT checkpointing is enabled by default. As shown in Fig. 6(a), the EHS device must be able to checkpoint volatile data at a designated memory space and restore them across power failure. However, we found that the checkpoint storage remained the same across power failure without updates, when Caphammer is launched, as described in Fig. 6(b). In this figure, the PC value must be 0x7568 across power failure; however, the victim restores the old value, 0x756A, which is a checkpointed PC value at the previous power-off time, i.e., the processor keeps repeating the task from the previous recovery point, leading to the DoS.In particular, we found this problem occurs in all software-based checkpoint/recovery schemes that divide the entire program into a series of recoverable tasks [36], [67].

Prior Works with Abnormal Termination. Caphammer also causes an abnormal termination as shown in Fig. 6(c). We found that the victim triggered the JIT checkpointing at the

Application	Scheme	Type of vulner.	Corrupted data	Location
aes	QuickRecall	a,c,d,e	RF metadata	random (a,c,d), AES_encryptData() in aec.c (c,e)
	Samoyed	a,c,d	RF	random (a,c,d)
door sensor	QuickRecall	a,c,d	RF sensor data	random (a,c,d) RXDATA
	Samoyed	a,c,d	RF	random (a,c,d)
bitcount	QuickRecall	a,c,d	RF	random (a,c,d)
	Samoyed	a,c,d	RF	random (a,c,d)
stringsearch	QuickRecall	a,c,d	RF	random (a,c,d)
	Samoyed	a,c,d	RF	random (a,c,d)
dijkstra	QuickRecall	a,c,d	RF	random (a,c,d)
	Samoyed	a,c,d	RF	random (a,c,d)
crc16	QuickRecall	a,c,d	RF,HWREG, CRCINIRES	random (a,c,d) crc.c (c)
	Samoyed	a,c,d	RF	random (a,c,d)
crc32	QuickRecall	a,c,d	RF,HWREG, CRCINIRES	random (a,c,d) crc.c (c)
	Samoyed	a,c,d	RF	random (a,c,d)
fir	QuickRecall	a,c,d	RF	random (a,c,d)
	Samoyed	a,c,d	RF	random (a,c,d)
dhrystone	QuickRecall	a,c,d	RF	random (a,c,d)
	Samoyed	a,c,d	RF	random (a,c,d)
fft	QuickRecall	a,c,d	RF,HWREG, DSPLIB_DATA	random (a,c,d), msp_cmplx_fft_fixed_q15.c
	Samoyed	a,c,d	RF	random (a,c,d)
basicmath	QuickRecall	a,c,d	RF	random (a,c,d)
	Samoyed	a,c,d	RF	random (a,c,d)
blinker	QuickRecall	a,c,d	RF	random (a,c,d)
	Samoyed	a,c,d	RF	random (a,c,d)

TABLE II: Security vulnerability analysis in QuickRecall and Samoyed. In the third and fifth column, a,c,d, and e represent abnormal termination, data corruption, DoS, and encryption failure, respectively. In the fourth column, RF represents register file.

moment of a power outage. A program counter (0x7568) and a stack pointer (0x721C) were safely updated at 0x033038 and 0x03303A in the designated storage, respectively. However, some data were not checkpointed (marked in red boxes). In particular, even though a buffer index value was originally 0 (located at 0x07214 in volatile memory space) before the power outage, it was incorrectly changed to 0x0063 across the outage by restoring a previously checkpointed value. In other words, the valid value was removed by the incorrect recovery of the power outage which results in wrong value restoration. In this case, the victim resumed the interrupted program from the power outage point, 0x7568, but used the wrong index value ending up causing the resulting pointer to access an illegal address outside of NVM space. Consequently, the victim suffered from an abnormal termination.

Prior Works with Encryption failure. If power failure occurs between encrypted data store and its associated metadata store, the victim would suffer from the data encryption failure in the wake of the power failure. As discussed in §IV-C, prior works implement the custom logic in the memory controller of the EHS device to accelerate the counter-mode encryption/decryption [10], [45] with JIT flushing support that ensures the power-failure-atomic write of both encrypted data and its associated metadata together to NVM. When the JIT flushing failed by Caphammer, the victim could not persist both the ciphertext and its associative metadata correctly. Figure 6(d) demonstrates the encryption failure problem. A ciphertext was encrypted with associated security metadata located at 0x001D00. However, due to the JIT flushing failure, the metadata was lost across power failure.

