

Received 20 August 2025, accepted 27 September 2025, date of publication 3 October 2025, date of current version 10 October 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3617792

RESEARCH ARTICLE

A Self-Adaptive Intrusion Detection System for Zero-Day Attacks Using Deep Q-Networks

MOUHAMMD ALKASASSBEH¹, EBTEHAL H. OMOUSH¹, MOHAMMAD ALMSEIDIN^{ID2}, AND AMJAD ALDWEESH^{ID3}

¹Computer Science Department, Princess Sumaya University for Technology, Amman 11941, Jordan

²Tafila Technical University, Tafila 66110, Jordan

³Department of Computer Science, College of Computing and IT, Shaqra University, Shaqra 11961, Saudi Arabia

Corresponding author: Amjad Aldweesh (a.aldweesh@su.edu.sa)

This work was supported by the Deanship of Scientific Research at Shaqra University.

ABSTRACT Zero-day attacks remain among the most formidable threats to modern cybersecurity, exploiting undisclosed vulnerabilities that bypass conventional detection mechanisms. In this paper, we propose a self-learning intrusion detection system (IDS) based on Deep Q-Networks (DQNs), a reinforcement learning approach capable of autonomously classifying network traffic without dependence on predefined signatures or labeled training data. Leveraging the UGRansome dataset, the model is evaluated under two experimental scenarios: a conventional random split and a more rigorous zero-day split, where ransomware families in the test set are entirely excluded from training. In binary classification, the proposed system achieves 97.6% accuracy (F1-score: 0.98) in the random split and 95.9% accuracy (F1-score: 0.96) in the zero-day setup, consistently prioritizing high recall for ransomware detection. In multiclass classification, it attains 97.0% and 92.0% accuracy in the random and zero-day splits, respectively. These results underscore the potential of reinforcement learning as a robust and adaptive foundation for real-time intrusion detection in evolving threat landscapes.

INDEX TERMS Zero-day attacks, intrusion detection systems (IDS), deep Q-networks (DQN), reinforcement learning, adaptive cybersecurity, self-learning models, anomaly detection, UGRansome dataset.

I. BACKGROUND

Zero-day attacks (ZDAs) remain one of the most challenging threats in modern cybersecurity, primarily due to their ability to exploit unknown vulnerabilities that have not yet been patched or documented. Unlike traditional cyber threats, which can often be identified through signature-based detection methods, ZDAs circumvent such systems by leveraging undisclosed flaws. This evasion capability necessitates the development of intrusion detection systems (IDSs) that can detect threats based on behavior rather than signatures.

To address this, various machine learning (ML) and deep learning (DL) approaches have been explored for adaptive anomaly detection. However, supervised learning models often require large volumes of labeled data, which may

The associate editor coordinating the review of this manuscript and approving it for publication was Amjad Mehmood^{ID}.

not be feasible for novel or evolving threats. In contrast, unsupervised learning techniques—such as Self-Organizing Maps (SOMs), clustering algorithms, and statistical anomaly detectors—offer promising alternatives by identifying deviations from established patterns without prior knowledge of attack types [1], [2].

Despite these advances, many of these systems still suffer from several limitations, including high false positive rates, poor adaptability to changing network behaviors, and high computational requirements. This is particularly problematic in environments that demand real-time decision-making and scalability across large volumes of traffic.

Reinforcement learning (RL) has emerged as a compelling alternative due to its self-learning capabilities and minimal reliance on labeled data [3], [4]. In RL, agents learn optimal policies by interacting with an environment and receiving rewards or penalties based on the outcomes of their

actions. Among RL algorithms, Deep Q-Networks (DQNs) have demonstrated particular promise in high-dimensional, partially observable spaces—making them well-suited to cybersecurity applications.

A DQN combines the exploration-based learning of RL with the feature extraction capabilities of deep neural networks. By learning Q-values that estimate the expected cumulative reward for taking specific actions in given states, DQNs enable IDS models to adapt to new threats dynamically. This adaptability is especially important in detecting ransomware and zero-day variants, which can differ significantly from previously known malware.

This research proposes a DQN-based intrusion detection framework designed to classify ransomware attacks using network flow data from the publicly available UGRansom dataset. Two experiments are conducted to evaluate the model's effectiveness: one using a conventional random split to establish baseline performance, and another simulating a zero-day condition where entire ransomware families in the test set are excluded during training. This setup closely mimics real-world threat dynamics and tests the system's ability to generalize.

Our methodology integrates data preprocessing, ransomware family-based zero-day splitting, custom RL environment design, DQN training, and model evaluation. The DQN model is optimized to minimize false negatives—a critical metric for ransomware detection—while maintaining high recall and accuracy. Experimental results show that the proposed system achieves up to 98% accuracy in detecting both known and previously unseen ransomware variants.

Building upon our previous work on reinforcement learning-based botnet detection [5], this study extends the applicability of adaptive, self-learning approaches to the more complex and evolving challenge of zero-day ransomware attacks. By leveraging the strengths of Deep Q-Networks, our system demonstrates the potential of RL for scalable, autonomous, and real-time threat detection in cybersecurity.

The remainder of this paper is organized as follows: Section II discusses the theoretical background of zero-day attacks and reinforcement learning. Section III reviews recent advancements in AI-based intrusion detection systems. Section IV outlines the proposed DQN-based framework and experimental setup. Section V presents and analyzes the results. Finally, Section VI concludes the paper and outlines potential directions for future research.

II. BACKGROUND

Zero-day attacks (ZDAs) are among the most sophisticated and damaging cyber threats, characterized by their exploitation of vulnerabilities that have not yet been disclosed or patched. Because these attacks target unknown flaws, they evade detection by traditional intrusion detection systems (IDSs), which rely primarily on signature-based mechanisms [6]. As a result, there is a growing need for intelligent and adaptive systems that can detect anomalies beyond predefined patterns.

Recent advancements in artificial intelligence (AI), particularly machine learning (ML) and deep learning (DL), have significantly influenced the development of modern IDSs. Supervised ML models, including decision trees, support vector machines (SVMs), and ensemble classifiers, have been employed to classify malicious traffic. However, these models often depend on large amounts of labeled data and may struggle to generalize to new or evolving threats [7]. To address these limitations, unsupervised learning methods such as clustering, statistical analysis, and Self-Organizing Maps (SOMs) have been proposed to detect outliers and anomalous behavior without labeled input [1], [2].

Deep learning models offer a promising alternative by learning complex data representations and capturing latent threat patterns in network traffic. Techniques such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and autoencoders have been used to extract high-level features from raw traffic data, improving detection accuracy and scalability [8], [9]. Additionally, attention mechanisms and generative models like GANs have enhanced the flexibility of DL-based IDSs.

Despite these advancements, most existing models face challenges such as poor adaptability to evolving threats, limited interpretability, high false positive rates, and computational inefficiencies—especially in real-time detection scenarios [10]. Moreover, many systems require retraining or manual intervention to remain effective, limiting their scalability in dynamic environments.

Reinforcement learning (RL) presents a promising solution to these challenges. Unlike supervised learning, RL enables agents to learn optimal behaviors by interacting with their environment and receiving feedback in the form of rewards or penalties [3], [4]. Deep Q-Networks (DQNs), which integrate deep learning with Q-learning, have been successfully applied in domains such as robotics, autonomous systems, and games. They are particularly suitable for cybersecurity tasks that involve sequential decision-making under uncertainty.

DQNs learn Q-values that estimate the expected cumulative reward of actions in specific states, allowing the model to make informed decisions even in high-dimensional or partially observable environments. This capability is crucial for detecting advanced threats such as ransomware, which often exhibit complex and time-dependent behaviors.

Although DQNs have demonstrated strong performance in various AI applications, their application to real-world intrusion detection—particularly for detecting zero-day ransomware—is still underexplored. This research aims to bridge that gap by designing a self-learning IDS that uses DQNs to autonomously detect ransomware in flow-based network traffic, even when confronted with previously unseen attack families.

A. ZERO-DAY ATTACK DETECTION CHALLENGES

Zero-day attacks (ZDAs) exploit previously unknown security vulnerabilities, making them difficult to detect and

mitigate. The following points summarize the main challenges in detecting ZDAs:

- **Unknown Signatures [11]:** Zero-day attacks leverage undisclosed vulnerabilities, rendering signature-based detection methods ineffective.
- **Lack of Training Data [11]:** The limited availability of labeled data for unseen threats hampers the effective training of supervised machine learning models.
- **Evasion Techniques [12]:** Attackers use polymorphism and obfuscation to disguise malicious behavior, enabling them to bypass traditional detection systems.
- **High False Positive Rates [13]:** Many IDS misclassify benign activity as malicious, leading to alert fatigue and increased burden on security personnel.
- **Rapid Evolution of Threats [11]:** Zero-day attacks evolve quickly, often outpacing the adaptability of conventional security mechanisms.
- **Real-Time Detection Limitations [14]:** Striking a balance between detection accuracy and real-time response remains a significant challenge.

B. REINFORCEMENT LEARNING APPROACH

Reinforcement Learning (RL) is a machine learning paradigm where agents learn optimal behaviors through interaction with an environment, guided by rewards or penalties [15]. Its adaptive and autonomous decision-making capabilities make RL particularly suitable for dynamic domains such as cybersecurity, especially for detecting complex threats like zero-day attacks. An RL system typically comprises the following components:

- **Agent:** The entity that makes decisions.
- **Environment:** The system or network with which the agent interacts.
- **State:** A representation of the current context or situation.
- **Action:** The set of possible decisions available to the agent.
- **Reward:** Feedback from the environment in response to the agent's actions.
- **Policy:** A strategy that guides the agent in selecting actions based on states.
- **Value Function:** A measure that estimates the long-term return of taking certain actions in given states.

The agent observes the current state, takes an action according to its policy, receives a reward, and updates its policy to improve future decision-making. Deep Reinforcement Learning (DRL) [16], an extension of RL, leverages deep neural networks to address high-dimensional state spaces and is widely used in sequential decision-making tasks. DRL approaches are broadly categorized into model-based and model-free methods. Model-free techniques, such as Q-learning and Deep Q-Networks (DQNs), learn policies directly from interactions without explicitly modeling the environment. In contrast, model-based approaches build an internal model of the environment's dynamics to plan and make decisions.

RL has shown great potential in cybersecurity applications. For instance, RL-based IDS can dynamically learn to detect anomalous patterns in network traffic and adapt to emerging attack strategies over time. It can also be applied in adversarial training, where RL agents generate adversarial inputs to evaluate and enhance the robustness of detection systems [16]. These characteristics make RL a promising direction for addressing the unique challenges posed by zero-day attacks.

