Defending Against EMI Attacks on Just-In-Time Checkpoint for Resilient Intermittent Systems

Jaeseok Choi
University of Central Florida
Orlando, FL, USA
jaeseok.choi@ucf.edu

Hyunwoo Joe
ETRI
Daejeon, South Korea
hwjoe@etri.re.kr

Changhee Jung
Purdue University
West Lafayette, IN, USA
chjung@purdue.edu

Jongouk Choi
University of Central Florida
Orlando, FL, USA
jongouk.choi@ucf.edu

***Abstract*—Energy harvesting systems have emerged as an alternative to battery-powered IoT devices. The systems utilize a just-in-time checkpoint protocol that stores volatile states when a power outage occurs, ensuring crash consistency. However, this paper uncovers a new security vulnerability in the checkpoint protocol, revealing its susceptibility to electromagnetic interference (EMI). If exploited, adversaries could cause denial of service or data corruption in victim devices. To defeat EMI attacks, this paper introduces GECKO, a compiler-directed countermeasure that operates on commodity platforms used in energy harvesting systems without requiring hardware support. Our experiments on real boards demonstrate that GECKO defeats the EMI attack with a trivial performance overhead by 6% on average.**

I. Introduction

The current landscape of the Internet of Things (IoT) is increasingly reliant on energy-efficient technologies, including limited battery lifespan, environmental impact during production, and disposal issues after periodic replacements. In response to these emerging problems, energy harvesting technology has been extensively studied and applied as an alternative to battery-powered IoT devices [14], [15], [30], [32], [35]–[37], [43], [46], [70], [93].

However, energy harvesting sources are inherently unreliable, leading to frequent power outages. To mitigate this issue, an energy harvesting system employs a capacitor as energy storage, buffering harvested energy until a sufficient amount is accumulated to run program; this is also called *intermittent system* [55], [67]. To ensure crash consistency across power failures, these systems utilize non-volatile memory (NVM) and employ some form of recovery mechanisms that checkpoint and restore volatile data, i.e., registers, across power outages.

The majority of prior works employ a just-in-time (JIT) checkpoint protocol that gives an illusion that volatile registers are non-volatile, allowing them to be persistent across power outages [11], [14], [67], [68], [101]. To achieve this, the prior works leverage an energy buffer and a voltage monitor. For example, when the system is powering off, they detect the power outage using a voltage monitor and checkpoint their volatile registers into designated NVM space by utilizing the energy buffered in a capacitor. On the other hand, when their capacitor is fully charged, they also detect it with the help of the voltage monitor and restore the checkpointed registers [11], [14], [67], [68]; this technology has been implemented in a processor known as nonvolatile processor (NVP) [14], [38], [58]–[64], [80], [83], [96], [98], [101]. The takeaway is that the voltage monitor is at the heart of intermittent systems because it is an essential part of detecting a power outage and checkpointing/restoring volatile states.[1]

Unfortunately, a voltage monitor is not always reliable [89] due to cold-start glitch [79], various loads [74], clock noise [3], [4], and electromagnetic interference (EMI) [29], [73], [92]. In particular, this paper reveals that adversaries can exploit EMI to remotely inject malicious signals into the voltage monitor of intermittent systems. By manipulating the output of the voltage monitor, attackers can repeatedly trigger false checkpoint signals, leading to a denial of service (DoS). More importantly, adversaries can induce checkpoint and recovery signals even when the energy buffer (capacitor) is not fully charged. In such cases, victim systems fail to checkpoint all volatile registers, resulting in data corruption and compromising the non-volatility of NVPs.

In this paper, we investigate the vulnerability of a voltage monitor in intermittent systems by injecting EMI signals into commodity platforms, such as TI-MSP430 [3], [4] or STM ARM Cortex-M [2], [7]), sold and used today as intermittent systems. This paper shows that a stabilized checkpoint signal can be intentionally induced and controlled causing the DoS and data corruption. Our experiments, which include both direct power injection (DPI) and remote signal injection, reveal that all nine platforms we analyzed are vulnerable to EMI attacks. To address the vulnerability, this paper also examines possible countermeasures [28], [44], [77], [84], [85], [90], [100]. However, we found that the solutions are not only costly but also impractical for power-hungry intermittent systems since they often require additional hardware supports and cause a significant performance overhead [28], [44], [77], [84], [85], [90], [100]. Therefore, there is a compelling need for a lightweight countermeasure that can thwart EMI attacks.

To this end, we introduce GECKO, a novel compiler-directed countermeasure for intermittent systems to defeat against EMI attacks. The key insight behind GECKO is that the system can thwart EMI attacks if it closes the attack vector (i.e., JIT checkpoint protocol). In other words, by eliminating the attack surface, adversaries can no longer manipulate the

[1]The mechanism is similar to Intel's Asynchronous DRAM Refresh (ADR), where all pending writes are flushed to NVM during a power loss by utilizing the buffered energy stored in a supercapacitor or battery [76]

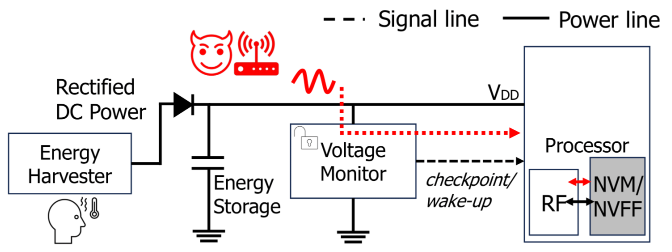


Fig. 1: High-level organization of intermittent system: It rectifies harvested energy, stores the energy in a capacitor, and powers a processor via a power line.

voltage monitor to trigger false checkpoint/recovery signals, which would otherwise result in DoS or data corruption.

However, GECKO may face a new challenge in achieving crash consistency, as the system must ensure that all volatile states are preserved across power outages without utilizing the protocol [11]–[13], [34], [39]–[41], [47]–[50], [54], [95], [97], [99]. To achieve correct power failure recovery without the JIT checkpoint protocol, GECKO leverages another type of power failure recovery solution, i.e., rollback recovery. GECKO compiler forms a series of idempotent regions [51], [53], ensuring that each region produces the same output across power outages. Thanks to the nature of idempotency, regardless of when a power failure occurs, GECKO ensures the program remains intact and recoverable.

The rollback recovery mechanism requires inserting a number of checkpoint stores within a given program region or task, inherently slowing down the performance (e.g., 50%∼400% slowdown [65]). To address the performance degradation issue, GECKO leverages a novel checkpoint pruning technique that can remove a majority of checkpoint stores in each program region without compromising the recovery guarantee [42], [52]. We found that checkpoint stores can be removed if they can be reconstructed when necessary. With that in mind, GECKO rather reconstructs checkpoint stores only when the systems are under attack, transiting the run-time overhead of checkpoint stores to the EMI attack recovery overhead, which is trivial in use cases (i.e. no attack). Furthermore, once the EMI attack is defeated, GECKO re-enables the JIT checkpoint protocol for better performance.

Through our experiments, we demonstrate that GECKO defeats EMI attacks with only an average 6% runtime overhead, achieving about an 80% reduction in checkpoint stores. The contributions of this paper can be summarized as follows:

- We demonstrate that EMI attacks cause critical security implications such as denial of service and data corruption in intermittent systems.
- We propose a compiler-directed solution called GECKO that can defeat the EMI attacks by exploiting idempotent processing with checkpoint pruning.
- Our real board experiments highlight the effectiveness and the practicality of GECKO. It incurs only a 6% runtime overhead on average.

II. Background and Motivation

A. *Intermittent System Architecture*

Intermittent systems capture ambient energy and store it in a capacitor as an energy buffer, utilizing this energy for computation without relying on a battery. However, the distinctive battery-less design makes them prone to frequent power failure. As a result, it becomes crucial for energy harvesting systems to ensure *crash consistency*, a mechanism that allows them to resume programs after frequent power outages [8], [14], [33], [86], [94].

To achieve correct power failure recovery, prior works have introduced a checkpoint-and-restore-based crash consistency solution, referred JIT checkpointing. The insight is that both the register and memory states at the resumption point are guaranteed to be the same as the states before power failure, and thus there is no crash consistency problem, achieving roll-forward recovery [38], [67], [86], [94]. For roll-forward recovery, the systems monitor the voltage level of the energy storage (i.e., a capacitor), using a voltage monitor. They then perform checkpoints of all volatile states to nonvolatile checkpoint storage just before a power interruption is anticipated. When power comes back, the systems restore the previously checkpointed states, enabling the interrupted program to resume. This approach has been adopted in commodity microcontrollers (MCUs) such as TI-MSP430 [3], [4], [6], [12], [38], [67].

B. *Roll-Forward Recovery: Just-In-Time Checkpoint*

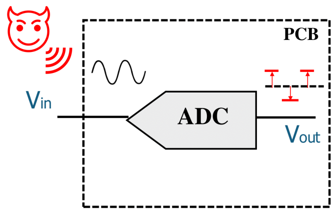
Intermittent systems primarily use NVM as their main memory without a cache. They provide hardware support for JIT checkpointing of volatile registers, ensuring the persistence of the entire system [69]. When the operating voltage of the system falls below a specific threshold (V_{backup}), it checkpoints the states of all registers into NVM. Upon the restoration of sufficient power (V_{on}), these saved register states are restored from the NVM, allowing the previously disrupted program to continue. It is assumed that the voltage monitor is reliable, and the voltage monitor is at the heart of intermittent systems because it is an essential part for detecting a power outage and checkpointing/restoring volatile states [38], [58]–[64], [67], [68], [80], [83].[2]

C. *Voltage Monitor*

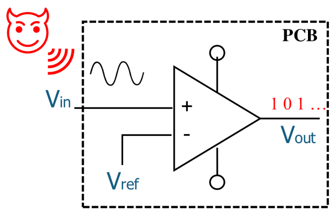
Typically, there are two types of voltage monitors used in intermittent systems: ADC- and comparator-based monitors.

Using ADC to Detect Power Loss. The majority of MCUs for intermittent systems typically feature a 12-bit or 10-bit analog-to-digital converter (ADC) peripheral, which is utilized to monitor the input voltage [3], [4], [38]. This input voltage is supplied by an energy harvester, as shown in Figure 1. In this setup, the input voltage, referred to as V_{CC}, is routed to the voltage monitor's ADC, as illustrated in Figure 2(a).

[2]One of the representative intermittent systems, NVP, use a hybrid register file (HRF) circuitry comprising standard flip-flops and NVFFs, allowing registers to be checkpointed into NVFF instead of NVM [63], [64], [80].