Vulnerability Report. Table II summarizes security implications caused by Caphammer in prior works [17], [59]. Overall, the hardware-based schemes (Samoyed and QuickRecall) suffer abnormal termination (a), data corruption (c), DoS (d), and encryption failure (e) in every benchmark application [17], [35] that we have tested when Caphammer is launched; any program point is vulnerable when Caphammer is launched (within 20k cycles). In particular, QuickRecall can also corrupt important data, such as security metadata, sensor data, and shared memory in a HW engine in aes, door sensor, crc16, crc32, and fft applications, by Caphammer; we found that a recent work called CatNap [60] has the same vulnerability—though the work is not opened to public.

V. Countermeasure

To defeat Caphammer, this paper introduces FanCap that detects the attack, ensures correct recovery, and provides a quality of service with the energy storage transformation.

Detection of Caphammer. To detect Caphammer, this paper leverages one essential observation, i.e., EHS devices do not boot the microcontroller (MCU) until their capacitor is fully charged [17], [49], [67]. In other words, when the device is ready to resume its program execution in the wake of power failure, the capacitor must always have fully buffered (charged) energy at the resumption point for the failure recovery. The implication is that the system can make as much progress as the full capacitor allows, even if no additional energy is harvested along the way. In particular, we refer to the assured progress time—for which the EHS device can sustain under the fully buffered energy—as the safe power-on period (SP).

Detection Strategy. By the definition of SP, the EHS device must be awake during the assured progress time. To check whether SP is secured at run time, FanCap OS leverages a watchdog timer. FanCap sets the timer interval to the SP and checks whether the timer has been expired within each power-on period. If the timer was not expired during one power-on period at all, e.g., the third power-off period in Fig. 7, FanCap considers the EHS device to be malfunctioning due to the capacitor degradation by Caphammer in that the SP turns out to be violated though it must not be in normal cases. To defeat the attack, FanCap transforms the energy storage organization.

How to Calculate Safe Power-On Period? To measure SP, it is required to know the capacitance since the EHS relies on only the capacitor when there is no harvested energy (§II-A). When the capacitor is the only power source for the device, it is possible to estimate the available energy input as follows: $Available\ Energy\ Input = \frac{1}{2}C_{buf} * (V_{max}^2 - V_{min}^2)$, where C_{buf} is capacitance given by users, V_{max} and V_{min} are the power-on and the power-off voltage level, respectively.

With the estimated available energy, FanCap can measure the SP by leveraging a simple model [20]: $E_{tot} = P_{tot}t = V_{dd}I_{leak}t + C_{msp}V_{dd}^2$, where V_{dd}, I_{leak}, and C_{msp} are input voltage to a microcontroller (MCU), leakage current, and the MCU capacitance, respectively. If any of them is unknown, FanCap could adapt the simple leakage current and capacitance model [66]. Given all this, FanCap finds the SP in a way that the available energy input should always be greater than the energy consumption.

However, the calculated SP may be inaccurate, causing detection errors. Especially, false negatives are problematic since they imply that the capacitor is already degraded, causing security breaches. To prevent such false negatives, FanCap

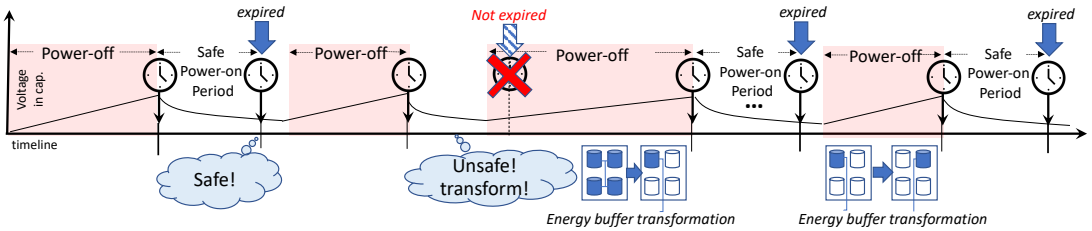


Fig. 7: FanCap overview: FanCap detects Caphammer with a watchdog timer. On detection, FanCap changes its operating mode.

