Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 272 (2025) 202–209

Procedia Computer Science

www.elsevier.com/locate/procedia

The 16th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2025)
October 28-30, 2025, Istanbul, Türkiye

NEMESIS: An Enhanced Hybrid Intrusion Detection System Leveraging Deep Q-Learning and Random Forest

Ahmed BENIDIR[a,*], Hakima OULD-SLIMANE[a], Nadjia KARA[b]

[a]*Dept. of Math. & Comp. Sci., Univ. of Quebec at Trois-Rivières, 3351 Blvd des Forges, Trois-Rivières, QC G8Z 4M3, Canada*
[b]*Dept. of Software Eng. & IT, ÉTS (École de technologie supérieure), 1100 Notre-Dame W., Montreal, QC H3C 1K3, Canada*

Abstract

Network intrusion detection systems (NIDS) face the growing difficulty posed by increasingly sophisticated and unseen attacks, which represent a dangerous threat due to their ability to exploit vulnerabilities that have not yet been identified. These attacks are inherently difficult to detect with conventional NIDS because such systems typically are built on known threat patterns or signatures, which are absent in unseen scenarios. Consequently, this greatly limits their efficacy in mitigating advanced threats, making networks susceptible to potential security breaches. To address these challenges, recent years have witnessed the emergence of various Reinforcement Learning (RL) approaches aimed at enhancing the automatic detection of network intrusions. These systems are equipped with autonomous agents that acquire the ability to learn independently and make decisions without requiring direct input or knowledge of human experts. In this paper, we propose a network intrusion detection mechanism that integrates a Deep Q Network-based model (DQN) with a supervised machine learning algorithm specifically designed for attack classification. Our model is characterized by meticulous fine-tuning of hyperparameters to optimize the performance of detection. Extensive experimental evaluations that take advantage of the NSL-KDD and CSE-CICIDS2017 datasets demonstrate that our hybrid approach significantly improves detection accuracy across various types of attack and outperforms other existing state-of-the-art solutions designed for similar purposes.

Keywords: Network Intrusion Detection; Reinforcement Learning; Supervised learning; CSE-CICIDS2017; NSL-KDD;

1. Introduction

Network intrusion detection systems (IDS) are a core technology in cyber defense Kilincer et al. [11], Venter and Eloff [20], classifying network traffic as benign or malicious. Signature-based IDS (SIDS) efficiently detects known

* Corresponding author. Tel.: +1-514-216-1109.
E-mail address: ahmed.benidir@uqtr.ca

1877-0509
10.1016/j.procs.2025.10.197

attacks with low false positives but fails against novel or polymorphic threats. Anomaly-based IDS (AIDS), often based on machine learning, address this limitation by modeling normal traffic behavior [10]. However, such models, including CNNs, remain prone to high false positive rates and misclassifications and require retraining when new attacks emerge.

Reinforcement learning (RL) offers an alternative by enabling adaptive agents capable of self-learning and identifying unseen intrusions without supervision [17]. Among RL methods, Q-learning is widely used [1], but its scalability issues in large state spaces have motivated the adoption of Deep Q-Networks (DQN) [15]. Several studies [13, 2] have explored DQN for intrusion detection, but most emphasize detection accuracy in outdated datasets such as NSL-KDD, without detailed implementation strategies or hyperparameter tuning.

Unlike prior hybrid approaches, our proposed NEMESIS framework explicitly separates detection tasks into two sequential steps: a DQN agent rapidly filters benign traffic, while a Random Forest (RF) classifier analyzes only potentially malicious samples. This design reduces computational overhead and enhances multiclass detection accuracy by focusing supervised learning on nonbenign traffic.

The main contributions are as follows.

- A hybrid IDS that combines a DQN agent with a Random Forest classifier.
- A detailed DQN architecture specification.
- Experimental validation in NSL-KDD and CSE-CICIDS2017 datasets.

The remainder of this paper is organized as follows. Section 2 reviews related works; Section 3 presents background concepts; Section 4 details NEMESIS; Section 5 discusses results; and Section 6 concludes the article.

2. Related Work

Several studies have combined deep reinforcement learning (DRL), notably Deep Q-Networks (DQNs), with supervised models for network intrusion detection. For example, [19, 13] use a DQN agent for initial filtering, followed by a supervised classifier for refined detection. A context-sensitive DQN-based IDS was proposed in [17], tested on NSL-KDD, AWID, and UNSW-NB15, showing that ensembles of supervised models can further improve performance. Similarly, [2] introduced a DQN with self-learning and hyperparameter tuning, achieving competitive multiclass detection in NSL-KDD. The work of [5] evaluated an adversarial DQN model on NSL-KDD and AWID, reaching an accuracy close to traditional SVM classifiers.