C. DEEP Q-NETWORK (DQN) [4], [17]

It is an RL algorithm that combines Q-learning with Deep Neural Networks (DNN) to enable agents to learn how to act optimally in complex, high-dimensional environments. Traditional Q-learning maintains a Q-table to estimate the value of state-action pairs, but this becomes impractical when the state space is too large. DQNs address this by using a DNN to approximate the Q-function, where the input is the current state and the output is a set of Q-values corresponding to all possible actions. To improve training stability and convergence, DQNs employ two important techniques:

- **Experience Replay:** Stores past experiences (state, action, reward, next_state) in a buffer and randomly samples batches for training. This breaks the correlation between consecutive samples.
- **Target Network:** A separate, periodically updated network is used to generate stable target Q-values during training, improving convergence.

The objective of training a DQN is to approximate the optimal action-value function $Q^*(s, a)$ by minimizing the discrepancy between predicted and target Q-values. A deep neural network with parameters θ is used to estimate the Q-function $Q(s, a; \theta)$. The network is trained by minimizing the loss function:

$$L(\theta) = \mathbb{E}_{(s, a, r, s') \sim D} \left[\left(r + \gamma \max_{a'} Q_{\text{target}}(s', a'; \theta^-) - Q(s, a; \theta) \right)^2 \right] \quad (1)$$

where r is the reward, $\gamma \in [0, 1]$ is the discount factor, s' is the next state, and θ^- are the parameters of a target network that is held fixed for a number of iterations and then updated. The expectation is taken over experiences (s, a, r, s') sampled uniformly from a replay buffer D , which stores past transitions to reduce correlation and stabilize training. By minimizing this loss, the DQN learns a policy that aims to maximize the expected cumulative reward over time.

III. RELATED WORK

IDS have been widely studied using both supervised and unsupervised ML approaches. Supervised techniques classify network traffic relying on labeled data. Supervised methods usually detect known attacks well but may struggle with ZDAs. Unsupervised techniques detect unknown or ZDAs without any prior knowledge of attack patterns and do not

require labeled datasets. This section categorizes related work based on these two learning approaches.

A. SUPERVISED LEARNING APPROACHES

Khan et al. [18] designed an effective NIDS that addresses unbalanced dataset issue. They used set of ML models trained on reliable flow-based data (CICIDS-2017). Firstly, the authors balanced the dataset using the SMOTE-Tomek Links technique and Then applied each DT, RF, XGBoost, KNN, NB, LR, and AdaBoost algorithms on it. The highest accuracy recorded by DT model with 0.9637 accuracy and 0.9633 F1-score and the AdaBoost model with 0.9637 accuracy and 0.9633 F1-score for multiclass classification. For binary classification, DT model has exhibited the highest test accuracy of 0.9996, followed by RF 0.9984, Adaboost 0.9977, and Xgboost 0.9957, with the highest average precision of 100% and ROC-AUC of 0.9996. The finding summarize with binary classification performs better when it takes more time to train each classifier than multiclass classification and the balanced CICIDS-2017 dataset improves the performance of DT and AdaBoost classifiers.

Srivastav et al. [7] proposed a ML approach for detecting the intrusion. The study assesses DT, RF, KNN, and LR models using a Kaggle-sourced dataset with 31 attributes. The preprocessing stage involved cleaning the data, normalization, and feature selection. The experimental results reveal that RF recorded the highest accuracy of 0.9865 and excels other classifiers in execution speed, precision, recall, and F1-score.

Amor et al. [19] applied the NB algorithm for intrusion detection. The experiment was conducted on the KDD'99 datasets, which are generally used for benchmarking intrusion detection problems. They analyze attacks at three levels: as whole attacks, grouped into four main categories, or simply as normal vs. abnormal behavior. They also compare the performance of NB with the DT technique. In all cases, classification accuracy is measured using the Percent of Correct Classification (PCC). The results indicate that NB performs well and is nearly as effective as DT. At the first level, which evaluates how well each model detects individual attacks, the accuracy in testing was 0.914 for DT and 0.912 for NB. At the second level, where the focus is on classifying connections into five categories (normal, DoS, R2L, U2R, and Probing), the accuracy was 0.9228 for DT and 0.9147 for NB. At the third level, which distinguishes between normal and abnormal connections, the accuracy was 0.9302 for DT and 0.9145 for NB.

Nhlapo et al. [20] detect zero-day and ransomware attacks by training various ML models on the UGRansome dataset. The ML models used were RF, XGBoost, Adaboost, SVM, NB, DT, Multilayer Perceptron (MLP) and Ensemble Method. Additionally, the SMOTE technique was used to improve the model performance. The results presented a very high performance for each RFC, XGBoost, and Ensemble Method, reaching 100% for all metrics, including accuracy,

precision, recall, and F1-score. In contrast, SVM and NB showed lower accuracy, scoring 0.69 and 0.78, respectively.

Kadri et al. [8] introduce enhanced preprocessing techniques alongside a comprehensive evaluation of traditional ML models and a DNN for real-time network intrusion detection. The preprocessing pipeline encompasses univariate analysis, feature selection, outlier handling, duplicates and missing value management, data normalization, and PCA-based dimensionality reduction. Using a PCA aims to simplify the utilized dataset by reducing the dimensionality without sacrificing important information. Feature selection improved model interpretability, reduced the risk of overfitting, and enhanced computational efficiency. They applied three classic classifiers: Extra Trees, GB, DT, and a DNN to detect intrusions in network traffic. Their experiments were performed on a nearly balanced network intrusion detection dataset that simulates real-world military network scenarios with various cyberattacks, including raw TCP/IP dump data, where each connection is labeled as either ‘Normal’ or ‘Anomalous’ (attack). It contains 41 features per connection, 38 numerical features, and 3 categorical features. The dataset was divided into train and test sets in the ratio 80:20. The performance of both traditional and deep learning models was evaluated using precision, recall, kappa, accuracy, F1 score, ROC curve analysis, and confusion matrices. Results indicate that the proposed method is efficient for real-time network intrusion detection and threat analysis. The DNN model enhanced detection capabilities, achieving the highest metrics across all evaluation parameters, with an AUC of 1.00, an accuracy of 0.995, and the lowest misclassification rate.

Tun et al. [21] evaluated the performance of NB for IDS using NSL-KDD and DARPA datasets. The study emphasizes the impact of data preprocessing, particularly discretization, on improving classification accuracy. The results demonstrate that NB with discretization excels the non-discretized version, recording 0.9713 accuracy on NSL-KDD and 0.9983 accuracy on DARPA. The study also examine feature selection, data cleaning, and transformation to enhance detection performance. Evaluation metrics, including precision, recall, and false positive rates, show that discretization minimize misclassification errors.

Alexis [22], in his PhD thesis, explores the use of Automated Machine Learning (AutoML) to identify zero-day cyber exploits targeting Modbus TCP/IP, a popular protocol used in Industrial Control Systems (ICS). Given Modbus's absence of embedded security features, such as authentication, encryption, and ICS, networks are Prone to various cyber threats, including Man-in-the-Middle (MITM), Denial of Service (DoS), and ransomware attacks. The thesis assessed machine learning models using the PyCaret library, analyzing their accuracy and efficiency in detecting unknown attacks. The dataset, sourced from Kaggle, was processed with various rescaling techniques, and statistical tests (Welch's ANOVA, Games-Howell) were applied to discover major differences between model performances.

The results showed that the best-performing models were non-rescaled Gradient Boosting 0.9879, Normalized Light Gradient Boosting 0.9875, and non-rescaled Light Gradient Boosting 0.9873.

Nagaraja et al. [23] performed regression analysis for intrusion detection. The experiment was conducted on KDD and NSL-KDD datasets. They applied dimensionality reduction techniques with different thresholds on the reduced data. The mean absolute percentage error (MAPE) and accuracy are calculated. The highest accuracy on the KDD dataset was 0.9864 at a threshold of 0.999, while the best MAPE was 0.0929 at a threshold of 0.999. On the NSL-KDD dataset, the highest accuracy reached 0.6455 at a threshold of 0.999, and the best MAPE was 0.452 at a threshold of 0.999. The results showed that regression analysis performed better on the KDD dataset compared to the NSL-KDD dataset.

Wisanwanichthan et al. [24] proposed a Double-Layered Hybrid Approach (DLHA) to detect anomalies and unseen attacks, especially uncommon attacks e.g., Remote2Local (R2L) and User2Root (U2R). DLHA anomaly-based IDS works to detect anomalous connections in a real-time manner. In the first layer, the researchers use NB to detect DoS and Probe attack and adopt SVM in the second layer to distinguish R2L and U2R from normal instances. The two layers work in a cascading manner. Their experiments were performed on NSL-KDD dataset. The results showed that DLHA excels many current state-of-the-art IDS techniques, and is significantly better than any single ML classifier by large margins. DLHA also displays an outstanding performance in detecting rare attacks by obtaining a detection rate of 0.966 and 1.0 from R2L and U2R, respectively.

B. SEMI-SUPERVISED LEARNING APPROACHES

Saurabh et al. [25] propose a Hybrid Multi-Stage Intrusion Detection System (HMS-IDS) for securing Industrial Internet of Things (IIoT) environments against ZDAs and Advanced Persistent Threats (APTs). Classical ML-based IDS struggle with unknown threats, as they primarily detect known attack signatures. HMS-IDS addresses this limitation by integrating supervised learning for detecting known attacks with unsupervised clustering for detecting unknown threats, specifically, K-Means clustering with cluster labeling (CL-K-Means), to identify unknown threats. The system is assessed on the CIC-ToN-IoT dataset, achieving 0.9949 accuracy in identifying well-known attacks and 0.9893 accuracy for unknown threats. The model's reliability is further tested on KDD-99 Cup, NSL-KDD, and CICIDS 2017 datasets. By leveraging feature selection and ensemble learning, HMS-IDS outperforms conventional IDS solutions in real-time threat detection for IIoT environments.

Sarwar et al. [26] examine the effectiveness of hybrid DL models in detecting network intrusions on imbalanced datasets. Three models were employed: CNN+DNN, Autoencoder+DNN, and DNN+XGBoost. The CSC IDS 2018 dataset was used and balanced using SMOTE,

FSMOTE, and CTGAN techniques. The SHAP feature selection and correlation were applied. The best performance was recorded for the Autoencoder with DNN and CNN with DNN models, especially with FFSMOTE and SMOTE.

Kim et al. [27] proposes a semi-supervised malware detection approach to overcome the limitations of signature-based and supervised learning methods in detecting zero-day malware. The method profiles benign characteristics, integrating autoencoding and one-class classification (OC) to identify previously unseen malware. Traditional OC classifiers may be limited with low detection rates, while autoencoders are sensitive to threshold settings. Traditional OC classifiers suffer from low detection rates, while autoencoders are sensitive to threshold settings. To address this, the authors leverage autoencoders for feature abstraction and OC classification for threshold-free decision-making. They examined four OC classification methods; the best-performing model was OCSVM, which achieved up to 0.916 accuracy. This result was considerably lower than those of conventional supervised learning techniques that yield at least 0.96 of accuracy in the fair comparison. Autoencoder-based detection outperformed OC classifiers, reaching 0.99 accuracy, but only with an optimal threshold, which is difficult to determine manually. To eliminate this dependency, the authors introduce AE-OCC, a threshold-free approach. The autoencoder learns benign patterns during training, and the OC classifier determines if new samples fit within this learned boundary positive 1 or not negative 1. Their experiments were conducted on Meraz18 and Drebin datasets, and the method presented with up to 0.971 accuracy.

