



Hardware Malware Detection using Federated Learning Technique

Sarthak Gupta^{*1}, Deepankar Bharadwaj², Amit Kumar³,
Abhay Pratap Singh Bhadauria⁴, Amrita Kumari⁵, Ashish Saini⁶

^{1,2}Department of Computer Science, IFTM University, Moradabad, India

^{3,5,6}Department of Computer Science, Quantum University, Roorkee, India

⁴Department of Computer Science, GLA, Mathura, India

Email: ^{*}sgmca1988@gmail.com, ²deepankar@iftmuniversity.ac.in, ³amit.cse@quantumeducation.in,
⁴abhaypratap00@gmail.com, ⁵amrita.cse@quantumeducation.in, ⁶ashish.cse@quantumeducation.in

Abstract. Hardware malware poses a significant threat to modern computing systems, exploiting vulnerabilities in hardware components such as microprocessors, firmware, and integrated circuits. Unlike traditional software-based threats, hardware malware operates at a lower level, making it highly stealthy, persistent, and challenging to detect. This type of malware refers to malicious modifications or logic inserted into hardware components, including microprocessors, firmware, and integrated circuits, during the design, manufacturing, or operational phases. These modifications can take the form of hardware Trojans, malicious firmware, or micro-architectural exploits that compromise system integrity. For example, hardware Trojans can remain dormant until triggered by specific conditions, allowing attackers to manipulate, exfiltrate, or destroy critical data without being detected. This paper provides a comprehensive overview of the evolving landscape of hardware malware, categorizing attack types such as hardware Trojans, malicious firmware, and side-channel exploits. It also explores state-of-the-art detection techniques, ranging from static and dynamic analysis to emerging machine learning approaches, while discussing mitigation strategies such as secure hardware design, supply chain auditing, and runtime defenses like Trusted Execution Environments. Apart from the above discussion, the article suggests an intrusion detection framework based on Federated Learning (FL) for detecting microarchitecture attacks such as Spectre, Meltdown, Fallout, and ZombieLoad. This approach allows several clients to train a global model without exchanging their actual data, hence preserving the privacy of the data. The use of a feature optimization method based on dimensionality reduction methods is included to improve the efficiency of the learning algorithm and decrease computational overheads. Experiments reveal that the suggested model performs very well with high accuracy in classifying various attacks.

Keywords: Federated Learning, Hardware malware, Hardware Trojans, Malicious firmware, Internet of Things.

1 Introduction

In recent years, the cybersecurity landscape has witnessed a significant shift from traditional software-based attacks to more sophisticated hardware-level threats. Hardware malware, a relatively new and evolving form of cyber-attack, has emerged as a critical concern for both researchers and industry professionals [1]. This type of malware operates at the hardware level, exploiting vulnerabilities in computer chips, firmware, and other physical components of computing systems. Unlike software malware, which can often be detected and mitigated through conventional security measures, hardware malware presents unique challenges due to its ability to persist even after software reinstallation or system resets [2]. Hardware platforms such as the IoT, embedded devices, and multi-server systems are particularly susceptible to cyberattacks. The complexity and scale of contemporary systems, often indicated by an increasing number of lines of code, further exacerbate the threat. Factors such as numerous software bugs, unsafe programming practices, improper configurations, and the ease of concealing malicious logic introduce significant vulnerabilities. Furthermore, the increasing level of network connectivity expands the attack surface, rendering all devices potential targets for adversaries. Consequently, the proliferation of IoT-based devices poses significant challenges to the cybersecurity community due to their inherent limitations in computational power and security capabilities.

The potential impact of hardware malware is far-reaching, affecting a wide range of devices from personal computers and smartphones to critical infrastructure systems and industrial control equipment. This paper initially presents an overview of hardware malware, exploring its various forms, attack vectors, and potential consequences. It further examines the technical aspects of how hardware malware operates, including its insertion methods during manufacturing processes or supply chain interventions. Additionally, current detection techniques, mitigation strategies, and ongoing research efforts aimed at strengthening hardware security are discussed [3]. Understanding the nature and implications of hardware malware is crucial for developing effective countermeasures and ensuring the integrity and security of modern computing systems. As technology continues to advance and reliance on interconnected devices grows, addressing the challenges posed by hardware malware becomes increasingly important for safeguarding individual privacy, organizational security, and national interests [4].

