Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.066551

ARTICLE

# A Novel Malware Detection Framework for Internet of Things Applications

**Muhammad Adil[1,*], Mona M. Jamjoom[2] and Zahid Ullah[3]**

[1]Department of Computer Science and Engineering, University at Buffalo, The State University of New York, Buffalo, NY 14260, USA
[2]Department of Computer Sciences, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, Riyadh, 11564, Saudi Arabia
[3]Information Systems Department, College of Computer and Information Sciences, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, 11432, Saudi Arabia

*Corresponding Author: Muhammad Adil. Email: muhammad.adil@ieee.org

Received: 11 April 2025; Accepted: 25 June 2025; Published: 30 July 2025

**ABSTRACT:** In today's digital world, the Internet of Things (IoT) plays an important role in both local and global economies due to its widespread adoption in different applications. This technology has the potential to offer several advantages over conventional technologies in the near future. However, the potential growth of this technology also attracts attention from hackers, which introduces new challenges for the research community that range from hardware and software security to user privacy and authentication. Therefore, we focus on a particular security concern that is associated with malware detection. The literature presents many countermeasures, but inconsistent results on identical datasets and algorithms raise concerns about model biases, training quality, and complexity. This highlights the need for an adaptive, real-time learning framework that can effectively mitigate malware threats in IoT applications. To address these challenges, (i) we propose an intelligent framework based on Two-step Deep Reinforcement Learning (TwStDRL) that is capable of learning and adapting in real-time to counter malware threats in IoT applications. This framework uses exploration and exploitation phenomena during both the training and testing phases by storing results in a replay memory. The stored knowledge allows the model to effectively navigate the environment and maximize cumulative rewards. (ii) To demonstrate the superiority of the TwStDRL framework, we implement and evaluate several machine learning algorithms for comparative analysis that include Support Vector Machines (SVM), Multi-Layer Perceptron, Random Forests, and k-means Clustering. The selection of these algorithms is driven by the inconsistent results reported in the literature, which create doubt about their robustness and reliability in real-world IoT deployments. (iii) Finally, we provide a comprehensive evaluation to justify why the TwStDRL framework outperforms them in mitigating security threats. During analysis, we noted that our proposed TwStDRL scheme achieves an average performance of 99.45 % across accuracy, precision, recall, and F1-score, which is an absolute improvement of roughly 3 % over the existing malware-detection models.

**KEYWORDS:** IoT applications security; malware detection; advanced machine learning algorithms; data privacy challenges

## 1 Introduction

Malware refers to software created to infiltrate or harm a computer system without the user's explicit consent. Malware has different types, and can be classified based on its specific actions, such as worms, backdoors, ransomware, trojans, spyware, rootkits, adware, etc. [1]. According to the Computer Economics reports, financial losses increased significantly every year due to malware attacks [2]. In 2016, the WannaCry ransomware attack affected computers in more than 150 countries, causing financial damage to various

organizations [3]. In 2024, the projected financial loss due to malware attacks is expected to reach $9.5 trillion globally, a 19% increase from the estimated $8 trillion in losses in 2023 [4]. Ransomware alone is projected to cost victims $265 billion annually by 2031. Moreover, it has been reported that cybercriminals are now using more advanced techniques to quickly compromise the system's security, with reduced average execution time from 60 days in 2019 to just 4 days in 2023 [5]. The financial sector is deeply concerned about technological security. To address this, they aim to design reliable and trustworthy countermeasures that can protect both individuals and institutions. These groups face significant risks from phishing attacks and social engineering tactics, which threaten their customer's trust and security.

In the literature, several antivirus software solutions such as Avast, Norton, McAfee, Kaspersky, AVG, and Bitdefender have been developed to counter various cyberattacks. These softwares rely on a short sequence of bytes, known as a signature, to detect malware within systems. However, they have significant limitations, especially when it comes to defending against zero-day attacks. These attacks exploit vulnerabilities that are unknown to these antivirus software. This challenge is further aggravated by the rise of malware generation toolkits, such as Zeus. These toolkits enable cybercriminals to generate thousands of malware variants through various obfuscation techniques, which makes the traditional signature-based detection technique less effective. In addition, malware is evolving daily, while signature generation is often human-driven, which creates a log in the detection process. All this makes it increasingly difficult to detect new malware with the help of older detection techniques. In light of these factors, relying solely on traditional antivirus approaches may not be enough, because the nature of malware attacks changes continuously. Therefore, more advanced and adaptive security solutions could be used to identify and mitigate emerging threats in real time.

In this paper, we propose a TwStDrl-enabled malware detection framework for IoT applications. We use a deep reinforcement learning algorithm to enable the system to learn and adapt in real time to evolving malware behavior. This real-time adaptability allows the framework to effectively detect and prevent malware infiltration in these applications. To evaluate the effectiveness of our model, we considered rival algorithms such as SVM, random forest, k-means, and other state-of-the-art solutions. This comparative analysis assesses how the TwStDrl-based framework outperforms existing methods in detecting and mitigating malware in IoT applications. In addition, the model is designed in a way to be lightweight and resource-efficient by considering the limited computational resources of IoT devices. The main contributions of this work are summarized below.

- In this work, we propose a TwStDrl-enabled framework that efficiently distributes malware detection tasks in two stages across IoT devices to reduce detection time. The framework primarily uses deep reinforcement learning algorithms, enabling real-time learning and adaptation to the evolving nature of malware threats.
- The two-step task scheduling strategy combines static and dynamic task management with presorting techniques, using Johnson's rule. This helps the model to adapt in real time and ensures the effective and timely completion of malware detection tasks across the network.
- Johnson's rule, used in the model, facilitates an optimal sequence for task processing. This optimization enhances both the detection and response rates to evolving security threats in IoT applications.
- In the subsequent phase, we implement various algorithms such as Support Vector Machine (SVM), Random Forest, K-Means, and Convolutional Neural Network (CNN) to evaluate the performance of the proposed model in comparison to these methods on different datasets.
- Finally, we compare the results of the TwStDrl algorithm with SOTA solutions and implemented algorithms to demonstrate why this algorithm is necessary and how it outperforms existing solutions.

**Paper Structure:** In this section, we cover the related work by providing readers with a glimpse into what has been achieved thus far in this domain. Moving on, Section 3 introduces our proposed model. In Section 4, we explore various algorithms and state-of-the-art schemes with their results that have been selected for comparison. Finally, Section 5 wraps up the paper by summarizing our findings and concluding remarks.

## 2 Related Work

In this section, we discuss the latest schemes used to counter malware attacks in IoT applications. Moreover, we highlight the limitations of existing techniques to establish a foundation for the proposed work in their presence. Given that, Glani et al. [6] proposed an advanced malware detection technique for IoT applications that targets hardware and source code vulnerabilities. In this model, the author used the Adler-32 hash function and Fibonacci search algorithms collectively to resolve this issue. However, the authors did not discuss the complexity and computation requirements of this model in real-time operation, which keeps it in a fuzzy state. In [7], the authors introduced convolutional neural networks (CNNs) for dynamic malware analysis considering edge devices. In this model, the authors converted the extracted behavior data into images to improve detection rate and accuracy. However, performing such a complex task on the edge of the network introduces significant computational complexity, which leads to slow response time during attacks. Htwe et al. [8] proposed a hybrid paradigm of feature selection method with a CART learning algorithm to tackle malware security vulnerability issues in IoT applications. The authors claimed 100% accuracy for the proposed model, with only botnet attacks. However, it is very hard to implement and use this model based on one attack detection. It could be interesting to check the results of the proposed model against different known and unknown attacks.

