Energy Attacks in the Battery-less Internet of Things: Directions for the Future

Luca Mottola[+], Arlsan Hameed[†], and Thiemo Voigt[*†]
[+]Politecnico di Milano (Italy), [*]RI.SE Computer Science (Sweden), [†]Uppsala University (Sweden)

ABSTRACT

We study how ambient energy harvesting may be used as an attack vector in the battery-less Internet of Things (IoT). Battery-less IoT devices rely on ambient energy harvesting and are employed in a multitude of applications, including safety-critical ones such as biomedical implants. Due to scarce energy intakes and limited energy buffers, their executions become *intermittent*, alternating periods of active operation with periods of recharging energy buffers. We reveal how, by exerting limited control on ambient energy one can create situations of *livelock*, *denial of service*, and *starvation*, without physical device access. We call these situations *energy attacks*. We detail, analyze, and quantitatively demonstrate how these attacks can be applied to battery-less IoT devices, and illustrate their consequences on a system's regular operation.

ACM Reference Format:
Luca Mottola[+], Arlsan Hameed[†], and Thiemo Voigt[*†], [+]Politecnico di Milano (Italy), [*]RI.SE Computer Science (Sweden), [†]Uppsala University (Sweden). 2024. Energy Attacks in the Battery-less Internet of Things: Directions for the Future. In *The 17th European Workshop on Systems Security (EuroSec '24), April 22, 2024, Athens, Greece.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3642974.3652283

1 INTRODUCTION

Ambient energy harvesting allows Internet of Things (IoT) devices to eliminate their dependency on traditional batteries [12]. This enables drastic reductions of maintenance costs and previously unattainable deployments, even in safety-critical settings such as biomedical implants [1, 18, 20, 25, 37].

Harvested energy is generally highly variable in time [12], yet energy buffers, such as capacitors, need to be miniaturized as well to limit device footprint, and therefore offer limited energy budgets. System shutdowns due to energy depletion are unavoidable and computing becomes *intermittent* [3]: periods of active execution and periods of energy harvesting are unpredictably interleaved.

Computing intermittently. Fig. 1 shows an example execution. The ambient charges the onboard capacitor until voltage V_{on} is reached that causes the device to power on. The device senses, computes, and communicates as long as the capacitor charge remains above a threshold V_{off}. The device then switches off, waiting for the capacitor to reach V_{on} again. This pattern may occur on tiny time scales; computing simple error correction codes on a battery-less IoT device may require as many as 16 active cycles [13].

EuroSec '24, April 22, 2024, Athens, Greece

ACM ISBN 979-8-4007-0542-7/24/04.
https://doi.org/10.1145/3642974.3652283

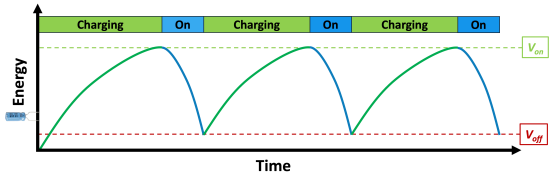


Figure 1: Example intermittent execution. *Periods of active computation and periods of charging the energy buffer alternate, leading to an intermittent computing pattern.*

Due to resource constraints, applications run with no operating system support [3]. When the device powers off at V_{off}, the system state would normally be lost. Intermittently-computing IoT systems use checkpointing [2, 9, 10, 13, 40, 53, 54, 61, 68] or task-based programming [21, 51, 52, 56, 62, 70] to create persistent state on non-volatile memory (NVM). These systems operate as the device approaches V_{off}, allowing devices to retain the application state across energy failures.

Operations on NVM, however, are extremely energy hungry [55]. Their energy cost may reach up to 350% the cost of the application processing, mainly due to the use of energy-hungry NVM technology [68]. Using persistent state to cross energy failures has further implications. When using FRAM as NVM, for example, wait cycles may be necessary to synchronize read/write operations with the MCU, further increasing energy consumption [64]. If the system employs stateful peripherals, their state is also to be retained across energy failures too [7, 11, 15, 54]. This increases the size of persistent state that must include information that may not be reflected in the system's main memory, adding to the energy overhead.

Energy harvesting as attack vector. This paper is about a *largely unexplored direction*: we study how a limited control on ambient energy may steer intermittent executions in unintended ways.

The straightforward scenario consists, for example, in physically blocking the solar radiation arriving at a solar panel that powers the device, eventually impeding forward progress. Crucially, we demonstrate that much more subtle situations exist. We generally call these situations *energy attacks*. While such attacks are more difficult to perform than just physically blocking a solar panel, they are much harder to detect. Attackers using these methods gain fine-grained control over their effects, which may help circumvent or bypass defense and detection mechanisms.

We define the attack model and systematically analyze possible energy attacks in Sec. 3. We experimentally demonstrate that energy attacks can exploit these vulnerabilities to create situations of *livelock*, *starvation*, and *denial of service*. Unlike the straightforward scenario above, we provide quantitative evidence that these attacks create situations that are deceptively similar to legitimate executions. In Sec. 4, we provide real-world evidence of how practical are energy attacks. We show that it takes no more than a few weeks for M.Sc. students with no specific training to exercise energy attacks.

The evidence we provide in this paper is a stepping stone to design detection and defense mechanisms, thus inspiring additional efforts in this area. We end the paper with a discussion of our work's limitations and of the feasibility of energy attacks in Sec. 5, and with pointers to follow-up research in Sec. 6 . Next, we illustrate necessary background information and survey related work.

2 BACKGROUND AND RELATED WORK

Our work intersects multiple areas. We provide here background information and survey related works.

Power attacks in data centers. Our work resonates with power attacks in data centers. With the increasing number of physical servers, their power distribution systems tend to approach peak capacities and power oversubscription is used to handle power provisioning. This works as long as servers do not peak simultaneously.

Malicious workloads may generate power spikes on multiple servers at the same time, which causes branch circuit breakers to trip, leading to power outages [31, 48]. Virtual machine provisioning [19] and side channels [19] are used to create abnormal behaviors. Detection techniques include machine learning applied to performance logs [19] and modeling user behaviors [48].