Even though there have been numerous studies conducted regarding hardware malware and their detection, most methods have been centralized in nature, requiring access to private data to perform tasks. This not only violates privacy but also makes scaling up difficult for IoT or edge computing environments where a lot of data needs to be processed efficiently. Thus, this paper presents an FL-based intrusion detection scheme, which can help detect attacks such as Spectre, Meltdown, Fallout, and ZombieLoad. This proposed technique uses federated learning (FL) to allow multiple clients to train a global detection model without having access to actual data. Moreover, feature optimization is used to improve detection accuracy and computation speed.

The main contributions of this work are summarized as follows:

- A FL-based intrusion detection framework tailored for distributed edge computing environments.
- Integration of feature optimization techniques to improve detection accuracy and reduce computational complexity.
- Effective detection of microarchitectural attacks including Spectre, Meltdown, Fallout, and ZombieLoad.
- A privacy-preserving collaborative learning mechanism that eliminates the need for centralized data sharing.
- Comprehensive experimental evaluation demonstrating high detection performance across multiple attack classes.

2. Taxonomy of Hardware Malware

2.1 Types of Hardware Malware

Hardware malware refers to malicious entities embedded in or targeting physical hardware components. Unlike software-based attacks, hardware malware exploits hardware-level vulnerabilities, making it particularly challenging to detect and mitigate [5]. Below are the key types of hardware malware, along with detailed explanations:

Hardware Trojans:

Hardware Trojans can degrade system performance, cause unexpected failures, or create security vulnerabilities [6]. They are a serious concern in modern electronics, especially due to the involvement of third-party vendors in the global supply chain. Detecting these Trojans is challenging because of their small size and stealthy behavior, but techniques such as side-channel analysis and formal verification are commonly used. To prevent such attacks, secure design methodologies and trusted manufacturing practices are essential [7].

Firmware malware:

Firmware malware is a type of malicious software that specifically targets the firmware of a device. Firmware is the low-level, essential software embedded within hardware components that controls how they function and interact with higher-level software, such as operating systems. Attackers identify and exploit vulnerabilities in the firmware of a specific device. This could involve exploiting insecure firmware updates, weak authentication mechanisms, or unpatched bugs. Examples of firmware include the BIOS/UEFI on a computer's motherboard, firmware in hard drives, network cards, or even embedded systems in IoT devices.

Side-channel (SC) attacks:

SC attacks exploit the physical attributes of a computing device to gather confidential data, such as passwords or encryption keys, without direct contact with the software or data on the computer. Side-channel attacks typically do not damage the system or require direct access to its internals [8]. Instead, they measure externally observable parameters. These attacks often exploit weaknesses in how an algorithm is implemented, not the algorithm itself. These attacks leverage unintended emissions or behavior of hardware devices during computation, making them a potent form of hardware-based threat.

Electromagnetic (EM):

EM malware is a category of malicious software that leverages or targets the electromagnetic emissions produced by electronic devices to conduct attacks. These emissions, generated naturally by the hardware during operation, can be exploited to either exfiltrate sensitive data or interfere with the functioning of the hardware [9]. This type of malware takes advantage of the natural or intentional EM radiation emitted by electronic devices during operation. All electronic devices emit electromagnetic (EM) radiation during operation, often as a byproduct of switching transistors, signal processing, or power regulation. The malware may encode data into modulated electromagnetic waves that can be captured by specialized receivers or common devices such as smartphones, radios, or antennas.

Chip-level backdoors:

Chip-level backdoors are introduced at the chip level are methods that are either deliberately or inadvertently incorporated into the hardware component to provide access, theft of information, or control over the computer system [10]. The backdoors at the hardware level are highly complex and hard to identify and counteract through conventional software-based security solutions [11-12].

3. Methodological Approach

3.1 Overview of the Methodological Framework

In this paper, a novel FL-based approach is used to detect hardware-based malware. The proposed approach allows multiple edge devices (clients) to collaborate in training a global detection model without any raw data being shared among clients. All clients conduct their local training processes with their respective datasets. However, all the model parameters from all clients are shared with the central server, which will generate a better global model after aggregating these local models. This methodology provides data protection and reduces computational costs in terms of communication overhead and scalability in edge environments. Figure 1 shows the proposed framework's structure.