Kim et al. [28] recognized that most network intrusion detection research relies on session-based features, making it difficult to detect intrusions before a session ends. To reduce detection delay, the authors propose a real-time packet-based detection method that classifies malicious traffic early. However, since initial traffic often resembles normal traffic, a fixed number of packets is typically needed for accurate classification. To overcome this, the study integrates a GAN with a Long LSTM classifier. Since standard GANs often fail during training, the authors use Wasserstein GAN with Gradient Penalty (WGAN-GP) to ensure stability. The GAN-based one-class classifier identifies packets that are difficult to classify, determining whether intrusion detection is feasible. Unlike traditional methods, it does not require a fixed number of packets per session, enabling immediate intrusion detection when a malicious packet arrives. The GAN discriminator is trained using only the packets misclassified by the LSTM classifier for the training dataset, so the characteristics of the packets that are highly likely to be misclassified are accurately learned by the GAN generator. Whenever a packet is received by NIDS/NIPS, the LSTM classifier is used to determine whether the session, including the packet, is malicious or not, and the result is verified using GAN. If the GAN determines low classification reliability, the result is ignored; if high, the session is processed accordingly. The active session cache is used to determine

whether a received packet is an existing session packet or a new session packet. This method has two advantages. First, the classifier can guarantee that it consumes only a fixed size of memory for classification, so it can support a high scalability in terms of the concurrent session number. Second, the fixed number of features makes the classifier design easy. The method was tested on ISCX2012, CIC2017, and CSE2018 datasets, with evaluations based on accuracy, precision, recall, F1-score, and detection speed (measured by packets needed per session). The results showed superior speed and early threat mitigation compared to existing techniques.

Carrera et al. [29] design an anomaly detection system capable of analyzing near real-time network traffic by utilizing classical unsupervised ML algorithms to identify anomalous network traffic promptly. The authors combined DL and Shallow Learning algorithms to maximize the prediction speed using DL techniques and, at the same time, maximize the accuracy index using robust Shallow Learning algorithms. The results were analyzed in terms of precision, recall, F1, AUC ROC, and accuracy. This paper explores the key features that various models prioritize when classifying an event as an attack, which was the SHAP methodology for XAI. The experiment implemented on KDD99, NSL-KDD, and CIC-IDS2017 datasets.

Salem et al. [9] present a Threat Intelligence Detection Model (TIDM) method for detecting ZDAs in large data flows. The model consisted of three components, which are an optimized filter (OptiFilter), an adaptive hybrid classifier, and alarm components. The OptiFilter component is contributed in its ability to capture data flows and process them in a queue of dynamic window size and create unlabeled connection vectors continuously. The hybrid classifier detects anonymous connections by using an Enhanced Growing Hierarchical SOM (EGHSOM) and a normal network behavior (NNB) model. The EGHSOM classifies the unlabeled connections in one of the three labels(normal, anomaly, or unknown), and the NNB model tests whether the unknown connections are “normal” or “anomalous.” The model updates in real-time continually. In the evaluation phase, the model’s performance is measured in both offline and online modes. The final results reveal on model’s ability to process massive data flows in real-time, classify unlabeled connections, reveal the label of unknown connections, and execute online updates successfully.

C. UNSUPERVISED LEARNING APPROACHES

Kumar et al. [30] proposes a new method for spotting unknown cyber-attacks (ZAs) by combining the heavy-hitters (HH) concept and graph technique. This model handles unknown high-volume attacks (HVA) and unknown low-volume attacks (LVA). Their approach extends detection abilities by leveraging modern network traffic from familiar attacks, enabling it to discover a broader range of threats. Unlike standard models, this system processes raw byte

streams from real-time network traffic without depending on certain network, source, or destination details. Performance evaluations show that it outperforms current methods in both binary and multi-class classification tasks. The system’s effectiveness is examined using real-time network logs and the CICIDS18 dataset. The results show higher performance, with an accuracy of 0.913 for binary classification and 0.903 for multi-class classification on real-time attack data. Furthermore, the model recorded an accuracy of 0.916 for binary classification when applied to the CICIDS18 dataset.

Afek et al. [31] Introduced a basic tool for automatic zero-day attack signature extraction for high-volume attacks. The Double Heavy Hitters algorithm (DHH) and Triple Heavy Hitters algorithms were used to extract the desired signatures. The algorithm runs in linear time, requiring one pass over the input and space dependent only on the predetermined size of the heavy hitters data structure. Furthermore, the authors provide an extended algorithm that is able to identify groups of signatures, often found together in the same packets, which further improves the quality of signatures generated by their system. Testing their system on captures of real-life attacks has detected high-volume attacks with very high recall and precision rates.

Li et al. [1] propose self-learning anomaly detection approach based on SOM. Their experiments relied on using a real-world dataset (MVTec AD) for unsupervised industrial anomaly detection and localization. The method utilizes pre-trained CNN (Wide-ResNet50-2) for feature extraction. The proposed SOMAD method accurately memorizes normal patterns and detects anomalies using Mahalanobis distance-based scoring. The result demonstrates that the SOMAD’s potential for zero-day ID, where it recorded 0.978 pixel-level AUROC, 0.933 PRO-score, and 0.979 image-level AUROC.

Singh et al. [32] proposed a framework for ML anomaly detection in order to improve the IDS capability. The focus was on an unsupervised approach for detecting each well-known unknown (ZDAs) using One-Class SVM (OCSVM) and active learning techniques. The CIC-IDS2017 dataset was used and compared with the UNSW NB15 and KDDCup ’99 datasets. This study utilized each of SVM and isolation-based clustering, to classify normal and suspicious network activity. Additionally, K-Means clustering was applied for structured data and DBSCAN for noisy data, enabling flexible anomaly detection. The result shows that this study exceeds exciting approaches in term of accuracy and efficiency.

Zhang et al. [33] present an unsupervised anomaly detection approach for NIDS. The AE was used in order to enhance the model’s ability to learn normal traffic patterns and identify deviations as anomalies. To enhance efficiency, the authors involve time correlation features and hierarchical clustering in data preprocessing, reducing both time and space complexity. The main model parts are the data preprocessing part, which filters raw network traffic, extracts statistical features, and clusters them, and

the AutoEncoder-based anomaly detection module, which reconstructs input traffic and flags anomalies based on reconstruction error. The model is trained on a dataset of 50,000 normal network traffic samples, collected from a cloud environment using the Tmarlin platform. The model testing phase includes four distinct datasets: (a) New malware attack dataset, gathered from Unit 42 Wireshark Quiz, representing novel malware behavior; (b) Worm attack dataset, generated via Flow Injection, including DCE RPC packet and DNS attacks; (c) System vulnerability dataset, containing Windows Locator Service buffer overflow vulnerabilities (CA-2003-03); and (d) Botnet dataset, derived from ISOT, featuring malicious DNS traffic generated by various botnets. Experimental results show 0.98 accuracy for worms and system vulnerabilities and 0.89 accuracy for botnets.

Shukla et al. [34] propose an unsupervised IDS designed for Industry 4.0 environments (UInDeSI4.0), addressing the limitations of securing cyber-physical systems (CPS) and industrial networks. The suggested model uses RF for feature selection. Then, these selected features are used to train an Isolation Forest (IF), an unsupervised anomaly detection approach that identifies cyber threats relying on anomaly scores, making it effective in scenarios where labeled attack data is missing. The model is evaluated on the UNSW-NB15 Industry 4.0 dataset, demonstrating a superior accuracy of 0.63 compared to classical IDS approaches while using only nine key features. Unlike DL-based approaches, which require high computational resources, UInDeSI4.0 offers a lightweight and efficient option, making it suitable for real-time deployment in industrial environments. The system also outperforms alternative feature selection methods, like PCA and Independent Component Analysis (ICA), in terms of accuracy and execution time. This research highlights the potential of tree-based unsupervised learning for efficient and scalable ID in modern industrial ecosystems.

Borgioli et al. [35] present self-learning network intrusion detection in cyber-physical and embedded systems. The proposed model used a One-Dimensional Convolutional AE (1D CNN AE). The learning process for normal network traffic patterns and detecting anomalies was by using reconstruction error. The packet-level approach enables more granular detection than flow-based IDS, reducing the likelihood of attackers evading detection. The model is trained only on benign traffic, allowing it to identify previously unseen attack types without needing attack labels. Random noise is introduced during the training phase to enhance resilience. The experiment was implemented on the EDGE-IIOTSET dataset, and it recorded 0.99 accuracy across multiple attack types, outperforming prior IDS solutions such as Kitsune IDS. Furthermore, it is optimized for real-time deployment on embedded edge devices, reaching packet processing rates of 1 Gbps with GPU acceleration. A heatmap-based explainability mechanism is also introduced to visualize packet sections that contribute to anomaly detection, improving interpretability.

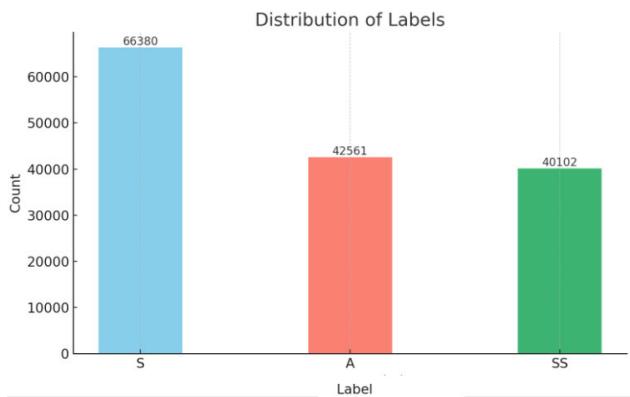
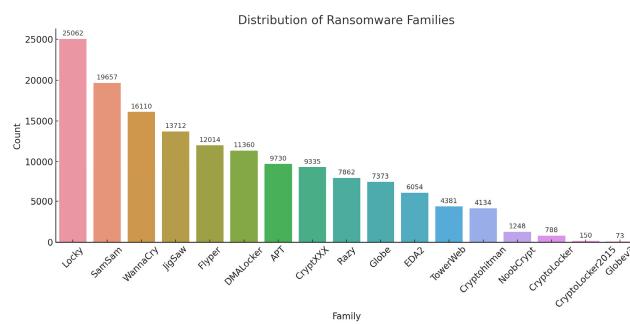
de Araujo-Filho et al. [36] introduce an unsupervised GAN-based IDS model to detect both well-known and ZDAs. The temporal convolutional networks (TCNs) and self-attention techniques were used in the GAN generator and discriminator. Various TCN and self-attention GAN architectures were assessed. Additionally, the IDS is configured to satisfy different requirements in order to trade-offs between detection accuracy and detection time. Furthermore, the study achieved an efficient service delivery with reduced end-to-end latency by leveraging edge computing and deploying the system on edge servers positioned closer to the monitored network nodes. They used the CICDDoS2019 dataset. The final results show that the proposed approach excels FID-GAN and ALAD state-of-the-art GAN-based IDSs in accuracy and speed. FID-GAN uses LSTM in both its GAN generator and discriminator, whereas ALAD relies on fully connected and regular convolutional networks. Notably, the proposed model with a TCN Block ($N = 2$) records the highest accuracy, 0.9707, precision of 0.9705, recall of 0.971, and F1-score of 0.97.

