

Comparative Behavioral Profiling of ARM based IoT Malware Families: A Statistical Characterization of Dynamic System Activity Reports

Derick Andre Arcenas¹, Zios Bug-os¹, Janrey A. Lagunda¹, Hannah Dalmacio¹,
Jorton A. Tagud¹, Jose C. Agoylo Jr.²

¹ FCIS, Southern Leyte State University – Main Campus, Sogod, Southern Leyte, Philippines

² FCIS, Southern Leyte State University – Tomas Oppus Campus, Southern Leyte, Philippines

Abstract - The rapid expansion of the Internet of Things (IoT) has increased the digital attack surface, making resource-constrained devices vulnerable to malware exploitation. Although dynamic analysis effectively captures malware behavior during execution, existing approaches often rely on diverse, high-dimensional features that complicate cross-architecture detection and limit straightforward statistical profiling. This study presents a quantitative descriptive-comparative analysis of behavioral profiles among ARM-based IoT malware families. A multi-stage Extract, Transform, Load (ETL) pipeline was employed to process dynamic System Activity Report (SAR) data from the CIC-YNU-IoTMal dataset. A 10% systematic random sample was analyzed using 14 key features representing CPU utilization, process activity, memory/I/O, and network throughput. Results revealed distinct behavioral signatures and hardware stress patterns across malware families. Tsunami exhibited elevated CPU utilization, Mirai demonstrated sustained runtime activity, Gafgyt showed greater memory and network dependence, while Agent maintained low resource consumption.

DOI: 10.18421/SAR93-04

<https://doi.org/10.18421/SAR93-04>

Corresponding author: Jose C. Agoylo Jr.,
FCIS, Southern Leyte State University – Tomas Oppus
Campus, Southern Leyte, Philippines

Email: jagoylo@southernleytestateu.edu.ph

Received: 04 August 2026.

Revised: 12 September 2026.

Accepted: 19 September 2026.

Published: 27 September 2026.

 ©2026 Derick Andre Arcenas et al.; published by UIKTEN. This work is licensed under the Creative Commons Attribution-NonCommercial 4.0 International License.

The article is published with Open Access at
<https://www.sarjournal.com/>

The findings establish a statistical foundation for lightweight, behavior-driven anomaly detection tailored to ARM-based IoT environments.

Keywords – IoT, ARM arch, statistical fingerprinting, resource utilization.

1. Introduction

In today's contemporary digital landscape, the Internet of Things (IoT) has seen rapid changes, it has connected millions of inexpensive devices to the internet, from smart home gadgets to industrial sensors. While this connectivity brings convenience, it also creates a huge attack surface for malware that targets underprotected systems. Running silently, such malware may hijack machines into coordinated networks to launch Distributed Denial of Service Attacks (DDos) which overwhelm services through traffic floods, or steal sensitive data, making effective detection and understanding of their behavior a critical research problem [3], [5].

Existing work shows that dynamic analysis which is the process of observing malware behavior while it runs is effective for capturing actions such as system calls, process activity, and resource usage [1], [3]. Automated profiling systems and cloud-based sandboxes have been used to collect rich behavioral data from IoT malware, including API calls and low-level system activity [1], [3]. However, much of this research focuses on system calls or mixed feature sets, and cross-architecture detection often becomes complex when many architectures and feature types are combined [6].

Existing work has successfully utilized dynamic analysis to capture broad system activity. There is a limited study in simple, statistical, and family-level behavioral profiling that focuses on the dynamic system activity of ARM-based IoT malware, even though ARM is the dominant architecture in IoT deployments [2], [10].

As the Internet of Things (IoT) era emerges, smart devices increasingly incorporated into daily life have turned into a prime target for hackers' malicious endeavors. IoT devices utilize different Unix-based architectures that adhere to ELF (Executable and Linkable Format) as their standard specification for binary files [7].

As such, this study focuses on ARM-based IoT malware families and uses the CIC-YNU-IoTMal dataset's dynamic system activity reports (SAR) to statistically characterize and compare their behavioral profiles. By analyzing CPU, process, and interrupt-level activity over time, the goal is to understand how different malware families load and stress ARM devices, and how these patterns differ from benign behavior.

2. Methodology

A. Research Design

This study employs a quantitative, descriptive-comparative design. This approach is suited for analyzing the CIC-YNU-IoTMal dataset, which can capture the systematic observation, manual feature selection, and statistical comparison of resource utilization patterns across different ARM-based malware families.