In [9], the author proposed an enhanced lightweight antivirus solution for embedded IoT devices. In this paradigm, they used graph theory with a measurement-based rational selection technique to appropriately set threshold levels for malware detection. However, the dependency on specific threshold parameters may affect the result statistics, and inhibit its use in real deployment. Deng et al. [10] proposed a transformer-based malware detection framework for IoT applications. The authors focused on the edge computing paradigm to address the limitation of centralized techniques and enhance the efficiency of malware detection by offloading computation-intensive tasks to nearby edge nodes. In [11], the authors proposed an Intelligent Trusted, and Secure Edge Computing (ITEC) malware detection approach for IoT applications. In this model, they used a signature-based pre-identification mechanism to detect malicious behaviors of untrusted third-party applications to categorize network traffic. However, the proposed model is sensitive to communication delay, throughput, and congestion issues, and could not be adopted in real IoT applications. Reference [12] introduced a novel CNN-based approach for malware detection in IoT Android applications. The author tested the proposed model on a specific dataset with defined rules and achieved remarkable results. However, it is still necessary to evaluate the proposed model on different datasets to see the statistical results, as machine learning models are susceptible to many internal and external threats.

In [13], the author proposed a graph compression algorithm with reachability relationship extraction (GCRR) to resolve the malware detection issue in Android applications. In the result analysis, the authors claimed that their model can detect malware in these applications 1.53% to 39.13% accurately compared to existing static methods. In [14], the author proposed a host-based anomaly detection system for IoT applications to resolve security problems. However, it is not clear how the proposed model is evaluated for different attacks. In [15], the authors introduced an event-aware Android malware detection system by considering the network traffic behavioral patterns. Moreover, the authors used event groups to capture higher-level semantics than API-level threats. However, the clustering complexity of the proposed model raises computation challenges, and may inhibit the deployment of this model. Vasan et al. [16] proposed

a robust cross-layer architecture-based malware detection scheme for IoT applications. In this model, the author used RNNs and CNNs collectively to improve the attack detection accuracy. However, this model was checked on a specific dataset during training and testing. Therefore, it is important to evaluate it on different datasets, considering the requirements of IoT applications. Shahid et al. [17] proposed a sparse autoencoder-based malware detection paradigm for IoT applications. This approach characterizes network behavior using statistics on the size and inter-arrival times of the first N packets in bidirectional TCP flow to improve the attack detection rate. However, a limitation of this method is specific characteristics of the smart home dataset, which was used for training, may not fully capture the diversity of IoT network behaviors in real-world scenarios.

In [18], the authors checked the application of one-class classification for malware detection in IoT applications using unsupervised learning techniques, while Almazroi [19] proposed a BERT-based feed-forward neural network framework to resolve the malware vulnerabilities detection problems in IoT applications. However, this model is complex and was checked on one dataset. Therefore, it is necessary to evaluate the proposed model on different datasets. Ahmad et al. [20] an advanced classification model for IoT malware detection using an optimized SVM algorithm. For optimization, they used nuclear reactor optimization (NRO), Artificial rabbits optimization (ARO), and particle swarm optimization (PSO) collectively to improve the classification accuracy. The authors claimed better results in the presence of state-of-the-art schemes. However, it is difficult to implement this model at the edges of IoT applications, where sensors and actuators will be working to collect and process data. In [21], the authors propose a serverless-based intelligent framework known as "CloudIntellMal" for detecting Android malware, while Sharma et al. [22], propose a multi-dimensional hybrid Bayesian belief network model to detect APT malware. This model is useful in the redressal of evasion attacks missed by signature-based solutions. In [23], the authors propose a graph–embedding framework known as IHODroid for Android malware detection. They evaluate it on the DREBIN dataset and demonstrate excellent results compared to existing baseline models. Moreover, we suggest the general readers to review article [24] to have a basic and advanced level understanding of malware and what has been done so far in the field and what is needed in the future. Following this, we added Table 1 in the paper to provide a more comprehensive analysis of the state-of-the-art schemes.

**Table 1:** Comparison between recent malware detection methods

| References | Year of publication | Machine learning/Deep learning algorithm | Application area |
|---|---|---|---|
| Shah et al. [25] | 2023 | Deep Learning-enabled malware detection system, using control flow graph (CFG) | Internet of Battlefield Things Applications |
| Smmarwar et al. [26] | 2023 | (AI)-empowered zero-day malware detection systems | Industrial Internet of Things Applications |
| Zhang et al. [27] | 2024 | Neural architecture search via proximal iterations (NASP) based malware detection systems | General Internet of Things Applications |
| El-Ghamry et al. [28] | 2023 | colony optimizer (ACO)-based malware detection systems | General Internet of Things Applications |
| Esmaeili et al. [29] | 2023 | GNN-based adversarial detector detects based malware detection systems | General Internet of Things Applications |

(Continued)

**Table 1 (continued)**

| References | Year of publication | Machine learning/Deep learning algorithm | Application area |
|---|---|---|---|
| Shafin et al. [30] | 2023 | Bidirectional long short-term memory, based malware detection systems | Smart City Internet of Things Applications |
| Sharma et al. [31] | 2023 | Ensemble learning based malware detection systems (Logistic Regression(LR), K-Nearest Neighbour(KNN), and eXtreme Gradient Boosting(XGB)) | General Internet of Things Applications |
| Devi and Arunachalam [32] | 2024 | Deep LSTM based malware detection systems | General Internet of Things Applications |
| Bajao and Sarucam [33] | 2023 | Evaluation of CNNs, LSTM, and GRUs based malware detection systems | General Internet of Things Applications |
| Mohammed et al. [34] | 2023 | An adaptive Malware Analysis Dynamic Machine Learning framework for malware detection | Healthcare Internet of Things Applications |

## 3 Proposed Model

In this section, we thoroughly talk about the proposed TwStDrl-enabled malware detection framework. For visual illustration, we added Fig. 1 in the paper.

The primary goal of our proposed model is to enable malware detection directly at the network edge. To achieve this, it combines static and dynamic analysis into a unified task-management framework, using a presorting technique based on Johnson's rule alongside deep-learning decision making. Detection tasks are then distributed across $N$ IoT devices and $M = 2$ processors (one for the static stage and one for the dynamic stage), where $N$ and $M$ denote the total number of connected devices and processors, respectively.

We address the resulting NP-hard scheduling problem by incorporating Johnson's rule optimization with deep-reinforcement-learning–based adaptive weighting. Task durations $(d_{i1}, d_{i2})$ are estimated from feature–importance scores produced in the first stage of the model. Since minimizing the total completion time of these two-step tasks is NP-hard, the presorting step orders static and dynamic tasks into an optimal sequence, setting the stage for efficient, dependency-aware scheduling across the IoT network.

Following this, a scheduling controller within the proposed model manages task execution at the network edge by continuously monitoring the system's state and selecting optimal scheduling actions based on data processing and threat analysis. The controller comprises three coordinated components such as Johnson's Rule–based presorting, load monitoring, and DRL action selection, which together ensure efficient task allocation. This framework is built on two key principles: **Concept 1:** It provides an optimal scheduling solution for single-processor devices, ensuring real-time malware detection. **Concept 2:** It guarantees that any subset derived from the task list remains optimally sequenced according to the model's defined rules.

Following Concept 1, our model improves malware detection accuracy across multiple processors by initially sorting them into a list. While Concept 2, allocates to each processor a task (dynamic data) to form subsets of the initial list (static dataset). This presorting guarantees that the parallel processing of tasks on

each processor is optimally arranged to identify malware patterns in the network. Moreover, these steps are summarized in Algorithm 1.