(a) ADC-based Voltage Monitor



(b) Comparator-based Voltage Monitor

Fig. 2: EMI attacks on voltage monitors

Subsequently, this input voltage is continually compared to a fixed reference voltage (V_{ref}) to detect any power loss within the system, indicated by the condition $V_{CC} < V_{ref}$. The reference voltage is usually generated by an on-chip reference module, which is supported by an internal capacitor or a battery. When a power loss is detected, and JIT checkpointing is required, the ADC monitor triggers a checkpoint function. This function is responsible for preserving the volatile states and ensuring correct power failure recovery.

Using Voltage Comparator to Detect Power Loss. The voltage monitor can also be constructed using a comparator in conjunction with a voltage reference to detect power loss [14], [86], [94]. As shown in Figure 2(b), the input voltage supply, V_{CC}, is fed to an external voltage comparator, and the configuration of the comparator involves setting the reference voltage, V_{ref}. The comparator compares the voltage signals at the + and - input terminals. If the + terminal is more positive than the - terminal, the comparator output is high. The comparator is configured to trigger an interrupt if the harvested voltage signal falls below the reference voltage, V_{ref}, i.e., an interrupt is triggered if V_{CC} falls below the threshold. Upon power loss detection, the interrupt service routine disables the voltage monitor and invokes a checkpoint and a shutdown function, which saves the volatile state information and turns off the device. On the other hand, when the input voltage is higher than a reboot voltage level (another reference voltage, i.e., V_{reboot}), the voltage monitor sends a wake-up signal and enables a checkpoint trigger, monitoring V_{CC}. Note that it provides a basic bridge between analog and digital domains and acts as a 1-bit ADC.

Real Board Implementation. For JIT checkpointing, Texas Instruments provides an open-source library called Compute Through Power Loss (CTPL) with low-power platforms [18]. CTPL enables a processor to checkpoint its volatile states and peripheral values in designated NVM space during a power outage. It is compatible with both ADC-based and comparator-based voltage monitors. The CTPL library allows control over the JIT checkpointing voltage threshold and sleep time; hereafter, this paper considers CTPL a type of NVP.

D. Electromagnetic Interference Attack

In recent years, prior studies have highlighted the consequences of EMI attacks on low-power sensor devices [28], [44], [77], [84], [85], [90], [100]. These studies demonstrate that attackers can remotely inject malicious signals into sensor devices, manipulating sensing data and posing significant threats to user privacy and safety. In particular, they found a low-power sensor device is more vulnerable to EMI attacks than general-purpose computers [84]. This occurs primarily because low-power sensor devices often lack effective noise filtering circuits, allowing adversaries to tune the transmitted EMI signal to the carrier frequency that matches the resonant frequency of the target sensor component, resulting in controlled sensing data. To this end, prior works introduce countermeasures such as low-power noise filters, secure circuitry, dual sensors, and data randomization (will be discussed in Section V-A).

With that in mind, we analyze a vulnerability in a voltage monitor and elaborate on how an attacker can remotely inject a malicious signal into intermittent systems. As described in Figure 1, the intermittent system is equipped with a voltage monitor and a capacitor as an energy buffer. The voltage monitor respectively sends a wake-up and a checkpoint signal when the energy buffer is fully charged and when a power outage is about to occur. In this design, this paper exposes that an attacker can intentionally generate the wake-up and checkpoint signals by injecting EMI signals into the voltage monitor when the victim device is active. Once the malicious signal enters, it is superimposed and digitized by an ADC as shown in Figure 2(a) or a voltage comparator as shown in Figure 2(b). Finally, an incorrect checkpoint or wake-up signal is induced in a processor, which can cause repeated power outages (i.e., DoS) or data corruption.

Types of EMI Attacks. EMI attacks can be categorized into two types: high-power EMI attacks and low-power EMI attacks. High-power EMI attacks involve disruptions, frying, and damaging the victim system [75]. Such a high-power EMI tool can lead to degradation or complete loss of the system's main functions, resulting in technical issues. Prior works have extensively proposed defense mechanisms against high-power EMI attacks [72], [75]. In this paper, we focus on low-power EMI attacks on intermittent systems, where the attacker manipulates the voltage monitor to cause false checkpoint and wake-up signals.

III. Threat Model

This paper assumes that adversaries aim to deceive the voltage monitor, causing repeated triggering of checkpoints in the victim or corrupting the JIT checkpoint protocol, resulting in checkpoint failure. We also assume that adversaries cannot physically access the hardware or software of the target system. Furthermore, this scenario excludes considerations of a malicious human operator who could directly influence or attack actual energy sources around the victim or manipulate its input power, including voltage thresholds.

Applications. Intermittent systems are typically used for applications in various domains such as wireless sensor nodes, implantable medical devices, highway toll cards, nano-satellites, and wearables [24], [26], [31], [56], [71], [78], [81], [82]. These devices harvest ambient energy and run an infinite loop that continually senses and triggers alarms in response to detected anomalies. For instance, wearable medical devices like continuous glucose monitors, designed for diabetic patients [25], [27], [88], exemplify this concept. These devices harvest energy from blood pressure and monitor various health parameters, including temperature and blood sugar/glucose. Notably, this monitoring occurs without the need for blood sampling, allowing continuous surveillance even during sleep or daily activities.

Weak Input Power. The actual harvested power is very weak (0∼2000μW [64]). In such a challenging environment, intermittent systems can function for brief periods (e.g., 15ms [65]), rapidly exhausting the buffered energy, before entering a long hibernation phase (e.g., more than 1s) [65].

Attack Scenario. Adversaries could launch an attack from one to several meters, depending on the power of their attack device and the susceptibility of the victim system. Malicious EMI signals can penetrate typical physical barriers, such as walls and windows, enabling the attack to be executed even from adjacent rooms [84]. Moreover, adversaries might sneakily leave or install a small remote control EMI-emitting device around the victim. During the attack, two parameters (frequency and amplitude of EMI signal) need to be adjusted.

Attack Device. Attackers can exploit various methods to generate and emit malicious EMI signals. They may opt for available signal generators, amplifiers, and antennas for attacks. Alternatively, adversaries might choose to procure or construct a specialized, compact, portable transmitter. For a proof-of-concept study, we used a RF signal generator with an antenna, equipped with gain control and a frequency range encompassing the desired attack frequencies. While the power of the EMI emitters utilized in our experiments is below 35 dBm, more sophisticated adversaries could leverage specialized equipment and techniques to enhance the effectiveness of their attacks [28], [44], [77], [84], [85], [90], [100].

Assumptions. To ensure the effectiveness of EMI attacks, attackers need to determine the specific frequency range that will be effective. This can be accomplished in two ways: through prior testing, or visual feedback. For prior testing, we assume that attackers can obtain an identical device to the target and experimentally determine the effective attack frequencies beforehand [44], [84], [91]. For visual feedback, we assume that attackers can visually monitor the target device's status to identify the effective frequency range. Once the attacker has established the appropriate frequency and power levels for the signals, ongoing feedback from the victim device is no longer necessary.

IV. Real System Demonstration

In this section, we analyze the vulnerability and the attack surface of intermittent systems by conducting DPI and remote EMI injection experiments on typical voltage monitors in commodity platforms. To generate EMI signals, we used an Agilent E4437B and a N5171B EXG X-Series RF Analog signal generators with a ZHL-4240 amplifier. For remote attacks, we used a directional antenna [5] with the generators.

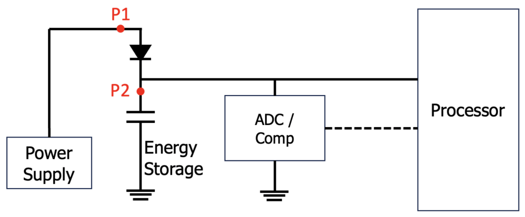


Fig. 3: The setup of direct power injections through different injection points (P1 and P2) in a typical intermittent system. In this design, the signal injection circuit is connected to the injection points P1 and P2.

A. Direct Power Injection Experiments

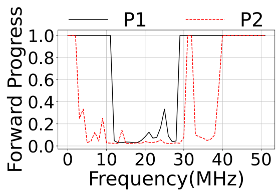
1) Experimental Settings: To measure and analyze the impact of remote EMI attacks on intermittent systems is challenging, given the unpredictable nature of induced EMI signals' path and strength. To address the challenge, we conducted direct power injection (DPI) experiments. DPI involves directly injecting EMI signals into specific points on the circuit. This method offers precise control over the power of the injected signals and helps prevent interference from unintentional EMI radiation on other parts of the circuit.

In our experiments, a DPI circuit was connected to each potential signal injection point on a target system. The setup, described in Figure 3, includes a capacitor as an energy storage for the intermittent system, powered by a +3.3V DC power source. Note that this environment is more resistant to EMI attacks than use cases where the system is powered by ambient energy sources, which typically provide much weaker input power (refer to Section III). In other words, if the DPI experiments are successful, it indicates that adversaries could easily launch EMI attacks on intermittent systems in real-world scenarios.

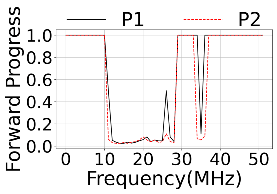
To induce DoS and data corruption, we utilized specific EMI signals capable of causing controllable checkpoint and recovery signals. Single-tone sine-wave EMI signals were injected into each injection point (P1 and P2) in the experimental circuitry. The frequency of the signals was swept from 1MHz to 1GHz, with a consistent power injection level of 20 dBm.

2) Attack Analysis: To see the impact of EMI signals, we measured the forward progress rate of current commodity intermittent systems under EMI attacks, i.e., how much progress each system can make under the attack. The rate was measured as follows: $R = T_{forward}/T_{guarantee}$, where R is a rate, $T_{forward}$ is the amount of time during which the system makes forward progress from the point where the attack is launched, and $T_{guarantee}$ is the guaranteed period during which the system can sustain.