Common with our problem is that energy is part of the attack vector. However, the technology is extremely different, for example, in terms of workloads and hardware platforms. In contrast to the attack model we describe in Sec. 3, attackers do not directly manipulate the energy provisioning channel and need access to the target data center or must be informed of its layout. The attacks we study, however, do not require physical access to the target device.

Security in battery-powered IoT. Resource-constrained IoT devices are difficult to secure due to resource constraints, which complicates the use of mainstream security mechanisms [65].

Battery-powered IoT devices enable peculiar attacks, for example, in an attempt to drain batteries [44, 59]. Low-power radios make IoT devices vulnerable to denial of service attacks, for example, due to intentional jamming [43]. Multi-hop networks require specialized network stacks that open to new kinds of attacks, in particular at the routing layer, motivating new security mechanisms ranging from hardware-based solutions [60] to methods for attack detection and mitigation that rely on machine learning [24, 63].

These approaches, unfortunately, falls short of expectations for battery-less IoT devices, where energy constraints are way more severe. Moreover, intermittent executions add a new dimension to the problem that requires specialized solutions.

Intermittent computing. The prevailing architecture includes a mixed-volatile MCU [64] with built-in NVM, and a capacitor to tame fluctuations of energy intake. Such device configuration is seen in both available platforms [36, 39] and concrete deployments [18, 26].

Specialized architectures also exist that use separate capacitors of different sizes as energy buffers [23]. This allows the system to strike a better trade-off between charging times and available energy. Smaller capacitors are the first to reach V_{on}; as this happens, tasks that consume little energy, such as probing low-power sensors, are immediately executed. Bigger capacitors take longer to reach V_{on}; their energy is eventually consumed by energy-hungry tasks, such as controlling actuators or radio operations.

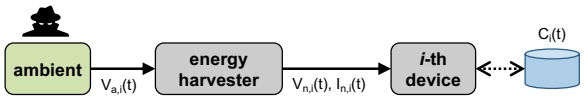


Figure 2: System and attack model.

Checkpointing [2, 9, 13, 53, 54, 61, 68] or task-based programming [21, 51, 56, 62, 70] is used to deal with energy failures. The former consist in replicating the application state on NVM, where it is retrieved back once the system resumes with sufficient energy. The latter offer abstractions to define and manage persistent state, while taking care of data consistency in case of repeated executions of non-idempotent code [68]. Software techniques are also used to handle peripheral states across energy failures [15] and to estimate energy consumption of intermittent programs [4].

Security and intermittent computing. The security scenario becomes uncharted territory here. The few existing solutions focus on securing persistent state.

As an example, Krishnan et al. [45] demonstrate that persistent state is vulnerable to sniffing, spoofing, or replay attacks. An attacker may simply sniff persistent state by reading the contents of the NVM using a debug port. Sniffing may be prevented by encrypting data on NVM, thus ensuring confidentiality, but does not prevent an attacker from spoofing, that is, tampering encrypted data, which threats the checkpoint integrity. By collecting several checkpoints, attackers may also replicate the application execution.

To secure persistent state, Asad et al. [8] experimentally evaluate the use of different encryption algorithms and ARM TrustZone protection. Krishnan et al. [47] build on this and propose a configurable checkpoint security setting that leverages application properties to reduce overhead. Ghodsi et al. [30] use lightweight algorithms [14] for securing checkpoints, ensuring confidentiality. Valea et al. [67] propose a SECure Context Saving hardware module inside the MCU. In contrast, Grisafi et al. [32] present a hypervisor to manage and protect checkpoints. Khrishnan et al. [46] present a generic secure protocol and apply Authenticated Encryption with Associated Data to protect checkpointing data.

Unlike the works above, we study new types of attacks realized by exerting control on ambient energy provisioning.

3 ENERGY ATTACKS

We present first the system and attack model we adopt. Then we present and analyze three novel energy attacks.

3.1 System and Attack Model

Fig. 2 illustrates the system and attack model we adopt. A resource-constrained intermittently-computing IoT device is equipped with multiple sensors and/or actuators, an MCU, a radio, and an energy management circuitry attached to the output of the energy harvester and used to charge the local energy buffer. Such configuration is seen in multiple battery-less IoT deployments [1, 18, 20, 37].

We model the energy coming from the ambient a and arriving at the energy harvester of node i as a continuous signal of voltage $V_{a,i}(t)$. This describes the energy content made available by the ambient to node i at time t. For simplicity, our description here considers a single energy source. The corresponding analysis, however, applies no matter the number of energy sources, as long as

the attack model is applicable to each of them. Relying on multiple energy sources [49] may be, nonetheless, a way to defend against energy attacks, as we hint in Sec. 6.

The energy harvester of node i takes $V_{a,i}(t)$ as input and transforms it into an energy signal of voltage $V_{n,i}(t)$ and current $I_{n,i}(t)$. The latter is a function of $V_{n,i}(t)$ and of the equivalent resistance offered by the charging circuitry at node i. The energy signal described by $V_{n,i}(t)$ and $I_{n,i}(t)$ charges the local energy buffer, eventually discharged while sensing, computing, or communicating. We model the charge available in the energy buffer of node i as $C_i(t)$.

The attacker has no physical access to the devices and no knowledge of the relation between $V_{a,i}(t)$ and $V_{n,i}(t)$ or $I_{n,i}(t)$. She can, however, sniff packets, inspect their content, and intervene along the path from the energy source to the energy harvester attached to the device, including directly controlling the energy source. This means the attacker can alter the value of $V_{a,i}(t)$ taken as input by the energy management circuitry at node i. We model this as a function $a_i(V_{a,i}(t))$, that is, a transformation a from the voltage domain to the same domain, specific to node i.

An elementary example to cause a denial of service at node i from t' onwards is $a_i(V_{a,i}(t)) = 0, t > t'$, that is, the harvester at node i receives no energy after t'. The energy buffer at node i progressively discharges because of application processing and capacitor leakage, until the device persists the state before entering the charging phase, as shown in Fig. 1. However, because $a_i(V_{a,i}(t)) = 0, t > t'$, that is, there is no energy arriving at node i later than t', $C_i(t)$ never reaches V_{on} again, and node i never resumes the operation.