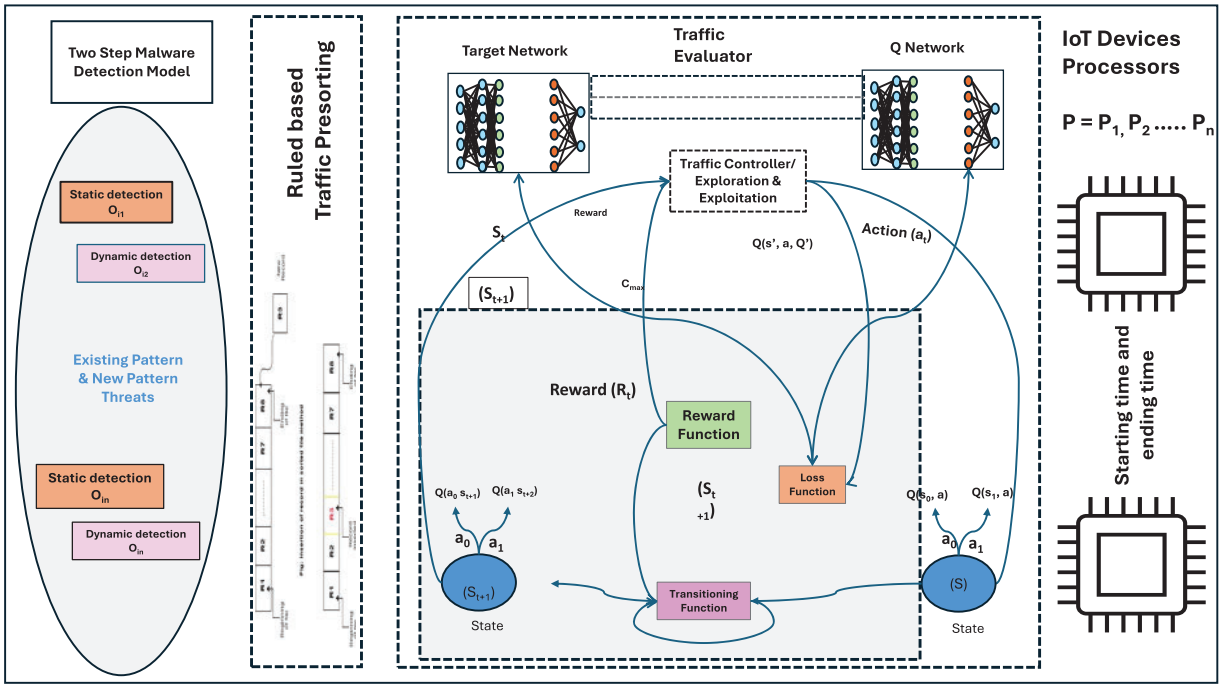


**Figure 1:** Architecture of the TwStDRL-enabled malware detection model, which integrates static and dynamic detection. The incoming traffic is first filtered based on predefined rules before being processed by the traffic evaluator. This evaluator, driven by a Q-network and a target network, continuously improves detection by balancing exploration (discovering new threats) and exploitation (using known patterns). A reward function enhances learning, while a loss function fine-tunes accuracy. IoT devices perform detection tasks according to a scheduled workflow by ensuring fast and efficient malware identification

**Algorithm 1:** Malware detection through two-step rule

**Require:** Task set $\mathcal{T} = \{T_{ask1}, T_{ask2}, \ldots, T_{askn}\}$, each with two-stage durations $\{d_{i1}, d_{i2}\}$
**Ensure:** Sorted task list $T'_{ask}$ for scheduling
Initialize $G_1 \leftarrow \{\}$, $G_2 \leftarrow \{\}$, and $T'_{ask} \leftarrow \{\}$
**for** each task $T_{aski} \in \mathcal{T}$ **do**
    **if** $d_{i1} \leq d_{i2}$ **then**
        $G_1 \leftarrow G_1 \cup \{T_{aski}\}$
    **else**
        $G_2 \leftarrow G_2 \cup \{T_{aski}\}$
    **end if**
**end for**
Sort $G_1$ by $d_{i1}$ in ascending order
Sort $G_2$ by $d_{i2}$ in descending order
$T'_{ask} \leftarrow G_1 \cup G_2 \rightarrow$ Append $G_2$ and $G_1$
**return** Updated $T'_{ask}$ Results

In Algorithm 1, we showed how tasks are divided into two groups, $G_1$ and $G_2$, based on the duration of their two operational phases. $G_1$ contains tasks where the first phase is shorter than or equal to the second ($d_{i_1} \leq d_{i_2}$), while $G_2$ includes tasks where the first phase is longer than the second ($d_{i_1} > d_{i_2}$). Each group is independently sorted into a predefined rule-based list before the two groups are strategically merged. This process of presorting and merging is important because it ensures that the tasks are efficiently assigned to IoT device processors. This approach not only improves individual processor efficiency but also enhances real-time malware detection across the network.

### *3.1 Operational Steps of the Edge-IoT Devices*

In IoT applications, detecting malware at the network edge is very important to ensure security. To achieve this, we use a two-step deep reinforcement learning (DRL) model that works in two phases: identification and mitigation of malware threats. The proposed model is formulated as a Markov Decision Process (MDP), which is defined by the following four-tuple representation:

$$\{S,\ A,\ P(s_t, a_t, s_{t+1}),\ R_t\} \tag{1}$$

- **State Space (S):** The state space at time $t$, denoted as $S$, encapsulates all possible malware detection scenarios based on the system's data. Each state transition is represented as $s_t, s_{t+1} \in S$.
- **Action Space (A):** The action space, $A$, comprises all feasible countermeasures against potential malware threats, including responses deployed across the network connected IoT devices.
- **Transition Probability ($P$):** The function $P(s_t, a_t, s_{t+1})$ defines the probability of transitioning from state $s_t$ to $s_{t+1}$ when executing action $a_t$. This captures the dynamic interactions between malware behavior and defensive responses.
- **Reward Function ($R$):** The reward, $R_t = R(s_t, a_t)$, quantifies the effectiveness of action $a_t$ at state $s_t$. The objective is to maximize security by minimizing malware impact through optimal policy learning.

In our proposed model, the state ($s_t$) captures all the important details needed to detect and prevent malware at the edge of the network. These details include the detection status of edge devices, network traffic characteristics, and system health, which are defined as below:

$$s_t = \left(Dt_{n_1}(t),\ Dt_{n_2}(t),\ Ct_{n_1}(t),\ Ct_{n_2}(t),\ Ut_m(t)\right) \tag{2}$$

In (2), $Dt_{n_1}(t)$ and $Dt_{n_2}(t)$ represent the detection times at Step 1 and Step 2 during malware identification process, while $Ct_{n_1}(t)$ and $Ct_{n_2}(t)$ denote the completion times for these steps. Initially, these values are set to zero and are updated dynamically as tasks are initiated and completed. Moreover, the $Ut_m(t)$ represents the overall network load on edge devices, which helps in assessing resource utilization during malware detection.

Given the current network state, the action $a_t$ determines how a device should handle the next malware threat based on the learned policy. The goal is to maximize malware detection and mitigation efficiency by optimizing action selection across states. The reward function $R_t$ evaluates the immediate impact of actions taken against malware threats by ensuring the network security. To further enhance optimization, we introduce a global objective function ($ct^*_{\max}$), which minimize the overall network load. First, we compute the total network utilization at time $t$ as:

$$U_t(t) = \sum_{k=1}^{m} u_k(t) \tag{3}$$

In (3), $ut_k(t)$ represents the load on node $k$ in the network. This metric is important for assessing real-time system performance to determine the instantaneous reward at time $t$. By integrating this reward function into the proposed model, we ensure the dynamic and adaptive malware detection and mitigation strategy.

