Micro-Auditing for Ransomware Detection in Resource-Constrained Industrial IoT Networks

Yuxiang Huang
University of Bristol
United Kingdom
yuxiang.huang@bristol.ac.uk

Calvin Brierley
University of Kent
United Kingdom
c.brierley@kent.ac.uk

James Pope
University of Bristol
United Kingdom
james.pope@bristol.ac.uk

Jiteng Ma
University of Bristol
United Kingdom
jiteng.ma@bristol.ac.uk

Antonio Di Buono
UKNNL
United Kingdom
antonio.dibuono@uknnl.com

Budi Arief
University of Kent
United Kingdom
b.arief@kent.ac.uk

George Oikonomou
University of Bristol
United Kingdom
george.oikonomou@bristol.ac.uk

ABSTRACT

The threat of ransomware on severely constrained industrial Internet of Things (IoT) deployments is real and difficult to defend against, especially because resource-constrained devices can be compromised and used to propagate malicious payloads with minimal observability. To support the development of ransomware detection and prevention countermeasures, this paper proposes a lightweight micro-auditing mechanism that captures diagnostic metrics derived from process scheduling, as well as from resource and network utilization patterns. The proposed mechanism minimizes computational, memory, and energy overhead through adaptive metric sampling, ensuring resource-efficient operation. We have developed this micro-auditing mechanism using the Contiki-NG operating system for IoT devices, and we use our implementation to derive memory and code footprint statistics as evidence of its lightweight nature. Using the Cooja simulator and an existing ransomware prototype, we examine the feasibility of our micro-auditing mechanism through a host of experiments on topologies of different densities, to quantify the speed and subtle nature of ransomware propagation. Our results highlight the fact that, due to its subtlety, this threat can elude traditional traffic- and power-based anomaly detectors. The micro-auditing mechanism not only enables device-level security auditing but also underpins our ongoing work on the development of countermeasures using a scalable framework for integrating machine learning classifiers, which could further refine threat discrimination and reduce false positives.

CCS CONCEPTS

• **Security and privacy → Embedded systems security**; **Malware and its mitigation**; *Mobile and wireless security*.

KEYWORDS

Industrial IoT, ransomware, auditing, constrained devices

1 INTRODUCTION

Wireless sensor networks (WSNs) serve as a foundational component of the Internet of Things (IoT), enabling critical applications such as environmental monitoring and industrial automation. However, the inherent resource limitations and decentralized architectures make WSNs potentially vulnerable to sophisticated attacks.

Ransomware, once confined to personal computers (PCs) and enterprise systems, has increasingly targeted IoT ecosystems, employing lightweight cryptographic payloads and peer-to-peer propagation to disrupt sensor networks. Traditional defense mechanisms, such as signature-based detection or centralized monitoring, are often inadequate in IoT environments due to their high overhead and inability to adapt to the dynamic and distributed network topologies. The stealthy propagation and persistent impact of ransomware in IoT systems underscore the urgent demand for novel approaches tailored to the unique constraints of these networks, ensuring robustness against both known and evolving threats.

In order to maintain high levels of digital security, it is prudent to test technology and its robustness to threats. In order to test security measures for this type of ransomware, in our previous work [6] we used Contiki-NG [9] to implement a prototype that demonstrated that the development of a ransomware of this nature is feasible, and we evaluated it to better understand the threat. This prototype leverages the inherent characteristics of microcontrollers and wireless communication protocols to propagate ransomware code across devices autonomously. Key features include: (1) machine-to-machine transmission during runtime, enabling self-replication without user interaction; (2) minimal resource footprint, ensuring compatibility with low-power devices; and (3) operational stealth, as it avoids triggering noticeable changes in network traffic or device power consumption. By embodying these features, this prototype reveals critical challenges in detecting subtle, resource-efficient ransomware in IoT networks.