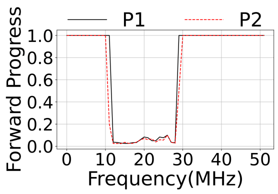
The results, as depicted in Figure 4, reveal that EMI signals at particular frequencies induce checkpoint signals, thereby causing the DoS. Depending on the frequency of the injected EMI signals, the checkpoint signals can be induced either frequently or rarely, enabling adversaries to manipulate the level of quality of service; we also found high frequencies (above 50MHz) did not cause any problem in all systems we



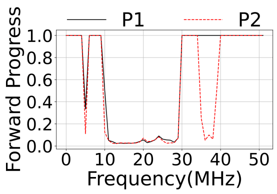
(a) MSP430F5529



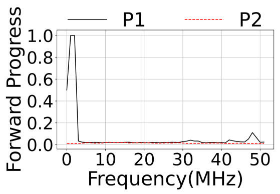
(b) MSP430FR2311



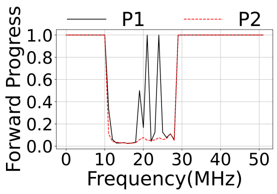
(c) MSP430FR2433



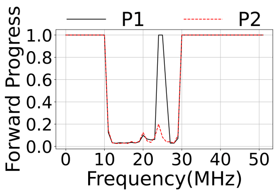
(d) /MSP430FR4133



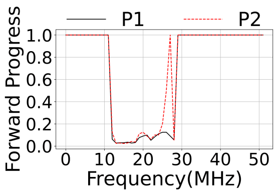
(e) MSP430FR5739



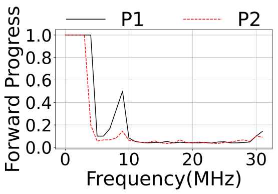
(f) MSP430FR5994



(g) MSP430FR6989

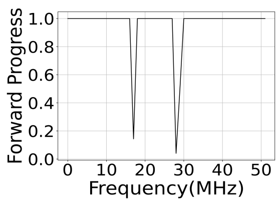


(h) MSP432P401R

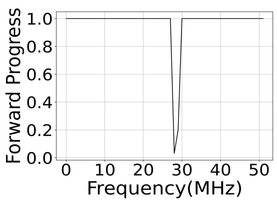


(i) STM32L552ZE

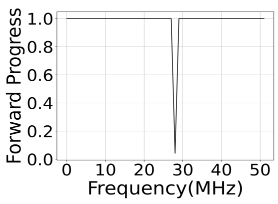
Fig. 4: DPI attack analysis on commodity micro-controllers with an ADC-based voltage monitor varying attack signal frequencies. The X-axis represents the attack signal frequency, while the Y-axis represents the forward progress rate.



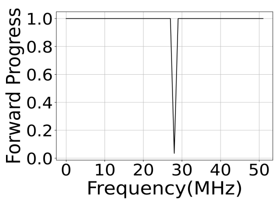
(a) MSP430F5529



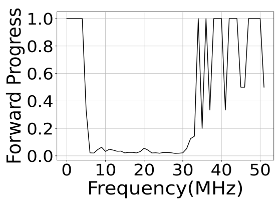
(b) MSP430FR2311



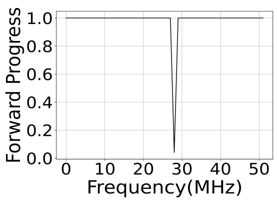
(c) MSP430FR2433



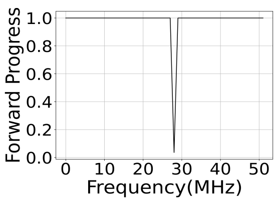
(d) /MSP430FR4133



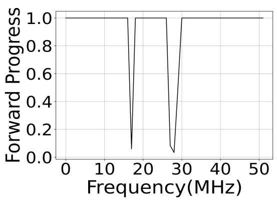
(e) MSP430FR5739



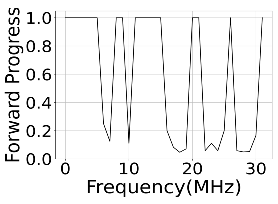
(f) MSP430FR5994



(g) MSP430FR6989



(h) MSP432P401R



(i) STM32L552ZE

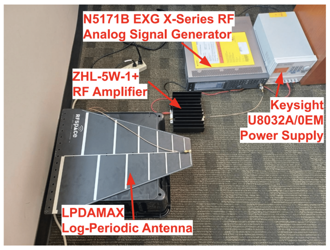
Fig. 5: Remote attack analysis on commodity micro-controllers with an ADC-based voltage monitor varying attack signal frequencies with 20dBm. The X-axis represents the attack signal frequency, while the Y-axis represents the forward progress rate.

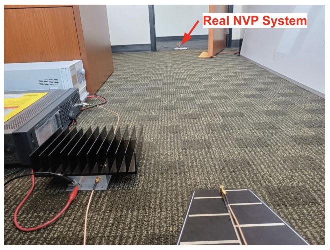
tested. Also, the DPI experiments in Figure 4 demonstrate that injecting signals into P2 causes lower forward progress rates in a broader frequency range compared to P1. This is mainly because the P2 signals can affect the ADC/Comparator more directly compared to P1 signals. This effect can be more significant when injected directly into the ADC/Comparator; however, it is out of scope in our threat model.

B. Remote EMI Attack Experiments

1) Experimental Settings: We further investigated remote EMI injections that can also control the output of the voltage monitor circuitry and thus induce the checkpoint/recovery signals from the victim monitor. For experiments, we used a +3.3V DC supply to power intermittent systems.

We transmitted single-tone sine-wave EMI signals, sweeping the frequency from 5MHz to 500MHz at intervals of 1MHz. The transmitting power was set at 35dBm—though we varied the level of the attack signal power from 0 to 35 dBm to estimate the correlation between the power of the attack


N5171B EXG X-Series RF Analog Signal Generator
ZHL-5W-1+ RF Amplifier
Keysight U8032A/0EM Power Supply
LPDAMAX Log-Periodic Antenna
(a) Experimental setting for remote attacks

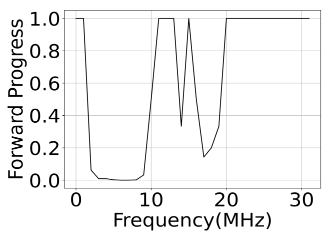


(b) Remote attack outside of a room

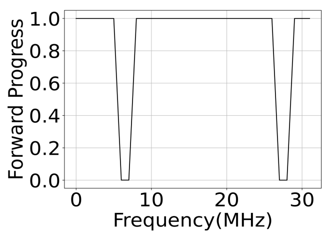
Fig. 6: Remote attack environment

Model	Voltage Monitor	ADC- $R_{min@0.1m}/Freq(MHz)$	Comp.-$R_{min@0.1m}/Freq(MHz)$	ADC-$F_{max@0.1m/Freq(MHz)}$
TI-MSP430FR2311	ADC	3.1% / 27MHz	N/A	41% / 27MHz
TI-MSP430FR2433	ADC	4.2% / 27MHz	N/A	41% / 27MHz
TI-MSP430FR4133	ADC	3.6% / 27MHz	N/A	42% / 28MHz
TI-MSP430F5529	ADC	4.0% / 27MHz	N/A	41% / 16MHz
TI-MSP430FR5739	ADC	1.8% / 27MHz	N/A	11% / 27MHz
TI-MSP430FR5994	ADC & Comp.	4.0% / 27MHz	$1.0*10^{-2}$% / 5MHz & 6MHz	28% / 27MHz
TI-MSP430FR6989	ADC & Comp.	3.6% / 27MHz	$1.2*10^{-2}$% / 27MHz	35% / 27MHz
TI-MSP432P (cortex-m4)	ADC & Comp.	3.3% / 27MHz	N/A	40% / 27MHz
STM32L552ZE (cortex-m33)	ADC & Comp.	4.8% / 17MHz	N/A	24% / 18MHz

TABLE I: Results of EMI attack experiments on real-world energy harvesting MCUs. ADC-R_{min} and Comp.-R_{min} represent the minimum forward progress rate under EMI attack with an ADC-based monitor and a comparator-based monitor, respectively. ADC-F_{max} represents the maximum checkpoint failure rate at a specific frequency.



(a) MSP430FR5994



(b) MSP430FR6989

Fig. 7: Remote attack analysis on commodity micro-controllers with a comparator-based voltage monitor varying attack signal frequencies. The X-axis represents the attack signal frequency, while the Y-axis represents the forward progress rate.

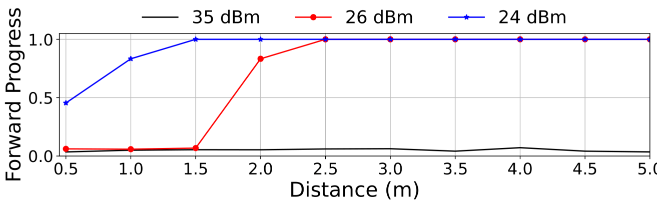
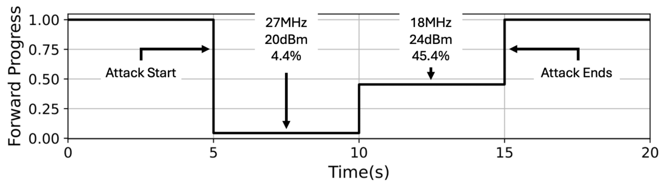
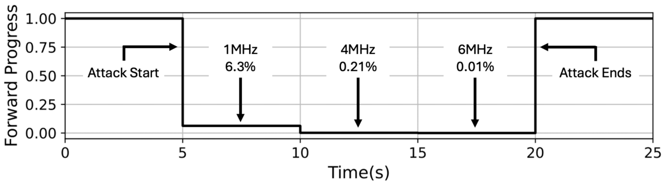


Fig. 8: Attack distance analysis: Results of remote attack experiments on intermittent systems within a 5-meter attack range.



(a) Remote attack analysis on MSP430FR5994 with an ADC-based voltage monitor



(b) Remote attack analysis on MSP430FR5994 with a comparator-based voltage monitor

Fig. 9: Remote attack analysis on MSP430FR5994 in real-time with an ADC-based voltage monitor and a comparator-based voltage monitor.

signal and the forward progress rate of the victim system. In our experiments, the target system was placed 5m away from an attacker's antenna; a 5m distance is long enough to attack a victim device sneakily in our attack scenario. Figure 6 describes the setting in detail.

2) Attack Analysis: We demonstrate adversaries can intentionally induce checkpoint/recovery signals and cause DoS and data corruption by remote EMI attacks in commodity intermittent systems.

Inducing a DoS with specific EMI signals. We monitored the forward progress rate of intermittent systems and recorded it with a laptop while sweeping the signal frequency. We observed that under EMI attacks, regardless of the type of voltage monitor (ADC-based or comparator-based), all intermittent systems that we tested suffered from DoS at certain frequencies as shown in Figure 5 and Figure 7, i.e., the forward progress rate is almost 0. Moreover, we found that, at the malicious frequencies, the systems were not able to finish I/O tasks; they got instead stuck indefinitely in the middle of (atomic) task execution such as sending a message or sensing. In this case, users cannot read the necessary information from the victim's device; more detail can be found in Table I, ADC-R and Comp-R columns.