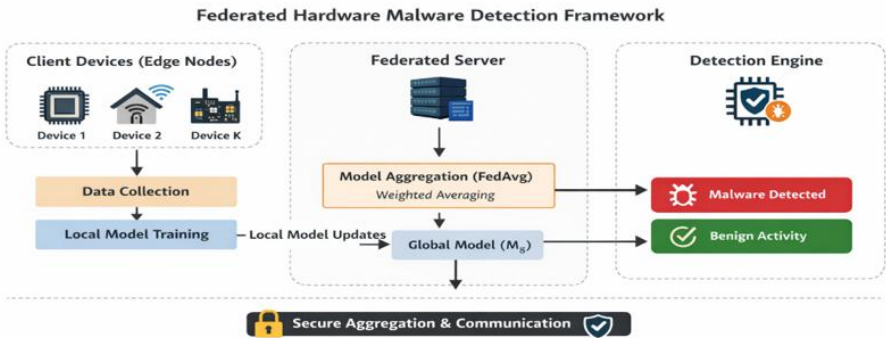


Fig. 1. Federated Hardware Malware Detection Framework

3.2 Mathematical Formulation of FL

The FL process aims to minimize a global objective function defined as Eq. (1). The optimization is performed through iterative aggregation of locally trained models using the Federated Averaging (FedAvg) algorithm, as shown in Eq. (2).

$$\min_w \sum_{k=1}^K \frac{n_k}{n} F_k(w) \quad \dots(1)$$

where K represents the number of participating clients, n_k denotes the number of data samples at client k , and $F_k(w)$ is the local loss function [13].

The global model parameters are updated using the FedAvg algorithm as follows:

$$w_{t+1} = \sum_{k=1}^K \frac{n_k}{n} w_k^t \quad \dots(2)$$

- w_{t+1} = Updated global model parameters
- K = Total number of participating clients
- n_k = Number of data samples at the client k
- $n = \sum_{k=1}^K n_k$ = Total number of samples across all clients
- $w_t^{(k)}$ = Model parameters from client k at round t

3.3 Feature Extraction and Preprocessing

Before the process of model training, the data set is pre-processed through methods like normalization and denoising [14-15]. Methods used for optimizing features involve algorithms like Principal Component Analysis (PCA) and Independent Component Analysis (ICA), which reduce the dimensions by selecting the best features from the data set.

3.4 Training Procedure

In the suggested FL architecture, there are several distributed nodes involved in the training phase. All of these nodes train their own model using their respective datasets. The trained models are then sent to the server for updating the global model [16]. This whole procedure is carried out repeatedly till convergence is attained. Each node learns from its own data, whereas the global model captures the cumulative knowledge of all the nodes.

3.5 FL Algorithm

The overall FL process is summarized as follows:

- Step 1: Initialize global model parameters.
- Step 2: Select participating clients for the current round.
- Step 3: Each client performs model training on its own dataset.
- Step 4: Clients upload the parameters to the server.
- Step 5: The server uses the uploaded parameters for model updating.
- Step 6: The updated global model is redistributed to clients.
- Step 7: Repeat the process until convergence.

4. Experimental Results and Detailed Analysis

4.1 Experimental Setup

The effectiveness of the FL-based attack detection algorithm is tested on a multi-class set of data that consists of both benign instances as well as attacks such as Spectre, Meltdown, Fallout, and ZombieLoad. Before the training of the system, the dataset is preprocessed with normalization and feature selection approaches to enhance its quality and decrease the data redundancy. The dataset is subsequently split into training and testing sets using an 80:20 ratio.

In the developed federated architecture, several distributed clients learn the models from the local datasets and contribute their outputs to the global performance of the system. The system’s performance is evaluated using standard metrics, precision (PR), recall (REC), F1-score (FS), and accuracy to assess the effectiveness of the classification process.

The proposed approach achieves high classification efficiency across all classes. The reason for that is associated with the proper feature selection and class separability of the selected dataset. From the confusion matrix, one can see that almost all the samples are correctly classified, with a few misclassified instances belonging to close classes.

The accuracy rate of the model is recorded to be at 99.95%, which demonstrates the efficacy of the suggested approach in identifying patterns of hardware attacks. Nonetheless, it should be noted that the evaluation process was carried out on a particular dataset, hence the need for testing using other real-life datasets.

Table 1. Confusion Matrix

	Benign	CrossSite	Fallout	Meltdown	Spectre V1	Spectre V2	Spectre V4	ZombieLoad
Benign	4320	2	0	0	0	0	0	0
CrossSite	0	443	0	0	0	0	0	0
Fallout	0	0	405	0	0	0	0	0
Meltdown	0	0	0	746	0	0	0	0
Spectre V1	0	0	0	0	424	0	0	0
Spectre V2	0	0	0	0	1	435	0	0
Spectre V4	0	0	0	0	0	1	479	0
ZombieLoad	0	0	0	0	0	0	0	734

Note: The bold values represent the True Positives for all classes.