It is worth noting that conventional anomaly detection systems – which prioritize abrupt volumetric traffic anomalies or bulk power-consumption deviations – are fundamentally ill-suited to identify protocol-compliant attacks that exploit fine-grained behavioral

Table 1: Existing Log and Audit Techniques for IoT Network

Technique	Record data	Security trace	Data size	Record frequency	Storage	On device processing	Platform
Microcontroller unit (MCU) Log [2]	State, debugging snapshots	No	Configurable	High	External Flash	Yes	MCU
Syslog [12]	System services, kernel snapshots	No	Moderate	Event-driven	Local file system	Yes	Linux-based
Auditd [11]	System calls, file/network events	Yes	Large	Event-driven	Local file system	Yes	Linux-based
AWS IoT Device Defender [4]	Device behavior metrics	Yes	Moderate	Minutes interval	Upload to cloud	No	MCU + server
Azure IoT [3]	Diagnostics messages	Yes	Moderate	Seconds Interval	Upload to application	No	MCU + PC

patterns. This deficiency is compounded by the decentralized architecture of IoT ecosystems, which hinders real-time monitoring and response mechanisms.

To address these systemic vulnerabilities, this paper proposes a resource-aware, on-device auditing framework that operates within the stringent energy and memory constraints of IoT devices. This novel micro-audit paradigm dynamically tracks security-relevant program behaviors through a combination of lightweight diagnostic metrics and adaptive sampling strategies. By modulating sampling rates, the system maintains optimal detection sensitivity while preserving resource efficiency, thereby enabling scalable deployment in large-scale IoT environments.

Contributions. The key contributions of this work include:

- *Ransomware infection evaluation*: we develop a ransomware prototype on the network test bench and quantify its propagation speed across networks of varying density, as well as its stealthy resource footprint in terms of central processing unit (CPU) utilization, radio activity, and traffic throughput.
- *Micro-audit mechanism development*: we introduce a lightweight auditing mechanism that captures diagnostic metrics and dynamically adjusts its sampling frequency to balance detection responsiveness with minimal computation and energy overhead.
- *Evaluation of detection effectiveness*: we demonstrate that our micro-audit approach reliably distinguishes ransomware infections from normal operation using only subtle metric deviations, while maintaining a small resource footprint suitable for resource-constrained IoT devices.

2 RELATED WORK

2.1 Embedded Logging System on Microcontrollers

Existing logging mechanisms on resource-constrained IoT devices generally record system events, states, and interactions to facilitate debugging and enhance operational visibility. These mechanisms employ a hierarchical severity model, comprising debug, info, warning, error, and critical, to filter and prioritize operational data, which can be batch-recorded locally. However, existing microcontroller logging techniques are inadequate for the lightweight security tracing required to detect anomalous intrusions in real time, as summarized in Table 1.

Redundant log records – primarily used for troubleshooting and offering limited utility in monitoring critical system security – generate data volumes that often exceed on-chip storage capacity, necessitating frequent offloading to external storage. However, frequently storing logging data to large-capacity external flash memory cards significantly shortens the battery's lifespan [2]. While some studies have sought to reduce energy consumption by triggering logging via external events and peripherals, they have failed to continuously monitor system behaviors at the security level, potentially overlooking abnormal intrusions [10].

2.2 Auditing Systems Implementation on Multi-Platform

WSN audit systems require a hierarchical three-layer architecture—application, edge, and sensing—to align framework structure with data reporting from remote to local levels.

2.2.1 Application Layer. At the highest layer, cloud platforms host resource-intensive security services, such as Amazon Web Services (AWS) IoT Device Defender, that leverage a centralized processing model to analyze aggregated network traffic and sensor data from distributed WSN nodes [4]. These services rely on continuous data uploads to perform system-level anomaly detection, identifying patterns indicative of large-scale data exfiltration and coordinated denial-of-service attacks [1].