For clarity, we added Algorithm 2 in the paper how the TwStDrl-enabled malware detection model works during the training process.

**Algorithm 2:** Training process of TwStDrl-Enabled malware detection model

**Require:** $\mathcal{T} = \{T_{task1}, T_{task2}, ..., T_{taskn}\}$
**Ensure:** $M$
Initialize $Q(S, a; \Theta)$, $Q'(S, a; \Theta')$
Set $\alpha$, $\epsilon$, $\gamma$
Initialize $\mathcal{D}$
**for** $eps = 1$ to $MaxEpisodes$ **do**
  **for** $T_{task} \in \mathcal{T}$ **do**
    **Task Scheduling via Johnson's Rule**
    Sort $T_{task}$ based on:
    If $Ct_{n_1}(t) \leq Ct_{n_2}(t)$, execute detection first.
    Else, execute mitigation first.
    **Select Action:**

$$a_t = \begin{cases} \text{random action}, & \epsilon \\ \arg\max_a Q(S_t, a; \Theta), & 1- \epsilon \end{cases}$$

    Execute $a_t$, observe $S_{t+1}$, $r_t$
    Store $(S_t, a_t, r_t, S_{t+1})$ in $\mathcal{D}$
    Sample $(S_t, a_t, r_t, S_{t+1})$ from $\mathcal{D}$
    Compute:

$$y_t = \begin{cases} r_t, & S_{t+1} \text{ terminal} \\ r_t + \gamma \max_{a'} Q'(S_{t+1}, a'; \Theta'), & \text{otherwise} \end{cases}$$

    Compute:

$$L(\Theta) = \mathbb{E}\left[(y_t - Q(S_t, a_t; \Theta))^2\right]$$

    Update:

$$\Theta \leftarrow \Theta - \alpha \nabla_\Theta L(\Theta)$$

    Periodically update: $\Theta' \leftarrow \Theta$
  **end for**
**end for**
**return** $M$

**Model Initialization:** The action-value function $Q(S, a; \theta)$ and its target counterpart $Q'(S, a; \theta')$ are initialized with weights $\theta$ and $\theta'$, respectively. The learning rate $\alpha$, exploration rate $\epsilon$, and discount factor $\gamma$ are set as hyperparameters to control the learning dynamics of the model. In addition, an experience replay buffer $\mathcal{D}$ is used to store past transitions, and enable stable training by reusing historical experience.

**Training Process:**

The training occurs over multiple episodes, where the model learns from a series of tasks denoted by $T$ in batches of 32 samples. The learning process uses an Adam optimizer with a learning rate of 0.001 and

discount factor $\gamma = 0.95$ for future rewards. Exploration is controlled through an $\varepsilon$-greedy strategy (initial $\varepsilon = 1.0$) that decays by a factor of 0.995 per episode until reaching $\varepsilon_{\min} = 0.001$, ensuring a progressive transition from exploration to exploitation.

Before selecting an action, tasks are pre-sorted using Johnson's Rule to optimize execution efficiency. The task sequence is determined based on detection and mitigation times as follows:

$$\begin{cases} \text{execute detection first}, & \text{if } Ct_{n_1}(t) \le Ct_{n_2}(t), \\ \text{execute mitigation first}, & \text{otherwise.} \end{cases} \tag{4}$$

The reward function combines binary classification accuracy (taking values of +1 or −1) with a load-balancing term scaled by (1 − device_load) to prioritize underutilized devices. Empirical analysis compared three variants—pure accuracy-based rewards (F1 = 0.89), load-scaled rewards (F1 = 0.92), and family-weighted rewards (F1 = 0.91)—demonstrating that the load-scaled version achieves the best trade-off between accuracy and utilization. Action selection follows

$$a_t = \begin{cases} \text{random action}, & \text{with probability } \varepsilon, \\ \arg\max_a Q(S_t, a; \theta), & \text{otherwise.} \end{cases} \tag{5}$$

Upon selecting an action $a_t$, the environment transitions to the next state $S_{t+1}$, and a corresponding reward $r_t$ is obtained based on the effectiveness of the malware detection response. The observed experience tuple $(S_t, a_t, r_t, S_{t+1})$ is stored in the replay buffer $\mathcal{D}$ for future learning.

**Q-Value Update:** To update the action-value function, a minibatch of stored transitions is sampled from $\mathcal{D}$. Before computing Q-values, the model applies Johnson's Rule to determine whether detection or mitigation tasks should be prioritized, ensuring an optimal action sequence. The target Q-value is then computed using the Bellman equation:

$$y_t = \begin{cases} r_t, & \text{if } S_{t+1} \text{ is a terminal state}, \\ r_t + \gamma \max\limits_{a'} Q'\left(S_{t+1}, a'; \theta'\right), & \text{otherwise.} \end{cases} \tag{6}$$

Here, the reward $r_t$ is influenced by the efficiency of task execution, considering both detection and mitigation completion times. By applying Johnson's Rule, the model prioritizes tasks that reduce overall malware response time, leading to more effective reinforcement learning.

**Loss Computation and Optimization:** The difference between the predicted Q-value and the target Q-value $y_t$ is computed as the loss function:

$$L(\theta) = \mathbb{E}\left[\left(y_t - Q(S_t, a_t; \theta)\right)^2\right]. \tag{7}$$

A gradient descent step is then applied to update the model parameters:

$$\theta \leftarrow \theta - \alpha \nabla_\theta L(\theta). \tag{8}$$

By incorporating Johnson's Rule in task prioritization, this optimization ensures that the model learns a task-aware policy, improving detection and mitigation efficiency.

**Target Network Synchronization:** To stabilize learning, the target network $Q'$ is periodically synchronized with the updated Q-network:

$$\theta' \leftarrow \theta. \tag{9}$$

**Completion of Training:** After executing the training process over multiple episodes, the optimized TwStDrl-enabled malware detection model is trained with an adaptive scheduling mechanism.

### *3.2 Experiment Setup*

In this section, we discuss the setup adopted for the implementation and evaluation of the proposed TwStDrl model. For implementation, we used Laptop Intel(R) Core(TM) i5-1035G1 CPU @ 1.00 GHz 1.19 GHz, 32.0 GB (31.8 GB usable) with Google Colab Pro version. In addition, we employed the TensorFlow and PyTorch frameworks to build and train the TwStDrl models. This framework's architecture is flexible and dynamic for the computation of random changes in the IoT traffic. The real-time learning process mitigates the risk of overfitting and helps to improve the model results.

### *3.3 Dataset*

For our experiments, we utilized the IoT-23 dataset, which comprises network traffic data of 23 different scenarios, including three benign and 20 malicious traffic patterns that belong to 11 distinct IoT botnet families. Specifically, it includes seven Mirai variants, two Torii instances, two Kenjiro samples, and one capture each of Gafgyt, Okiru, Hakai, IRCBot, Linux.Mirai, Linux.Hajime, Muhstik, and Hide-and-Seek. Given that diversity, it provides a comprehensive representation of real-world IoT network activities, which makes it well-suited for evaluating the performance of the TwStDrl model. To ensure robust model training and validation, we partitioned the dataset into three subsets such as 70% for training, 10% for validation, and 20%.

The benign data was collected from real-world IoT devices such as smart door locks, smart sensors, smart LED lamps, and other actuators commonly used in IoT applications. These devices provide an accurate representation of typical IoT network behavior, and form a baseline for detecting malicious activities. Given the large size of the IoT-23 dataset, our analysis focuses on differentiating malicious and benign traffic to ensure an effective evaluation of our proposed model. A detailed characterization of the dataset is provided in Table 2.