B. Data Source

This study uses the CIC-YNU-IoTMal dataset. The researchers specifically focused on malware that targets ARM-based devices. Industry-data from Strategic Market Research (2025) shows that ARM represents 55% of the IoT market. As such, the focus ensure that our results accurately reflect malware behavior on the most widely deployed hardware.

C. Data Preprocessing and Feature Selection

To ensure data quality, a multi-stage Extract, Transform, Load (ETL) pipeline was implemented:

- **Stream Extraction and Sampling:** Due to the large volume of the sar.csv dataset, a stream-processing approach was utilized. Data was read in chunks of 100,000 records. To maintain a representative spread of activities while optimizing for analysis performance, a 10% systematic random sample was extracted from each chunk and using a fixed random_state=42 for reproducibility.
- **Data Cleaning and Label Refinement:** A filtering layer was applied to remove records labeled as "Unknown" in the MalwareFamily column. This ensures the analysis focuses exclusively on confirmed malware behaviors and prevents label noise from skewing the statistical profiles.

- **Manual Feature Selection:** The study narrowed the scope from the full SAR report to 14 "Essential" features representing four core system domains:

- **CPU Utilization:** cpu-load[0].usr, sys, and idle.
- **Process Activity:** proc (process creation) and cswch (context switches).
- **Memory & I/O:** memused-percent, active memory, disk[0].util-percent, and io.tps.
- **Network Throughput:** Received (rxkB) and transmitted (txkB) traffic.

D. Data Analysis

The behavioral profiling is conducted through:

- **Statistical Characterization:** Seaborn and Matplotlib for exploratory data analysis, generating heatmaps and distribution plots to visualize resource consumption such as CPU vs Network varies by Malware Family.
- **Comparative Profiling:** Descriptive statistics are calculated for the "Essential" features. The results build the "family signature", that allows the study to determine if specific families such as Mirai exhibit unique, statistically significant footprints in system activity compared to benign samples.

E. Ethical Consideration

This study adhered to ethical principles in the collection, processing, analysis, and reporting of cybersecurity data. Since the research utilized the publicly available CIC-YNU-IoTMal dataset containing pre-collected dynamic system activity reports, no personal information, human participants, or direct interaction with individuals was involved. The researchers used the dataset solely for academic and research purposes and respected the terms and conditions governing its use. Malware-related data were handled responsibly and analyzed within a controlled research context, with no attempt to deploy, modify, or use the malware samples for unauthorized activities.

3. Results and Discussion

A. Dataset Distribution and Sample Balance

In Figure 1, the distribution of sample records across the evaluated ARM-based Internet of Things (IoT) malware families and benign baseline traffic is shown with a logarithmic transformation scale, mainly to handle the heavy class imbalances that are there in the evaluated subset from the multi-layer dataset.

From the empirical numbers it looks like Mirai really dominates the total record count, and it is followed closely by Benign samples, with those two named classifications vastly outpacing the rest of the categories. On the other hand, families such as DarkNexus and Generic sit more in the mid-tier volume, while strains like Tsunami and Agent end up having the smallest volumetric profiles of captured runtime actions. This kind of heavily skewed data layout kind of reflects what we see in modern botnets when they are deployed in production networks, where a small set of aggressive code leaks tends to over-saturate the infected systems.

Also, this overall distribution trait is backed [9], where their evolutionary analysis followed how the public leakage of core source repositories directly gave rise to highly modular variants. Those variants then mirror dominant execution signatures, and that ends up creating systemic imbalances during dynamic execution profiling and multi-class classification setups.