Inducing a checkpoint failure with specific EMI signals. EMI attacks can lead to data corruption due to checkpoint failures. When attackers generate RF signals at 27 MHz, they can remotely trigger checkpoint and recovery signals in a victim system repeatedly. In particular, when the recovery signals are triggered within a malicious voltage range (V_{fail}), where $V_{off} < V_{fail} < V_{backup}$, the system executes the JIT checkpoint protocol without sufficient energy, thereby resulting in checkpoint failures. Notably, the false checkpoint/recovery signals can only be induced while the victim is powered on. To measure the checkpoint failure rate, we define a performance metric as follows: $F = N_{fail}/N_{checkpoints}$, where F is the failure rate, N_{fail} is the number of checkpoint failures, and $N_{checkpoint}$ is the total number of checkpoints. The result can be found in Table I, ADC-F column.

EMI power and distance relationship. We further investigate the relationship between the power of the attack signal and the attack distance. Figure 8 demonstrates that an EMI remote attack can be launched 0∼5m away from a victim device outside of a room (even when a door is closed). Attackers need to control the magnitude of the EM signal to gain effective control of the victim device. Theoretically, the distance of an EMI attack is directly proportional to the power of the EMI signals [84]. The experiment also demonstrates that the distance over which the malicious signal can propagate may exceed 5 meters as the power of the signal increases.

EMI attack analysis in real-time. Attackers can exploit precise frequency and power adjustments to control forward progress rates, making their actions less detectable. Figure 9 illustrates various attack scenarios conducted at different times, showing how adversaries can deliberately control the forward progress rate by exploiting different RF signal frequencies. As described in Figure 9(a) and Figure 9(b), adversaries can control the aggressiveness of their attacks by adjusting the attack signal frequency, making the attacks more stealthy by minimizing their impact and delaying detection by the victim system.

EMI attack analysis on other components. We conducted additional experiments to see whether other components within a processor are vulnerable to EMI attacks. We found that components, such as registers, cache, and internal clocks, are resistant to remote EMI attacks mainly because they are typically enclosed within a microprocessor, offering natural shielding and minimizing attack surface.[3] In contrast, a voltage monitor is connected to an external capacitor, which is decoupled from a processor, thereby increasing the attack surface as shown in Fig.3.

V. Countermeasures

This section explores conventional countermeasures with hardware supports (Section V-A1) and software supports (Section V-A2). To this end, this section introduces GECKO, a compiler-directed countermeasure (Section V-B).

A. Conventional Countermeasures

1) Hardware Solution: Within a decade, many solutions that incorporate both hardware and software have been published. However, to the best of our knowledge, these prior solutions are not suitable for power-hungry intermittent systems. One might suggest securing the JIT checkpoint protocol by incorporating hardware components with a voltage monitor, such as filters or differential comparators, to mitigate the EMI signals [28], [44], [77], [84], [85], [90], [100]. However, the filters are designed solely to reduce noise and are incapable of thwarting EMI attacks completely. Moreover, incorporating additional hardware components such as capacitors and conductors [44], [77], [90], or developing custom hardware circuitry like hardware anomaly detectors [84], [85], results in increased power consumption and space requirements in devices. These solutions, while effective in detecting attacks, are not only costly but also power-intensive, thus being overkill for intermittent systems. Also, other conventional physical countermeasures against EMI attacks include fully shielding the voltage monitor and its signal line to the processor with aluminum foil or placing the victim device inside a Faraday cage to block EMI completely. Unfortunately, these solutions are impractical for typical energy harvesting applications, such as wearable or healthcare devices, and are not cost-effective.

2) Software Solution: There have been several efforts to address the EMI attacks through software-based approaches. Fang et al. [28] conducted a study employing a fusion model trained with a Feature Extraction Unit (FEU) and Long Short-Term Memory (LSTM) to detect EMI attacks. The prior work from Zhang at al. [100] proposed a method involving the utilization of random secret bits corresponding to the activation and deactivation of analog sensors. The output of the sensors is then compared to the expected output to detect the presence of intentional EMI attacks, e.g., if the outputs are not matched, then it assumes the device is under attack.

Software-based solutions are more cost-effective and power-efficient compared to hardware-based schemes. However, they can introduce a significant performance overhead due to the need for non-trivial model training [28], new instruction set architectures, secret bit sequence generation, and repeated power cycling, which can degrade the quality of service [100]. More importantly, they cannot be applied to intermittent systems since they are unable to achieve crash consistency across frequent power outages, which is critical in an energy harvesting domain; Table II compares the prior works to our proposed countermeasure in more detail.

B. GECKO *Compiler*

To this end, this paper introduces GECKO, a compiler-directed countermeasure for secure intermittent systems without requiring user intervention or incurring any additional hardware support. The key insight is that intermittent systems can thwart EMI attacks if they close the attack vector (i.e., JIT checkpoint protocol). By closing the attack surface, adversaries can no longer manipulate the checkpoint protocol. Also, as discussed in Section IV-B2, by reducing the attack surface by disabling the voltage monitor, the system is able to maintain forward progress across EMI attacks.

To achieve crash consistency without the JIT checkpointing, GECKO leverages another type of power failure recovery solution, i.e., rollback recovery. GECKO inserts checkpoint stores into a given program by leveraging idempotent processing [20], [21], [87]. It generates a sequence of idempotent (recoverable) program tasks or regions that can be effectively recovered in case of power failure. Note that the rollback recovery is robust against the EMI attacks because the rollback recovery does not require the use of the voltage monitor which is the vulnerable component of the EMI attack. By closing the attack surface, the attacker cannot exploit the voltage monitor to induce fake signals.

However, the rollback recovery mechanism requires the insertion of a number of checkpoint stores in a given program

[3] Prior works demonstrated that such components can also be vulnerable to remote EMP or EMFI attacks [9], [10]; however, such attacks are beyond the scope of our study, as they often require close proximity and carry a higher risk of detection (Sec. II-D).

Prior works	Target	HW/SW	Energy Efficiency	Power Failure Recovery	Intermittent System Applicability
Ghost Talk [44]	Microphones	Hybrid	Low	No	N/A
Rocking Drones [77]	Drones	Hybrid	Low	No	N/A
Trick or Heat [84]	Incubators	Hardware	Low	No	N/A
SoK [90]	Analog Sensors	Hybrid	Low	No	N/A
Detection of EMI [100]	Temperature Sensors, Microphones	Software	High	No	N/A
Transduction Shield [85]	Pressure Sensors, Microphones	Hybrid	Low	No	N/A
Detection of Weak EMI [28]	Sensors from IIoT	Software	Low	No	N/A
GECKO	**Voltage Monitor**	**Software**	**High**	**Yes**	**Applicable**

TABLE II: Comparison of prior solutions for EMI attack mitigation. While previous works introduce solutions for mitigating EMI attacks, they are not applicable to intermittent systems due to the lack of power failure recovery mechanisms.

region or task, inherently slowing down the performance (e.g., 50%∼400% slowdown [10], [16], [57], [65], [67], [87]. Moreover, they often require a new programming language with user intervention, which is challenging for end-users to implement [16], [17], [57], [65]. To defeat EMI attacks without causing such issues, GECKO combines the benefits of JIT checkpointing and idempotent processing while avoiding their downsides. Unlike the original JIT checkpoint protocol, GECKO checkpoints registers during program execution in case of EMI attacks. Unlike the idempotent processing, GECKO does not rollback to the beginning of the interrupted program region across power failures, but resumes the interrupted program from the same point where it was stopped in the same way as the JIT checkpointing as long as the device is not under attacks.

The high-level idea is that GECKO keeps a given program always intact from checkpoint failure like a software-based solution. Upon detecting EMI attacks, GECKO disables the JIT checkpoint protocol and rolls back to the entry of the interrupted program region by restoring compiler-assisted checkpointed registers. On the other hand, in the absence of attacks, GECKO keeps running with the JIT checkpointing on the systems.

VI. GECKO Implementation

A. Detection of EMI Attacks

GECKO dynamically detects EMI attacks in a reactive manner. This approach is chosen for its accuracy and performance benefits compared to a proactive approach, which tends to slow down performance as a software-based crash consistency solution (Section V-A2). To detect the attack reactively, GECKO exploits two different mechanisms: (1) an acknowledgment (ACK) based detection and (2) a timer-based detection. First, GECKO utilizes an ACK as a barrier of the JIT checkpoint protocol. After checkpointing all registers in NVM/NVFF, GECKO persists the ACK, treating it as if it were the last register to be checkpointed, and toggles the ACK at each reboot time (e.g., 0→1→0 ...). In this way, if the ACK has not changed across power failure, GECKO assumes that the system is under attack, potentially failing to checkpoint and causing data corruption.

Second, GECKO forms a series of program regions in such a way that each region must be completed within one capacitor charge cycle by considering its minimum time bound of power-on period.[4] The insight behind this is that it can detect EMI attacks if a power outage occurs more than once in the same program region. Based on the insight, GECKO incorporates a verification step into the boot protocol to confirm whether the system has completed at least one program region following a power outage. If it turns out that no regions are completed, GECKO assumes that the system is under DoS attacks. Upon detection in either case, it disables the JIT checkpoint protocol and relies on the idempotent processing, which allows the program to re-enter the interrupted idempotent program region and restores all compiler-assisted checkpoint registers.

Discussion. One might argue that an adversary could remotely power on and off a victim device by spoofing malicious RF signals, potentially leading to a denial-of-service (DoS) issue. However, we found that this scenario is not feasible. This is mainly because RF attack signals are too weak to control the power line effectively (refer to Sec. III). Instead, an energy harvester within the victim system collects the attack signals as ambient energy (i.e., RF) and stores them in a capacitor until it is fully charged. This harvester-device platform is commonly used in current intermittent system platforms [1], [3], [8], [10], [38], [65], [66], [86].

B. Idempotent Processing with Lightweight Checkpointing

GECKO compiler allocates separate checkpoint storage in NVM, which is different from the storage used for JIT checkpoint protocol. Utilizing this separate checkpoint storage in NVM, GECKO leaves checkpoint stores at compilation time, creating a series of idempotent program regions. Thanks to the double checkpoint storage, GECKO can ensure that a given program remains intact even in the face of EMI attacks.

For idempotent processing with lightweight checkpointing, GECKO exploits five-step compiler passes. First, GECKO compiler front-end translates the source code into LLVM intermediate representation (IR) [45] and applies traditional compiler optimizations on the IR. Second, the optimized IR goes through an idempotent region formation pass [22]. Third, after constructing the idempotent regions, GECKO analyzes the size of each idempotent region to ensure forward progress execution. To measure the size of each region, GECKO leverages the worst-case execution time (WCET) analysis [12] and checks for a given region (r) whether WCET(r) is less

[4]Given the fully-buffered energy, GECKO analyzes how long the system can sustain its execution under the worst-case power consumption mode [12].