**Table 2:** Detailed overview of IoT-23 dataset for selected scenarios

| Scenario type | Time duration in (h) | Total No. of Packets | Traffic flow |
|---|---|---|---|
| Malware analysis phase-1 | 112 | 1,686,000 | 1,008,749 |
| Malware analysis phase-2 | 2 | 1,309,000 | 238 |
| Malware analysis phase-3 | 36 | 496,000 | 156,104 |
| Malware analysis phase-4 | 24 | 233,000 | 23,146 |
| Malware analysis phase-5 | 24 | 23,000 | 10,404 |
| Malware analysis phase-6 | 24 | 50,000 | 3287 |
| Malware analysis phase-7 | 24 | 50,000 | 3210 |
| Honeypot analysis phase-1 | 5.4 | 398,000 | 1383 |
| Honeypot analysis phase-2 | 24 | 21,000 | 461 |
| Honeypot analysis phase-3 | 1.4 | 8276 | 139 |

### *3.4 Data Preparation*

To achieve better results, it is important to prepare the data beforehand. First, we build a data pipeline that handles both malware and normal traffic by utilizing a dataset containing multiple malware families. We create a `Label` column (1 for malware, 0 for benign) for the binary classifier, and preserve each sample's family name in a separate `Family` column for auxiliary analysis. Unnecessary columns are removed, and missing values are filled with zeros.

Next, we convert non-numeric fields into model-ready formats. IP addresses are encoded as `float64` values by processing each octet. Protocol types (the `proto` field) are one-hot encoded. Port numbers (`id.resp_p`) are binned into the three standard IANA ranges—well-known (0–1023), registered (1024–49,151), and ephemeral (49,152–65,535)—using quantile discretization. This preserves key network-behavior patterns while reducing dimensionality.

To ensure a fair evaluation across all eleven malware families, we perform stratified splitting at two levels: maintaining the malware-to-benign ratio for the binary task, and preserving each family's relative proportion for family-level analysis. Within each split, we apply Z-score normalization to continuous features only, using training-split statistics to prevent data leakage.

Finally, we address class imbalance by (1) weighting malware samples three times more heavily than benign ones during binary training; (2) sampling more frequently from rare families during the DRL phase; and (3) adding dynamic rewards that penalize misclassifications of low-frequency families.

## 4 Comparative Analysis

In this section, we discuss the comparative performance of the proposed algorithm against other established algorithms that have been considered in this research to practically check their results on the IoT2023 datasets. In the first phase, we trained the selected algorithms, including our proposed model, to ensure they were adequately prepared for evaluation.

### *4.1 Proposed Model Convergence Result Statistics*

This training process included hyperparameter tuning and cross-validation to optimize model performance and mitigate overfitting. Each algorithm was subjected to a consistent training process with the utilization of the same training data split to ensure a fair comparison. This approach allowed us to assess the generalization capabilities of each model under similar conditions by providing a robust basis for evaluation. Following the training phase, we conducted a comprehensive evaluation of the algorithms based on several things to see the model convergence. These metrics are crucial for understanding the effectiveness of each model in detecting malware, particularly in the context of IoT environments where false positives can lead to significant operational disruptions. The comparative analysis not only highlighted the strengths and weaknesses of the proposed model relative to existing algorithms but also provided an understanding of the specific characteristics of the datasets that influenced performance. By analyzing the results, we identified areas for improvement in our model and gained a deeper understanding of the challenges associated with malware detection in diverse IoT scenarios. The results obtained for TwStDrl are shown in Fig. 2.

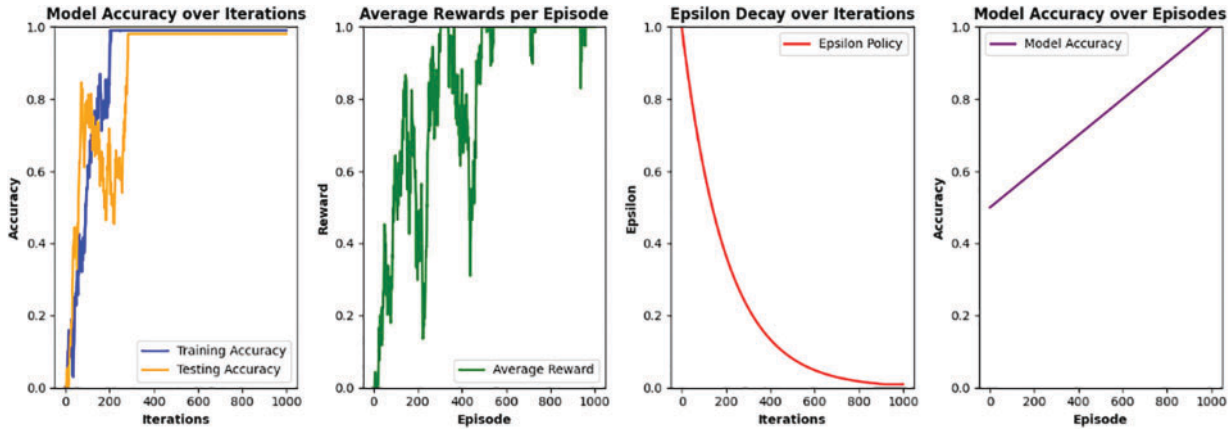


**Figure 2:** TwStDrl model convergence result statistics

### *4.2 SVM and Random Forest Algorithm Result Statistics*

In this section, we discuss the training and validation accuracies of the Support Vector Machine (SVM) and Random Forest algorithms, as well as their corresponding training and validation losses on the IoT2023 dataset. This evaluation is important for getting a deep understanding of the performance of these comparative algorithms in the context of our proposed model. By analyzing both accuracy and loss, we can gain a sense of understanding how well each algorithm works against seen and unseen data, which are basic elements to illustrate each model's robustness for real-world IoT applications. Furthermore, the training accuracy provides an indication of how well the model fits the training data, while validation accuracy serves as a benchmark for performance on new, unseen data instances. In addition to accuracy, monitoring training and validation losses throughout the training process allows us to evaluate the convergence behavior of each algorithm. A decreasing loss indicates that the model is learning effectively, while fluctuations or increases in validation loss can show potential issues with model stability or generalization. Moreover, the results obtained from this process are shown in Fig. 3 for SVM and Random Forest algorithms.

### *4.3 CNN and K-Means Algorithm Result Statistics*

In this section, we examine the training and validation accuracies of the Convolutional Neural Network (CNN) and K-Means clustering algorithms, along with their respective training and validation losses on the IoT2023 dataset. Getting the results of these metrics will help while evaluating the effectiveness of these algorithms in comparison to our proposed model. Moreover, in the literature, we noted that CNNs perform better on structured data like images and sequences, while K-means is useful in unsupervised learning scenarios because it offers valuable results by grouping similar data points based on feature similarity. Therefore, it is important to check and evaluate their results in the presence of the proposed model to acknowledge why our model is better suited for IoT applications in the presence of these algorithms. Furthermore, checking both training and validation losses is important for analyzing the learning process of these models. Moreover, the decline in training loss indicates that the model is effectively learning from the training data, while validation loss serves as an indicator of how well the model is likely to perform on unseen data. Moreover, the results obtained during experiment analysis are shown in Fig. 4.