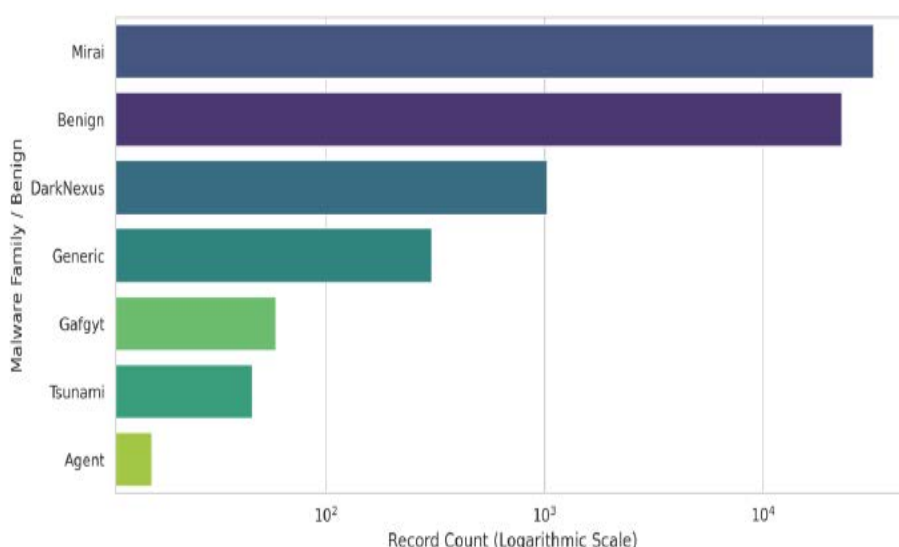


Figure 1. Distribution of Sample Records across ARM IoT Malware Families

B. Comparative CPU Utilization Profiles

Figure 2 presents a comparison of CPU User, CPU System, and CPU Idle metrics for both the malware group and a control group. These findings are shown in a type of graph or chart that allows us to observe the trends. Mirai and Tsunami malware exhibit elevated CPU User figures, suggesting that they are intensely operating in the user space. This indicates that a significant amount of code is being executed in the user space. Currently, the CPU System metrics indicate that Tsunami utilizes the operating system's kernel space reliably.

This indicates that Tsunami depends significantly on the operating system to handle its tasks. A distinction can be observed between the malware and the control group. This demonstrates that analyzing the usage of the operating system can assist in detecting malware. This is what [3] discovered in their research, where they demonstrated that examining how processes utilize resources can assist us in detecting malware, even if the malware attempts to conceal itself. CPU User and CPU System figures resemble fingerprints, serving as identifiers for malware. This can assist us in locating malware even when it attempts to avoid being discovered.

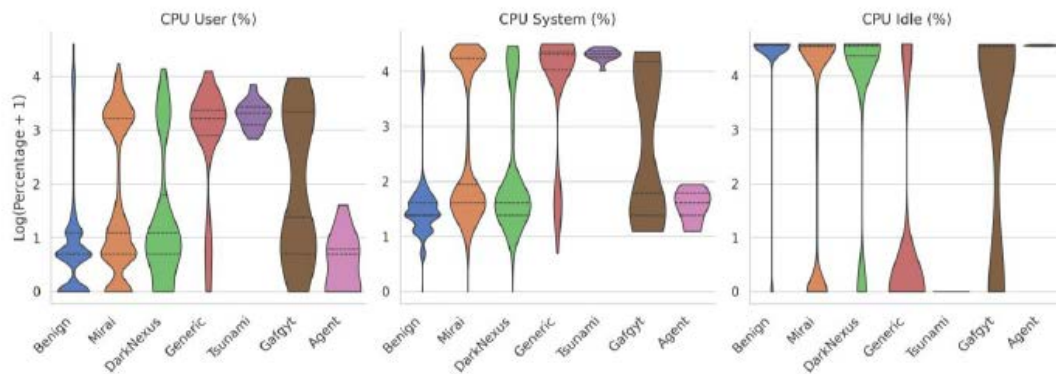


Figure 2. Comparative CPU Utilization Profiling on ARM Architecture (Log-Scaled)

C. Resource Footprint Intersection

Figure 3 shows a graph that compares the rate of context switches per second on the axis with the amount of network traffic sent on the vertical axis.

The scatter plot reveals two different ways in malware behavior.