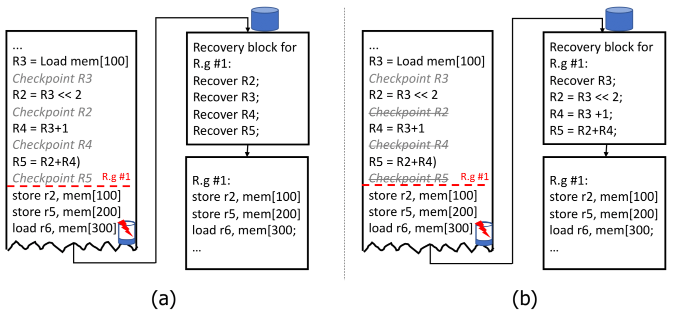


Fig. 10: Comparison between (a) GECKO without pruning and (b) GECKO with pruning optimization.

than the minimum time bound of the power-on period. If the execution time of a given region exceeds a threshold, GECKO will flag it. Fourth, for any given vulnerable idempotent region whose execution time is longer than a threshold, GECKO splits it into smaller regions so that each region can be completed within a single power-on period [12]. Then, the compiler inserts checkpoint stores, loops back to the WCET analysis step, and re-checks until all regions meet the timing requirement.

Region formation. For idempotent region formation, GECKO employs alias analysis to identify all possible memory anti-dependencies [22], [42], [52]. To ensure correct execution, every execution path from an anti-dependent load/store pair must cross at least one region boundary. GECKO avoids cutting memory writes preceding anti-dependencies, forming Write-After-Read-After-Write (WARAW) dependencies, since these still maintain idempotence [22]. However, as WCET analysis compiler pass may inadvertently cut these dependencies and compromise idempotence, GECKO conducts an additional idempotent region formation pass after the WCET pass to ensure idempotence for all regions.

Loop and I/O operation. It is challenging to analyze loops and I/O operations with compiler support. To this end, GECKO places a region boundary in the loop header to construct correct idempotent regions as prior works [12], [87], [99]. For I/O operations, GECKO compiler also relies on intra-procedural analysis, i.e., GECKO places region boundaries at any interrupts, asynchronous events, and function calls. In other words, GECKO treats them as separate regions.

C. Checkpoint Pruning

The overhead of idempotent processing correlates to the number of checkpoints executed at runtime. To minimize the number of checkpoint stores, GECKO leverages a novel compiler analysis, i.e., checkpoint pruning [42], [52], that can identify unnecessary checkpoints based on the following insight: some checkpoint stores can be pruned as long as they can be reconstructed across power failure. In other words, given a register input r of a program region Rg, it is unnecessary to checkpoint all updates for r, as long as r's value can be safely reconstructed.

To reconstruct pruned checkpoint values, GECKO generates recovery blocks and executes them in the wake of power failure. The recovery blocks are similar to traditional program slices but come with additional requirements. After successfully building the blocks, GECKO removes the corresponding checkpoints. Then, in the wake of power failure, GECKO runtime will reconstruct the original value by executing the recovery blocks. It is important to note that although the recovery block execution overhead for checkpoint reconstruction seems to be the same as the original checkpoint overhead, this is not true. The recovery block execution only occurs when the system is under attack, causing its overhead only in specific scenarios. In other words, checkpoint pruning allows GECKO to shift the runtime overhead of idempotent processing to EMI attack recovery, i.e., the overhead is minimized in the absence of EMI attacks, effectively optimizing the performance.

Figure 10 demonstrates how GECKO works. Figure 10(a) illustrates the control flow graph divided into idempotent regions (R.g #1) using a region partitioning algorithm based on Ratchet [87], with red dashed lines indicating region boundaries. GECKO checkpoint stores to provide guaranteed recovery where the register inputs to R.g #1 are R2-5. Figure 10(b) minimizes the performance overhead with checkpoint pruning as the resultant recovery block for R.g #1. For example, the values of R2, R4, and R5 can be reconstructed by executing the recovery block. In this example, GECKO can achieve over 80% checkpoint reduction by pruning the checkpoint stores; this paper will discuss about the impact of checkpoint pruning in Section VII.

D. Checkpoint Integrity

For correct recovery, GECKO must ensure the integrity of the checkpoints. Otherwise, the input registers of an interrupted region, which are checkpointed in earlier regions, may be overwritten by the checkpoints of the region. Prior works achieve the checkpoint integrity with double-buffering [65], [87]. It declares two register file arrays (e.g., `int RF[2][16]`) as checkpoint storages and, at each region boundary, flips the first boolean array index variable. This allows one of the two arrays to be intact while the other is written by centralized checkpointing during which power outage can occur.

Unfortunately, GECKO cannot directly use this simple technique. Since some checkpoints are pruned by GECKO, each register should have its own checkpoint storage index variable, that must be flipped every time it is checkpointed, to use double-buffering. This seems to be a significant overhead. For example, checkpointing 16 registers with double-buffering costs: *16 CheckpointStores + 16 IndexStores + 16 IndexLoads*.

To overcome this challenge, GECKO statically assigns a different array index to neighboring checkpoints of the same register. This can be reduced to a simple 2-coloring problem which treats the array index as a color [12], [42], [52]. In particular, the coloring might fail at a join point in the control flow graph of the program. GECKO solves this problem by selectively inserting additional checkpoints to the problematic control flow path. The rationale here is to ensure that the checkpoints of a given register over the incoming paths to the join point always share the same color, i.e., checkpoint storage index.

GECKO colors the checkpoint using a pre-order depth-first search starting from one path. Then, when it tries to color along the other path, a conflict may happen, i.e., two neighboring checkpoints can be assigned the same storage index for a register. To resolve the conflict, GECKO creates a new region between the conflicting regions and inserts an additional checkpoint, that saves the problematic register to a different index.

E. Recovery Block

Constructing recovery blocks is a key part of checkpoint pruning to ensure correct power failure recovery. To determine whether the checkpoints can be eliminated, GECKO checks the integrity of the recovery blocks' data and their control flow. In other words, if and only if the recovery blocks can reconstruct the value of register inputs, GECKO can eliminate the checkpoint stores in a given program.

To generate recovery blocks, GECKO analyzes the program dependence graph since the recovery blocks must depend on data dependence and control dependence backtracking [42], [52]. First, GECKO leverages the data dependence backtracking to ensure that the resulting blocks can recompute the values of register inputs from static checkpoints. In the process, GECKO traverses in reverse through the vertices of the dependency graph by following data-dependent edges in a depth-first search method. The notation $v \xrightarrow[\delta^d]{r} v'$ is used to represent that v relies on data from v', where v' defines the register r utilized by v. When provided with a register input r for a region Rg, GECKO conducts backtrack traversal along a series of vertices ($RgE \xrightarrow[\delta^d]{r_1} v_1 \xrightarrow[\delta^d]{r_2} \dots \xrightarrow[\delta^d]{r_n} v_n$), where RgE indicates the entry point of region Rg dependent on r_1, and v_n marks the final node in a path. The backtracking can be terminated in three conditions: when the vertex v_n has no data-dependent edge, when the vertex v_n has already been in a set of the minimum checkpoint set, and when the vertex v_n is unsafe. Second, GECKO utilizes the control dependence backtracking to guarantee that the control flow in the recovery blocks matches the original control flow. Assume a vertex v_i is associated with a group of vertices ($\mathbb{V}$) that are data-dependent on register r, denoted as $\forall v'_i \in \mathbb{V}$, $v_i \xrightarrow[\delta^d]{r} v'_i$. After successfully tracing all the data-dependent paths via v_i, GECKO ensures the integrity of control flow to ensure that the recovery slice produces the expected value of r at v_i.

F. Back to Normal!

Once the EMI attack is mitigated, GECKO re-enables the JIT checkpoint protocol. Given that EMI attacks can potentially occur at any moment, GECKO attempts to reactivate the protocol at reboot times while monitoring for signs of an attack. GECKO checks whether the voltage monitor sends the checkpoint signal within the initial region following a power outage. If not signaled, GECKO assumes that the threat has been mitigated. However, if the assumption turned out to be incorrect, i.e., the attack was not defeated, GECKO disables the JIT checkpoint protocol again and rolls back to a recent idempotent recovery point. This incurs no harm because an idempotent program is inherently resilient and ensures correct recovery, and thus the attempted JIT checkpointing does not lead to incorrect recovery.

VII. Evaluation

A. Experimental Setting

To evaluate the impact of GECKO, we conducted experiments with NVP, the idempotent compiler (Ratchet [87]), and GECKO, having a 1 mF supercapacitor as an energy buffer on a real evaluation board (MSP430FR5994), whose vulnerable frequency range was the smallest among all devices we tested as shown in Fig. 5. NVP is a basis/representative of all recent intermittent system solutions that are equipped with the JIT checkpointing mechanism, and Ratchet is a compiler-directed rollback recovery scheme that does not require a new programming language or user intervention. To implement NVP, we used TI-CTPL library [18] on the evaluation board. In this environment, we analyzed the security vulnerability by injecting EMI signals using all schemes.

B. Performance Analysis

To analyze the performance overhead of GECKO, we measured the execution time of each application on NVP, Ratchet, and GECKO. We set NVP as the baseline and compared the performance of each scheme to the baseline. We conducted experiments in three scenarios: (1) when the system is free from power outages (Section VII-B1), (2) when the system is under EMI attacks (Section VII-B3), and (3) when the system is in real energy harvesting environment (Section VII-B4).

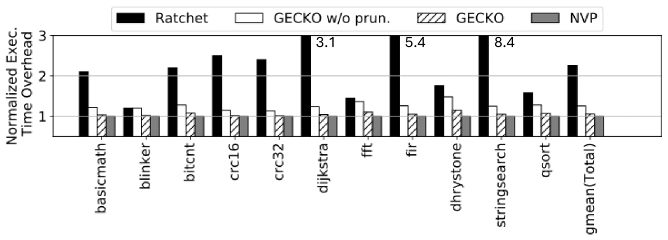


Fig. 11: Normalized execution time overhead of GECKO with and without (w/o) the pruning. The baseline is NVP.

1) Performance Analysis Without Power Outage: Figure 11 describes the performance overhead of each recovery scheme compared to the baseline. When there is no attack, Ratchet causes a performance overhead of approximately 2.4x because it executes checkpoint stores as demonstrated in the prior work [87]. On the other hand, GECKO causes about 6% performance overhead on average, compared to the baseline. This is possible because the compiler-assisted checkpoint overhead is negligible thanks to the checkpoint pruning technique. We also measured the performance overhead of GECKO without the pruning optimization. GECKO without the optimization causes about 30% performance overhead.