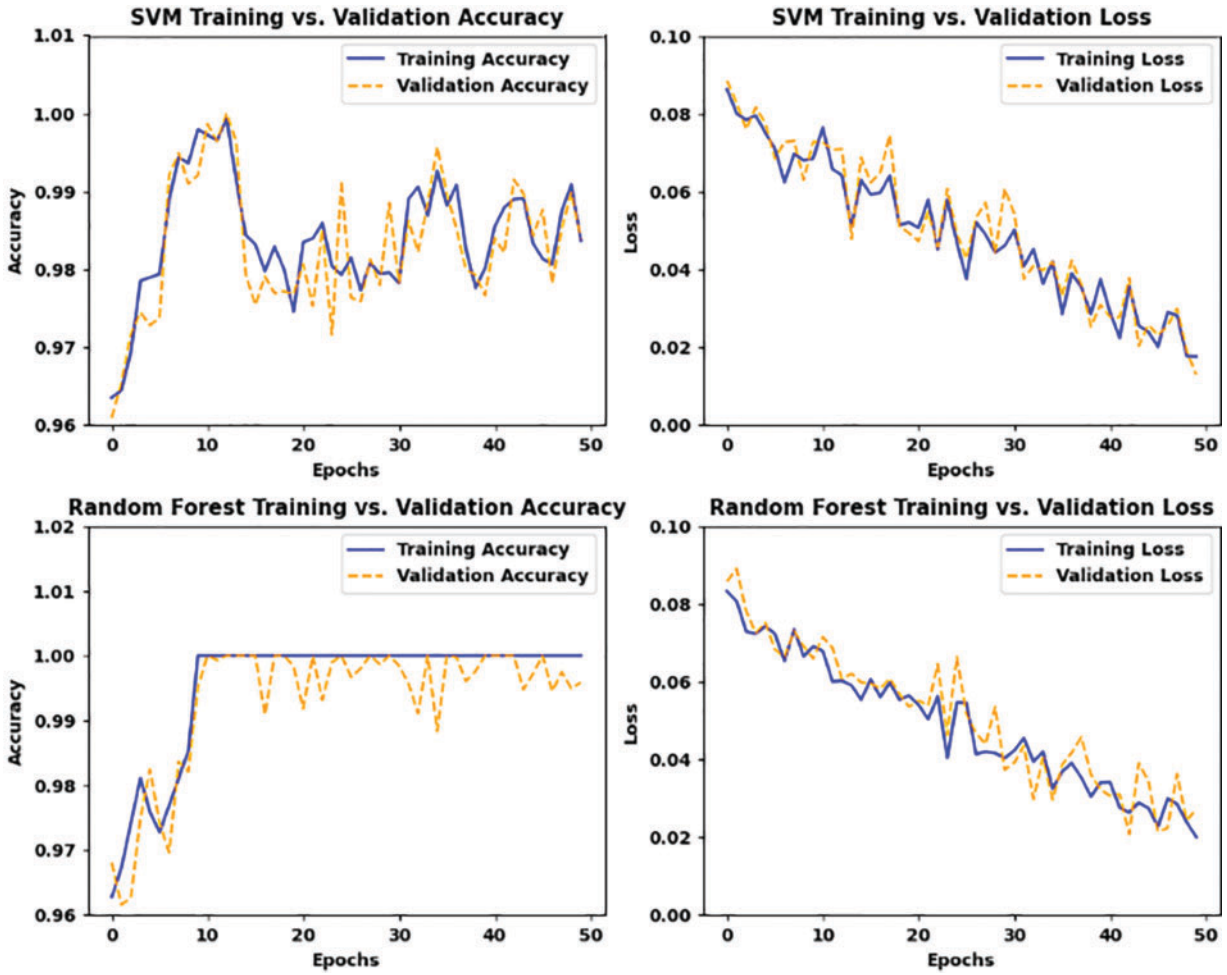


**Figure 3:** SVM and random forest result statistics

### *4.4 Overall Result Statistics*

In this section, we discuss the training and validation accuracies of the different algorithms considered in our study, along with their corresponding training and validation losses. We present Table 3 in the paper to summarize the performance metrics, including F1 score, precision, and recall, which are important for evaluating the algorithms' capabilities in handling imbalances in IoT applications. Moreover, incorporating these metrics into our analysis allows for a more detailed comparison between different considered algorithms by identifying strong and weak. This comprehensive evaluation not only helps in selecting the most suitable model for IoT malware detection but also contributes to the ongoing discourse on improving algorithmic performance in the context of evolving cybersecurity threats.

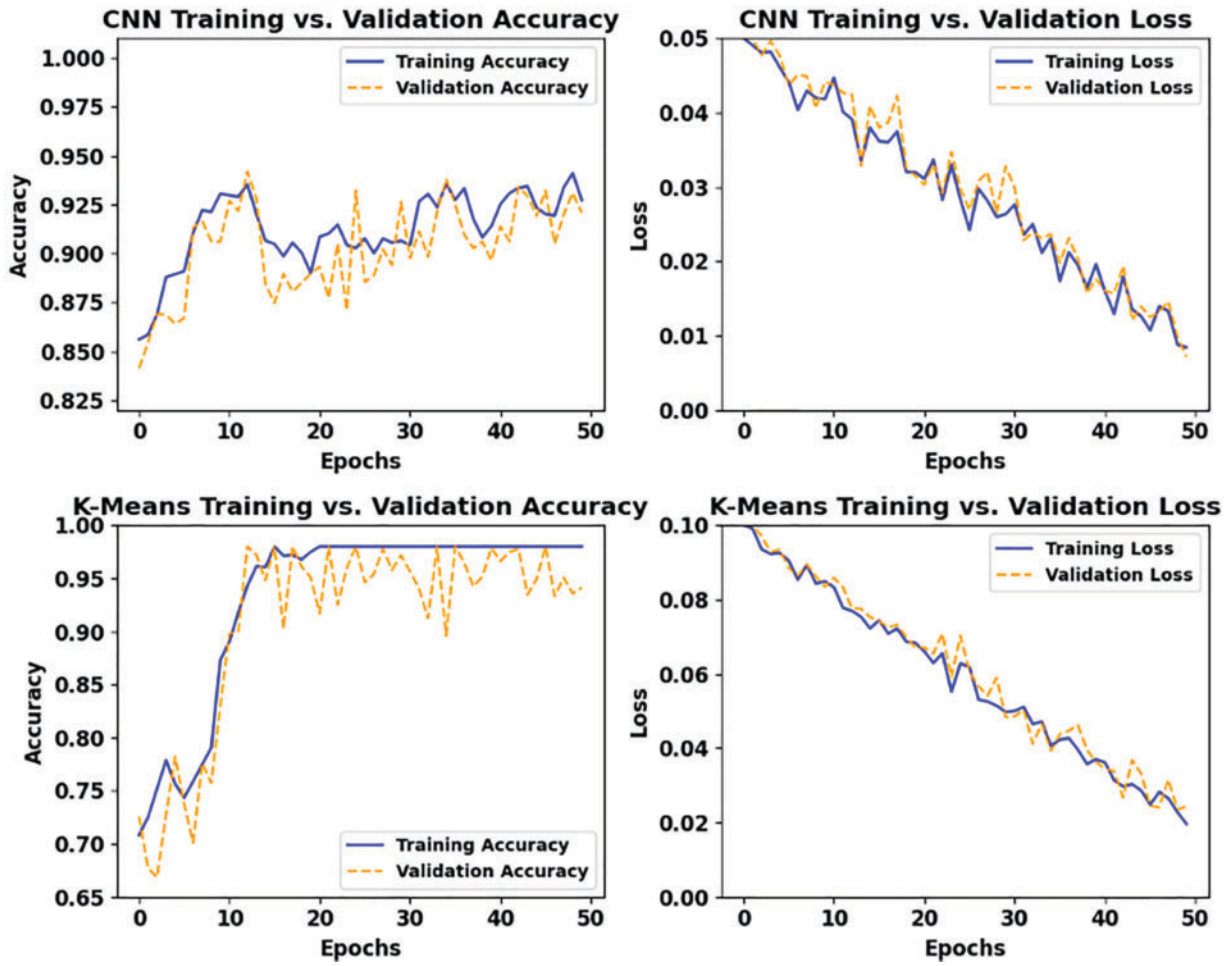


**Figure 4:** CNN and K-Means algorithm result statistics

**Table 3:** Comparative analysis with implemented algorithms

| Model /Algorithm | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| SVM | 0.93 | 0.93 | 0.93 | 0.93 |
| Random Forest | 0.92 | 0.92 | 0.92 | 0.92 |
| CNN | 0.97 | 0.98 | 0.98 | 0.98 |
| K-Means | 0.96 | 0.95 | 0.96 | 0.95 |
| TwST-DRL | 0.99 | 0.997 | 0.998 | 0.993 |