In general, these approaches demonstrate the potential of hybrid DQN–supervised IDSs, but often without a clear separation of roles between modules, leading to computational overhead and limited adaptability in dynamic environments.

3. Background

3.1. Network-based intrusion detection system

Intrusions threaten the confidentiality, integrity, and availability of information systems. Network-based IDS (NIDS) monitor traffic and complement firewalls by detecting threats beyond signature rules [12]. They are mainly divided into signature-based IDS (SIDS), effective for known attacks but blind to novel threats, and anomaly-based IDS (AIDS), which rely on machine learning or statistical models to detect deviations from normal traffic [10]. Both approaches face challenges such as false positives, evolving attack strategies, and frequent model updates.

3.2. Supervised learning in IDS

Supervised IDS train classifiers on labeled data after preprocessing and feature selection [10]. Algorithms such as decision trees, SVM, KNN, neural networks, or random forests predict whether new traffic is benign or malicious. The main challenge is to achieve strong generalization to unseen attacks while minimizing false alarms.

3.3. Reinforcement learning

Reinforcement learning (RL) is a reward-based paradigm where an agent interacts with an environment and learns policies that map states to actions to maximize cumulative rewards. RL problems are typically modeled as Markov Decision Processes (MDPs) defined by states, actions, rewards, agent, and environment.

Q-Learning

Q-learning is a widely used RL algorithm [15], where the agent learns the expected reward for each state–action pair (Eq. 1). However, it struggles with scalability in high-dimensional spaces, motivating the use of deep neural approximations.

$$Q(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0{=}s, a_0{=}a\right],$$
$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right). \tag{1}$$

Deep Q-Network

Deep Q-Networks (DQNs) extend Q-learning by approximating the Q-function with neural networks, enabling learning in large continuous state spaces [14]. In intrusion detection, DQNs process network traffic features as input and output action values that guide the agent's decisions. Figure 1 illustrates the agent–environment interaction.

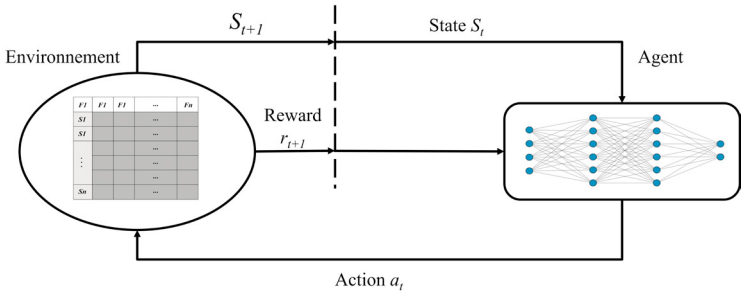

Fig. 1. DQN model based on agent–environment interaction [2]

4. Our proposed Approach: NEMESIS

In the following subsection, we describe the NEMESIS approach, including the DQN model and the datasets used.

Although DQN models alone have shown promising results in network intrusion detection tasks, they typically focus on binary classification (benign vs. attack) and are often limited in their ability to differentiate between multiple types of attacks, as demonstrated in [2] by testing a DQN agent on the NSL-KDD dataset. Our hybrid approach, NEMESIS, addresses this limitation by introducing a two-stage decision process.

1. **Stage 1 – DQN Agent (Binary Classification):** The DQN agent acts as a lightweight and fast filter, determining whether the incoming network traffic is benign or potentially malicious. This reduces the computational load and limits the scope of more complex analysis to relevant samples only.
2. **Stage 2 – Random Forest Classifier (Multiclass Classification):** When the DQN agent flags a sample as non benign, the Random Forest classifier is then activated to perform a fine-grained multiclass classification, identifying the exact attack type (up to 15 classes in CICIDS2017).

This architectural separation brings several benefits.

- Improved precision in multiclass detection: The RF classifier excels at handling unbalanced multiclass data. Benefits from focusing on only suspected attacks, avoiding dilution from benign samples.
- Efficient learning feedback loop: The RF classifier provides precise labels to the DQN agent, which are used to calculate more accurate rewards during training. This tight reward feedback loop improves the agent's learning stability and convergence.
- Better generalization across datasets: The hybrid model shows stronger generalization capacity when tested on both the NSL-KDD and CICIDS2017 datasets, outperforming standalone DQN models in accuracy and recall, particularly for rare attack types.

In contrast, standalone DQN models often struggle with:

- High false positive rates in multiclass settings.
- Slow convergence due to sparse and less informative rewards.
- Inflexibility in handling non-stationary attack distributions or class imbalance.