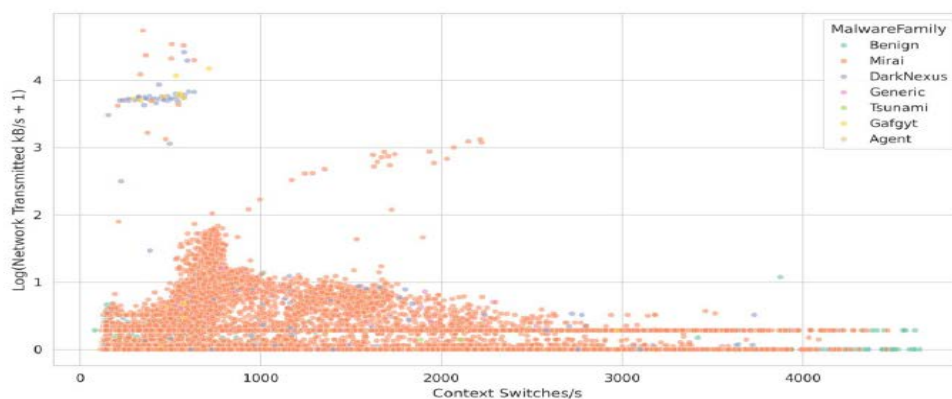


Figure 3. Process Scheduling vs. Network Transmission

Firstly, this is visible as a cluster in the upper-left quadrant comprising families like Mirai, DarkNexus, and Gafgyt, is characterized by high network traffic and low context-switching rates. This pattern is indicative of Denial-of-Service (DoS) attacks, where execution threads run in tight loops to maximize packet throughput while avoiding the scheduling overhead of the operating system. The second modality spans horizontally along the lower baseline, where malware variants, predominantly Mirai, exhibit incredibly high context-switching rates exceeding 4,000 per second with minimal network transmission, suggesting highly concurrent local activity like device scanning or brute-forcing. This distinct trade-off between task scheduling and network throughput underscores the importance of multi-layered behavioral analysis. [4] findings are comparable to this outcome. They examined how malware functions on devices connected to the internet of things. It was discovered that the frequency of communication and the volume of network traffic can be utilized to determine when the infection is under remote control.

D. The Behavioral Fingerprint Matrix

Figure 4 shows a heatmap that summarizes the intensity of 14 System Activity Report (SAR) features for threat groups. This helps create a fingerprint for each threat family. The matrix reveals that Agent works as a quiet background process using the CPU when it is not busy. Gafgyt uses a lot of memory and network resources. Tsunami uses a lot of user processing and system resources at the time. This helps identify hardware stress patterns. These patterns can improve machine learning models without needing to break down binary code. [8], Highlight the importance of these matrices. They say that IoT systems have CPU types and instruction sets. This makes traditional signature matching not work well. As such, it's necessary to focus on behavioral and system resource traits.

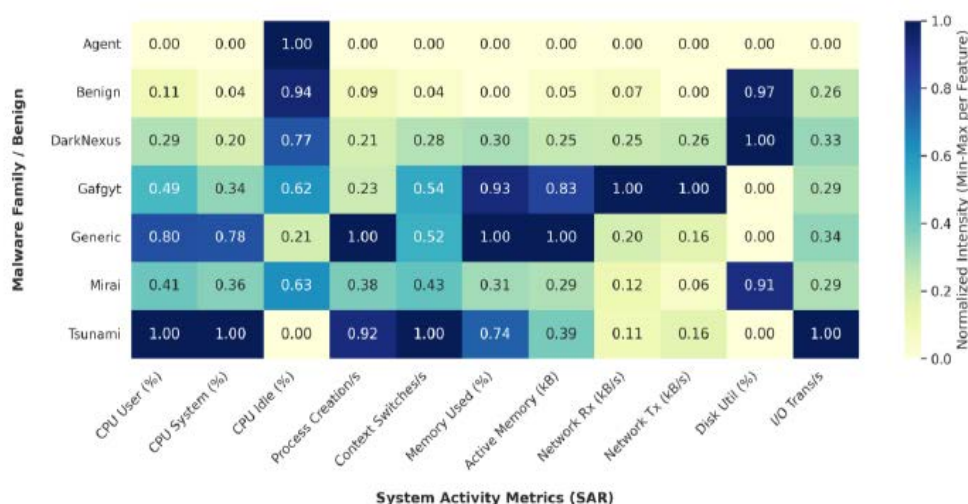


Figure 4. Comparative Behavioral Fingerprint of ARM IoT Malware Families

4. Conclusion and Recommendation

This study examines the behavioral profiling of ARM-based IoT malware families using dynamic system activity reports from the CIC-YNU-IoTmal dataset. It is primarily focused on a group of system activity characteristics like CPU usage, process behavior, memory and I/O patterns, and network throughput, to observe how different malware families operate during execution. With this way, the study was able to point out the different resource usage patterns of each malware family, and then compare them with benign system activity, too. The results indicate that every malware family has its own behavior pattern. While Tsunami exhibited high CPU user and system activity, Mirai emerged as one of the most prevalent malware families in the dataset. Agent operated more smoothly, resembling a quiet background process, whereas Gafgyt seemed to demand greater memory and network resource usage. The agent ran more calmly. These results suggest that malware families can be recognized not just by their code, but also by how they affect the device's system resources.