### *4.5 Comparative Analysis with Rival Algorithms*

In this section, we discuss the result statistics of the proposed model with rival algorithms as presented in Table 4 by focusing on their performance metrics. Moreover, this comparative analysis highlights the effectiveness of the proposed model in the presence of different malware detection models. This table also provides a useful understanding of these different models' performance in terms of accuracy, precision, recall, and F1-score. To explore, the accuracy metric shows the overall correctness of the model's predictions,

while precision demonstrates the proportion of true positive results among all positive predictions. Recall, on the other hand, measures the model's ability to identify all relevant instances, which is particularly important for malware detection and can have serious consequences, while the F1-score serves as a harmonic mean of precision and recall to offer a balanced view of the model's performance, especially in scenarios where class distributions are imbalanced. The results from our proposed model, TwStDrl, demonstrate a significant improvement in performance metrics compared to existing models, achieving an accuracy of 99.0%, precision of 99.70%, recall of 99.80%, and F1-score 99.30%. These results indicate that our approach not only excels in detecting malware but also minimizes false positives to improve the robustness of the model.

**Table 4:** Comparative analysis with rival malware detection models on the IoT-23 dataset

| Reference and year of publications | Model or adopted algorithm | Accuracy% | Precision% | Recall% | F1-score% |
|---|---|---|---|---|---|
| Dzulqarnain [35], 2019 | Hybrid machine learning model | 98.41% | 97.56% | 98.83% | 98.60% |
| Xing et al. [36], 2022 | Deep learning (Auto-encoder) | 96.22% | 96.14% | 96.20% | 96.17% |
| Jamal et al. [37], 2022 | Artificial neural network based framework | 97.08% | 94.00% | 93.00% | 93.50% |
| Bhayo et al. [38], 2023 | SVM, decision tree and Naive Bayes | 97.60% | 96.30% | 98.20% | 97.40% |
| Sanchez et al. [39], 2024 | Deep learning (LSTM) | 96.20% | 95.90% | 96.50% | 96.10% |
| Sahu et al. [40], 2021 | Deep learning (LSTM) | 95.68% | 95.31% | 96.42% | 95.86% |
| Zhu et al. [41], 2020 | Ensemble learning framework | 94.92% | 97.04% | 96.94% | 96.99% |
| Sun et al. [42], 2024 | 96.40% | 96.80% | 95.90% | 96.40% | |
| This Paper (TwStDrl) | Two-Step-Deep reinforcement learning | 99.0% | 99.70% | 99.80% | 99.30% |

## 5 Conclusion

In this paper, we propose an advanced two-step deep reinforcement learning-enabled framework, known as "TwStDrl" to resolve malware detection and prevention issues in IoT applications. Our TwStDrl framework is capable of identifying malware at the network's edge while protecting the user's important data privacy. We implement and evaluate well-known algorithms such as SVM, Random Forest, CNN, and K-Means to verify the experimental performance of the proposed model. During analysis, we found that the proposed model outperforms these algorithms in terms of comparative metrics. Additionally, we benchmarked the TwStDrl model with state-of-the-art schemes and found that it is better than them in terms of accuracy, precision, validation rates, etc. Based on these findings, we are confident that TwStDrl will be useful for IoT applications to detect malware in real time with great efficiency.

**Acknowledgement:** This work is supported by Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2025R104), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

**Funding Statement:** This work is supported by Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2025R104), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

**Author Contributions:** Muhammad Adil performed the experiments research methodology, prepared the initial draft of the manuscript, and gave final approval of the version to be published, while Mona M. Jamjoom conducted an analysis of the research findings, contributed to editing and revising the manuscript, and provided project management and supervision. Zahid Ullah provided research resources, contributed to the analysis, critically reviewed the manuscript for technical content, and approved the final version. All authors reviewed the results and approved the final version of the manuscript.

**Availability of Data and Materials:** Please find the link of dataset "https://www.stratosphereips.org/datasets-iot23" (accessed on 24 June 2025).

**Ethics Approval:** Not applicable.

**Conflicts of Interest:** The authors declare no conflicts of interest to report regarding the present study.

## References

1. Raff E, Nicholas C. A survey of machine learning methods and challenges for Windows malware classification. arXiv:2006.09271. 2020.
2. Conti M, Gangwal A, Ruj S. On the economic significance of ransomware campaigns: a Bitcoin transactions perspective. Comput Secur. 2018;79:162–89.
3. Hsiao SC, Kao DY. The static analysis of WannaCry ransomware. In: 2018 20th International Conference on Advanced Communication Technology (ICACT). 2018 Feb 11–14; Chuncheon-si, Gangwon-do, Republic of Korea. p. 153–8.
4. Kesler E. A transdisciplinary approach to cybersecurity: a framework for encouraging transdisciplinary thinking. arXiv:2405.10373. 2024.
5. Neprash HT, McGlave CC, Cross DA, Virnig BA, Puskarich MA, Huling JD, et al. Trends in ransomware attacks on US hospitals, clinics, and other health care delivery organizations, 2016–2021. In: JAMA Health Forum. Chicago, IL, USA: American Medical Association; 2022. e224873 p.
6. Glani Y, Ping L, Shah SA. AASH: a lightweight and efficient static IoT malware detection technique at source code level. In: 2022 3rd Asia Conference on Computers and Communications (ACCC). 2022 Dec 16–18; Shanghai, China. p. 19–23.
7. Jeon J, Park JH, Jeong YS. Dynamic analysis for IoT malware detection with convolution neural network model. IEEE Access. 2020;8:96899–911. doi:10.1109/access.2020.2995887.
8. Htwe CS, Thwin MMS, Thant YM. Malware attack detection using machine learning methods for IoT smart devices. In: 2023 IEEE Conference on Computer Applications (ICCA 2023). 2023 Feb 27–28; Yangon, Myanmar. p. 329–33.
9. Buttyan L, Nagy R, Papp D. Simbiota++: improved similarity-based IoT malware detection. In: 2022 IEEE 2nd Conference on Information Technology and Data Science (CITDS). 2022 May 16–18; Debrecen, Hungary. p. 51–6.
10. Deng X, Wang Z, Pei X, Xue K. TransMalDe: an effective transformer-based hierarchical framework for IoT malware detection. IEEE Trans Netw Sci Eng. 2024;11(1):140–51. doi:10.1109/tnse.2023.3292855.
11. Deng X, Chen B, Chen X, Pei X, Wan S, Goudos SK. A trusted edge computing system based on intelligent risk detection for smart IoT. IEEE Trans Ind Inform. 2023;20(2):1445–54. doi:10.1109/tii.2023.3245681.
12. Dhanya K. Obfuscated malware detection in IoT Android applications using Markov images and CNN. IEEE Syst J. 2023;17(2):2756–66. doi:10.1109/jsyst.2023.3238678.