Table 2. Performance Evaluation of the Proposed IDS

Class	PR (%)	REC (%)	FS (%)
Benign	1.0	1.0	1.0
CrossSite Attack	1.0	1.0	1.0

Fallout Attack	1.0	1.0	1.0
Meltdown Attack	99.98	99.97	99.98
Spectre V1 Attack	1.0	1.0	1.0
Spectre V2 Attack	99.97	99.98	1.0
Spectre V4 Attack	1.0	1.0	1.0
ZombieLoad Attack	99.99	99.98	1.0
Accuracy	99.95	99.96	99.95

5. Conclusion

In this paper, a FL-based system is presented that can detect malware at the hardware level, specifically those attacks at the microarchitecture level like Spectre, Meltdown, and ZombieLoad. The proposed system proves its effectiveness with high classification accuracy and performance across many attack categories. The data privacy and scalability of the system are ensured through distributed learning without the exchange of raw data. Even though the presented system works well in a particular dataset, future work includes testing the proposed system in different types of datasets under adversarial conditions.

References

1. Chaudhary, D., Rajasegarar, S., & Pokhrel, S. R. (2025). Towards adapting federated and quantum machine learning for network intrusion detection: A survey. arXiv preprint. <https://arxiv.org/abs/2509.21389>

2. Sáez-de-Cámara, X., Ray, P. P., & Jindal, A. (2023). Clustered federated learning architecture for network anomaly detection in large-scale heterogeneous IoT networks. arXiv preprint. <https://arxiv.org/abs/2303.15986>

3. Nguyen, T. D., Marchal, N., Miettinen, M., Sadeghi, A., Asokan, N., & Tarkoma, S. (2018). D²IoT: A federated self-learning anomaly detection system for IoT. In Proceedings of the IEEE/ACM International Conference on Internet of Things Design and Implementation (IoTDI) (pp. 1–12).

4. Ahmad, B. A. (2020). Real-time detection of Spectre and Meltdown attacks using machine learning. arXiv preprint. <https://arxiv.org/abs/2006.01442>

5. Pan, Z., & Mishra, P. (2021). Automated detection of Spectre and Meltdown attacks using explainable machine learning. In Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust (HOST) (pp. 1–6).

6. Tong, Z., Qi, Y., & Liu, J. (2022). Attack detection based on machine learning for Spectre and Meltdown variants. arXiv preprint. <https://arxiv.org/abs/2208.14062>

7. Fedorchenko, E., Novikova, E., & Shulepov, A. (2022). Comparative review of intrusion detection systems based on federated learning: Advantages and open challenges. Algorithms, 15(7), 247.

8. Sanal Kumar, K. P., Rizwan, M., & Singh, P. (2021). Security and privacy-aware artificial intrusion detection system using federated machine learning. *Computers & Electrical Engineering*, 96, 107496.
9. Weber, D., Neumann, I. M., & Eisenbarth, T. (2021). Osiris: Automated discovery of microarchitectural side channels. In *Proceedings of the USENIX Security Symposium* (pp. 1–18).
10. Bukhari, B., Khan, A. A., & Jan, M. A. (2024). Advancements in securing federated learning with intrusion detection systems using neural networks and feature engineering techniques. *Artificial Intelligence Review*.
11. Kumar, A., Kumar, V., & Bhadauria, A. P. S. (2025). Optimizing intrusion detection in edge computing network: A hybrid ML approach with recursive feature elimination. *International Journal of Intelligent Engineering and Systems*, 18(1). <https://doi.org/10.22266/ijies2025.0229.04>
12. Kumar, A., Kumar, V., & Bhadauria, A. P. S. (2026). Real-time hybrid machine learning-based next-generation intrusion detection system for edge computing networks. *Turkish Journal of Engineering*, 9(3), 600–611.
13. McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the Artificial Intelligence and Statistics (AISTATS)* (pp. 1273–1282).
14. Kumar, A., Kumar, V., Mahto, M. K., & Bhadauria, A. P. S. (2026). RDE-GAI-IDS: Real-time distributed ensemble and generative-AI-based intrusion detection system to detect threats in edge computing networks. In *Generative Artificial Intelligence for Next-Generation Security Paradigms* (pp. 335–358).
15. Kumar, A., Kumar, V., & Bhadauria, A. P. S. (2026). A dual approach to intrusion detection in edge environments: Offline and online perspectives. *SN Computer Science*, 7(5), 399.
16. Kumar, V., Kumar, V., & Kumar, A. (2024). A lightweight machine learning-based intrusion detection system for edge computing. *International Journal of Engineering Sciences & Emerging Technologies*, 11(4).

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (<http://creativecommons.org/licenses/by-nc/4.0/>), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