The choice to combine a DQN agent with a Random Forest classifier is not a simple intuition, but rather a structured experimental approach. First, we evaluated different supervised models (SVM, KNN, Decision Tree and RF) on our datasets, taking into account the criteria of accuracy, robustness to class imbalance, and computation time. Random Forest stood out for several reasons: It is known for its ability to efficiently handle unbalanced multiclass data [4], tolerates noise well and exhibits good generalization, while remaining fast to train and infer. Our own comparative tests showed that RF offered the best performance/speed compromise in our data.

NEMESIS comprises a DQN agent with a variety of components and a supervised classifier. The primary function of this classifier is to assign the *state_vector* to its corresponding class. This classification process plays a crucial role by providing accurate feedback to the environment through correctly labeled classes, thereby facilitating a mechanism for rewarding or penalizing the agent. The ultimate goal is for the agent to refine its learning process to minimize instances of misclassification. The details concerning the various DQN components are described in the following sections.

1. **Environment** The environment provides the agent with states and rewards. Here, it is built from the NSL-KDD and CSE-CICIDS2017 datasets after preprocessing, normalization, and resampling. The features (40 for NSL-KDD, 68 for CICIDS2017) form the state space, while labels are used to compute the rewards.
2. **Agent** The agent is a deep Q-Network (DQN) that learns a value-based policy by interacting with the environment. Observes states, selects actions, and receives rewards. Training starts with exploration (e.g. ϵ-greedy), then gradually shifts toward greedy action selection as the policy improves.
3. **States** States are feature vectors provided by the environment (40 or 68 features depending on the dataset). These vectors represent network traffic and are the input to the DQN during training and evaluation.
4. **Actions** Actions are agent decisions, derived from Q-values on the state vector. The agent evaluates these outputs against thresholds to classify the traffic as benign or malicious.
5. **Rewards** Rewards are feedback signals: positive for correct classifications and negative otherwise. Their values depend on the true labels and prediction probabilities, guiding the agent to improve detection accuracy.

4.1. Datasets

We used two benchmark datasets for intrusion detection: *NSL-KDD* and *CSE-CICIDS2017*. NSL-KDD is an improved version of KDD'99, with redundant records removed to avoid biased learning. It contains 41 features per connection and labels: Normal, DoS, Probe, R2L, and U2R. CICIDS2017 provides richer traffic with 79 features labeled as *Benign* or various *Attack types* [8, 16]. The preprocessing included removal of duplicates, NaN and infinite values, label encoding of categorical attributes [2], Min–Max scaling, and binary relabeling (Benign vs. Attack) for the DQN experiments. The splits were 70/15/15 for DQN and 80/20 for Random Forest. Both data sets are highly imbalanced.

Table 1. Agent and neural network parameters

Parameters	Description	Values
num_timesteps	Number of timesteps to train DQN	300k/400k
num_iter	Number of iteration to improve Q-values in DQN	4
hidden_layers	Number of hidden layers: Setting weights, producing outputs, based on activation function	3
num_units	Number of hidden unit to improve the accuracy of prediction and training	128, 64, 32
activation function	Non-linear activation function	ReLU
gamma γ	Discount factor for target prediction	0.3
Learning rate	The learning rate controls the size of weight updates	Dynamic
epsilon ϵ	Degree of randomness for performing actions	Dynamic
batch-size (bs)	A batch of dataset's records fetched for processing	16

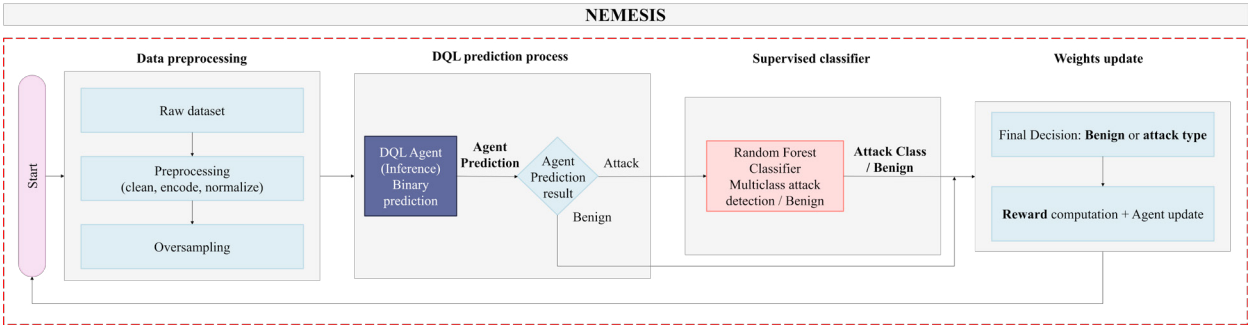

Fig. 2. NEMESIS architecture

To improve rare attack detection, we applied oversampling only in training / validation: *SMOTE* for CICIDS2017 [6] and *RandomOverSampler* for NSL-KDD. This choice balances classes without biasing the test set [7, 18]. After balancing, Random Forest training used equalized class distributions (1,816,702 samples/class for CICIDS2017; 67,343/class for NSL-KDD), while DQN was trained on binary data.