The typical Azure IoT solution enables efficient resource monitoring across heterogeneous systems by leveraging OS-native logging tools, thereby minimizing performance overhead [3]. However, its reliance on a computer-based collection node limits applicability in IoT scenarios, as many embedded systems lack native logging capabilities for dataset collection [7]. Cloud-based auditing introduces latency due to data transmission dependencies and may fail to detect localized, time-sensitive threats before escalation.

2.2.2 Edge Layer. At the intermediate layer, gateway routers deploy lightweight audit systems to analyze regional network traffic and sensor data over WSNs. These systems operate at the edge, reducing reliance on cloud connectivity by preprocessing data locally.

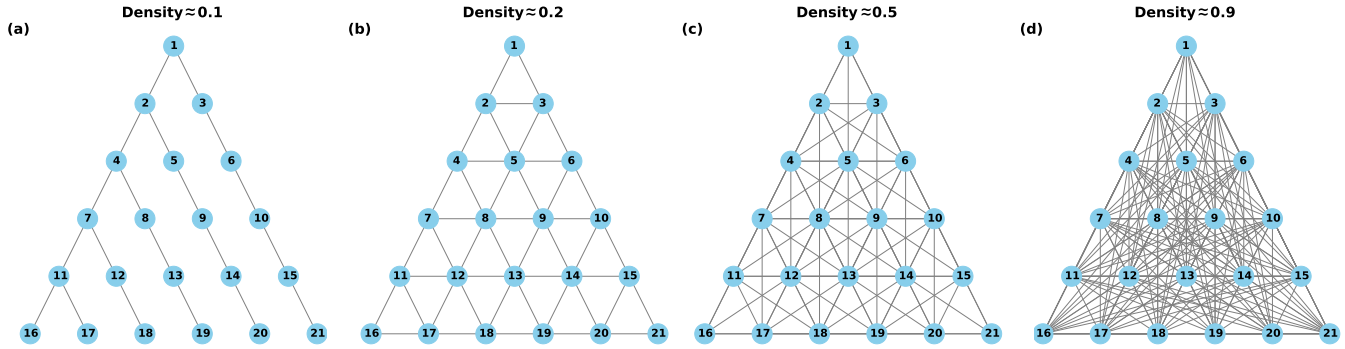

Figure 1: Network test bench with varied density levels. The sub-figures show: (a) target density ≈ 0.1, actual density = 0.0952; (b) target density ≈ 0.2, actual density = 0.2143; (c) target density ≈ 0.5, actual density = 0.5000; (d) target density ≈ 0.9, actual density = 0.9571.

For example, the audited Linux subsystem provides an adaptive safeguarding framework to detect malicious events, support incident response, and perform forensic investigations [11].

However, similar techniques can introduce severe performance overhead, slow down workflow, generate redundant events, and rapidly increase log file sizes [12]. Furthermore, its dependence on external energy meters or side channel sensors limits its feasibility in embedded systems without such an infrastructure [8].

2.2.3 Sensing Layer. At the lowest layer of WSNs, ultra-lightweight audit systems are deployed directly on resource-constrained sensor nodes to validate readings in real time with minimal overhead. One example of such an approach analyzes the microcontroller's instantaneous power-consumption waveform to extract discriminative features and identify malfunctions. Although effective for gross anomalies, this power-analysis method struggles to detect faults that induce only subtle changes in consumption, as minor fluctuations often fall within the device's normal operating range and thus evade detection [5].

An alternative strategy leverages temperature "fingerprints" inherent to microcontrollers' thermal profiles for intrusion detection – benefiting from universality across heterogeneous Industrial IoT deployments – yet remains highly susceptible to ambient temperature fluctuations and incapable of identifying malicious code that leaves the thermal signature largely unchanged [13]. Moreover, the severe power, computation, and memory constraints of in-node MCUs impose strict limits on the complexity of any anomaly-detection algorithm, necessitating a careful balance between detection capability and resource consumption.

3 MOTIVATION FOR AN AUDITING SYSTEM

3.1 Ransomware Prototype in Network

To help increase robustness against these threats, we used our ransomware proof-of-concept [6] to create autonomously-propagating code that can infect devices throughout a deployment in a manner that is difficult to detect. Once activated, infected devices exhibit behavioral anomalies intended to coerce users into paying ransom.