conservatively estimates the SP rather than considering the worst-case power-consuming scenario. In other words, FanCap does not consider additional power consumption from peripherals or the maximum current consumption mode of devices. Thanks to the conservative estimation of SP, it switches to a quarantine mode in a proactive way before any damage occurs; we found out that when SP is violated, there is about 2% capacitance degradation (§VI-C)—that is normally not happened without Caphammer attacks. Due to its proactive detection, FanCap may cause false positives. However, even in the event of false positives, FanCap incurs a trivial performance overhead since it only reorganizes energy storage formation when the capacitors are being degraded, and switches back to the normal mode when they are recovered—though we did not observe false positives in our experiments (§VI).

FanCap is not currently leveraging the worst-case power analysis since it makes SP shorter—for given capacitor size—compared to the average analysis, thereby increasing the possibility of false negatives (checkpoint-failure occurs). FanCap could have used the best-case power analysis to ensure the absence of false negatives. However, this would incur too many false positives due to the huge gap between the resulting SP (bounded by static analysis) and the real best-case (longest) progress time. To this end, FanCap chooses the average time as a middle-ground approach. Nevertheless, FanCap can always detect Caphammer except only one case where the best-case execution progress is made for all intermittent cycles, which is practically impossible.

Why Safe Power-On Period (SP)? Users could notice Caphammer by monitoring the power failure frequency since the attack increases the frequency. However, the frequency-based detection is not only expensive but also inaccurate. First, it requires additional hardware support such as a persistent timer [24] to measure the power failure frequency. Unfortunately, since the persistent timer also leverages the capacitor-based JIT checkpointing for timekeeping (including power-off time), it causes additional capacitor charging time and has the same vulnerability issue. In addition, another type of persistent timer [11], that leverage SRAM remanence decay, cannot accurately measure the power failure frequency against spoofing attacks. When attackers do not supply any power as part of spoofing attacks, the timer can lose all data in SRAM, i.e., the timer cannot measure the power-off time.

More importantly, attackers can easily fool the frequency-based detection by varying the frequency in an arbitrary manner. That is, even if the naive detection successfully once notices Caphammer, attackers can change the frequency at their disposal and eventually launch Caphammer bypassing the detection logic. To address the problem, FanCap leverages the SP-based detection that is accurate without requiring the expensive hardware support.

Energy Storage Transformation FanCap has an unique capability that shields the capacitor (i.e., energy storage) from Caphammer through an energy transformation mechanism. Initially, FanCap establishes the energy storage configuration as a parallel–series switched capacitor circuit, ensuring reliable and energy-efficient storage. This approach contrasts with the use of a single capacitor, which can lead to higher operating voltages and ripple currents [34], i.e., the parallel-series capacitor banks are more reliable than the single capacitor bank against Caphammer. This initial configuration is referred to as the *normal mode* in this paper.

Upon detecting an Caphammer attack (or capacitor degradation), FanCap alerts users and recovers the energy storage from degradation by exploiting its self-recovery characteristic (§II-D). FanCap transforms the capacitor bank organization by connecting only one capacitor to use and disconnecting all the others, while keeping the same capacitance as the normal mode organization [52]—we call this is a *quarantine* mode. When FanCap detects the attacks, FanCap transforms the energy storage to the quarantine mode. Although the circuit seems to be changed, the overall capacitance of the transformed energy storage remains the same. Hence, no matter how many times the energy storage organization is reconfigured, FanCap seldom affects the execution time of program.