Oluwadare et al. [2] conducted a survey on the most widely used techniques of self-learning algorithms-based IDS. The survey paper assesses these methods based on their popularity, strengths, limitations, and resource requirements. The reviewed techniques involved LSTM, AE, RBM, DBN, MLPs, RBFNs, GANs, RNNs, CNNs, and SOMs. As a result, the authors suggest a framework to enhance IDS capabilities against both well-known and unknown attaches by leveraging the LSTM and AE. The framework aims to enhance anomaly detection and mitigate the challenges posed by ZDAs.

D. DATASET CONTRIBUTIONS

Nkongolo et al. [37] built an up-to-date and modern network traffic dataset named UGRansome for anomaly detection in 2021. The dataset contains 149043 data points and 14 columns. UGRansome is an imbalanced dataset with a non-uniform distribution of instances. It was labelled into three classes as follows: anomaly (A), signature (S), and synthetic signature (SS), as shown in Figure 1, which illustrates the distribution of prediction labels—S, SS, and A—used in the UGRansome dataset to categorize network activity. Figure 2 presents the distribution of ransomware families within the dataset. The name of the zero-day threat. For instance, WannaCry, Locky, APT, Globe, JigSaw, EDA2, and SamSam. Also, Figure 2 highlights the dominance of certain families, such as Locky and WannaCry, reflecting their prevalence in real-world ransomware incidents.

The UGRansome includes real malware and netflow patterns extracted from publicly available datasets. The UGR'16 dataset was used because it was designed in 2018. The ransomware dataset built in 2019 was also utilized to retrieve salient malware patterns. The main asset of the proposed dataset is its implication in the detection of zero-day attacks and anomalies that have not been explored before and cannot be recognized by known threat signatures.

**FIGURE 1.** Distribution Of class Labels (S, SS, A).**FIGURE 2.** Distribution Of Ransomware families.

The Random Forest algorithm was applied to UGRansome, and the result revealed an improvement in zero-day threat detection. The UGRansome dataset is formatted in CSV format and available on the Kaggle site.

IV. METHODOLOGY

This section presents the design and implementation of the proposed DQN-based intrusion detection system (IDS), which is evaluated using multiple experiments on the UGRansome dataset. The methodology comprises data preprocessing, environment modeling, agent design, training configuration, and evaluation strategies.

A. DATASET AND PREPROCESSING

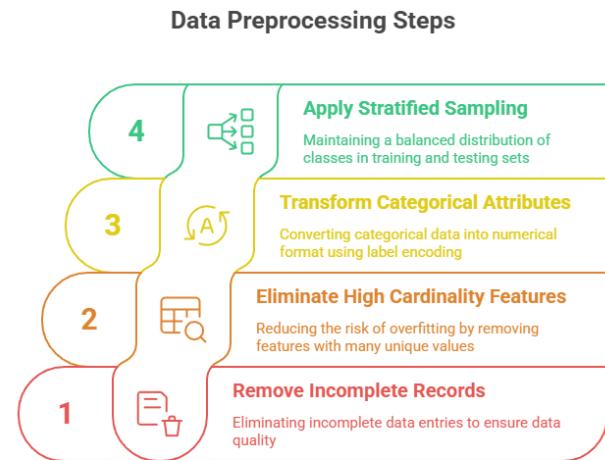
In our experiments, we used the UGRansome dataset [16], which is widely recognized for evaluating ransomware and zero-day attack detection techniques. This dataset consists of 149,043 network flow records, each described by 14 features. These features include attributes such as time, protocol, flag status, malware family, cluster information, threat indicators, and financial data like USD and BTC values. Each network flow is labeled with one of three categories: Anomaly (A), Signature (S), or Synthetic Signature (SS). Table 1 summarizes the UGRansome Dataset.

At the beginning, several preprocessing steps were applied to the UGRansome dataset. First, incomplete records were removed to ensure data quality. Subsequently, features with

TABLE 1. Summary of the UGRansome dataset.

Property	Description
Dataset Name	UGRansome [16]
Application	Ransomware and zero-day detection
Total Records	149,043 network flows
Number of Features	14
Selected Features	Time, Protocol, Flag, Family, Clusters, Threats, USD, BTC
Label Types	Anomaly (A), Signature (S), Synthetic Signature (SS)

high cardinality, such as SeedAddress, ExpAddress, and IPAddress, were eliminated to reduce the risk of overfitting. In addition, categorical attributes, including Protocol, Flag, Family, and Threats, were then transformed into a numerical format using the label encoding technique. Stratified sampling was employed to maintain a balanced distribution of classes in both training and testing sets. Finally, all numerical features were normalized using Min-Max scaling to a range of [0, 1]. Figure 3 presents the steps of dataset preprocessing.

**FIGURE 3.** Steps of dataset preprocessing.

Every network flow record is handled as a separate *state* that the learning agent detects. Given the features of this condition, the agent has to decide on a suitable *action*, which in this regard relates to labeling the flow as either benign or ransomware. The surroundings react with a *reward signal*—a scalar value reflecting the correctness of the selected action—once the agent decides. While an incorrect classification results in a lower or negative reward, guiding the agent to change its decision-making strategy in next interactions, a correct classification results in a positive reward, motivating the agent to reinforce that behavior. Through constant environmental interaction and reward feedback, the agent learns to increase its classification accuracy and fit changing network traffic over time. The environment consists of:

- **States:** Feature vectors representing each network flow.

- **Actions:** Discrete class predictions (e.g., benign, ransomware family).
- **Rewards:** Positive for correct predictions and negative for misclassifications.

The asymmetric reward function emphasizes minimizing false negatives, crucial in ransomware defense. The utilized asymmetric reward is presented in Table 2.

TABLE 2. Reward scheme.

Prediction	Outcome	Reward
True Positive (TP)	Correct ransomware detection	+1.0
True Negative (TN)	Correct benign detection	+0.1
False Positive (FP)	Benign misclassified as ransomware	-0.5
False Negative (FN)	Missed ransomware detection	-1.0

To balance exploration and exploitation during training, we use a ϵ -greedy exploration policy. In order to promote exploration of the surroundings, the agent initially sets ϵ to 1.0 and begins with a high exploration rate. With each step, ϵ progressively drops by a decay factor of 0.995, enabling the agent to favor the exploitation of learned strategies more and more. The agent maintains some degree of exploration during the learning process thanks to this decay, which lasts until it reaches a minimum value of 0.1. The DQN agent learns from mini-batches sampled from a replay buffer. Q-value updates use the Mean Squared Error (MSE) between predicted and target Q-values, with periodic updates to a target network to stabilize training.

This work utilizes a Deep Q-Network (DQN) aimed at effectively mapping network traffic characteristics to optimal actions. The architecture consists of an input layer corresponding to the flow-based feature count in the dataset. Two completely connected hidden layers, each with ReLU activation functions to provide non-linearity and raise learning capacity, are compared to this. Each layer has 64 neurons. By generating Q-values matching every conceivable class, the output layer helps the agent to make decisions depending on its present condition.

Experiments on a system running Windows 11 Pro on an Intel Core i7-8565U CPU, 16 GB RAM, and an NVIDIA MX250 GPU evaluated the DQN's performance. Python 3.10 and PyTorch 2.0 were used in development together with widely used libraries, which include pandas, NumPy, scikit-learn, and matplotlib. Table 3 summarizes key hyperparameters used during training, including learning rate, discount factor, batch size, number of episodes, and replay memory size. These values had been selected especially to ensure efficient convergence and training stability.

B. HYPERPARAMETERS

We developed four experimental settings meant for both binary and multiclass classification scenarios in order to fully assess the performance of our model. The initial setup (Exp. 1), which employs randomly shuffled training and testing splits for binary classification, enables one to

TABLE 3. Hyperparameters used in DQN training.

Parameter	Value
Learning Rate (α)	0.001
Discount Factor (γ)	0.99
Batch Size	64
Episodes	10
Steps per Episode	20,000
Epsilon (Start/End)	1.0 / 0.1
Decay Rate	0.995
Loss Function	MSE
Replay Memory Size	10^5
Optimizer	Adam

evaluate the baseline performance of the model under normal circumstances. Families utilized in the training phase are not included in the testing phase in the second setup (Exp. 2), therefore creating a more challenging scenario employing a zero-day setting. This configuration tests the generalization capacity of the model against hitherto unheard-of malware families.

Concerning classification among known families for multiclass evaluation, the third configuration (Exp. 3) uses both training and testing data, including instances from every class. We analyze the ability of the model to distinguish among numerous classes in a well-known environment. Conversely, the fourth configuration (Exp. 4) examines the model under multiclass zero-day settings in which the testing set comprises malware families not seen during training. This last configuration clarifies the durability and adaptability of the model in real-world situations when new malware types are always developing. Figure 4 presents the experiment setup. Algorithm 1 summarizes the training proposed DQN-based IDS.

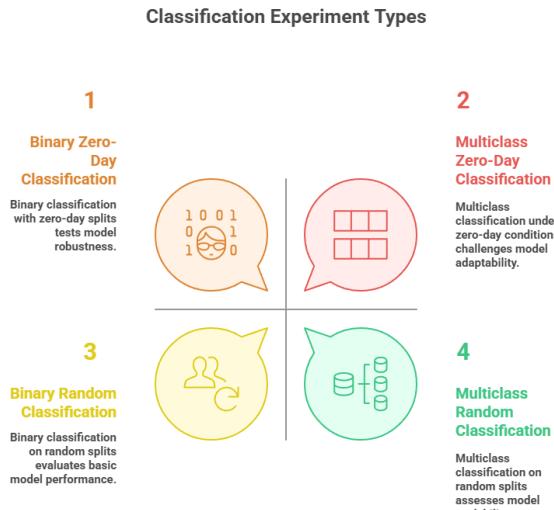
V. EXPERIMENTS AND RESULTS

This section presents the experimental setup and results obtained using the proposed DQN-based intrusion detection system.