Assuming an initial compromise via physical access, our ransomware prototype dynamically injects malicious code upon receipt and redirects program flow using a function-pointer hijack. The ransomware payload then executes seamlessly at runtime, overriding normal operations and propagating itself to neighboring devices. Throughout this process, each device remains fully functional and appears inactive. Once the ransomware has self-propagated to a predetermined number of nodes, it activates its attack mode by deploying its payload (including the ransom note). To ensure persistence, the ransomware overwrites the device's flash at the original process-function address with its malicious routine, guaranteeing survival across reboots and sustained control.

The prototype exhibits two key characteristics: spreading capability and stealth. It propagates through machine-to-machine communication, infecting without producing noticeable changes in network traffic or device power consumption, thereby making detection difficult.

3.2 Simulated Network Testbed

The testbed uses the Cooja simulator to model a 21-node data collection IEEE 802.15.4 network of devices arranged in a tree topology: one node acts as data sink and 20 nodes are configured as periodic traffic sources. Each node is configured with a 50-meter radio range and runs the time slotted channel hopping (TSCH) media access control (MAC) protocol over IPv6 Low-Power Wireless Personal Area Network (6LoWPAN), carrying user datagram protocol (UDP) datagrams end-to-end. Upward routing is handled by classic IPv6 routing protocol for low-power and lossy networks (RPL) in non-storing mode, establishing a single destination oriented directed acyclic graph (DODAG) rooted at the traffic source. Each source generates a 40-byte sensory packet that traverses the RPL DAG towards the sink. Data packets are generated at 1Hz to emulate routine monitoring traffic. This setup allows for an evaluation of protocol performance under realistic low-power, lossy channel conditions while preserving timing and link-layer behaviors inherent to IEEE 802.15.4 TSCH. These are summarized in Table 2.

Ransomware spreads across the network more quickly as density increases. To capture this, we assess performance under four distinct density levels, as depicted in Fig. 1. For all experiments, we keep the same tree structure, node count and wireless channel configuration. For each experiment, we modify node density by changing the

Table 2: Network Simulation Configuration

Nodes	21 Cooja nodes (1 sink, 20 sources)
Transmission range	50 meters
Routing	RPL classic
Transport layer	UDP
Network layer	6LoWPAN
MAC layer	TSCH
Physical layer	IEEE 802.15.4
Sensory data packet size	40 bytes
Sensory data report frequency	1 Hz

physical distance between nodes such that communication links are created or removed. Nodes, and the presence (or the lack thereof) of pairwise links (edges in the network graph) remain constant throughout each respective experiment. The network density (ND) for each network instance corresponds to the total number of edges over the maximum number of edges possible in the graph. This is subsequently determined via Eq. 1:

$$ND = \frac{|E|}{\frac{|V|(|V|-1)}{2}} = \frac{2|E|}{|V|(|V|-1)} \quad (1)$$

where $|E|$ represents the number of edges and $|V|$ denotes the number of nodes in the network.

We model four different densities: Very high ($ND \approx 0.9$), medium, low and very low (0.5, 0.2 and 0.1 respectively). Due to the number of nodes in the graph, the above values represent density rounded to a single decimal point.

3.3 Infection Spreading over Network

To evaluate the spread speed of the ransomware across different network structures, scripts in the Cooja simulator were used to ensure that the initial node launching the malicious code was identical across all cases, with propagation commencing simultaneously. The infection time of each node was recorded until all 20 nodes in the network were infected.

As shown in Fig. 2, networks with lower density took longer to become fully infected, while denser network structures facilitate faster ransomware propagation. In this experiment, propagation in denser networks was nearly simultaneous due to the greater number of edges. However, regardless of the network density, the proposed ransomware successfully propagated and infected entire network through the communication links among devices.