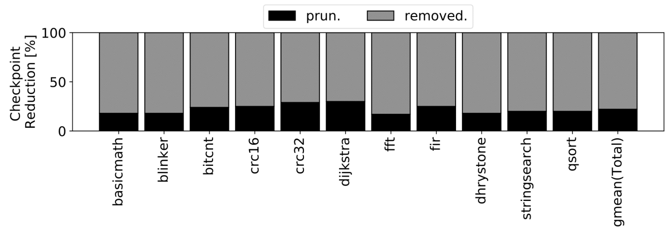


Fig. 12: Checkpoint reduction analysis: Gray boxes represent the checkpoint stores that can be removed by the checkpoint pruning technique.

2) Checkpoint Pruning Analysis: To analyze the performance improvement of checkpoint pruning in more detail, we measured the number of checkpoint stores of both non-optimized and optimized (pruned) versions of GECKO. Figure 12 shows that when checkpoint pruning is enabled, it can reduce about 80% of checkpoint stores that are required for GECKO non-optimized version.

3) Performance Analysis With EMI Attacks: We conducted our experiments in realistic energy harvesting environmental setting with EMI attacks. For energy harvesting environments, we employed a power generator constructed using MSP430FR5969 and interfaced it with our evaluation board via GPIO pins. A RF power trace was utilized, inducing a power outage at a frequency of 1Hz. While the power generator are providing some power to the evaluation board, we remotely generated malicious EMI signals. To evaluate the performance of each scheme, we measured the throughput of each scheme when they were under EMI attacks. For throughput, we counted the number of application completion times during one minute, i.e., $\frac{\#.of.comp}{1min.}$; we set the NVP without EMI attacks as a baseline.

We found that GECKO can make forward progress while NVP and Ratchet fail due to the DoS problem. Specifically, NVP was not able to reboot when its registers were corrupted by EMI attacks, and some regions in Ratchet are too long to be completed within one capacitor charge cycle, thereby leading to the DoS problem. On the other hand, GECKO kicked in the idempotent processing successfully once it detected the EMI attacks, providing a service even under EMI attacks. Overall, GECKO has a 41% throughput on average compared to the baseline, i.e., NVP's original throughout

EMI Attack Detection. To check whether GECKO can detect EMI attacks without losing data in various dynamic environments, we conducted experiments involving the injection of malicious EMI signals. Six different attack scenarios were constructed for the experiments: (a) no attack, (b) attack at 40 minutes, (c) attack at 30 minutes, (d) attacks at 20 and 40 minutes, (e) attacks at 15, 30, and 35 minutes, and (f) attacks at 10, 25, and 40 minutes, as shown in Figure 13a, 13b, 13c, 13d, 13e, and 13f, respectively. We found that when EMI attack was launched, GECKO immediately detected it by checking the ACK across power failure (Section VI-A). GECKO could also restore the correct checkpoint stores by reconstructing the required values and kept providing a service. On the other hand, when the EMI attack was ended, GECKO successfully got back to JIT checkpointing after ensuring its functionality as shown in Figure 13.

4) Performance Analysis In Real Energy Harvesting Environment: To assess the performance overhead of GECKO in a real energy harvesting environment, we conducted experiments using a Powercast P2110-EVB RF energy harvester [1], in the same way as prior works [12], [23], [68]. Powercast TX91501-3W transmitter, emitting an RF signal at a center frequency of 915 MHz, was used to power the evaluation board.

For performance analysis, we ran benchmark applications on three different schemes: Ratchet, NVP, and GECKO, as depicted in Figure 14. Ratchet showed the worst performance among the tested schemes, primarily due to a number of checkpoint stores. On the other hand, GECKO caused about 6% performance slowdown compared to NVP. This small overhead caused by GECKO was because it incurred only a few checkpoint stores without executing any recovery blocks.

C. Code Size Analysis

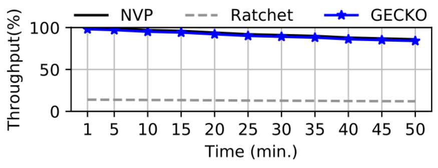
	basicmath	bitcnt	blink	crc16	crc32	dhrystone	dijkstra	fft	fir	qsort	string.	avg
# ckpt.	150	83	6	20	58	139	108	303	41	59	1128	81

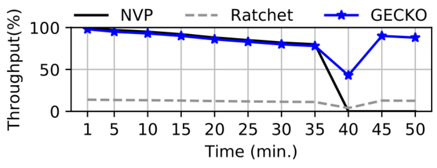
TABLE III: The total number of checkpoint stores generated by GECKO in each application

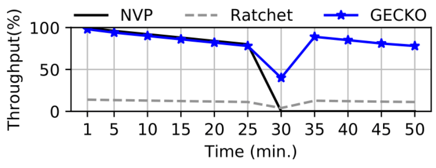
On average, GECKO incurs only 6% binary size overhead. Additionally, the number of recovery blocks for each application is approximately 7, with an average of 6 instructions in each block. In the wake of a power failure, GECKO needs to locate the correct recovery block if an EMI attack is detected. To manage the recovery blocks, GECKO has a look-up table consisting of about 130 instructions. Although it seems like a costly recovery process, GECKO does not cause any pressure on the memory usage at run-time because the recovery cost is applied only when an EMI attack is detected. In other words, GECKO does not execute the recovery blocks with the look-up table when there is no attack. Also, we counted the number of checkpoint stores generated by GECKO in each application (Tab.III). GECKO caused about 81 stores on average.

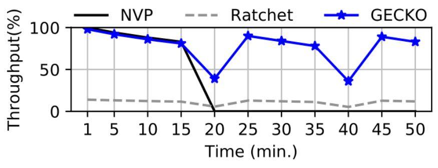
D. Capacitor Size Variation

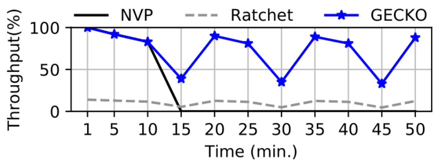
For the capacitor size sensitivity analysis, we conducted additional experiments within the same environment used for performance analysis, varying the capacitor size. In particular, we set the minimum capacitor size to 1 mF and the maximum capacitor size to 10 mF. Although it was possible to decrease the capacitor size further, smaller capacitors were found to degrade performance in a prior work [14]. We configured the checkpoint voltage thresholds accordingly; all capacitors were set to buffer the same amount of energy regardless of capacitance. With this configuration, we measured the performance of NVP and GECKO, varying the capacitor size: 1 mF, 2 mF, 5 mF, and 10 mF.


(a) Scenario 1


(b) Scenario 2


(c) Scenario 3


(d) Scenario 4


(e) Scenario 5

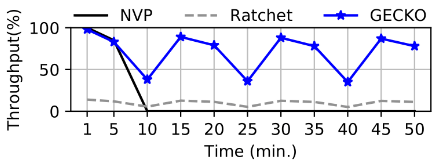

(f) Scenario 6

Fig. 13: Attack detection and recovery analysis varying attack patterns and recovery solutions. 0% throughput represents a denial of service.

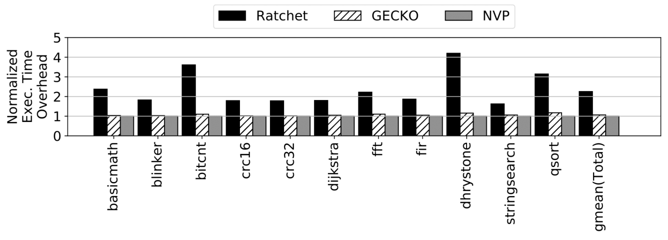


Fig. 14: Performance results in energy harvesting situation. We compare GECKO with Ratchet and NVP. Y-axis shows the normalized performance overhead time compared to NVP as baseline.

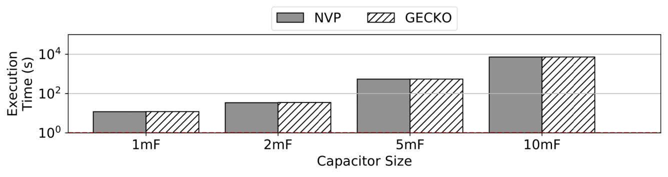


Fig. 15: Total execution time of each solution on average varying capacitor size.

Figure 15 shows the total execution time of GECKO and NVP on average, where lower values indicate better performance. Overall, the performance of GECKO consistently matches that of NVP. Both schemes perform optimally when the capacitor size is 1 mF. However, as the capacitor size increases, their execution time exponentially rises. This is primarily due to the substantial increase in charging time with larger capacitor sizes.

VIII. Other Related Works

Recently, researchers have identified security vulnerabilities in intermittent systems and proposed various countermeasures. *Cronin et al.* exposed that adversaries can deliberately wear out NVFFs in NVP to cause data corruption, i.e., wear out attacks [19]. They assume that adversaries can execute malicious code to rapidly consume energy and trigger frequent checkpoints, wearing out the checkpoint storage. To mitigate this vulnerability, they propose rotating the checkpoint storage location, which helps to reduce the wear-out rate. On the other hand, *Choi et al.* found that the JIT checkpoint mechanism can be corrupted when a capacitor (i.e., energy storage) is degraded [10]. They also discovered that such degradation of capacitor can be used as an attack vector, which they named as, Caphammer [9]. In use cases or attack scenarios, the capacitor can be degraded over time, and thus it may not be able to store a sufficient amount energy for correct checkpointing. To address the issue, they proposed a runtime system that can detect the degradation and recover the degraded capacitor. Despite prior attacks and solutions, current intermittent systems remain vulnerable to EMI attacks that are more precise and farther attacks than the high EMP attacks. To the best of our knowledge, GECKO is the first compiler-directed approach to defeat against these attacks.

IX. Conclusion

This paper exposes a new security vulnerability in intermittent systems. We found that EMI attacks can control a voltage monitor in the systems, resulting in data corruption and DoS issues. To defeat the attacks, this paper introduces GECKO, a compiler-directed mitigation solution without requiring hardware support. For performance optimization, GECKO leverages a checkpoint pruning technique with recovery blocks. Our experiments demonstrate that GECKO can successfully thwart EMI attacks and achieve correct recovery against them, causing only 6% performance overhead.

Acknowledgement

The authors thank Purdue and UCF CompArch members, ETRI Mobility UX Research Section, and anonymous reviewers for their valuable feedback. This work is in part supported by the IITP Grant from the Korea Government (MSIT) under 2021-0-00360 and 24HR3120 in ETRI, and by the NSF grants CNS-2314680, CNS-2314681, and CCF-2153749.

References

[1] "Powercast hardware." [Online]. Available: http://www.powercastco.com.