The attacker has access to application's source code or can reverse-engineer that from the binaries [66]. Codebases for battery-less IoT systems, including and especially the ones used in real deployments [1, 18, 25], are often public [6], including hardware drivers [7, 11, 15]. Compilers [28] often require the entire source code to perform full-program optimizations.

3.2 Evidence of Vulnerabilities

We systematically analyze and experimentally demonstrate three vulnerabilities. Two target single devices; their individual placement in space is therefore immaterial. The third one affects a whole network. Energy attacks exploiting these vulnerabilities may lead to *livelocks*, *starvation*, or *denial of service*.

Unless otherwise specified, we consider a TI MSP430FR5969 running at 1 MHz as target MCU and set V_{on} to 3.3 V and V_{off} to 1.8 V, which is the most energy-efficient setting [5]. Whenever communication is required, we employ a CC1101 transceiver. We use different methodologies and tools, including real hardware, numerical simulations, and time-accurate emulation.

Attack #1: static activation thresholds ⇒ livelock. The first vulnerability is based on how working parameters of system software for intermittent computing are determined. That allows an attacker to create a situation of livelock, that is, a condition where the system keeps repeating the same set of operations, and yet makes no progress in the long run [22].

Established design processes for intermittent systems recommend setting the activation threshold V_{on} by striking a balance between charging times and energy content when the system is at V_{on} [2, 10, 13, 23, 36, 40, 54, 61]. The former suggests a lower

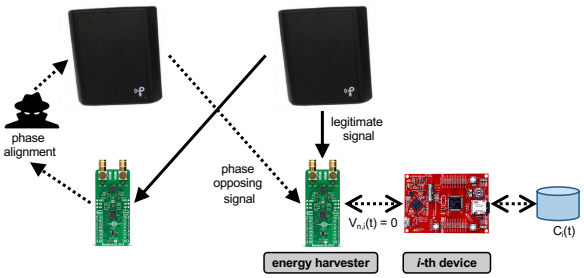


Figure 3: Generating an opposing RF energy signal by using a second Powercast transmitter-receiver pair. *The opposing signal creates destructive interference at the device under attack, canceling out the energy contribution of the legitimate Powercast transmitter.*

V_{on}, whereas the latter pushes for a higher V_{on}. In most existing systems [2, 10, 13, 40, 54, 61], V_{on} is statically set before deployment and does not necessarily guarantee that the energy content is sufficient to make progress in the application logic and persist the new state before an imminent energy failure. The ambient is supposed to provide some energy also during the active times, partially replenishing the energy buffer while the system progresses [42].

An attacker may, however, systematically block the energy source at a node i while the device is computing, that is, he creates a transformation a such that $a_i(V_{a,i}(t)) = 0, t' > t > t''$, where t' and t'' are the points in time where the system reaches V_{on} and V_{off}, respectively. As this transformation targets the individual device, it does not require any specific placement of a node with respect to the others or to the energy source. We demonstrate this situation using a TI MSP430FR5969 Launchpad and two Powercast transmitter-receiver pairs, running HarvOS [13], an existing checkpointing system for intermittent computing. The same situation can be achieved, for example, by controlling the incident solar radiation [12].

While the first Powercast transmitter normally powers the device, the attacker implements function a by generating an opposing signal with the second Powercast receiver-transmitter pair, as shown in Fig. 3. We achieve this by generating a signal in phase opposition with the regular one, yielding *destructive interference* [50, 58]. We describe in Sec. 4 how phase alignment can be empirically achieved. By doing so, the attacker cancels out the energy contribution of the legitimate Powercast transmitter. The attacker identifies t' and t'' by correlating wireless traffic to different points in the code.

Fig. 4 shows an example execution once the phase alignment is achieved. Without any contribution of energy during the active times and a V_{on} setting that does not account for this, the system approaches V_{off} with insufficient energy to persist state, that is, no new checkpoint is created. When the system reaches V_{on} again, HarvOS resorts to the previous checkpoint, that is, the one that does not include the progress achieved between t' and t''. The previous operations are then executed again, and with the attacker replaying the same function a once more, the system approaches V_{off} again with insufficient energy to persist the state. As long as the attacker keeps doing so, the system continues to restart from the same checkpoint, each time executing the same operations, and yet making no progress in the long run.

Note that in the case of RF energy harvesting, this attack may occur not just without physical access to the device, but also without

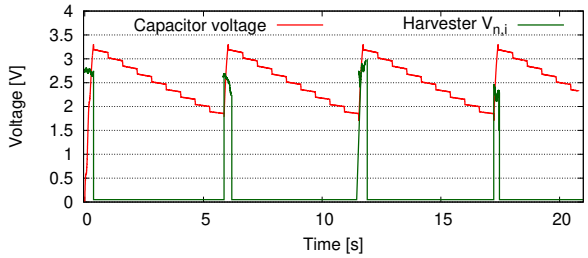


Figure 4: Livelock situation caused by systematically blocking the energy source while the device is active. *The discharge pattern is identical every time the device is computing: it always resumes from the same checkpoint and repeats the same set of instructions, without ever making progress since there is no energy left to persist the state when reaching V_{off}.*

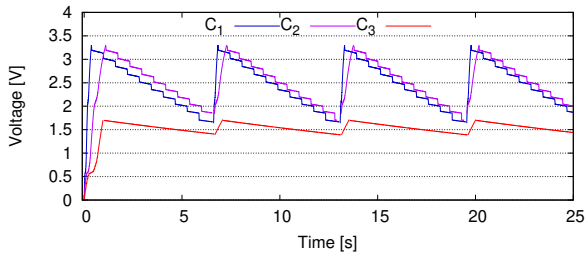


Figure 5: A case of starvation generated by purposely suspending energy provisioning. *The largest capacitor C_3 is never fully charged and the associated task never executes. This task stops consuming data from a non-volatile queue that eventually overflows.*

being anywhere close to it. As long as the attacker is in the (wireless) range of both the legitimate Powercast transmitter and of the device under attack, as shown in Fig. 3, she may detect both the incoming energy wave and regular network packets. With this information, the attacker may eventually generate the canceling energy signal using the procedure outlined in Sec. 4. In our setup, we manage to achieve this as long as the attacker is on either the same, or on adjacent floors than the device under attack, in a regular university building occupying approximately a 120 m x 50 m horizontal surface. Situations when the ambient provides no energy during executions may happen, yet not systematically [22, 61].