A. EXPERIMENT 1: BINARY CLASSIFICATION (RANDOM SPLIT)

In this experiment, our goal was to general ransomware classification. The dataset was train/test randomly split using an 70/30 ratio, ensuring that all ransomware families were represented in both sets. The dataset comprised 82,663 samples labeled as class S and 66,380 samples collectively labeled as classes SS and A. This scenario reflects a traditional supervised learning setting where the model is exposed to examples from all attack types. All ransomware families were included in both sets. After preprocessing, a custom RL environment was constructed where the DQN agent classified each flow as benign (S) or ransomware (S and A) as follows:

- **State:** At time step t , the agent observes a feature vector $s_t = x_t \in \mathbb{R}^d$.

**FIGURE 4.** Experiment setup classification.

- **Action:** $a_t \in \{0, 1\}$, where $a_t = 1$ indicates ransomware and $a_t = 0$ indicates benign.
- **Reward:**

$$r_t = \begin{cases} +1.0 & \text{if } a_t = 1 \wedge y_t = 1 \quad (\text{TP}) \\ +0.1 & \text{if } a_t = 0 \wedge y_t = 0 \quad (\text{TN}) \\ -0.5 & \text{if } a_t = 1 \wedge y_t = 0 \quad (\text{FP}) \\ -1.0 & \text{if } a_t = 0 \wedge y_t = 1 \quad (\text{FN}) \end{cases}$$

- **Transition:** The environment moves to the next sample: $s_{t+1} = x_{t+1}$.
- **Termination:** $t = N \Rightarrow$ episode terminates

The RL agent was trained using the hyperparameters summarized in Table 3.

1) DEEP Q-NETWORK ARCHITECTURE

The Q-function was approximated using a fully connected neural network $Q(s, a; \theta)$, where the input is the state vector $s \in \mathbb{R}^d$ and the output is a vector of Q-values for each possible action $a \in \{0, 1\}$. The network architecture is defined as:

$$\begin{aligned} Q(s, a; \theta) &= \text{MLP}(s) \\ &= f_3 \left(W_3 \cdot f_2 \left(W_2 \cdot f_1(W_1 \cdot s + b_1) + b_2 \right) + b_3 \right) \end{aligned} \quad (2)$$

where:

- $W_1 \in \mathbb{R}^{64 \times d}$, $W_2 \in \mathbb{R}^{64 \times 64}$, $W_3 \in \mathbb{R}^{2 \times 64}$
- f_1, f_2 are ReLU activation functions
- f_3 is the identity function (no activation)

Two identical networks are maintained:

- **Policy network:** $Q_{\text{policy}}(s, a; \theta)$ for action selection
- **Target network:** $Q_{\text{target}}(s, a; \theta^-)$ used for stable target value estimation, where $\theta^- \leftarrow \theta$

Algorithm 1 Training DQN-Based IDS

Input: Dataset \mathcal{D} , action space \mathcal{A} , learning rate α , discount γ , batch size B , replay memory size N , initial ϵ , minimum ϵ_{\min} , decay κ , episodes E , update frequency T_{update} .

Output: Trained Q-network Q_θ

Initialize Q-network Q_θ and target network $Q_{\theta^-} \leftarrow Q_\theta$;

Initialize replay buffer $\mathcal{M} \leftarrow \emptyset$;

for each episode $e \in \{1, \dots, E\}$ **do**

for each step t in episode **do**

Observe state s_t (flow features);

Select action a_t using ϵ -greedy policy;

Execute a_t , observe reward r_t (Table 2), next state s_{t+1} ;

Store transition (s_t, a_t, r_t, s_{t+1}) in \mathcal{M} ;

Sample mini-batch from \mathcal{M} ;

Compute target $y_i = r_i + \gamma \max_{a'} Q_{\theta^-}(s'_i, a')$;

Update Q-network by minimizing MSE loss;

Update $\epsilon \leftarrow \max(\epsilon \cdot \kappa, \epsilon_{\min})$;

if $t \bmod T_{\text{update}} = 0$ **then**

| Update target network $\theta^- \leftarrow \theta$;

end

end

return Q_θ

2) DQN AGENT TRAINING

The Deep Q-Network agent was trained over E_{\max} episodes using ϵ -greedy exploration. At each time step t in episode e , the agent selects an action a_t based on:

$$a_t = \begin{cases} \text{random}(a) & \text{with probability } \varepsilon \\ \arg \max_a Q(s_t, a; \theta) & \text{with probability } 1 - \varepsilon \end{cases}$$

The environment transitions to the next state s_{t+1} , and the tuple $(s_t, a_t, r_t, s_{t+1}, d_t)$ is stored in the replay buffer \mathcal{D} . When $|\mathcal{D}| \geq B$, a minibatch $\mathcal{B} \subset \mathcal{D}$ of size B is sampled, and the network is trained using the temporal difference (TD) loss:

$$\begin{aligned} y_t &= \begin{cases} r_t & \text{if } d_t = \text{True} \\ r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-) & \text{otherwise} \end{cases} \\ \mathcal{L}(\theta) &= \frac{1}{B} \sum_{(s_t, a_t, r_t, s_{t+1}, d_t) \in \mathcal{B}} (y_t - Q(s_t, a_t; \theta))^2 \end{aligned}$$

The target network parameters θ^- are updated with the policy network parameters θ at the end of each episode:

$$\theta^- \leftarrow \theta$$

The exploration rate decays multiplicatively:

$$\varepsilon \leftarrow \max(\varepsilon \cdot \varepsilon_{\text{decay}}, \varepsilon_{\min})$$

3) EVALUATION STRATEGY

The trained DQN model is evaluated on a held-out test set. Let \hat{y}_i denote the predicted label for the i -th test sample, and y_i denote the true label. The overall accuracy is computed as:

$$\text{Accuracy} = \frac{1}{N} \sum_{i=1}^N \mathbb{I}(\hat{y}_i = y_i)$$

where N is the total number of test samples, and $\mathbb{I}(\cdot)$ is the indicator function that returns 1 when the condition is true and 0 otherwise.

To further assess model performance, a confusion matrix $C \in \mathbb{N}^{K \times K}$ is computed, where K is the number of classes. The entry C_{ij} indicates the number of times a sample from class i was predicted as class j .

Figure 5 illustrates the DQN loss curve observed during training, demonstrating how the agent's prediction errors decrease over successive episodes. Following training, the

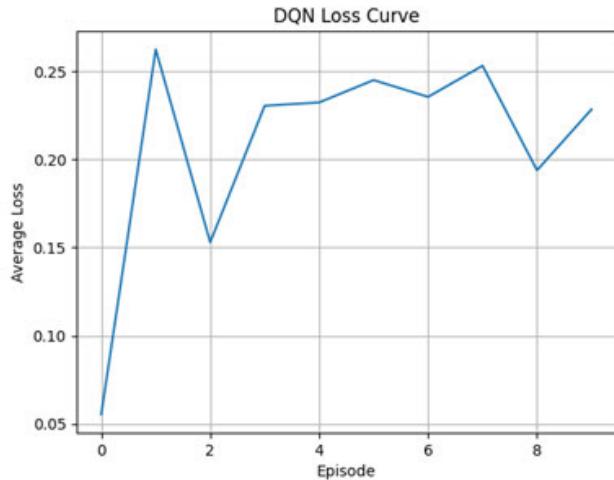


FIGURE 5. Training loss curve for Experiment 1 (Random split).

model was evaluated on the test set. Due to inherent randomness in the training process, such as weight initialization, data shuffling, and action selection during exploration, the results may vary slightly with each run. The best-performing model achieved an accuracy of **97.6%**. As shown in Figure 6, the model accurately distinguishes between known attack (S) (benign) and ransomware traffic flows (A and SS).

The experiment 1 results are summarized in Table 4:

TABLE 4. Classification report for Experiment 1.

Class	Precision	Recall	F1-Score	Support
Benign	1.00	0.95	0.97	19914
Ransomware	0.96	1.00	0.98	24799
Accuracy		0.98		
Macro Avg	0.98	0.97	0.98	44713
Weighted Avg	0.98	0.98	0.98	44713

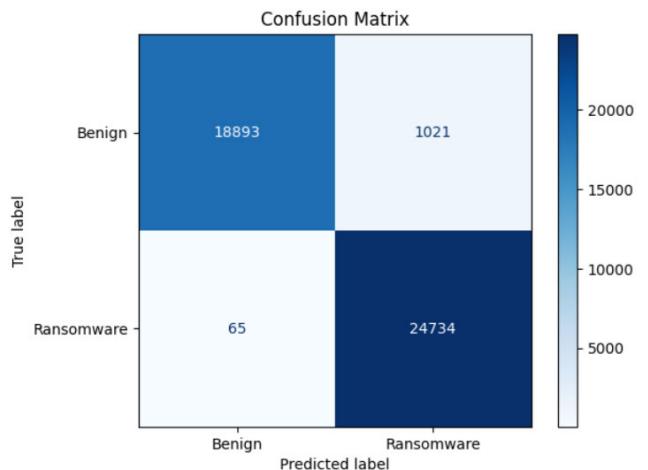


FIGURE 6. Confusion matrix for Experiment 1 (Random split).

The model achieved 97.6% accuracy, with high precision and recall for both benign and ransomware classes. It correctly detected nearly all ransomware instances (recall = 1.00) while maintaining very few false positives. The results indicate strong, balanced performance suitable for reliable ransomware detection.

B. EXPERIMENT 2: BINARY CLASSIFICATION (ZERO-DAY FAMILY SPLIT SIMULATION)

In this experiment we improved the model by simulating a realistic zero-day detection scenario. Instead of random splitting, the dataset train/test split using an 70/30 ratio based on ransomware family (unseen families). In other words, strict isolation of ransomware families. This tested the model's ability to detect attacks from new, previously unseen types. Let \mathcal{D} denote the set of all ransomware families in the dataset. We partitioned \mathcal{D} into disjoint subsets:

$$\begin{aligned}\mathcal{F}_{\text{train}} \cup \mathcal{F}_{\text{test}} &= \mathcal{D} \\ \mathcal{F}_{\text{train}} \cap \mathcal{F}_{\text{test}} &= \emptyset\end{aligned}$$

Samples were then split such that:

$$\begin{aligned}\mathcal{F}_{\text{train}} &= \{(x_i, y_i) \mid f_i \in \mathcal{F}_{\text{train}}\} \\ \mathcal{F}_{\text{test}} &= \{(x_i, y_i) \mid f_i \in \mathcal{F}_{\text{test}}\}\end{aligned}$$

This ensures that all ransomware families in the test set were completely unseen during training. In this experiment the DQN agent was trained using the same architecture, hyperparameters, and training procedure as in Experiment 1. Despite having no prior exposure to the test ransomware families, the model successfully generalized to the unseen patterns. The zero-day experiment yielded a detection accuracy of **95.9%**, confirming the model's robustness in identifying novel threats, as illustrated in Figure 7 and Table 7.

To further assess the model's learning behavior in this setting, the training loss curve is shown in Figure 8.