3.4 Resource Utilization for Infection

Traditional detection techniques often rely on spotting abrupt changes in resource consumption. To evaluate resource utilization on device nodes during ransomware infection, we measured (1) CPU load for basic program execution, (2) radio-module activity required for wireless communication, and (3) aggregate data throughput across the IoT network.

CPU utilization was computed as the fraction of time the microcontroller spent in active mode, where power draw is highest,

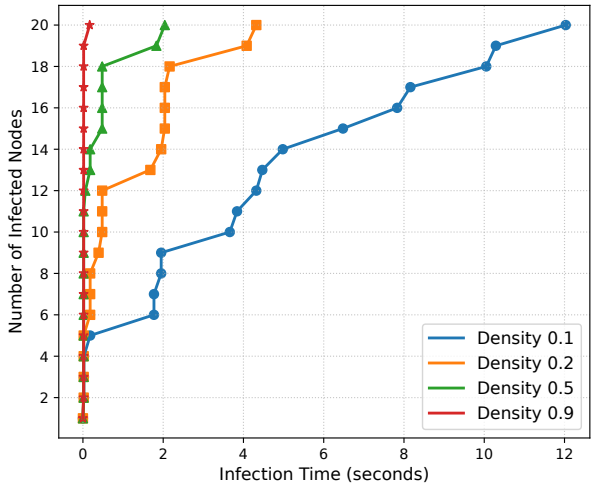

Figure 2: Infection speed comparison for different densities.

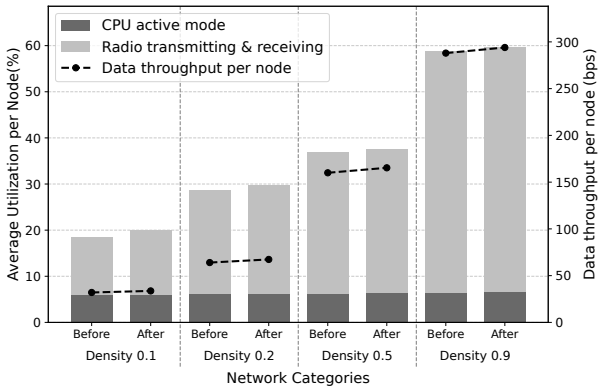

Figure 3: Resource utilization: before & after infection.

relative to the time spent in sleep or deep sleep states. In Contiki-NG, radio usage is tracked separately and follows the TSCH schedule, which precisely controls when the radio turns on for transmission or reception. After joining all nodes to the TSCH network, we quantified radio-usage ratio for each node by counting the number of occupied TSCH slot.

Across networks with varying densities, the average CPU and radio-usage metrics before versus after infection showed only marginal differences, as depicted in Fig. 3. Moreover, the ransomware's compact code footprint produced no discernible change in network throughput. Consequently, conventional anomaly detectors, which are designed to detect spikes in resource consumption or traffic patterns, are unlikely to detect this stealthy threat.

4 MICRO-AUDIT DESIGN AND EVALUATION

Given the limitations of existing detection methods, we developed a lightweight micro-audit mechanism tailored for resource-constrained devices to monitor security-relevant program activities and surface anomalies during ransomware infection.

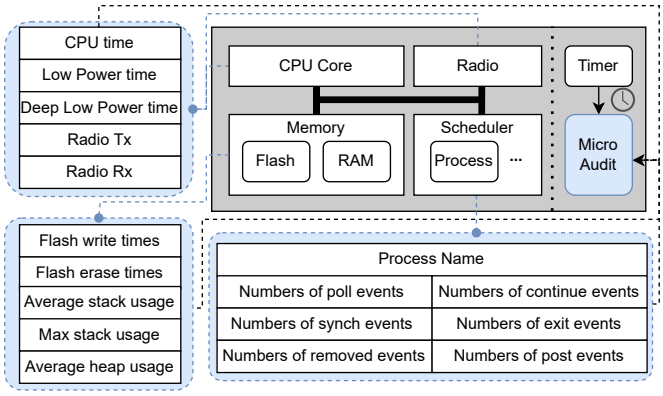

Figure 4: Micro audit statistics on microcontrollers.