Attack #2: skewed energy management ⇒ starvation. When applied to a multi-capacitor architecture, this attack creates a situation we characterize as starvation. An attacker may ration the energy arriving at the harvester so that the smaller capacitors reach V_{on} more often than the bigger ones, while relying on the larger leakage of the latter to further slow down their charging.

This makes producer tasks, such as probing low-power sensors usually powered by the smaller capacitors, push data in the local data buffers more rapidly than tasks powered by larger capacitors, such as radio transmissions, which consume the data. The local buffers eventually overflow. To tune this attack, one can use simple energy profiling tools [2]. From an outsider perspective, the loss of data due to this attack is indistinguishable from other data losses, for example, due to packet losses during wireless transmissions.

To gain a precise understanding of the execution, we investigate this attack using a custom version of the Siren MSP430 emulator [27] we develop, which implements a multi-capacitor hardware architecture [23] and can re-play existing energy traces from an RF energy source [61]. The entire codebase of the simulator and the data enabling the study that follows are available [34]. We use three different capacitors C_1, C_2, and C_3, with $C_3 > C_2 > C_1$. We use C_1 for sensing from a low-power temperature sensor, C_2 to locally process the data, and C_3 for radio operation.

Fig. 5 shows the voltage levels at the three capacitors in an example execution. By periodically interrupting energy provisioning for sufficiently long that C_3 is never fully charged, the local data buffer eventually overflows. To achieve this in practice, the same technique as in the previous attack is applicable, which only requires generating a canceling signal that prevents RF energy harvesting to happen. The same considerations about the location of the attacker compared to the device under attack apply here as well.

A situation where buffers between tasks overflow is surprisingly easy to reach, due to technology limitations. Currently available NVMs are extremely limited in size; therefore, the local data buffers are normally dimensioned to store just a handful of entries and it does not take long until they fill up. These occurrences are sometimes reported also for systems that operate normally [1], yet this attack accelerates the likelihood of these occurrences.

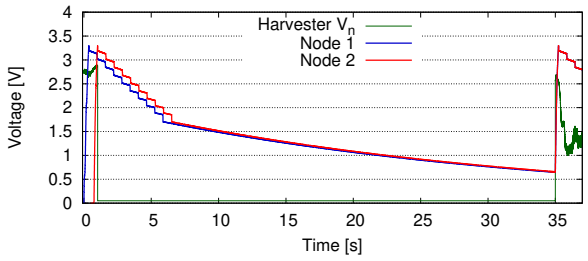
Attack #3: unwanted synchronization ⇒ denial of service. We investigate whether vulnerabilities exist that impact the network as opposed to single devices. We eventually figure out that energy attacks may be setup so that the transmissions of two or more devices systematically collide, degrading system performance.

In most IoT applications [1, 18, 20, 37], homogeneous devices are deployed in large numbers with identical hardware and execute the same sense-process-transmit loop [1, 18, 20, 37]. Co-located devices are therefore subject to almost identical energy patterns from the ambient [12, 38, 57]. An example is when relying on light sources [38, 57]. In these settings, $V_{a,i}(t)$ is the same for all i.

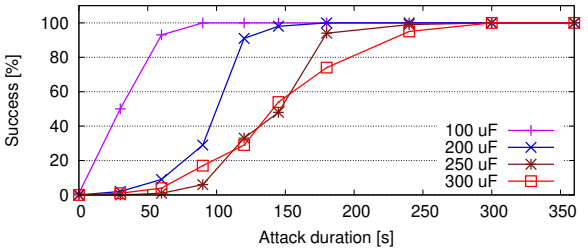
The simplest way of taking advantage of this configuration is first to block the energy intake for the nodes under attack, that is, $a_i(V_{a,i}(t)) = 0, \forall t > t'$. Eventually, $C_i(t''') = 0$ for some $t''' > t'$ for all nodes i under attack, because even if the devices do not compute, capacitors self-discharge because of leakage currents. At this point the attacker removes the blockage, restoring the original energy intake with $a_i(V_{a,i}(t)) = \mathbb{1}, \forall t > t'''$. All capacitors start charging in the same way and devices start operating simultaneously when they reach V_{on}. At this point, they are synchronized.

To investigate this situation, we develop a custom discrete-event simulator using **`SimPy`**. The simulator includes accurate numerical models of the essential electronics, including the energy harvester, the energy management circuitry attached to the output of the energy harvester, the capacitor with its leakage, and a load that models the IoT device. The input to the simulator is a real-world voltage trace from a solar panel deployed indoor [4]. As for the second attack, the entire codebase of the simulator, along with the data enabling the study that follows, is available [35].

Fig. 6(a) shows an execution of two devices 1 and 2 where the green curve represents $V_{n,1}(t) = V_{n,2}(t)$, whereas the red and blue curves represent $C_1(t)$ and $C_2(t)$, respectively. The attacker blocks the energy source at time $t' = 1$ s; both C_1 and C_2 decrease rapidly while the devices continue the execution. When device 2 dumps the state on NVM and switches off at time $t'' = 7$ s, being $a_1(V_{a,1}(t)) =$



(a) A case of two nodes synchronizing by blocking the energy source.



(b) Attack duration Δt and success probability against capacitor size.

Figure 6: Denial of service created by artificially synchronizing nodes. *By exploiting capacitor leakage, an attacker synchronizes nodes so their packet transmissions systematically collide.*

$a_2(V_{a,2}(t)) = 0, t' < t < t''', t''' = 35$ s, and provided the duration of the blockage $\Delta t = t''' - t'$ is sufficiently large, capacitor leakage eventually leads to $C_1(t) \approx C_2(t) \approx 0$.