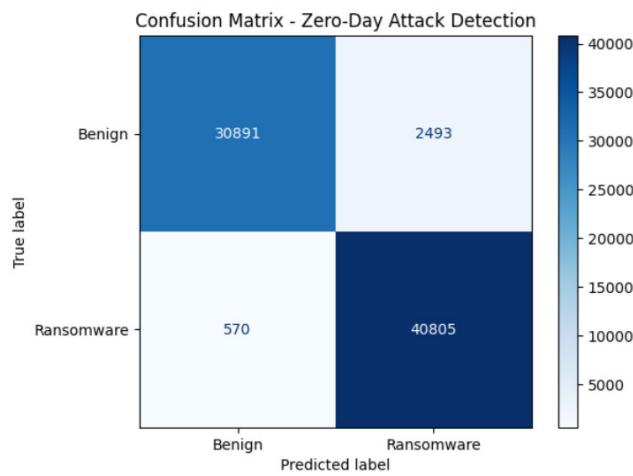


FIGURE 7. Confusion matrix for Experiment 2 (Zero-Day Ransomware families).

TABLE 5. Classification report for Experiment 2.

Class	Precision	Recall	F1-Score	Support
Benign	0.98	0.93	0.95	33384
Ransomware	0.94	0.99	0.96	41375
Accuracy	0.96			
Macro Avg	0.96	0.96	0.96	74759
Weighted Avg	0.96	0.96	0.96	74759

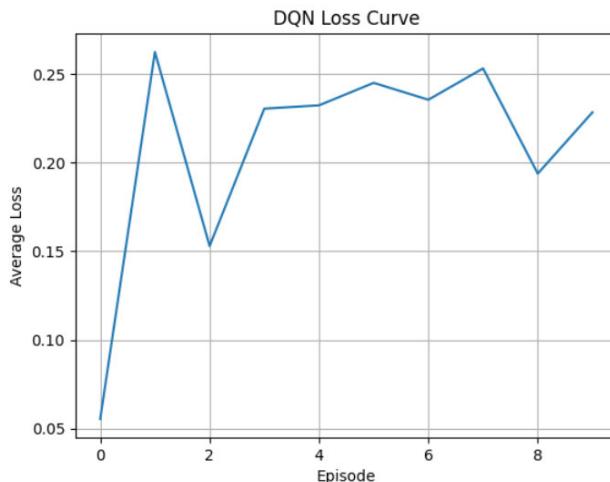


FIGURE 8. Training loss curve for Experiment 2 (Zero-Day Ransomware families).

C. EXPERIMENT 3: MULTI-CLASS (RANDOM SPLIT)

This experiment investigate the feasibility of using RL-DQN for classifying ransomware behavior into multiple categories. The DQN model is implemented as a fully connected feedforward neural network, with the output layer consisting of three neurons—each corresponding to one of the target classes. We applied the same training setup in the previous

experiments. The dataset was randomly split into training (70%) and testing (30%) subsets using stratified sampling to maintain class balance. The results was summarized in each of Figure 9, Figure 10 and Table.

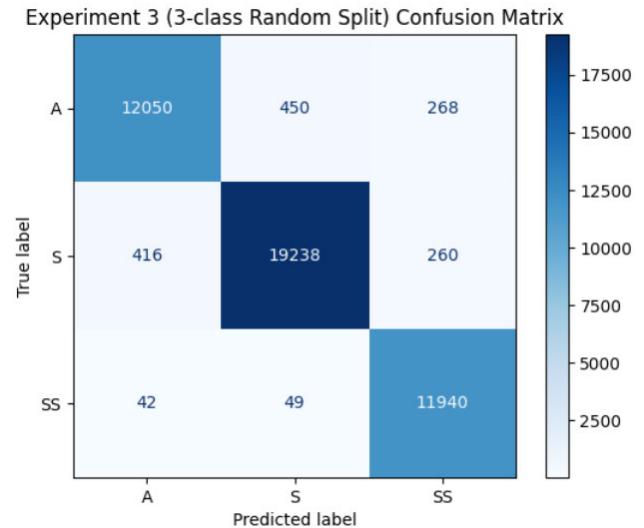


FIGURE 9. Confusion matrix for Experiment 3 (Multiclass) random split.

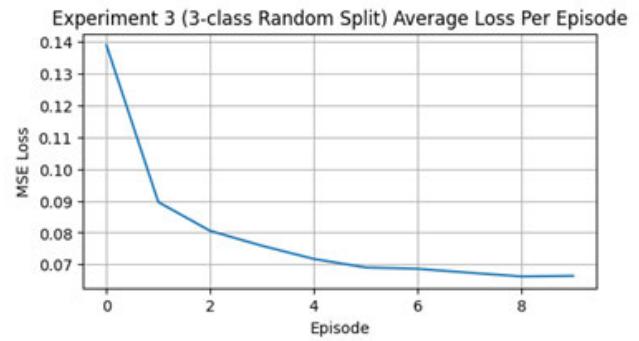


FIGURE 10. Training loss curve for Experiment 3 Multiclass (Random split).

D. EXPERIMENT 4: MULTI-CLASS (ZERO-DAY FAMILY SPLIT)

This experiment was conducted using the same configuration as the previous one, except that the data was split according to the zero-day family strategy, as in Experiment 2. Additionally, the output layer of the model was modified to contain three neurons, corresponding to the three class labels in the dataset. The results are summarized in Figure 11, Figure 12, and Table 7.

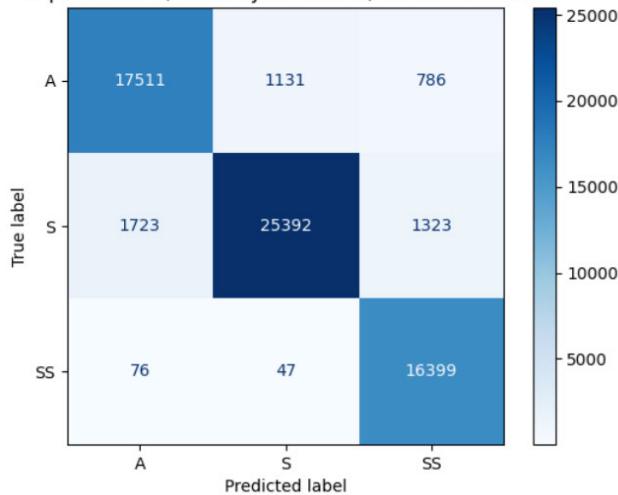
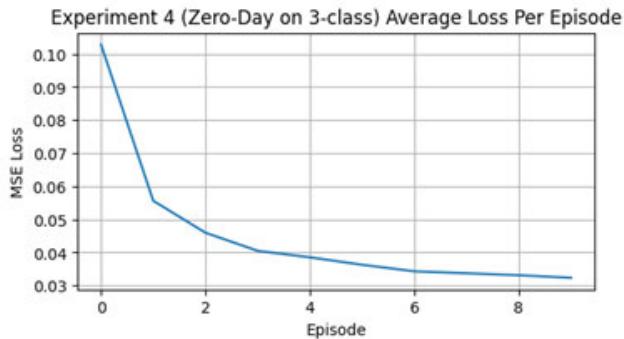
E. COMPARATIVE ANALYSIS OF EXPERIMENTS

Table 8 provides a summarized comparison of the four experiments conducted to evaluate the performance of a RL-DQN model for ransomware detection under varying classification and data-splitting scenarios. Across all experiments, the

TABLE 6. Classification report for multiclass prediction (Random split).

Class	Precision	Recall	F1-Score	Support
A	0.96	0.94	0.95	12768
S	0.97	0.97	0.97	19914
SS	0.96	0.99	0.97	12031
Accuracy	0.97			
Macro Avg	0.97	0.97	0.97	44713
Weighted Avg	0.97	0.97	0.97	44713

Experiment 4 (Zero-Day on 3-class) Confusion Matrix

**FIGURE 11.** Confusion matrix for Experiment 4 (Multiclass) Zero-day family split.**FIGURE 12.** Training loss curve for Experiment 4 Multiclass (Zero-day family split).

model demonstrates consistently high accuracy, confirming the effectiveness of reinforcement learning for this task.

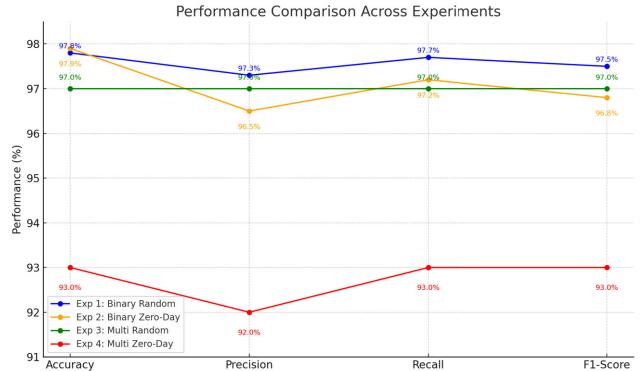
In binary classification settings (Experiments 1 and 2), the model performs exceptionally well, achieving accuracies of 97.6% and 95.9% respectively. Notably, in Experiment 2, where the agent was tested on entirely unseen ransomware families (zero-day simulation), the model maintained nearly identical performance to the randomly split setting. This

TABLE 7. Classification report for multiclass prediction (Zero-Day family split).

Class	Precision	Recall	F1-Score	Support
A	0.91	0.90	0.90	19428
S	0.96	0.89	0.92	28438
SS	0.89	0.99	0.94	16522
Accuracy	0.92			
Macro Avg	0.92	0.93	0.92	64388
Weighted Avg	0.92	0.92	0.92	64388

TABLE 8. Comparison of Ransomware detection experiments.

Experiment	Accuracy	Best Recall	F1-Score Avg
Binary class (Random Split)	97.6%	Ransomware (1.00)	0.98
Binary class (Zero-Day Family Split)	95.9%	Ransomware (0.99)	0.96
Multiclass (Random Split)	97.0%	SS (0.99)	0.97
Multiclass (Zero-Day Family Split)	92.0%	SS (0.99)	0.94

**FIGURE 13.** Performance comparison: Random split vs. Zero-Day.**TABLE 9.** Comparison of our work with Nhlapo et al. [20].

Aspect	Our work on RL-DQN	Nhlapo [20]
Classification Type	Multiclass	Multiclass
Technique	RL-DQN	RFC, XGBoost, NB, SVM
Evaluation Setting	Random + Zero-Day	Random
Accuracy	97.6% (random), 95.9% (zero-day)	Up to 100%
Adaptability	Learns dynamically from the environment	Fixed classifier trained on full data

suggests strong generalization capabilities and the potential for robust deployment in real-world scenarios where novel threats continuously emerge.

TABLE 10. Summary of challenges across different methods.