4.1 Micro Audit Design

Fundamentally, the micro-auditing system collects resource utilization statistics and stores them for the purposes of analysis. The Contiki-NG OS schedules execution via processes and events calling; accordingly, our micro-audit system recorded per-process event-handling statistics over configurable time windows, establishing a baseline of normal execution patterns. Deviations from these baselines, such as unexpected event sequences or frequencies, were flagged as potential anomalies.

We collect process-scheduler statistics around processes and counts of events of each different type, as illustrated in Fig. 4. To persist across reboot cycles, the ransomware prototype attempts to save itself onto the infected device's internal flash. Flash erase/write operations could indicate anomalous activity, therefore statistics around those are also monitored by the micro-auditing system. Auditing statistics recorded the power consumption including the CPU core and radio activities. Contiki-NG already features capability of collecting very rich network-related statistics at all layers of the network stack, and the micro-auditing system is designed such that it can leverage this pre-existing capability.

Given that the proposed micro-audit mechanism targets resource-constrained IoT devices, two key criteria were emphasized: (i) maintaining a lightweight footprint to minimize deployment overhead, and (ii) ensuring detection efficacy by capturing distinctive metrics that reliably reveal the proposed ransomware.

4.2 Resource Overhead Assessment

The device resource overhead grew with network density in the earlier experiments. Consequently, we selected the network with a density of 0.9, representing the highest resource load, as our benchmark. Measurement were conducted with and without the audit mechanism enabled, capturing: (1) power consumption, (2) average and peak stack usage during program execution, and (3) firmware footprint after compilation, including the size of the executable code segment in ROM (.text) and the allocation for uninitialized *globals* and *statics* (.bss). These comparisons quantified the computational and memory overhead imposed by our micro-audit mechanism.

The micro-audit system features a dynamically adjustable auditing frequency. With auditing enabled, we evaluated three distinct

Table 3: Evaluation of resource overhead for network density 0.9 under varying audit conditions.

Evaluation metrics		Without audit	With audit		
			per 60s	per 10s	per 1s
Average current consumption (mA)		9.8	10.2	10.8	12.3
Average stack usage (bytes)		136	144	145	149
Max stack usage (bytes)		508	512	524	548
Firmware footprint (KB)	*.text*	95.43	96.35	—	—
	.bss	12.99	13.31	—	—

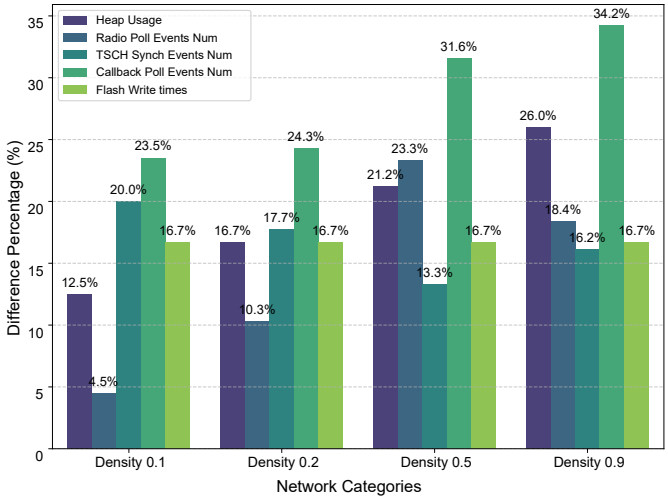

Figure 5: Noticeable resource usage increase after infection.

recording intervals: a low-frequency mode suitable for daily monitoring to minimize power draw, and progressively higher frequencies triggered upon threat indications to collect richer diagnostic data for anomaly analysis.