In particular, FanCap lets each capacitor bank take turns powering the EHS device by controlling the switches accordingly in the quarantine mode; other unused capacitors wait for a turn in quarantine and get recovered thanks to the resilience nature of a capacitor (§II-D). In this way, although one of the capacitors was under attack in each energy storage transformation, the degraded capacitor can be recovered during the quarantine period, while others are used. An important issue is when to pick the next capacitor from the quarantine to be used in the upcoming power-on period, putting the one that was used/degraded in quarantine. In fact, this capacitor scheduling process should be repeated until all the capacitors are recovered, defeating Caphammer.

FanCap performs the capacitor scheduling in a round-robin (RR) manner; each capacitor gets a single time quantum. When every capacitor finishes its quarantine and defeats Caphammer, FanCap gets back on track by reforming the capacitor banks to their original organization. To make sure the recovery of a degraded capacitor, this paper defines the quarantine period as the time for which the capacitor has not been used since the last use point. Thus, the quarantine period

must be the same as a single time quantum of the RR capacitor scheduling. At a high level, the following shows how FanCap schedules capacitors to defeat the detected Caphammer.

At design time, to ensure a sufficiently long quarantine period, FanCap leverages the recovery model of a capacitor used in the EHS device [43] (§II-D) defined as: $C_{recovery}(t, T, V_{end}) = a * exp(-\frac{t}{\tau_1}) + b * exp(-\frac{t}{-\tau_2})$, where a and b characterize the capacitor state, and τ_1 and τ_2 are the time constants governing the recovery rate of the capacitor. We get these parameters from the device manual, and thus FanCap figures out the time taken for a degraded capacitor to get recovered using the above recovery model.

To see if the current quarantine period finishes, FanCap measures the accumulated power-on time across outages. FanCap checks whether the time is greater than the quarantine period along the way. If so, FanCap signals the energy storage to schedule the next capacitor putting the used one in quarantine. During each time quantum, FanCap ensures SP in every capacitor bank organization. If SP is violated before the time quantum ends, FanCap schedules the next capacitor to be used in the following power-on period as an exceptional case. This process continues until all capacitors are fully recovered. However, if SP cannot be ensured in all organizations, FanCap assumes that the capacitor banks are damaged. Note that repeating the quarantine mode can wear out capacitor switches, which use flash memory. To mitigate this issue, once all capacitors are recovered, FanCap stops rotating the capacitor banks and returns to the original parallel-series switched capacitor circuit organization.

For energy storage transformation, FanCap leverages the JIT checkpointing mechanism with controllable switches. If capacitor scheduling is required, FanCap will turn on/off controllable state-retaining switches that are connected to MCU through GPIO pins [8] at a power-off time. In particular, FanCap turns on/off the capacitor switches after checkpointing data; otherwise, it can fail checkpointing because the energy storage transformation can directly cause a power failure. Although it seems that FanCap requires increasing the checkpoint voltage level to ensure both JIT checkpointing and energy storage transformation, the voltage level adjustment is unnecessary in reality. Since the capacitor switches are controllable with a way lower voltage level than NVM, FanCap can start the energy storage transformation after finishing the data checkpointing by using the residue energy.

Applicability. FanCap is scalable, allowing flexible adaptation to various EHS devices. For large and complex devices, such as multi-core solutions [31], battery-less sensors [7], [58], and mobile gaming devices [47], which utilize multiple capacitor banks for energy storage, FanCap can reconfigure the capacitor bank organization through programmable capacitor switches. For lightweight EHS devices utilizing a single capacitor bank [37], [44], [53], [54], FanCap disconnects the capacitor bank and operates in quarantine mode upon detecting Caphammer. Without relying on the attacked capacitor, FanCap disables the JIT checkpoint mechanism but enables a software-based crash consistency solution, executing a sequence of recoverable tasks. However, the solution may encounter the DoS issue in quarantine mode, where any lengthy task could potentially become stagnated, as discussed in §IV-C. To address this issue, FanCap reconnects the capacitor once it is recovered.

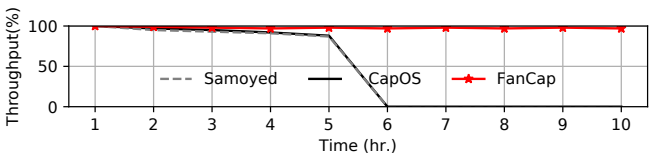


Fig. 8: Attack detection and recovery analysis: 0% throughput represents a denial of service.