13. Niu W, Wang Y, Liu X, Yan R, Li X, Zhang X. GCDroid: android malware detection based on graph compression with reachability relationship extraction for IoT devices. IEEE Internet Things J. 2023;10(13):11343–56. doi:10.1109/jiot.2023.3241697.
14. Breitenbacher D, Homoliak I, Aung YL, Elovici Y, Tippenhauer NO. HADES-IoT: a practical and effective host-based anomaly detection system for IoT devices (extended version). IEEE Internet Things J. 2021;9(12):9640–58. doi:10.1109/jiot.2021.3135789.
15. Lei T, Qin Z, Wang Z, Li Q, Ye D. Evedroid: event-aware android malware detection against model degrading for IoT devices. IEEE Internet Things J. 2019;6(4):6668–80. doi:10.1109/jiot.2019.2909745.
16. Vasan D, Alazab M, Venkatraman S, Akram J, Qin Z. MTHael: cross-architecture IoT malware detection based on neural network advanced ensemble learning. IEEE Trans Comput. 2020;69(11):1654–67. doi:10.1109/tc.2020.3015584.
17. Shahid MR, Blanc G, Zhang Z, Debar H. Anomalous communications detection in IoT networks using sparse autoencoders. In: 2019 IEEE 18th International Symposium on Network Computing and Applications (NCA); 2019 Sep 26–28; Cambridge, MA, USA. p. 1–5.
18. Shi T, McCann RA, Huang Y, Wang W, Kong J. Malware detection for Internet of Things using one-class classification. Sensors. 2024;24(13):4122. doi:10.3390/s24134122.
19. Almazroi AA, Ayub N. Deep learning hybridization for improved malware detection in smart Internet of Things. Sci Rep. 2024;14(1):7838. doi:10.1038/s41598-024-57864-8.
20. Ahmad I, Wan Z, Ahmad A, Ullah SS. A hybrid optimization model for efficient detection and classification of malware in the Internet of Things. Mathematics. 2024;12(10):1437. doi:10.3390/math12101437.
21. Mishra P, Jain T, Aggarwal P, Paul G, Gupta BB, Attar RW, et al. CloudIntellMal: an advanced cloud based intelligent malware detection framework to analyze android applications. Comput Electr Eng. 2024;119(7):109483. doi:10.1016/j.compeleceng.2024.109483.
22. Sharma A, Gupta BB, Singh AK, Saraswat VK. A novel approach for detection of APT malware using multi-dimensional hybrid Bayesian belief network. Int J Inf Secur. 2023;22(1):119–35. doi:10.1007/s10207-022-00631-5.
23. Li T, Luo Y, Wan X, Li Q, Liu Q, Wang R, et al. A malware detection model based on imbalanced heterogeneous graph embeddings. Expert Syst Appl. 2024;246(27):123109. doi:10.1016/j.eswa.2023.123109.
24. Sharma A, Gupta BB, Singh AK, Saraswat VK. Advanced persistent threats (APT): evolution, anatomy, attribution and countermeasures. J Ambient Intell Humaniz Comput. 2023;14(7):9355–81. doi:10.1007/s12652-023-04603-y.
25. Shah IA, Mehmood A, Khan AN, Elhadef M, Khan AuR. HeuCRIP: a malware detection approach for Internet of Battlefield Things. Cluster Comput. 2023;26(2):977–92. doi:10.1007/s10586-022-03618-y.
26. Smmarwar SK, Gupta GP, Kumar S. AI-empowered malware detection system for Industrial Internet of Things. Comput Electr Eng. 2023;108:108731. doi:10.1016/j.compeleceng.2023.108731.
27. Zhang X, Hao L, Gui G, Wang Y, Adebisi B, Sari H. An automatic and efficient malware traffic classification method for secure Internet of Things. IEEE Internet Things J. 2023;11(5):8448–58. doi:10.1109/jiot.2023.3318290.
28. El-Ghamry A, Gaber T, Mohammed KK, Hassanien AE. Optimized and efficient image-based IoT malware detection method. Electronics. 2023;12(3):708. doi:10.3390/electronics12030708.
29. Esmaeili B, Azmoodeh A, Dehghantanha A, Srivastava G, Karimipour H, Lin JCW. A GNN-based adversarial Internet of Things malware detection framework for critical infrastructure: studying Gafgyt, Mirai and Tsunami campaigns. IEEE Internet Things J. 2023;11(16):26826–36. doi:10.1109/jiot.2023.3298663.
30. Shafin SS, Karmakar G, Mareels I. Obfuscated memory malware detection in resource-constrained IoT devices for smart city applications. Sensors. 2023;23(11):5348. doi:10.3390/s23115348.
31. Sharma A, Babbar H, Vats AK. Ransomware attack detection in the Internet of Things using machine learning approaches. In: 2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC 2023); 2023 May 4–6; Salem, India. p. 553–9.
32. Devi RA, Arunachalam A. Enhancement of IoT device security using an improved elliptic curve cryptography algorithm and malware detection utilizing deep LSTM. High-Confidence Comput. 2023;3(2):100117. doi:10.1016/j.hcc.2023.100117.

33. Bajao NA, Sarucam JA. Threats detection in the Internet of Things using convolutional neural networks, long short-term memory, and gated recurrent units. Mesopotamian J Cyber. 2023;2023:22–9. doi:10.58496/mjcs/2023/005.
34. Mohammed MA, Lakhan A, Zebari DA, Abdulkareem KH, Nedoma J, Martinek R, et al. Adaptive secure malware efficient machine learning algorithm for healthcare data. CAAI Trans Intell Technol. 2023. doi:10.1049/cit2.12200.
35. Dzulqarnain D. Investigating IoT malware characteristics to improve network security [master's thesis]. Enschede, Netherland: University of Twente; 2019.
36. Xing X, Jin X, Elahi H, Jiang H, Wang G. A malware detection approach using autoencoder in deep learning. IEEE Access. 2022;10:25696–706. doi:10.1109/access.2022.3155695.
37. Jamal A, Hayat MF, Nasir M. Malware detection and classification in IoT network using ANN. Mehran Univ Res J Eng Technol. 2022;41:80–91.
38. Bhayo J, Shah SA, Hameed S, Ahmed A, Nasir J, Draheim D. Towards a machine learning-based framework for DDoS attack detection in software-defined IoT (SD-IoT) networks. Eng Appl Artif Intell. 2023;123(1):106432. doi:10.1016/j.engappai.2023.106432.
39. Sanchez PMS, Celdrán AH, Bovet G, Pérez GM. Adversarial attacks and defenses on ML- and hardware-based IoT device fingerprinting and identification. Future Gener Comput Syst. 2024;152(12):30–42. doi:10.1016/j.future.2023.10.011.
40. Sahu AK, Sharma S, Tanveer M, Raja R. Internet of Things attack detection using hybrid deep learning model. Comput Commun. 2021;176(3):146–54. doi:10.1016/j.comcom.2021.05.024.
41. Zhu H, Li Y, Li R, Li J, You Z, Song H. SedmDroid: an enhanced stacking ensemble framework for Android malware detection. IEEE Trans Netw Sci Eng. 2020;8(2):984–94. doi:10.1109/tnse.2020.2996379.
42. Sun Z, An G, Yang Y, Liu Y. Optimized machine learning enabled intrusion detection system for Internet of Medical Things. Franklin Open. 2024;6(11):100056. doi:10.1016/j.fraope.2023.100056.