As summarized in Table 3, increasing the audit frequency yields a corresponding rise in runtime overhead. Nevertheless, the additional resource consumption remains modest and acceptable compared to an unaudited baseline. This finding demonstrates that deploying the micro-audit mechanism does not significantly increase the resource usage on devices, confirming its lightweight nature and suitability for resource-constrained environments.

4.3 Micro-Audit–Driven Ransomware Detection

As part of our ongoing research, to be documented in a future output, we are working on leveraging micro-auditing data for the purposes of developing detection and prevention countermeasures.

Using the micro-audit method, ransomware activity is assessed by detecting significant deviations in monitored metrics. As expected, higher auditing frequencies provide more detailed measurements. Under routine, low-frequency auditing, we quantified the magnitude of change in key metrics before and after infection.

Across networks with varying density, our detection indicators (leveraging the auditing statistics, heap usage, radio activity, TSCH synchronization events, callback-event counts, and flash writing) exhibited clear shifts that can reliably signal ransomware infection.

Since Cooja does not model real-device stack behavior, heap memory was used for dynamically allocating and freeing temporary variable buffers, thereby reflecting RAM activity on the device. As shown in Fig. 5, a corresponding increase in heap usage was observed after the infection. It is assumed that the attacker would install a reserved callback function as a backdoor, redirecting normal system processes to execute the injected payload. This hijack behavior manifested as a surge in callback-event counts.

Owing to the (typical) self-propagating nature of ransomware, infected nodes multicast payloads to neighboring nodes, driving up physical-layer radio-poll events and MAC-layer TSCH synchronization events. Additionally, attempts to make the ransomware persistent – by writing it to the flash memory, for instance – result in a higher number of write operations to critical memory regions.

5 LIMITATIONS AND ONGOING WORK

The micro-auditing mechanism produces pronounced shifts in detection indicators following a ransomware infection. However, legitimate activities, such as firmware over-the-air (OTA) updates, can induce similar metric fluctuations (e.g., flash erase/write operations) and trigger false alarms. To address this, we are currently in the process of developing audit-driven detection algorithms using machine learning models to discriminate benign updates from malicious anomalies, and thus improve detection accuracy.

We are targeting to deploy the ransomware detection mechanism on the device itself to enable real-time alerts and autonomous self-defense, while periodically forwarding audit data to a central server for deeper analysis. To balance responsiveness and overhead, the design adaptively adjusts the audit-data transmission rate: baseline reporting has a low transmission rate, but the rate escalates when on-device indicators suggest a potential attack. Investigation of the accuracy, precision and recall of centralized detection mechanisms under conditions of varying auditing data volume is also part of our ongoing work.

As audit data from leaf nodes traverse intermediate routers en route to the sink, forwarding nodes expend energy to handle the extra traffic. We therefore plan to evaluate the trade-off between detection efficacy and communication burden to ensure that the added audit transmissions do not excessively tax network resources.

Once infected, it is reasonable to assume that an attacker may wish to tamper auditing mechanisms. Disabling auditing would likely lead to an easy detection, therefore a more stealthy approach would be to tamper auditing statistics such that they reflect normal, before infection operating conditions. Our ongoing work is investigating the accuracy of centralized detection under conditions of uncertainty about auditing data trustworthiness.

6 CONCLUSION

With focus on enhancing the security of digital solutions, in this paper we have introduced a novel, lightweight micro-auditing mechanism tailored for resource-constrained IoT devices to detect stealthy ransomware infections. The ransomware was demonstrated to propagate rapidly and remains covert across networks of varying densities, effectively evading traditional traffic- or power-based anomaly detectors. The proposed auditing approach adaptively identifies malicious activities by monitoring diagnostically significant statistical metrics, including heap dynamics, TSCH/radio patterns, and flash access anomalies, at optimized sampling frequencies. This approach minimizes computational and energy overhead while ensuring timely detection of stealthy attacks.