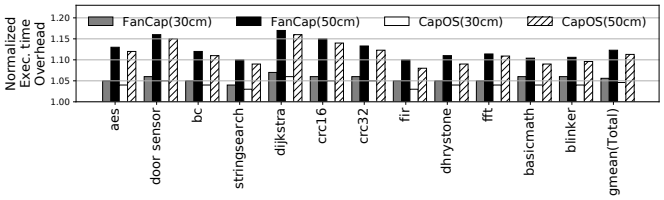


Fig. 9: Normalized execution time overhead of FanCap and CapOS without Caphammer attacks. The baseline is Samoyed.

VI. Evaluation

A. Evaluation Setting

We implemented FanCap on MSP430FR5994 [2]. For security analysis, we compared FanCap with Samoyed [59] and CapOS [22], while launching Caphammer. In particular, CapOS is the state-of-the-art for addressing capacitor degradation by using an acknowledgment (ACK) as a checkpoint barrier. CapOS persists the ACK, treating it as the last register to be checkpointed after saving all registers in NVM [22]. When the ACK becomes corrupted, CapOS identifies the capacitor issue and initiates a software-based crash consistency scheme.

For performance analysis, we used the same set of applications tested in Sec. IV-E [20], [22] and measured the execution time of each application running FanCap, Samoyed, and CapOS. To set up the energy harvesting environment, we employed a Powercast RF generator positioned at distances of 30 cm and 50 cm from the evaluation board equipped with a P2110-EVB RF energy harvester. We used 50mF capacitor as an energy buffer and configured the voltage thresholds in the same manner as described in Sec. IV-D.

B. Security Analysis

We conducted experiments by running benchmarks with FanCap, Samoyed, and CapOS while launching Caphammer; we used the spoofing model for the attack (§IV). While Caphammer was ongoing, we measured the average throughput of each benchmark. Throughput was calculated by counting the number of application completions within one hour: $\frac{\text{\# of completions}}{\text{1 hour}}$. We set the throughput of Samoyed without Caphammer as the baseline. From the experiments, we found that FanCap continued program execution, while Samoyed and CapOS experienced abnormal termination or DoS after 5 hours of Caphammer, across all tested benchmarks, as shown in Fig. 8. In particular, we found that CapOS could not defend

against Caphammer. This is because when the victim failed to trigger the JIT checkpointing under Caphammer, the ACK remained uncorrupted. As a result, CapOS could not detect Caphammer, leading to the DoS.

C. Performance Analysis

Figure 9 describes the normalized run-time overhead of FanCap and CapOS, compared to the baseline. FanCap causes an execution time overhead of about 6% and 13%, while CapOS causes about 5% and 11% on average, compared to Samoyed, when the energy source is placed at 30 cm and 50 cm, respectively. The figure shows that when an energy source is far away from the harvesting board, FanCap results in more performance overhead. This is because the overhead caused by its OS module increases when the harvested power becomes poor and causes more frequent power failure; FanCap did not encounter any false positives in all the applications we tested. We also measured the energy-delay product (EDP) of FanCap and Samoyed. We found that FanCap caused about 12∼37% overhead compared to Samoyed.

D. Sensitivity Analysis

For sensitivity analysis, we also tested Samoyed and FanCap with a different checkpoint voltage threshold. We set the threshold as 2.2V for Samoyed—that is way higher than the original margin (i.e., 2.00003V as stated in §IV-C). Due to the high margin, Samoyed caused >2 slowdown compared to FanCap, but it took more than 5 days to launch Caphammer in Samoyed, while FanCap successfully defeated Caphammer. We tested them on another type of evaluation board, STM32L series [4], with Cortex-M4 and Cortex-M33 processors [1], [5]. We found that FanCap successfully defended against Caphammer and incurred an execution time overhead of approximately 10% compared to the baseline, whereas other solutions were vulnerable to Caphammer.