In general, the study shows that dynamic analysis is useful for understanding ARM-based IoT malware behavior. CPU usage, process scheduling, memory use, disk activity, and network traffic can act as useful indicators for malware profiling. This research can serve as a basic scaffold for future behavior-based malware detection studies, especially for IoT devices that use ARM architecture.

However, the study is limited to selected ARM based malware samples and selected system activity report features.

As a result, the following kinds of studies could be conducted in the future:

1. Broaden the dataset scope – incorporate additional malware families, integrate benign samples, and include various IoT architectures to achieve more comprehensive results rather than a limited focus.
2. Add more behavioral features – include more system activities such as file operations, system calls, power consumption, and device level logs, (not only the ones used here).
3. Apply machine learning techniques – test classification or clustering models to determine if the observed behavior patterns can identify and assist in grouping malware families effectively.
4. Compare across different architectures – examine how ARM based malware behavior is, different from malware behavior on other hardware architectures.
5. Conduct deeper family level analysis – analyze each malware family separately, to understand its unique behavior and its system impact, more clearly.

Ultimately, this study provides a foundation for comprehending ARM-based IoT malware using statistical and behavioral analysis. This also shows that dynamic system activity reports can reveal helpful malware patterns when examining how malware utilizes system resources during operation. This also encourages the creation of more sophisticated malware detection techniques, which concentrate not only on the appearance of the infection but also on its actual behavior once it has entered the system.

References:

- [1]. C. Y. Chen and S. W. Hsiao, "IoT malware dynamic analysis profiling system and family behavior analysis," in *Proc. 2019 IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 6013–6015. doi: 10.1109/BigData47090.2019.9005981.
- [2]. S. Dadkhah, *et al.*, "CIC-YNU-IoTMal: A comprehensive multilayer dataset for static and dynamic analysis of IoT malware behavior," *Information Systems*, vol. 102722, 2026.
- [3]. J. Jeon, J. H. Park, and Y. S. Jeong, "Dynamic analysis for IoT malware detection with convolution neural network model," *IEEE Access*, vol. 8, pp. 96899–96911, 2020. doi: 10.1109/ACCESS.2020.2995887.
- [4]. Đ. D. Jovanović and P. V. Vuletić, "Analysis and characterization of IoT malware command and control communication," *Telfor Journal*, vol. 12, no. 2, pp. 80–85, 2020. doi: 10.5937/telfor2002080J.
- [5]. S. Kumar, P. Ahlawat, and J. Sahni, "IoT malware detection using static and dynamic analysis techniques: A systematic literature review," *Security and Privacy*, vol. 7, no. 6, Art. no. e444, 2024. doi: 10.1002/spy2.444.
- [6]. Q.-D. Ngo, H.-T. Nguyen, V.-H. Le, and D.-H. Nguyen, "A survey of IoT malware and detection methods based on static features," *ICT Express*, vol. 6, no. 4, pp. 280–286, 2020. doi: 10.1016/j.icte.2020.04.005.
- [7]. A. D. Raju, I. Y. Abualhaol, R. S. Giagone, Y. Zhou, and S. Huang, "A survey on cross-architectural IoT malware threat hunting," *IEEE Access*, vol. 9, pp. 91686–91709, 2021. doi: 10.1109/ACCESS.2021.3091427.
- [8]. M. Venkatasubramanian, A. H. Lashkari, and S. Hakak, "IoT malware analysis using federated learning: A comprehensive survey," *IEEE Access*, vol. 11, pp. 5004–5018, 2023. doi: 10.1109/ACCESS.2023.3235389.
- [9]. H. Wang, *et al.*, "An evolutionary study of IoT malware," *IEEE Internet of Things Journal*, vol. 8, no. 20, pp. 15422–15440, 2021. doi: 10.1109/JIOT.2021.3063840.
- [10]. Y. Zhao and A. Kuerban, "MDABP: A novel approach to detect cross-architecture IoT malware based on PaaS," *Sensors*, vol. 23, no. 6, Art. no. 3060, 2023. doi: 10.3390/s23063060.