[2] "Cortex-m4, revision r0p0, technical reference manual," http://infocenter.arm.com/help/topic/com.arm.doc.ddi0439b/DDI0439B_cortex_m4_r0p0_trm.pdf, 2010, accessed: 2017-11-08.

[3] "Msp430fr5994launchpad development kit (mspexp430fr5994)," Mar 2016. [Online]. Available: http://www.ti.com/lit/ug/slau678a/slau678a.pdf

[4] "Msp430fr59xx mixed-signal microcontrollers (rev. f)," Mar 2017. [Online]. Available: http://www.ti.com/lit/ds/symlink/msp430fr5969.pdf

[5] "Rfspace lpdamax wide-band pcb log periodic antenna," 2018. [Online]. Available: http://rfspace.com/RFSPACE/Antennas_files/LPDA-MAX.pdf

[6] "Msp430fr599x mixed-signal microcontrollers datasheet (rev.d)," Jan 2021. [Online]. Available: https://www.ti.com/lit/ds/symlink/msp430fr5994.pdf?ts=1726190814187&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FMSP430FR5994

[7] "Arm cortex-m33 in a nutshell," https://www.st.com/content/st_com/en/arm-32-bit-microcontrollers/arm-cortex-m33.html, Sep 2023, accessed: 2023-11-08.

[8] A. Bhattacharyya, A. Somashekhar, and J. S. Miguel, "Nvmr: non-volatile memory renaming for intermittent computing," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, 2022, pp. 1–13.

[9] J. Choi, J. Choi, H. Joe, and C. Jung, "Caphammer: Exploiting capacitor vulnerability of energy harvesting systems," *ACM SIGBED International Conference on Embedded Software*, 2024.

[10] J. Choi, H. Joe, and C. Jung, "Capos: Capacitor error resilience for energy harvesting systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2022.

[11] J. Choi, H. Joe, Y. Kim, and C. Jung, "Achieving stagnation-free intermittent computation with boundary-free adaptive execution," in *Real-Time & Embedded Technology and Applications Symposium*, 2019.

[12] J. Choi, L. Kittinger, Q. Liu, and C. Jung, "Compiler-directed high-performance intermittent computation with power failure immunity," in *Real-Time & Embedded Technology and Applications Symposium*, 2022.

[13] J. Choi, Q. Liu, and C. Jung, "Cospec: Compiler directed speculative intermittent computation," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2019.

[14] J. Choi, J. Zeng, D. Lee, C. Min, and C. Jung, "Write-light cache for energy harvesting systems," in *Proceedings of the 50th Annual International Symposium on Computer Architecture*, 2023, pp. 1–13.

[15] Y.-W. Chong, W. Ismail, K. Ko, and C.-Y. Lee, "Energy harvesting for wearable devices: A review," *IEEE Sensors Journal*, vol. 19, no. 20, pp. 9047–9062, 2019.

[16] A. Colin and B. Lucia, "Chain: Tasks and channels for reliable intermittent programs." in *In Proceedings of the 2016 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*. ACM, 2015, pp. 514–530.

[17] A. Colin and B. Lucia, "Termination checking and task decomposition for task-based intermittent programs," in *Proceedings of the 27th International Conference on Compiler Construction*. ACM, 2018.

[18] W. Cooper, "What Is Compute through Power Loss?" May 2015.

[19] P. Cronin, C. Yang, and Y. Liu, "A collaborative defense against wear out attacks in non-volatile processors," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 2018, pp. 1–6.

[20] M. de Kruijf and K. Sankaralingam, "Idempotent processor architecture," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2011, pp. 140–151.

[21] M. de Kruijf and K. Sankaralingam, "Idempotent code generation: Implementation, analysis, and evaluation," in *Code Generation and Optimization (CGO), 2013 IEEE/ACM International Symposium on*. IEEE, 2013, pp. 1–12.

[22] M. A. De Kruijf, "Compiler construction of idempotent regions and applications in architecture design," Ph.D. dissertation, Madison, WI, USA, 2012.

[23] J. de Winkel, C. Delle Donne, K. S. Yildirim, P. Pawełczak, and J. Hester, "Reliable timekeeping for intermittent computing," in *Proceedings of ASPLOS*, 2020.

[24] B. Denby and B. Lucia, "Orbital edge computing: Nanosatellite constellations as a new class of computer system," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 939–954.

[25] E. Digitale, "Technology equality gap for kids' diabetes treatment is growing," 2021. [Online]. Available: https://scopeblog.stanford.edu/continuousglucosemonitorbloodsugartestsmartphoneapp/

[26] J. S. Douglas, "Preliminary analysis of energy harvesting assault pack," *US Army CERDEC CP&ID Power Division, accessed Nov*, vol. 26, p. 2017, 2015.

[27] S. Emaminejad, W. Gao, E. Wu, Z. A. Davies, H. Yin Yin Nyein, S. Challa, S. P. Ryan, H. M. Fahad, K. Chen, Z. Shahpar, S. Talebi, C. Milla, A. Javey, and R. W. Davis, "Autonomous sweat extraction and analysis applied to cystic fibrosis and glucose monitoring using a fully integrated wearable platform," *Proceedings of the National Academy of sciences*, vol. 114, no. 18, pp. 4625–4630, 2017.

[28] K. Fang, T. Wang, X. Yuan, C. Miao, Y. Pan, and J. Li, "Detection of weak electromagnetic interference attacks based on fingerprint in iiot systems," in *Future Generation Computer Systems Volume 126*, 2021, pp. 295–304.

[29] D. I. Gorman, M. R. Guthaus, and J. Renau, "Architectural opportunities for novel dynamic emi shifting (demis)," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, 2017, pp. 774–785.

[30] A. Haeberlin, A. Zurbuchen, S. Walpen, J. Schaerer, T. Niederhauser, C. Huber, H. Tanner, H. Servatius, J. Seiler, H. Haeberlin *et al.*, "The first batteryless, solar-powered cardiac pacemaker," *Heart rhythm*, vol. 12, no. 6, pp. 1317–1323, 2015.

[31] C. Haynes, J. Douglas, and J. Collazo, "Relationship between metabolic cost and power output with a prototype energy harvesting assault pack," *Journal of Science and Medicine in Sport*, vol. 20, p. S102, 2017.

[32] J. Hester, K. Storer, L. Sitanayah, and J. Sorber, "Towards a language and runtime for intermittently-powered devices," *sleep*, vol. 9, p. 10, 2016.

[33] M. Hicks, "Clank: Architectural support for intermittent computation," in *In Proceedings of ISCA '17*. ACM, 2017.

[34] S.-Y. Huang, J. Zeng, X. Deng, S. Wang, A. Sifat, B. Bharmal, J.-B. Huang, R. Williams, H. Zeng, and C. Jung, "Rtailor: Parameterizing soft error resilience for mixed-criticality real-time systems," in *2023 IEEE Real-Time Systems Symposium (RTSS)*, 2023, pp. 344–357.

[35] B. Islam and S. NIRJON, "Zygarde: Time-sensitive on-device deep inference and adaptation on intermiently-powered systems," *ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2020.

[36] S. Islam, J. Deng, S. Zhou, C. Pan, C. Ding, and M. Xie, "Enabling fast deep learning on tiny energy-harvesting iot devices," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2022, pp. 921–926.

[37] S. Islam, S. Zhou, R. Ran, Y.-F. Jin, W. Wen, C. Ding, and M. Xie, "Eve: Environmental adaptive neural network models for low-power energy harvesting system," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, 2022, pp. 1–9.

[38] H. Jayakumar, A. Raha, and V. Raghunathan, "Quickrecall: A low overhead hw/sw approach for enabling computations across power cycles in transiently powered computers," in *International Conference on VLSI Design and International Conference on Embedded Systems*, 2014.

[39] J. Jeong and C. Jung, "Pmem-spec: persistent memory speculation (strict persistency can trump relaxed persistency)," in *ACM ASPLOS*, 2021.

[40] J. Jeong, J. Zeng, and C. Jung, "Capri: Compiler and architecture support for whole-system persistence," in *International Symposium on High-Performance Parallel and Distributed Computing*, 2022.

[41] H. Kim, J. Zeng, Q. Liu, M. Abdel-Majeed, J. Lee, and C. Jung, "Compiler-directed soft error resilience for lightweight gpu register file protection," in *PLDI '20: 41st ACM SIGPLAN International Conference on Programming Language Design and Implementation*, ser. PLDI 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 989–1004. [Online]. Available: https://doi.org/10.1145/3385412.3386033

[42] H. Kim, J. Zeng, Q. Liu, M. Abdel-Majeed, J. Lee, and C. Jung, "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*, 2020.

[43] V. Kortbeek, A. Bakar, S. Cruz, K. S. Yildirim, P. Pawełczak, and J. Hester, "Bfree: Enabling battery-free sensor prototyping with python," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 4, pp. 1–39, 2020.

[44] D. F. Kune, J. Backes, S. S. Clark, D. Kramer, M. Reynolds, K. Fu, Y. Kim, and W. Xu, "Ghost talk: Mitigating emi signal injection attacks against analog sensors," in *2013 IEEE Symposium on Security and Privacy*. IEEE, 2013, pp. 145–159.

[45] C. Lattner and V. Adve, "Llvm: A compilation framework for lifelong program analysis & transformation," in *Proceedings of the International Symposium on Code Generation and Optimization*, ser. CGO '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 75–.

[46] S. Lee, B. Islam, Y. Luo, and S. Nirjon, "Intermittent learning: On-device machine learning on intermittently powered system," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 3, no. 4, pp. 1–30, 2019.

[47] 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)*. IEEE, 2016, pp. 1–6.

[48] Q. Liu, J. Izraelevitz, S. K. Lee, M. L. Scott, S. H. Noh, and C. Jung, "ido: Compiler-directed failure atomicity for nonvolatile memory," in *International Symposium on Microarchitecture (MICRO)*, 2018.

[49] Q. Liu, C. Jung, D. Lee, and D. Tiwari, "Clover: Compiler directed lightweight soft error resilience," in *Proceedings of the 16th ACM SIGPLAN/SIGBED Conference on Languages, Compilers and Tools for Embedded Systems 2015 CD-ROM*, ser. LCTES'15. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2670529.2754959

[50] Q. Liu, C. Jung, D. Lee, and D. Tiwari, "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*. IEEE, 2016, pp. 228–239.

[51] Q. Liu, C. Jung, D. Lee, and D. Tiwari, "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*, 2016, pp. 228–239.

[52] Q. Liu, C. Jung, D. Lee, and D. Tiwari, "Compiler-directed lightweight checkpointing for fine-grained guaranteed soft error recovery," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2016.