Discussion. Although we used electrostatic double-layer supercapacitors for our experiments, we believe FanCap can work on different types of supercapacitors, such as electrostatic double-layer capacitors, pseudo-capacitors, and hybrid capacitors. This is because all of them are susceptible to over-voltages and charging/discharging [46], which are the primary attack surfaces exploited by Caphammer. Also, we discovered that different types of capacitors may have different lifespans under Caphammer. Based on this finding, we plan to develop a new energy storage architecture with various types of capacitors; we leave it as our future work.

VII. Other Related Works

To address the capacitor aging/degradation problem, prior works proposed new capacitor materials and novel sizing/-packaging solutions. Recently, *Pamete et al.* characterized the performance degradation of supercapacitors, studying electrode degradation and electrolyte decomposition. Based on their experiments, they suggested using new materials, such as ionic liquids and solid-state electrolytes, to achieve better supercapacitor performance and longevity [15]. On the other hand, *Chen et al.* proposed methods to reduce operating power fluctuations and optimize capacitor pack sizes. Furthermore, they also found that balancing multi-cell capacitor banks can reduce the capacitor aging and degradation rate [48]. Despite these efforts to minimize degradation, Caphammer, to the best of our knowledge, can eventually damage the capacitors. FanCap is the first countermeasure designed to defeat Caphammer.

VIII. Summary

This paper discovers that energy harvesting systems are vulnerable to Caphammer i.e., a capacitor hammering attack, degrading the capacitance readily and stealthily. To defeat Caphammer, this paper introduces FanCap, that can detect the attack and prevent it by transforming the capacitor banks. Our experimental results demonstrate that FanCap can successfully thwart Caphammer with an average slowdown of 6%.

Acknowledgement

The authors thank Purdue and UCF CompArch members as well as the anonymous reviewers for their valuable comments. This work is in part supported by the NSF grants CNS-2314680, CNS-2314681, and CCF-2153749.

References

[1] Arm cortex-m4 32b mcu+fpu, up to 256kb flash+32kb sram, timers, 4 adcs (12/16-bit), 3 dacs, 2 comp., 1.8 v operation. http://www.mouser.com/catalog/specsheets/stmicroelectronics_dm00058405.pdf, Sep 2013.
[2] Msp430fr5994launchpad development kit, Mar 2016.
[3] bq25570 nano power boost charger and buck converter for energy harvester powered applications, March 2019.
[4] Stm32l series ultra-low-power. http:/https://www.st.com/resource/en/brochure/brstm32ulp.pdf, July 2022. Accessed: 2024-05-28.
[5] Arm cortex-m33 in a nutshell, Sep 2023. Accessed: 2023-11-08.
[6] Erin Digitale. Technology equality gap for kids' diabetes treatment is growing, 2021.
[7] A. Alsubhi et al. Stash: Flexible energy storage for intermittent sensors. *ACM Transactions on Embedded Computing Systems*, 2024.
[8] A. Colin et al. A reconfigurable energy storage architecture for energy-harvesting devices. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 767–781. ACM, 2018.
[9] A. Curtiss et al. Facebit: Smart face masks platform. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(4):1–44, 2021.
[10] A. Krishnan et al. Secure intermittent computing protocol: Protecting state across power loss. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 734–739. IEEE, 2019.
[11] A. Rahmati et al. {TARDIS}: Time and remanence decay in {SRAM} to implement secure protocols on embedded devices without clocks. In *21st {USENIX} Security Symposium)*, pages 221–236, 2012.
[12] A. Sample et al. Design of an rfid-based battery-free programmable sensing platform. *IEEE transactions on instrumentation and measurement*, 57(11):2608–2615, 2008.
[13] C. Kulkarni et al. Prognostic techniques for capacitor degradation and health monitoring. In *The Maintenance& Reliability Conference*, 2011.
[14] D. Fiala et al. Detection and correction of silent data corruption for large-scale high-performance computing. In *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pages 1–12. IEEE, 2012.
[15] E. Pameté et al. The many deaths of supercapacitors: degradation, aging, and performance fading. *Advanced Energy Materials*, 13:2301008, 2023.
[16] F. Zheng et al. Study on effects of applied current and voltage on the ageing of supercapacitors. *Electrochimica Acta*, 276:343–351, 2018.
[17] H. Jayakumar et al. Quickrecall: A low overhead hw/sw approach for enabling computations across power cycles in transiently powered computers. In *VLSI Design and 2014 13th International Conference on Embedded Systems*, pages 330–335. IEEE, 2014.