At time t''', the attacker removes the blockage, thus $a_1(V_{a,1}(t)) = a_2(V_{a,2}(t)) = \mathbb{1}, t > t'''$. From now on, the two capacitors charge up in the same way, as they are subject to the same ambient energy: the red and blue curves in Fig. 6(a) almost perfectly overlap. The two devices reach V_{on} at the same time and restart their execution by going through the same operations at the same times, leading to transmitting simultaneously, generating a packet collision.

The attacker has no information on C_1 or C_2. The attack may be unsuccessful if Δt is too small; for example, because C_1 and C_2 do not arrive at roughly the same energy content, which is a prerequisite for generating the synchronous execution afterwards. The attacker may use increasing Δt until successful. Even if the nodes are not perfectly synchronized, moving their transmissions closer in time puts increasing pressure on collision avoidance and backoff techniques, which are confronted with an artificial situation that would otherwise be extremely rare. Network throughput and packet latency are consequently degraded.

Many traits of this scenario are found in real deployments [1, 18, 20, 37]. Relying on the assumption that energy intakes at co-located nodes are similar is not just common, but even used as a basis to implement communication protocols in networks of intermittently-computing IoT devices [29]. With this condition, playing this attack in practice may be as simple as blocking the single energy source that the nodes under attack rely on. As an example, in an indoor setting that uses light sources to harvest energy, controlling the lights by tapping into any of the modern building management systems [41, 69] may provide a suitable entry point.

Two key parameters that determine the probability of success are Δt and capacitor size. Using our simulation tool, we evaluate their impact on the attack's success probability with two nodes. The results are shown in Fig. 6(b). As expected, the probability of a successful attack increases with Δt. This is because the longer is the time the attacker blocks the energy source, the higher are the chances that eventually $C_1(t) \approx C_2(t) \approx 0$. With a smaller capacitor, smaller Δt are sufficient for a successful attack, as $C_1(t) \approx C_2(t) \approx 0$ is reached faster. Further quantitative data is available [35].

4 ENERGY ATTACKS IN PRACTICE

Although energy attacks on real deployments have, to the best of our knowledge, not been observed, we provide here evidence that such attacks could occur in practice and offer an indication of the skills required and corresponding level of expertise.

Setting. Prior to the systematic study presented earlier, we recruit eight computer engineering M.Sc. students[1]. They attended courses in low-power wireless networks, embedded programming, and IoT but have no earlier training on battery-less IoT devices. We hand them reading material on existing techniques [13, 52] and provide a TI MSP430FR5969 Launchpad attached to either a 40 x 40 mm polycrystalline silicon solar panel or to a Powercast receiver, plus a two-capacitor energy subsystem [23] that we emulate using an Arduino Uno board as energy controller. Each Powercast receiver is paired to a single Powercast transmitter. Each students can choose which energy harvester to use.

We give the students the C implementation of a typical sense-process-transmit loop [1, 18, 20, 37]. After a single day of training, the students can run the code on the Launchpads. Six students use HarvOS [13] and only minimally refactor the original application code. Out of the six students, four use the solar panel and the other two use the Powercast system. The remaining two students use the solar panel and and split the code in tasks, using Alpaca [52], an existing task-based programming abstraction.

We challenge the students with the following goal: *without changing the code, disrupt the application without completely halting the system or physically accessing the device.* The students work independently and are not allowed to exchange ideas, information, or code. We also make sure they cannot rely on backchannel information from each other, for example, by sniffing wireless transmissions. Besides observing their work, we conduct semi-structured interviews at the end of every day. The transcripts are available [33].

Outcome. The students recognize that, without physical access to the device, energy harvesting is a potential attack vector. What they eventually setup are, interestingly, rudimentary instances of some of the energy attacks we demonstrate in Sec. 3.

After two full days of experimentation, five of the six students using HarvOS figure out they can break the system by creating a livelock, similar to attack #1. During the interview, four students explicitly mention"livelock", while the fifth student intuitively describes what is, in fact, a livelock. They spend the following days trying to create the conditions that lead to the livelock. Depending on the harvesting technology, they end up conceiving different techniques to achieve this.

Two students using the solar panel succeed halfway through the third day by intentionally alternating periods of blocking the energy source with periods of regular operation, based on information from

[1]We obtained IRB approval from their institution. They are not compensated.

sniffed packets indicating at what stage is the execution. The third student using the solar panel succeeds using a similar technique half a day later. After two weeks, one student succeeds in making the attack automatic. Using a light sensor on a separate device, he estimates the amount of energy harvested by the target device and accordingly tunes the periods of energy blockage.

After about five days of work, the other two students using HarvOS and the Powercast system create a setup with a second Powercast receiver-transmitter pair generating a signal in phase opposition with the regular one, yielding destructive interference similar to Sec. 3. To identify the "right" phase, they attach the second receiver to a laptop that controls the second Powercast transmitter. The laptop replicates the regular signal obtained from the attached Powercast receiver with a variable phase within a $[-\lambda/2, \lambda/2]$ interval, at small increments of δ. Sniffing packets from the target device allows them to determine when to stop the process, which corresponds to when they hit the "right" phase that almost cancels out the original energy signal [50, 58]. By doing so periodically, they eventually achieve the same effect as with the solar panel.

It takes a bit more for the two students using solar panels and Alpaca to realize they can fiddle with the multiple capacitors. After about two weeks, each of them creates a setup that blocks the energy arriving at the largest capacitor to slow down the task that takes energy from that. This, combined with the larger leakage compared to the smaller capacitor, makes the task powered by the larger capacitor progressively starve. As the tasks powered by the smaller capacitors continue producing data almost at the regular rate, yet the tasks powered by the larger capacitors do not consume data at comparable pace, the buffers between the tasks eventually overflow, creating an instance of attack #2.

Take-away. In a matter of weeks, seven computer engineering M.Sc. students out of eight, with no previous exposure to the technology at stake, managed to successfully setup two of the energy attacks described earlier. The students were randomly selected and had no special equipment available other than what is normally found in an embedded system lab of any technological university. Likely because research efforts in intermittent computing concentrate on issues such as maintaining forward progress, understanding the related security issues is much less investigated. This includes both investigating known attacks and exploring new types of threats.