[53] Q. Liu, C. Jung, D. Lee, and D. Tiwari, "Compiler-directed soft error detection and recovery to avoid due and sdc via tail-dmr," *ACM Trans. Embed. Comput. Syst.*, vol. 16, no. 2, dec 2016. [Online]. Available: https://doi.org/10.1145/2930667

[54] Q. Liu, C. Jung, D. Lee, and D. Tiwarit, "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)*. IEEE, 2016, pp. 1–12.

[55] B. Lucia, V. Balaji, A. Colin, K. Maeng, and E. Ruppel, "Intermittent computing: Challenges and opportunities," in *LIPIcs-Leibniz International Proceedings in Informatics*, vol. 71. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[56] B. Lucia, B. Denby, Z. Manchester, H. Desai, E. Ruppel, and A. Colin, "Computational nanosatellite constellations: Opportunities and challenges," *GetMobile: Mobile Computing and Communications*, vol. 25, no. 1, pp. 16–23, 2021.

[57] B. Lucia and B. Ransford, "A simpler, safer programming and execution model for intermittent systems," in *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation*. ACM, 2015, pp. 575–585.

[58] K. Ma, J. Li, X. Li, Y. Liu, Y. Xie, M. Kandemir, J. Sampson, and V. Narayanan, "Iaa: Incidental approximate architectures for extremely energy-constrained energy harvesting scenarios using iot nonvolatile processors," *IEEE Micro*, vol. 38, no. 4, pp. 11–19, 2018.

[59] K. Ma, X. Li, M. T. Kandemir, J. Sampson, V. Narayanan, J. Li, T. Wu, Z. Wang, Y. Liu, and Y. Xie, "Neofog: Nonvolatility-exploiting optimizations for fog computing," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, 2018, pp. 782–796.

[60] K. Ma, X. Li, J. Li, Y. Liu, Y. Xie, J. Sampson, M. T. Kandemir, and V. Narayanan, "Incidental computing on iot nonvolatile processors," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2017, pp. 204–218.

[61] K. Ma, X. Li, H. Liu, X. Sheng, Y. Wang, K. Swaminathan, Y. Liu, Y. Xie, J. Sampson, and V. Narayanan, "Dynamic power and energy management for energy harvesting nonvolatile processor systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 4, pp. 1–23, 2017.

[62] K. Ma, X. Li, S. R. Srinivasa, Y. Liu, J. Sampson, Y. Xie, and V. Narayanan, "Spendthrift: Machine learning based resource and frequency scaling for ambient energy harvesting nonvolatile processors," in *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2017, pp. 678–683.

[63] K. Ma, X. Li, K. Swaminathan, Y. Zheng, S. Li, Y. Liu, Y. Xie, J. J. Sampson, and V. Narayanan, "Nonvolatile processor architectures: Efficient, reliable progress with unstable power," *IEEE Micro*, vol. 36, no. 3, pp. 72–83, 2016.

[64] K. Ma, Y. Zheng, S. Li, K. Swaminathan, X. Li, Y. Liu, J. Sampson, Y. Xie, and V. Narayanan, "Architecture exploration for ambient energy harvesting nonvolatile processors," in *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on*. IEEE, 2015, pp. 526–537.

[65] K. Maeng, A. Colin, and B. Lucia, "Alpaca: Intermittent execution without checkpoints," *Proceedings of the ACM on Programming Languages*, vol. 1, no. OOPSLA, pp. 1–30, 2017.

[66] K. Maeng and B. Lucia, "Adaptive dynamic checkpointing for safe efficient intermittent computing," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 2018.

[67] K. Maeng and B. Lucia, "Supporting peripherals in intermittent systems with just-in-time checkpoints," in *ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2019.

[68] K. Maeng and B. Lucia, "Adaptive low-overhead scheduling for periodic and reactive intermittent execution," in *ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2020.

[69] D. Narayanan and O. Hodson, "Whole-system persistence," in *Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems*, 2012, pp. 401–410.

[70] E. Nwafor, A. Campbell, D. Hill, and G. Bloom, "Towards a provenance collection framework for internet of things devices. in 2017 ieee smartworld, ubiquitous intelligence & computing, advanced & trusted computed, scalable computing & communications, cloud & big data computing, internet of people and smart city innovation (smartworld/scalcom/uic/atc/cbdcom/iop/sci)," 2017.

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

[72] W. A. Radasky, C. E. Baum, and M. W. Wik, "Introduction to the special issue on high-power electromagnetics (hpem) and intentional electromagnetic interference (iemi)," *IEEE Transactions on electromagnetic compatibility*, vol. 46, no. 3, pp. 314–321, 2004.

[73] A. Richelli, L. Colalongo, and Z.-M. Kovács-Vajna, "Emi effect in voltage-to-time converters," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 4, pp. 1078–1082, 2020.

[74] E. Ruppel, M. Surbatovich, H. Desai, K. Maeng, and B. Lucia, "An architectural charge management interface for energy-harvesting systems," in *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2022, pp. 318–335.

[75] F. Sabath, "What can be learned from documented intentional electromagnetic interference (iemi) attacks?" in *2011 XXXth URSI General Assembly and Scientific Symposium*. IEEE, 2011, pp. 1–4.

[76] S. Scargall, *Programming Persistent Memory*. Intel, 2020.

[77] Y. Son, H. Shin, D. Kim, Y. Park, J. Noh, K. Choi, J. Choi, and Y. Kim, "Rocking drones with intentional sound noise on gyroscopic sensors," in *24th USENIX Security Symposium (USENIX Security 15)*, 2015, pp. 881–896.

[78] D. J. Sorin, "Computer architecture for orbital edge computing," *IEEE Computer Architecture Letters*, vol. 53, no. 04, pp. 7–8, 2020.

[79] F. Su, Y. Liu, Y. Wang, and H. Yang, "A ferroelectric nonvolatile processor with 46μ s system-level wake-up time and 14μ s sleep time for energy harvesting applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 3, pp. 596–607, 2017.

[80] F. Su, K. Ma, X. Li, T. Wu, Y. Liu, and V. Narayanan, "Nonvolatile processors: Why is it trending?" in *Design, Automation & Test in*

Europe Conference & Exhibition (DATE), 2017. IEEE, 2017, pp. 966–971.

[81] J. Swanner, J. Bito, G. Nichols, X. He, J. Hewett, and M. M. Tentzeris, "Integrating multiple energy harvesting systems for department of defense applications," in *EESAT Conference-Evolution & Revolution*, 2017.

[82] M. K. Talarico, C. A. Haynes, J. S. Douglas, and J. Collazo, "Spatiotemporal and kinematic changes in gait while carrying an energy harvesting assault pack system," *Journal of biomechanics*, vol. 74, pp. 143–149, 2018.

[83] S. K. Thirumala, A. Raha, H. Jayakumar, K. Ma, V. Narayanan, V. Raghunathan, and S. K. Gupta, "Dual mode ferroelectric transistor based non-volatile flip-flops for intermittently-powered systems," in *Proceedings of the International Symposium on Low Power Electronics and Design*, 2018, pp. 1–6.

[84] Y. Tu, S. Rampazzi, B. Hao, A. Rodriguez, K. Fu, and X. Hei, "Trick or heat? manipulating critical temperature-based control systems using rectification attacks," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2301–2315.

[85] Y. Tu, V. S. Tida, Z. Pan, and X. Hei, "Transduction shield: A low-complexity method to detect and correct the effects of emi injection attacks on sensors," in *ASIA CCS '21: Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, 2021, pp. 901–915.

[86] Y. Wang, Y. Liu, S. Li, D. Zhang, B. Zhao, M.-F. Chiang, Y. Yan, B. Sai, and H. Yang, "A 3us wake-up time nonvolatile processor based on ferroelectric flip-flops," in *Proceedings of the ESSCIRC*, 2012.

[87] J. V. D. Woude and M. Hicks, "Intermittent computation without hardware support or programmer intervention," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. Savannah, GA: USENIX Association, 2016, pp. 17–32.

[88] Z. Xiao, X. Tan, X. Chen, S. Chen, Z. Zhang, H. Zhang, J. Wang, Y. Huang, P. Zhang, L. Zheng, and H. Min, "An implantable rfid sensor tag toward continuous glucose monitoring," *IEEE journal of biomedical and health informatics*, vol. 19, no. 3, pp. 910–919, 2015.

[89] M. Xie, M. Zhao, C. Pan, J. Hu, Y. Liu, and C. J. Xue, "Fixing the broken time machine: consistency-aware checkpointing for energy harvesting powered non-volatile processor," in *Proceedings of the 52nd Annual Design Automation Conference*. ACM, 2015, p. 184.

[90] C. Yan, H. Shin, C. Bolton, W. Xu, Y. Kim, and K. Fu, "Sok: A minimalist approach to formalizing analog sensor security," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 233–248.

[91] C. Yan, W. Xu, and J. Liu, "Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicle," in *DEF CON 24*.

[92] M.-L. Yeh, W.-R. Liou, H.-P. Hsieh, and Y.-J. Lin, "An electromagnetic interference (emi) reduced high-efficiency switching power amplifier," *IEEE Transactions on Power Electronics*, vol. 25, no. 3, pp. 710–718, 2009.

[93] K. S. Yildirim and P. Pawelczak, "On distributed sensor fusion in batteryless intermittent networks," in *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 2019, pp. 495–501.

[94] J. Zeng, J. Choi, X. Fu, A. P. Shreepathi, D. Lee, C. Min, and C. Jung, "Replaycache: Enabling volatile cachesfor energy harvesting systems," in *International Symposium on Microarchitecture*, 2021.

[95] J. Zeng, S.-Y. Huang, J. Liu, and C. Jung, "Soft error resilience at near-zero cost," in *Proceedings of the 38th ACM International Conference on Supercomputing*, ser. ICS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 176–187. [Online]. Available: https://doi.org/10.1145/3650200.3656605

[96] J. Zeng, J. Jeong, and C. Jung, "Persistent processor architecture," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 1075–1091. [Online]. Available: https://doi.org/10.1145/3613424.3623772

[97] J. Zeng, H. Kim, J. Lee, and C. Jung, "Turnpike: Lightweight soft error resilience for in-order cores," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, 2021, pp. 654–666.

[98] J. Zeng, T. Zhang, and C. Jung, "Compiler-directed whole-system persistence," in *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, 2024, pp. 961–977.

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

[100] Y. Zhang and K. Rasmussen, "Detection of electromagnetic interference attacks on sensor systems," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 203–216.

[101] Y. Zhou, J. Zeng, J. Jeong, J. Choi, and C. Jung, "Sweepcache: Intermittence-aware cache on the cheap," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, 2023, pp. 1059–1074.