[18] H. Kim et al. Compiler-directed soft error resilience for lightweight gpu register file protection. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 989–1004, 2020.

[19] J. Choi et al. Compiler-directed high-performance intermittent computation with power failure immunity. In *2022 IEEE 28th Real-Time and Embedded Technology and Applications Symposium*, pages 40–54.

[20] J. Choi et al. Achieving stagnation-free intermittent computation with boundary-free adaptive execution. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 331–344, 2019.

[21] J. Choi et al. Cospec: Compiler directed speculative intermittent computation. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 399–412. ACM, 2019.

[22] J. Choi et al. Capos: Capacitor error resilience for energy harvesting systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(11):4539–4550, 2022.

[23] J. Choi et al. Write-light cache for energy harvesting systems. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pages 1–13, 2023.

[24] J. Hester et al. Persistent clocks for batteryless sensing devices. *ACM Transactions on Embedded Computing Systems*, 15(4):1–28, 2016.

[25] J. Jeong et al. Capri: Compiler and architecture support for whole-system persistence. In *the 31st International Symposium on High-Performance Parallel and Distributed Computing*, pages 71–83, 2022.

[26] J. Zeng et al. Replaycache: Enabling volatile cachesfor energy harvesting systems. In *54th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 170–182, 2021.

[27] J. Zeng et al. Turnpike: Lightweight soft error resilience for in-order cores. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 654–666, 2021.

[28] J. Zeng et al. Persistent processor architecture. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 1075–1091, 2023.

[29] J. Zeng et al. Compiler-directed whole-system persistence. In *Proceedings of the 51th Annual International Symposium on Computer Architecture*, 2024.

[30] J. Zeng et al. Soft error resilience at near-zero cost. In *Proceedings of the 38th ACM International Conference on Supercomputing*, pages 176–187, 2024.

[31] K. Akhunov et al. Adamica: Adaptive multicore intermittent computing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(3):1–30, 2022.

[32] K. Ma et al. Architecture exploration for ambient energy harvesting nonvolatile processors. In *Proceedings of 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, HPCA '15, pages 526–537, Piscataway, NJ, USA, 2015. IEEE Press.

[33] K. Yildirim et al. Ink: Reactive kernel for tiny batteryless sensors. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, pages 41–53, 2018.

[34] M. Chen et al. Stacked switched capacitor energy buffer architecture. *IEEE Transactions on Power Electronics*, 28(11):5183–5195, 2013.

[35] M. Guthaus et al. Mibench: A free, commercially representative embedded benchmark suite. In *Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on*, pages 3–14. IEEE, 2001.

[36] M. Kiwan et al. Alpaca: Intermittent execution without checkpoints. In *Proceedings of the 2016 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications*. ACM, 2017.

[37] M. Monjur et al. Soundsieve: Seconds-long audio event recognition on intermittently-powered systems. In *Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services*, pages 28–41, 2023.

[38] Q. Jiang et al. Glitchhiker: Uncovering vulnerabilities of image signal transmission with iemi. In *USENIX Security*, volume 23, 2023.

[39] Q. Liu et al. Clover: Compiler directed lightweight soft error resilience. *ACM Sigplan Notices*, 50(5):1–10, 2015.

[40] Q. Liu et al. Compiler-directed lightweight checkpointing for fine-grained guaranteed soft error recovery. In *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 228–239. IEEE, 2016.

[41] Q. Liu et al. Low-cost soft error resilience with unified data verification and fine-grained recovery for acoustic sensor based detection. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–12. IEEE, 2016.