By dynamically adjusting audit granularity in response to network conditions and threat severity, the mechanism achieves a critical balance between detection efficacy and resource preservation. This methodology not only complements conventional anomaly detection techniques but also lays a foundation for a scalable framework with the potential to integrate machine learning classifiers, thereby further refining threat discrimination and reducing false positives in future implementations.

ACKNOWLEDGMENT

This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC), Grant Numbers EP/X036871/1 and EP/X036707/1.

REFERENCES

[1] Smriti Bhatt, Thanh Kim Pham, Maanak Gupta, James Benson, Jaehong Park, and Ravi Sandhu. 2021. Attribute-Based Access Control for AWS Internet of Things and Secure Industries of the Future. *IEEE Access* 9 (Jul 2021), 107200–107223.

[2] Luke J. Bradley and Nick G. Wright. 2020. Optimising SD Saving Events to Maximise Battery Lifetime for Arduino™/Atmega328P Data Loggers. *IEEE Access* 8 (Nov 2020), 214832–214841.

[3] Luca Calderoni, Dario Maio, and Lorenzo Tullini. 2022. Benchmarking Cloud Providers on Serverless IoT Back-End Infrastructures. *IEEE Internet of Things Journal* 9, 16 (Jan 2022), 15255–15269.

[4] Rajakumaran Gayathri, Shola Usharani, Miroslav Mahdal, Rajasekharan Vezhavendhan, Rajiv Vincent, Murugesan Rajesh, and Muniyandy Elangovan. 2023. Detection and Mitigation of IoT-Based Attacks Using SNMP and Moving Target Defense Techniques. *Sensors* 23, 3 (Feb 2023).

[5] Kento Hasegawa, M. Yanagisawa, and N. Togawa. 2018. Detecting the Existence of Malfunctions in Microcontrollers Utilizing Power Analysis. *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)* (Sep 2018), 97–102.

[6] Yuxiang Huang, Calvin Brierley, Adel ElZemity, James Pope, Jiteng Ma, Antonio Di Buono, Budi Arief, and George Oikonomou. 2025. Ransomware in Resource-Constrained Industrial IoT Networks: There Actually is a Threat. In *21st International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*. 763–770.

[7] Changjong Kim and Sunggon Kim. 2023. Optimizing Logging and Monitoring in Heterogeneous Cloud Environments for IoT and Edge Applications. *IEEE Internet of Things Journal* 10, 24 (2023), 22611–22622.

[8] Fangyu Li, Yang Shi, Aditya Shinde, Jin Ye, and Wenzhan Song. 2019. Enhanced Cyber-Physical Security in Internet of Things Through Energy Auditing. *IEEE Internet of Things Journal* 6, 3 (Feb 2019), 5224–5231.

[9] George Oikonomou, Simon Duquennoy, Atis Elsts, Joakim Eriksson, Yasuyuki Tanaka, and Nicolas Tsiftes. 2022. The Contiki-NG Open Source Operating System for Next Generation IoT Devices. *SoftwareX* 18 (2022), 101089.

[10] Ferran Reverter. 2020. Toward Non-CPU Activity in Low-Power MCU-Based Measurement Systems. *IEEE Transactions on Instrumentation and Measurement* 69, 1 (Nov 2020), 15–17.

[11] Wadlkur Kurniawan Sedano and Muhammad Salman. 2021. Auditing Linux Operating System with Center for Internet Security (CIS) Standard. In *2021 ICIT*. 466–471.

[12] R. Sekar, Hanke Kimm, and Rohit Aich. 2024. eAudit: A Fast, Scalable and Deployable Audit Data Collection System*. In *2024 IEEE Symposium on Security and Privacy (SP)*. 3571–3589.

[13] Tingting Wang, Kai Fang, Wei Wei, Jinyu Tian, Yuanyuan Pan, and Jianqing Li. 2023. Microcontroller Unit Chip Temperature Fingerprint Informed Machine Learning for IIoT Intrusion Detection. *IEEE Transactions on Industrial Informatics* 19, 2 (Aug 2023), 2219–2227.