5 DISCUSSION

The feasibility of energy attacks depends not just on the abilities of the attacker, but also on the energy source.

Certain energy sources are simple to control, such as RF energy: a Powercast system can be hooked to a regular machine and controlled programmatically. Other sources may be controlled under given conditions. Light, for example, is controllable in indoor environments: an attacker may gain control of the lighting infrastructure in a building, as it is often assumed in visible light communications [41, 69], and use that to convey the attack. This setup should, however, work in combination with other light sources that cannot be as easily controlled, for example, solar radiation from the outside. Energy sources also exist where the physical medium cannot change sufficiently rapidly, for example, temperature gradients. These sources are unlikely to be attack vectors.

An orthogonal dimension relates to time. Depending on the source, the attacker may need a variable number of attempts before an attack succeeds. For instance, how rapidly attack #1 using RF energy is going to succeed depends on the choice of δ, which drives the search of the phase corresponding to the opposing signal. If δ is too large, the procedure may never yield the conditions for the attack to succeed; the attacker then chooses a smaller δ and repeats the procedure. A similar consideration applies to attack #2 and Δt.

Lacking source code information may make setting up certain energy attacks extremely laborious, whereas the same information is not as fundamental in other cases. For attack #3, for example, it would suffice to know that the application logic is the same across all devices and unfolds as the usual sense-process-transmit loop.

6 OUTLOOK

We studied how exerting limited control on ambient energy provisioning to battery-less IoT devices may be used as an attack vector. We detailed, analyzed and provided experimental evidence of three types of novel energy attacks causing livelock, denial of service, and starvation. We also provided evidence of the skills required and corresponding expertise to exercise these attacks in practice.

Our work fosters follow-up efforts in at least two directions. First, energy attacks must be *detected*. This means understanding when ambient energy provisioning does not follow the "natural" patterns, which appears as a case of anomaly detection [17]. Three peculiar requirements exist: detecting energy attacks *i)* accurately and *ii)* with low latency, while doing so *iii)* right on the IoT devices, as opposed to an external system, to spare the energy overhead of radio operations. The problem is difficult because, for example, energy forecasting techniques [16] are not applicable, in that instead of predicting future energy supplies, we are to understand when current energy supplies follow abnormal patterns.

Second, the system should *defend* against energy attacks, which is a manifold problem. For example, key performance metrics are inherently application-specific. Defense techniques may be designed that generally tame the negative effects of energy attacks, independent of their specific nature, or be tailored to specific attacks. As the feasibility of energy attacks is tied to the energy source, combining multiple energy sources [49] may be an option to defend, for example. How exactly to combine energy-rich sources that may be attack vectors, with energy-poor sources that are exceedingly difficult to employ as such, looks however non-trivial.

Acknowledgments. The authors wish to thank Andrea Terrifico, Gianluca Marveli, Giovanni Macchi, Simone Salvi, Andrea Saviotti, Lorenzo Tarelli, Julia Stivallato, and Golsa Serati for contributing to the user study in Sec. 4. This work is partially supported by the Swedish Science Foundation (SSF), by the Swedish Research Council (grant 2021-04968), and by the National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.3 - Call for tender No. 1561 of 11.10.2022 of Ministero dell'Università e della Ricerca (MUR); funded by the European Union - NextGenerationEU.

REFERENCES

[1] M. Afanasov et al. 2020. Battery-Less Zero-Maintenance Embedded Sensing at the Mithræum of Circus Maximus. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SENSYS)*.

[2] S. Ahmed et al. 2019. Efficient Intermittent Computing with Differential Checkpointing. In *Proceedings of the ACM International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES)*.

[3] S. Ahmed et al. 2024. The Internet of Batteryless Things. *Commun. ACM* 67, 3 (2024).

[4] S. Ahmed, A. Bakar, N. A. Bhatti, M. H. Alizai, J. H. Siddiqui, and L. Mottola. 2019. The Betrayal of Constant Power × Time: Finding the Missing Joules of Transiently-powered Computers. In *Proceedings of the International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES)*.

[5] Saad Ahmed, Qurat Ul Ain, Junaid Haroon Siddiqui, Luca Mottola, and Muhammad Hamad Alizai. 2020. Intermittent Computing with Dynamic Voltage and Frequency Scaling.. In *Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN)*.

[6] Mahdi Amiri-Kordestani and Hadj Bourdoucen. 2017. A survey on embedded open source system software for the internet of things. In *Free and Open Source Software Conference*.

[7] A. R. Arreola, D. Balsamo, G. V. Merrett, and A. S. Weddell. 2018. RESTOP: Retaining External Peripheral State in Intermittently-Powered Sensor Systems. *Sensors* (2018).

[8] H. A. Asad et al. 2020. On securing persistent state in intermittent computing. In *International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems*.

[9] D. Balsamo, A. S. Weddell, A. Das, A. R. Arreola, D. Brunelli, B. M. Al-Hashimi, G. V. Merrett, and L. Benini. 2016. Hibernus++: A Self-Calibrating and Adaptive System for Transiently-Powered Embedded Devices. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2016).

[10] D. Balsamo, A. S. Weddell, G. V. Merrett, B. M. Al-Hashimi, D. Brunelli, and L. Benini. 2015. Hibernus: Sustaining Computation During Intermittent Supply for Energy-Harvesting Systems. *IEEE Embedded Systems Letters* (2015).

[11] G. Berthou, T. Delizy, K. Marquet, T. Risset, and G. Salagnac. 2018. Sytare: a Lightweight Kernel for NVRAM-based Transiently-Powered Systems. *IEEE Trans. Comput.* (2018).

[12] N. A. Bhatti, M. H. Alizai, A. A. Syed, and L. Mottola. 2016. Energy Harvesting and Wireless Transfer in Sensor Network Applications: Concepts and Experiences. *ACM Transactions on Sensor Networks* (2016).

[13] N. A. Bhatti and L. Mottola. 2017. HarvOS: Efficient Code Instrumentation for Transiently-powered Embedded Sensing. In *Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*.