Challenge	ML-Based (Supervised)	ML-Based (Unsupervised)	ML-Based (Reinforcement)	DL-Based Methods	Anomaly-Based Methods
Computation & Resources	<ul style="list-style-type: none"> Poor real-world performance due to processor limitations Resource-intensive training 	Requires GPU hardware for inference; challenging for router-like devices	<ul style="list-style-type: none"> High computation and network costs Implementation challenges due to ignored processing requirements 	<ul style="list-style-type: none"> Long training time Huge computational needs Unsuitable for on-device detection 	<ul style="list-style-type: none"> Time-consuming with large data High computational resource usage
Data Quality & Processing	Performance affected by data quality, class imbalance, noise, and outliers	Limited attack detection due to insufficient experimental data	Risk of manual labeling errors	Limited real-world effectiveness due to insufficient data	Feature loss during processing impacts detection accuracy
Model Training		Parameter adjustment challenges across multiple corpora		<ul style="list-style-type: none"> Generalization issues Feature selection challenges Resource-intensive retraining 	Requires large normal data samples for reliable training
Accuracy & Detection	<ul style="list-style-type: none"> Poor novel threat detection Weakened by obfuscation Persistent false alarms 			<ul style="list-style-type: none"> Specific attack identification issues High false alarm rates Minority class challenges 	Decision boundary classification issues
Adaptability & Flexibility	Poor cross-dataset performance	Limited to specific anomaly patterns		<ul style="list-style-type: none"> Poor threat evolution adaptation Limited by pre-existing corpora Unreliable with input variations 	Lacks scalability
Real-Time & Network Adaptability	Sensitive to network changes			Poor performance in dynamic environments due to batch training	

Multiclass experiments (Experiments 3 and 4) tested the model's ability to distinguish between multiple ransomware behaviors. While accuracy slightly decreased to 97.0% in the random-split setting (Exp 3), and further to 92.0% in the zero-day family split setting (Exp 4), the model still retained strong classification performance. The slight performance drop in Experiment 4 reflects the increased difficulty of detecting and differentiating between unseen ransomware types across multiple categories. However, the RL-DQN model still achieved an F1-Score average of 0.94 and a near-perfect recall of 0.99 for class SS, indicating its sensitivity to certain threat types.

Figure 13 presents a side-by-side performance comparison across the four conducted experiments. The results show that the highest metrics are achieved in the binary classification experiments (Exp 1 and Exp 2), with accuracy, recall, and F1-scores consistently above 96.5%. Notably, even in the challenging zero-day setting (Exp 2), the DQN model retains strong performance, with only minor drops in precision and F1-score compared to the random split. Multiclass experiments (Exp 3 and Exp 4) show a natural decline in performance, especially under zero-day conditions (Exp 4), where the model faces unseen ransomware families with more granular class distinctions. However, the model still achieves 93% accuracy in Exp 4, which is a strong result given the complexity of the task. Overall, this figure highlights the robustness of our RL-based framework, particularly its generalization capacity in both binary and multiclass zero-day detection scenarios.

F. COMPARISON AND DISCUSSION

To evaluate the effectiveness of our reinforcement learning-based detection approach, we compare it with the recent study by Nhlapo and Nkongolo [20], which also employs the UGRansome dataset for ransomware detection. While

both studies utilize the same dataset, the methodologies and evaluation settings differ substantially. Our work explores both binary and multiclass classification scenarios, incorporating realistic zero-day family splits, while their study is limited to traditional supervised learning on known threats.

Specifically, our experiments include four settings: binary classification with random and zero-day splits (Experiments 1 and 2), and multiclass classification with similar split types (Experiments 3 and 4). In the most realistic setting—Experiment 2 (Binary Zero-Day)—our RL-DQN model achieved an accuracy of 97.6%, demonstrating strong generalization to unseen ransomware families. Even under the more complex multiclass zero-day scenario (Experiment 4), the model maintained 92.0% accuracy, with a high recall of 0.99 for class SS. These results reflect the RL-DQN agent's ability to adapt to previously unseen behaviors, learning to distinguish ransomware patterns through interaction and reward feedback. In contrast, Nhlapo and Nkongolo [20] apply ensemble classifiers such as RFC, XGBoost, Naïve Bayes, and SVM, focusing primarily on supervised training with known data distributions. Although their ensemble models report 100% accuracy, these results are achieved under traditional settings without explicit zero-day validation. Moreover, they adopt a fixed multiclass labeling strategy without considering task-specific grouping or model robustness to novel families. Table 8 illustrates how our agent performs across different configurations and under more rigorous evaluation conditions. Our work emphasizes model robustness, realistic threat simulation, and generalization, providing a more comprehensive assessment of detection capabilities than static supervised classifiers. In summary, while Nhlapo and Nkongolo [20] demonstrate high performance under conventional settings, our reinforcement learning framework excels in zero-day detection

TABLE 11. Summary of related work.

Reference	Year	Approach	Algorithms/Methods	Dataset(s)	Key Contributions/Findings
Khan et al. [18]	2024	Supervised	SMOTE-Tomek, DT, RF, XGBoost, KNN, NB, LR, AdaBoost	CICIDS-2017	Balanced dataset improved DT and AdaBoost accuracy; binary classification performed better
Srivastav et al. [7]	2023	Supervised	DT, RF, KNN, LR	Kaggle dataset	RF achieved highest accuracy (0.9865), outperforming others in speed and precision
Amor et al. [19]	2004	Supervised	NB, DT	KDD'99	NB performed comparably to DT at different attack classification levels
Nhlapo et al. [20]	2024	Supervised	RF, XGBoost, Adaboost, SVM, NB, DT, MLP, Ensemble	UGRansome	RF, XGBoost, and Ensemble reached 100% in accuracy metrics
Kadri et al. [8]	2025	Supervised	Extra Trees, GB, DT, DNN, PCA-based preprocessing	Military network dataset	DNN achieved highest accuracy (0.995), effective for real-time detection
Tun et al. [21]	2023	Supervised	NB with discretization	NSL-KDD, DARPA	Discretization significantly improved NB accuracy (0.9983)
Alexis [22]	2025	Supervised	AutoML (PyCaret), Gradient Boosting	Modbus TCP/IP	Identified best models for zero-day exploits in ICS networks
Nagaraja et al. [23]	2021	Supervised	Regression with dimensionality reduction	KDD, NSL-KDD	Regression analysis performed better on KDD dataset
Wiswananichthan et al. [24]	2021	Supervised	DLHA with NB and SVM	NSL-KDD	Achieved high detection for rare attacks (R2L, U2R)
Saurabh et al. [25]	2025	Semi-supervised	HMS-IDS with supervised learning and K-Means	CIC-ToN-IoT, KDD-99, NSL-KDD, CICIDS2017	Achieved high accuracy for known (0.9949) and unknown (0.9893) threats
Sarwar et al. [26]	2025	Semi-supervised	CNN+DNN, Autoencoder+DNN, DNN+XGBoost	CSC IDS 2018	Autoencoder+DNN and CNN+DNN excelled with FMSMOTE
Kim et al. [27]	2023	Semi-supervised	AE-OCC	Meraz18, Drebin	AE-OCC achieved 0.971 accuracy, eliminating threshold dependency
Kim et al. [28]	2022	Semi-supervised	GAN (WGAN-GP) + LSTM	ISCX2012, CIC2017, CSE2018	Enabled early, real-time intrusion detection at packet-level
Carrera et al. [29]	2022	Semi-supervised	Deep Learning + Shallow Learning with SHAP	KDD99, NSL-KDD, CIC-IDS2017	Combined high prediction speed with high accuracy models
Salem et al. [9]	2022	Semi-supervised	TIDM	Real-time network flows	Designed for real-time IDS, updating continuously
Kumar et al. [30]	2025	Unsupervised	Heavy-Hitters, Graph techniques	CICIDS18, real-time logs	Detected unknown attacks effectively in binary/multi-class classification
Afek et al. [31]	2019	Unsupervised	Double/Triple Heavy Hitters	Real-life attack captures	One-pass, linear-time algorithm for attack signature extraction
Li et al. [1]	2021	Unsupervised	SOM with CNN-based features	MVTec AD	Achieved high anomaly detection with 0.978 AUROC
Singh et al. [32]	2021	Unsupervised	One-Class SVM, Active Learning, K-Means, DBSCAN	CIC-IDS2017, UNSW-NB15, KDD Cup 99	Efficiently detected zero-day attacks with high accuracy
Zhang et al. [33]	2023	Unsupervised	Autoencoder + Hierarchical Clustering	Cloud-based traffic	High accuracy for worms and system vulnerabilities
Shukla et al. [34]	2023	Unsupervised	IF with RF-based feature selection	UNSW-NB15 Industry 4.0	Lightweight IDS suitable for real-time industrial environments
Borgioli et al. [35]	2024	Unsupervised	1D CNN AE	EDGE-IIOTSET	Achieved 0.99 accuracy with real-time deployment capability
de Araujo-Filho et al. [36]	2023	Unsupervised	GAN-based IDS using TCN	CICDDoS2019	Outperformed existing GAN-based IDS models
Oluwadare et al. [2]	2024	Unsupervised (Review)	Survey of self-learning algorithms	—	In-depth analysis of self-learning methods for IDS
Nkongolo et al. [37]	2021	Dataset	—	UGRansome	Modern dataset capturing zero-day threats

and multiclass generalization—offering a more adaptable and resilient approach for modern cybersecurity challenges. Table 9 outlines the main differences.

Despite accuracy and generalization, it is important to mention that an important aspect of deploying our suggested RL-DQN approach in practical intrusion detection systems is computational efficiency. Reinforcement learning models typically require higher training costs compared to traditional supervised classifiers; however, once trained, inference can be performed with low latency, making them suitable for real-time detection. The implemented experiments show that

the trained RL-DQN agent classifies traffic samples within milliseconds, which is acceptable for online IDS pipelines.

The interpretability of DQN-driven alerts is another aspect that is more critical for real-world deployment. Security analysts may find it difficult to comprehend the reasoning behind a specific detection decision because reinforcement learning models are frequently thought of as “black boxes.” This can be addressed by highlighting the traffic features that had the biggest impact on the model’s action using post-hoc interpretability techniques (such as feature importance analysis, LIME, or SHAP).

VI. CONCLUSION AND FUTURE WORK

This study presented a self-learning intrusion detection system powered by Deep Q-Networks (DQNs), designed to effectively identify both known and zero-day ransomware attacks. By formulating intrusion detection as a reinforcement learning task, the system learns optimal classification policies through continuous interaction with network traffic data, without the need for predefined signatures or labeled training samples. Comprehensive experiments were conducted across binary and multiclass classification settings, evaluated under both random splits and challenging zero-day family splits—where ransomware variants in the test set were completely unseen during training. Results demonstrate that the DQN-based IDS achieves strong and consistent performance, with up to 97.6% accuracy in the random split and 95.9% accuracy under zero-day conditions in the binary classification setting. Furthermore, in the multiclass scenario, the model maintains 97.0% and 92.0% accuracy in the random and zero-day evaluations, respectively. The system also preserved high recall for ransomware classes, highlighting its potential to detect novel threats effectively. These findings underscore the promise of reinforcement learning as a foundation for next-generation IDS, capable of evolving in real time to combat emerging cyber threats. As future work, we plan to extend the action space to incorporate open-set recognition, explore alternative RL algorithms such as actor-critic or PPO, and deploy the system in real-time network environments to assess its responsiveness and scalability under live traffic conditions.