[42] Q. Liu et al. ido: Compiler-directed failure atomicity for nonvolatile memory. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 258–270. IEEE, 2018.

[43] R. Chaari et al. Capacitance recovery analysis and modelling of supercapacitors during cycling ageing tests. 82:37–45, 2014.

[44] S. Lee et al. Intermittent learning: On-device machine learning on intermittently powered system. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(4):1–30, 2019.

[45] S. Liu et al. Janus: Optimizing memory and storage support for non-volatile memory systems. In *2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture*, pages 143–156. IEEE, 2019.

[46] S. Liu et al. Review on reliability of supercapacitors in energy storage applications. *Applied Energy*, 2020.

[47] V. Kortbeek et al. Bfree: Enabling battery-free sensor prototyping with python. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(4):1–39, 2020.

[48] X. Chen et al. Aging and degradation of supercapacitors: Causes, mechanisms, models and countermeasures. *Molecules*, 28:5028, 2023.

[49] Y. Wang et al. A 3us wake-up time nonvolatile processor based on ferroelectric flip-flops. In *ESSCIRC, 2012 Proceedings of the*, pages 149–152. IEEE, 2012.

[50] Y. Zhou et al. Sweepcache: Intermittence-aware cache on the cheap. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 1059–1074, 2023.

[51] Z. Xiao et al. An implantable rfid sensor tag toward continuous glucose monitoring. *IEEE journal of biomedical and health informatics*, 2015.

[52] S. Franco and J. Kang. *Electric circuits fundamentals*. Oxford University Press, 1995.

[53] B. Islam and S. Nirjon. Scheduling computational and energy harvesting tasks in deadline-aware intermittent systems. In *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 95–109. IEEE, 2020.

[54] B. Islam and S. Nirjon. Zygarde: Time-sensitive on-device deep inference and adaptation on intermiently-powered systems. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT/UBICOMP'20)*, 4(3):1–29, 2020.

[55] J. Jeong and C. Jung. Pmem-spec: persistent memory speculation (strict persistency can trump relaxed persistency). In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 517–529, 2021.

[56] C. Kulkarni. *A physics-based degradation modeling framework for diagnostic and prognostic studies in electrolytic capacitors*. Vanderbilt University, 2013.

[57] Q. Liu and C. Jung. Lightweight hardware support for transparent consistency-aware checkpointing in intermittent energy-harvesting systems. In *2016 5th Non-Volatile Memory Systems and Applications Symposium (NVMSA)*, pages 1–6. IEEE, 2016.

[58] Y. Luo and S. Nirjon. Smarton: Just-in-time active event detection on energy harvesting systems. In *2021 17th International Conference on Distributed Computing in Sensor Systems*, pages 35–44. IEEE, 2021.

[59] K. Maeng and B. Lucia. Supporting peripherals in intermittent systems with just-in-time checkpoints. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 1101–1116. ACM, 2019.

[60] K. Maeng and B. Lucia. Adaptive low-overhead scheduling for periodic and reactive intermittent execution. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 1005–1021, 2020.

[61] H. Markus and A. Gabino. Energy storage using supercapacitors: How big is big enough? *AnalogDialogue*, 54(3), 2020.

[62] H. Mendis and P. Hsiu. Accumulative display updating for intermittent systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 18(5s):1–22, 2019.

[63] MICROCHIP. Xlp 16-bit development board – for low power pic mcu prototyping.

[64] Powercast. Lifetime power lifetime power energy harvesting energy harvesting development kit for wireless sensors, 2017.

[65] D Pritchard. Wearable energy harvesting for charging portable electronic devices by walking. 2020.

[66] A. Sinha and A. Chandrakasan. Jouletrack-a web based tool for software energy profiling. In *In Proceedings of the 38nd Annual Design Automation Conference*, DAC '01, 2001.

[67] Joel Van Der Woude and Matthew Hicks. Intermittent computation without hardware support or programmer intervention. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 17–32, Savannah, GA, 2016. USENIX Association.

[68] Y. Zhang and C. Jung. Featherweight soft error resilience for gpus. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 245–262. IEEE, 2022.