[14] Julia Borghoff et al. 2012. PRINCE–a low-latency block cipher for pervasive computing applications. In *International Conference on the Theory and Application of Cryptology and Information Security*.

[15] A. Branco, L. Mottola, M. H. Alizai, and J. H. Siddiqui. 2019. Intermittent Asynchronous Peripheral Operations. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (SENSYS)*.

[16] Alessandro Cammarano, Chiara Petrioli, and Dora Spenza. 2012. Pro-Energy: A novel energy prediction model for solar and wind energy-harvesting wireless sensor networks. In *International Conference on Mobile Ad-Hoc and Sensor Systems (MASS)*.

[17] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)* 41, 3 (2009).

[18] Q. Chen, Y. Liu, G. Liu, Q. Yang, X. Shi, H. Gao, L. Su, and Q. Li. 2017. Harvest Energy from the Water: A Self-Sustained Wireless Water Quality Sensing System. *ACM Transactions on Embedded Computing Systems* (2017).

[19] Shenglei Chen, Dongyang Ou, Congfeng Jiang, Jing Shen, Li Yan, and Shuangshuang Guo. 2020. Power Attack and Detection Technology in Data Centers: A Survey. In *International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI)*.

[20] H. Chiang, J. Hong, K. Kiningham, L. Riliskis, P. Levis, and M. Horowitz. 2018. Tethys: Collecting Sensor Data without Infrastructure or Trust. In *Proceedings of the 3rd IEEE/ACM International Conference on Internet-of-Things Design and Implementation (IoTDI)*.

[21] A. Colin and B. Lucia. 2016. Chain: Tasks and Channels for Reliable Intermittent Programs. In *Proceedings of the ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*.

[22] A. Colin and B. Lucia. 2018. Termination Checking and Task Decomposition for Task-based Intermittent Programs. In *Proceedings of the International Conference on Compiler Construction (CC)*.

[23] A. Colin, E. Ruppel, and B. Lucia. 2018. A Reconfigurable Energy Storage Architecture for Energy-Harvesting Devices. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*.

[24] Kelton AP da Costa, João P Papa, Celso O Lisboa, Roberto Munoz, and Victor Hugo C de Albuquerque. 2019. Internet of Things: A survey on machine learning-based intrusion detection approaches. *Computer Networks* 151 (2019).

[25] Bradley Denby, Krishna Chintalapudi, Ranveer Chandra, Brandon Lucia, and Shadi Noghabi. 2023. Kodan: Addressing the Computational Bottleneck in Space. In *Proceedings of theInternational Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*.

[26] F. Fraternali, B. Balaji, Y. Agarwal, L. Benini, and R. Gupta. 2018. Pible: Battery-Free Mote for Perpetual Indoor BLE Applications. In *Proceedings of the 5th Conference on Systems for Built Environments*.

[27] M. Furlong, J. Hester, K. Storer, and J. Sorber. 2016. Realistic Simulation for Tiny Batteryless Sensors. In *Proceedings of the International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems (ENSSYS)*.

[28] David Gay, Philip Levis, Robert Von Behren, Matt Welsh, Eric Brewer, and David Culler. 2003. The nesC language: A holistic approach to networked embedded systems. *Acm Sigplan Notices* 38, 5 (2003).

[29] Kai Geissdoerfer and Marco Zimmerling. 2022. Learning to Communicate Effectively Between Battery-free Devices. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*.

[30] Zahra Ghodsi, Siddharth Garg, and Ramesh Karri. 2017. Optimal checkpointing for secure intermittently-powered IoT devices. In *International Conference on Computer-Aided Design (ICCAD)*.

[31] Albert Greenberg, James Hamilton, David A Maltz, and Parveen Patel. 2008. The cost of a cloud: research problems in data center networks.

[32] Michele Grisafi, Mahmoud Ammar, Kasim Sinan Yildirim, and Bruno Crispo. 2022. MPI: Memory Protection for Intermittent Computing. *IEEE Transactions on Information Forensics and Security* 17 (2022).

[33] Arslan Hameed et al. 2023. Exploratory Study Interviews. https://www.dropbox.com/s/3rt1z2acsbntc41/interviews-anon.pdf?dl=0

[34] Arslan Hameed et al. 2023. Starvation Attack Simulator and Data. https://www.dropbox.com/s/gh634z2ascacce23/priority-attack.zip?dl=0

[35] Arslan Hameed et al. 2023. Synchronization Attack Simulator and Data. https://www.dropbox.com/s/39123z2asdantd01/synch-attack.zip?dl=0

[36] J. Hester and J. Sorber. 2017. Flicker: Rapid Prototyping for the Batteryless Internet-of-Things. In *Proceedings of the International Conference on Embedded Network Sensor Systems (SENSYS)*.

[37] N. Ikeda, R. Shigeta, J. Shiomi, and Y. Kawahara. 2020. Soil-Monitoring Sensor Powered by Temperature Difference between Air and Shallow Underground Soil. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT)* (2020).

[38] Bashima Islam, Yubo Luo, and Shahriar Nirjon. 2023. Amalgamated Intermittent Computing Systems. In *Proceedings of the International Conference on Internet of Things Design and Implementation (IoTDI)*.

[39] N. Jackson, J. Adkins, and P. Dutta. 2019. Capacity over Capacitance for Reliable Energy Harvesting Sensors. In *Proceedings of the 18th International Conference on Information Processing in Sensor Networks (IPSN)*.

[40] H. Jayakumar, A. Raha, W. S. Lee, and V. Raghunathan. 2015. QuickRecall: A HW/SW Approach for Computing Across Power Cycles in Transiently Powered Computers. *ACM Journal on Emerging Technologies in Computing Systems* (2015).

[41] Aleksandar Jovicic, Junyi Li, and Tom Richardson. 2013. Visible light communication: opportunities, challenges and the path to market. *IEEE Communications Magazine* 51, 12 (2013).

[42] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava. 2007. Power Management in Energy Harvesting Sensor Networks. *ACM Transactions on Embedded Computing Systems* (2007).