ETHICAL APPROVAL

This study did not involve human participants or animals, and therefore, ethical approval is not applicable.

COMPETING INTERESTS

The authors declare that they have no competing interests.

AUTHORS' CONTRIBUTIONS

- **Mouhammd AlKasassbeh:** Conceptualization, Methodology, Formal Analysis, Supervision.
- **Ebtahal H. Omoush:** Conceptualization, Methodology, Surveying, Writing - Original Draft.
- **MOHAMMAD ALMSEIDIN:** Methodology, Surveying.
- **Amjad Aldweesh:** Conceptualization, Methodology and Surveying.

AVAILABILITY OF DATA AND MATERIALS

There are not datasets used in this work.

REFERENCES

- [1] N. Li, K. Jiang, Z. Ma, X. Wei, X. Hong, and Y. Gong, “Anomaly detection via self-organizing map,” in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2021, pp. 974–978.
- [2] S. Oluwadare, Z. ElSayed, and O. Adekoya, “A brief review of unsupervised learning algorithms for zero-day attacks in intrusion detection systems,” in *Proc. IEEE 3rd Int. Conf. Comput. Mach. Intell. (ICMI)*, Apr. 2024, pp. 1–6.
- [3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 1st ed., Cambridge, MA, USA: MIT Press, 1998.
- [4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [5] M. Alauthman, N. Aslam, M. Al-Kasassbeh, S. Khan, A. Al-Qerem, and K.-K. R. Choo, “An efficient reinforcement learning-based botnet detection approach,” *J. Netw. Comput. Appl.*, vol. 150, Jan. 2020, Art. no. 102479.
- [6] A. Armijos and E. Cuenca, “Zero-day attacks: Review of the methods used based on intrusion detection and prevention systems,” in *Proc. IEEE Colombian Caribbean Conf. (C3)*, Nov. 2023, pp. 1–6.
- [7] S. Srivastav, K. Guleria, and S. Sharma, “Machine learning based predictive model for intrusion detection,” in *Proc. Int. Conf. Signal Process., Comput., Electron., Power Telecommun. (IConCEPT)*, May 2023, pp. 1–5.
- [8] R. Vadisetty and A. Polamarasetti, “Enhancing intrusion detection systems with deep learning and machine learning algorithms for real-time threat classification,” in *Proc. Asian Conf. Intell. Technol. (ACOIT)*, Sep. 2024, pp. 1–6.
- [9] M. Salem and A.-K. Al-Tamimi, “A novel threat intelligence detection model using neural networks,” *IEEE Access*, vol. 10, pp. 131229–131245, 2022.
- [10] L. Y. Por, Z. Dai, S. J. Leem, Y. Chen, J. Yang, F. Binbeshr, K. Y. Phan, and C. S. Ku, “A systematic literature review on AI-based methods and challenges in detecting zero-day attacks,” *IEEE Access*, vol. 12, pp. 144150–144163, 2024.
- [11] R. Ahmad, I. Alsmadi, W. Alhamdani, and L. Tawalbeh, “Zero-day attack detection: A systematic literature review,” *Artif. Intell. Rev.*, vol. 56, no. 10, pp. 10733–10811, Feb. 2023.
- [12] Y. Guo, “A review of machine learning-based zero-day attack detection: Challenges and future directions,” *Comput. Commun.*, vol. 198, pp. 175–185, Jan. 2023.
- [13] R. Duraz, D. Espes, J. Francq, and S. Vaton, “SECL: A zero-day attack detector and classifier based on contrastive learning and strong regularization,” in *Proc. 19th Int. Conf. Availability, Rel. Secur.*, New York, NY, USA, Jul. 2024, pp. 1–12.
- [14] A. A. Hammad, S. R. Ahmed, M. K. Abdul-Hussein, M. R. Ahmed, D. A. Majeed, and S. Algburi, “Deep reinforcement learning for adaptive cyber defense in network security,” in *Proc. Cognit. Models Artif. Intell. Conf.*, New York, NY, USA, May 2024, pp. 292–297.
- [15] J. Note, E. Mullalli, and B. Cico, “Machine learning algorithms for cyber attack detection and classification,” in *Proc. Int. Conf. Comput. Syst. Technol.*, New York, NY, USA, Jun. 2024, pp. 29–36.
- [16] S. Hore, J. Ghadermazi, D. Paudel, A. Shah, T. Das, and N. Bastian, “Deep PackGen: A deep reinforcement learning framework for adversarial network packet generation,” *ACM Trans. Privacy Secur.*, vol. 28, no. 2, pp. 1–33, Feb. 2025.
- [17] N. Sanghi, *Reinforcement Q-Learning Tutorial*. New York, NY, USA: Apress, 2021.
- [18] F. A. Khan, A. A. Shah, N. Alshammry, S. Saif, W. Khan, M. O. Malik, and Z. Ullah, “Balanced multi-class network intrusion detection using machine learning,” *IEEE Access*, vol. 12, pp. 178222–178236, 2024.
- [19] N. B. Amor, S. Benferhat, and Z. Elouedi, “Naive Bayes vs decision trees in intrusion detection systems,” in *Proc. ACM Symp. Appl. Comput.*, New York, NY, USA, Mar. 2004, pp. 420–424.
- [20] S. J. Nhlapo and M. N. W. Nkongolo, “Zero-day attack and ransomware detection,” 2024, *arXiv:2408.05244*.
- [21] T. Tun, K. K. Wai, and M. S. Khaing, “Performance of machine learning using preprocessing and classification for intrusion detection system,” in *Proc. IEEE Conf. Comput. Appl. (ICCA)*, Feb. 2023, pp. 260–265.
- [22] M. H. Alexis, “Machine learning for detecting modbus zero-day exploits,” Ph.D. thesis, George Washington Univ., 2025.
- [23] A. Nagaraja, U. Boregowda, and V. Radhakrishna, “Regression analysis for network intrusion detection,” in *Proc. Int. Conf. Data Sci.*, New York, NY, USA, 2021, pp. 173–179.
- [24] T. Wisanwanichthan and M. Thammawichai, “A double-layered hybrid approach for network intrusion detection system using combined naive Bayes and SVM,” *IEEE Access*, vol. 9, pp. 138432–138450, 2021.

- [25] K. Saurabh, V. Sharma, U. Singh, R. Khondoker, R. Vyas, and O. P. Vyas, "HMS-IDS: Threat intelligence integration for zero-day exploits and advanced persistent threats in IIoT," *Arabian J. Sci. Eng.*, vol. 50, no. 2, pp. 1307–1327, Jan. 2025.
- [26] M. N. Sarwar, M. S. Arman, T. Bhuiyan, and F. B. Rafiq, "Optimizing intrusion detection with hybrid deep learning models and data balancing techniques," in *Proc. IEEE 4th Int. Conf. AI Cybersecurity (ICAIC)*, Feb. 2025, pp. 1–6.
- [27] C. Kim, S.-Y. Chang, J. Kim, D. Lee, and J. Kim, "Automated, reliable zero-day malware detection based on autoencoding architecture," *IEEE Trans. Netw. Service Manage.*, vol. 20, no. 3, pp. 3900–3914, Sep. 2023.
- [28] T. Kim and W. Pak, "Early detection of network intrusions using a GAN-based one-class classifier," *IEEE Access*, vol. 10, pp. 119357–119367, 2022.
- [29] F. Carrera, V. Dentamaro, S. Galantucci, A. Iannaccone, D. Impedovo, and G. Pirlo, "Combining unsupervised approaches for near real-time network traffic anomaly detection," *Appl. Sci.*, vol. 12, no. 3, p. 1759, Feb. 2022.
- [30] V. Kumar and D. Sinha, "A robust intelligent zero-day cyber-attack detection technique," *Complex Intell. Syst.*, vol. 7, no. 5, pp. 2211–2234, Oct. 2021.
- [31] Y. Afek, A. Bremler-Barr, and S. L. Feibish, "Zero-day signature extraction for high-volume attacks," *IEEE/ACM Trans. Netw.*, vol. 27, no. 2, pp. 691–706, Apr. 2019.
- [32] R. Singh and G. Srivastav, "Novel framework for anomaly detection using machine learning technique on CIC-IDS2017 dataset," in *Proc. Int. Conf. Technol. Advancements Innov. (ICTAI)*, Nov. 2021, pp. 632–636.
- [33] Y. Zhang, K. Wen, and X. Wang, "An unsupervised network anomaly detection model and implementation," in *Proc. 3rd Int. Conf. Artif. Intell., Autom. Algorithms*, New York, NY, USA, Jul. 2023, pp. 120–125.
- [34] A. K. Shukla, S. Srivastav, S. Kumar, and P. K. Muhuri, "UInDeSI4.0: An efficient unsupervised intrusion detection system for network traffic flow in Industry 4.0 ecosystem," *Eng. Appl. Artif. Intell.*, vol. 120, Apr. 2023, Art. no. 105848.
- [35] N. Borgioli, F. Aromolo, L. T. X. Phan, and G. Buttazzo, "A convolutional autoencoder architecture for robust network intrusion detection in embedded systems," *J. Syst. Archit.*, vol. 156, Nov. 2024, Art. no. 103283.
- [36] P. F. De Araujo-Filho, M. Naili, G. Kaddoum, E. T. Fapi, and Z. Zhu, "Unsupervised GAN-based intrusion detection system using temporal convolutional networks and self-attention," *IEEE Trans. Netw. Service Manage.*, vol. 20, no. 4, pp. 4951–4963, Dec. 2023.
- [37] M. Nkongolo, J. P. van Deventer, and S. M. Kasongo, "UGRansom1819: A novel dataset for anomaly detection and zero-day threats," *Information*, vol. 12, no. 10, p. 405, Sep. 2021.



MOUHAMMD ALKASASSBEH received the degree from the School of Computing, Portsmouth University, U.K., in 2008. He holds a Full Professorship with the Computer Science Department, Princess Sumaya University for Technology. His research interests include network traffic analysis, network fault detection, network fault and anomaly classification, and the application of machine learning within the realm of computer networking and network security.

EBTEHAL H. OMOUSH is currently a Full Professor of computer science.



MOHAMMAD ALMSEIDIN received the Ph.D. degree from the Department of Information Technology, University of Miskolc, in 2020. He is currently an Assistant Professor with the Department of Computer Science, Tafile Technical University. His research interests include fuzzy systems, network traffic analysis, and machine learning. His current focus is on fuzzy rule interpolation and network security.



AMJAD ALDWEESH received the bachelor's degree in computer science, the M.Sc. degree (Hons.) in advanced computer science and security from The University of Manchester, and the Ph.D. degree in blockchain and smart contracts technology from Newcastle University.

He is a Computer Associate Professor interested in the blockchain and smart contracts technology and cybersecurity.

• • •