[43] John Kanwar, Niclas Finne, Nicolas Tsiftes, Joakim Eriksson, Thiemo Voigt, Zhitao He, Christer Åhlund, and Saguna Saguna. 2021. JamSense: Interference and Jamming Classification for Low-power Wireless Networks. In *IFIP Wireless and Mobile Networking Conference (WMNC)*.

[44] Konrad-Felix Krentz, Christoph Meinel, and Hendrik Graupner. 2017. Countering Three Denial-of-Sleep Attacks on ContikiMAC. In *International Conference on Embedded Wireless Systems and Networks (EWSN)*.

[45] A. Krishnan and P. Schaumont. 2018. Exploiting Security Vulnerabilities in Intermittent Computing. In *SPACE International Conference*.

[46] A. Krishnan, C. Suslowicz, D. Dinu, and P. Schaumont. 2019. Secure intermittent computing protocol: Protecting state across power loss. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*.

[47] Archanaa S. Krishnan and Patrick Schaumont. 2022. Benchmarking And Configuring Security Levels In Intermittent Computing. *ACM Transactions on Embedded Computing Systems* (2022).

[48] Chao Li, Zhenhua Wang, Xiaofeng Hou, Haopeng Chen, Xiaoyao Liang, and Minyi Guo. 2016. Power attack defense: Securing battery-backed data centers. *ACM SIGARCH Computer Architecture News* 44, 3 (2016).

[49] Huicong Liu, Hailing Fu, Lining Sun, Chengkuo Lee, and Eric M Yeatman. 2021. Hybrid energy harvesting technology: From materials, structural design, system integration to applications. *Renewable and sustainable energy reviews* 137 (2021).

[50] Qingzhi Liu, Kasım Sinan Yildirim, Przemysław Pawełczak, and Martijn Warnier. 2016. Safe and secure wireless power transfer networks: Challenges and opportunities in RF-based systems. *IEEE Communications Magazine* 54, 9 (2016), 74–79.

[51] B. Lucia and B. Ransford. 2015. A Simpler, Safer Programming and Execution Model for Intermittent Systems. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*.

[52] K. Maeng, A. Colin, and B. Lucia. 2017. Alpaca: Intermittent Execution Without Checkpoints. *Proceedings of the ACM Programming Languages* (2017).

[53] K. Maeng and B. Lucia. 2018. Adaptive dynamic checkpointing for safe efficient intermittent computing. In *Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*.

[54] K. Maeng and B. Lucia. 2019. Supporting Peripherals in Intermittent Systems with Just-in-Time Checkpoints. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*.

[55] Andrea Maioli and Luca Mottola. 2021. ALFRED: Virtual Memory for Intermittent Computing. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems (SENSYS)*.

[56] A. Y. Majid, C. Delle Donne, K. Maeng, A. Colin, K. S. Yildirim, B. Lucia, and P. Pawelczak. 2020. Dynamic Task-Based Intermittent Execution for Energy-Harvesting Devices. *ACM Transactions on Sensor Networks* (2020).

[57] Amjad Yousef Majid, Patrick Schilder, and Koen Langendoen. 2020. Continuous sensing on intermittent power. In *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*.

[58] M Yousof Naderi, Prusayon Nintanavongsa, and Kaushik R Chowdhury. 2014. RF-MAC: A medium access control protocol for re-chargeable sensor networks powered by wireless energy harvesting. *IEEE Transactions on Wireless Communications* 13, 7 (2014).

[59] Van-Linh Nguyen, Po-Ching Lin, and Ren-Hung Hwang. 2019. Energy depletion attacks in low power wireless networks. *IEEE Access* 7 (2019).

[60] Jorge Portilla, Andrés Otero, Eduardo de la Torre, Teresa Riesgo, Oliver Stecklina, Steffen Peter, and Peter Langendörfer. 2010. Adaptable security in wireless sensor networks by using reconfigurable ECC hardware coprocessors. *International Journal of Distributed Sensor Networks* 6, 1 (2010).

[61] B. Ransford, J. Sorber, and K. Fu. 2011. Mementos: System Support for Long-running Computation on RFID-scale Devices. *ACM SIGARCH Computer Architecture News* (2011).

[62] E. Ruppel and B. Lucia. 2019. Transactional Concurrency Control for Intermittent, Energy-harvesting Computing Systems. In *Proceedings of the International Conference on Programming Language Design and Implementation (PLDI)*.

[63] Syeda Manjia Tahsien, Hadis Karimipour, and Petros Spachos. 2020. Machine learning based solutions for security of Internet of Things (IoT): A survey. *Journal of Network and Computer Applications* 161 (2020).

[64] Texas Instruments. 2017 (last access: May 13th, 2020). MSP430-FR5969 datasheet. https://www.ti.com/lit/ds/symlink/msp430fr5969.pdf.

[65] V. A. Thakor, M. A. Razzaque, and M. R. A. Khandaker. 2021. Lightweight cryptography algorithms for resource-constrained IoT devices: A review, comparison and research opportunities. *IEEE Access* 9 (2021).

[66] Sharath K Udupa, Saumya K Debray, and Matias Madou. 2005. Deobfuscation: Reverse engineering obfuscated code. In *Working Conference on Reverse Engineering (WCRE)*.

[67] Emanuele Valea, Mathieu Da Silva, Giorgio Di Natale, Marie-Lise Flottes, Sophie Dupuis, and Bruno Rouzeyre. 2018. SI ECCS: SECure context saving for IoT devices. In *International Conference on Design & Technology of Integrated Systems In Nanoscale Era (DTIS)*.

[68] J. Van Der Woude and M. Hicks. 2016. Intermittent Computation Without Hardware Support or Programmer Intervention. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI)*.

[69] Weitao Xu, Jin Zhang, Jun Young Kim, Walter Huang, Salil S Kanhere, Sanjay K Jha, and Wen Hu. 2019. The design, implementation, and deployment of a smart lighting system for smart buildings. *IEEE Internet of Things Journal* 6, 4 (2019), 7266–7281.

[70] K. S. Yildirim, A. Y. Majid, D. Patoukas, K. Schaper, P. Pawelczak, and J. Hester. 2018. InK: Reactive Kernel for Tiny Batteryless Sensors. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems (SENSYS)*.