The DQN agent architecture was empirically tuned for accuracy and efficiency. We used a three-layer MLP (128–64–32) with ReLU activations, batch size 16, and a dynamic learning rate ensuring stable convergence. The discount factor was set to $\gamma = 0.3$, and the training was carried out in 300k (CICIDS2017) and 400k (NSL-KDD) time steps. The binary action space (benign / suspicious) and the reward function emphasize accurate detection while minimizing false alarms, enabling adaptation to evolving threats. The final hyperparameters are summarized in Table 1.

Training follows a ϵ-greedy policy [3, 15], where actions are selected, rewards calculated, and Q-values are updated iteratively. During inference, the agent first performs binary classification: benign traffic is discarded, while suspicious samples are forwarded to the Random Forest for multiclass identification. This two-stage process reduces computational cost and improves detection accuracy. The workflow is shown in Figure 4.1, and the procedures are described in Algorithms 1 and 2.

5. Experimental results

Experiments were carried out on a system with 105GB RAM, a 10-core Intel Xeon Platinum 8358 CPU (2.60 GHz) and an NVIDIA A100 GPU, running Ubuntu. The implementation used Python 3.11.7 and Stable-Baselines3 v2.3.2. The results were evaluated using standard machine learning metrics, and the evaluation metrics include accuracy, precision, recall, and F1-score.

Algorithm 1: Training of Deep Q-Learning Agent in Custom Environment

Input : Preprocessed dataset D, DQN parameters (episodes, γ, ϵ, batch size, etc.)
Output: Trained agent policy for binary intrusion detection

```
Normalize(D)
Initialize agent parameters
batch_size ← 16
States ← sample mini-batches from D
Initialize model
for timestep ← 1 to T do
    Reset environment state
    for iter ← 1 to num_iterations do
        // Step 1: Select action
        for i ← 1 to batch_size do
            if with probability ϵ then
                A_i ← random action from action space
            else
                Q_i ← model.predict(state_i)
                A_i ← arg max(Q_i)
            end
        end
        ϵ ← ϵ × decay_rate
        // Step 2: Compute rewards
        for i ← 1 to batch_size do
            R_i ← compute reward(A_i, true label)
        end
        // Step 3: Q-value update
        Q' ← model.predict(next_state)
        for i ← 1 to batch_size do
            Q_target[i] ← R_i + γ · max(Q'[i])
        end
        // Step 4: Train model
        model.train(state, Q_target)
        Update state ← next_state
    end
end
```

Algorithm 2: Hybrid Inference: DQN (binary) + RF (attack class)

Input : Test set $\{(x_i, y_i)\}_{i=1}^{n}$, trained `DQN`, trained `RF`
Output: Final predictions {benign or attack class} and DQN reward updates

```
for i ← 1 to n do
    x ← preprocess_sample(x_i)
    state ← extract_state_vector(x)
    binary_pred ← DQN.predict(state)   // 0 = benign, 1 = attack
    if binary_pred = 0 then
        final_pred ← "benign"
    else
        attack_class ← RF.predict(state)
        final_pred ← attack_class
    end
    true_label ← y_i
    if binary_pred = 0 and true_label = "benign" then
        reward ← +2        // True negative
    else if binary_pred = 1 and true_label ≠ "benign" then
        if final_pred = true_label then
            reward ← +5        // Correct classification
        else
            reward ← −2        // Attack detected, wrong type
    else
        reward ← −5 // False positive or false negative
    end
    DQN.update(state, reward)
end
```

5.1. Performances: individual classifiers

The classifiers (DQN and RF) are evaluated separately and both are compared to similar models in the state of the art.
The prediction of the DQN model and the results of the random forest classification are presented in table 2.

The experimental evaluation highlights the effectiveness of NEMESIS for both binary and multiclass intrusion detection. The standalone DQN agent shows strong recall (97.3% on NSL-KDD), capturing nearly all attacks but with moderate precision due to false positives. When combined with the RF classifier, the multiclass identification

Table 2. DQN Agent and RF Classifier Results

Dataset	Model	Accuracy	Precision	Recall	F1-score
CICIDS17	Agent	95.04%	96.64%	93.32%	94.95%
	RF	98.82%	99.44%	98.82%	99.09%
NSLKDD	Agent	91.18%	90.24%	97.34%	93.66%
	RF	76.60%	81.33%	76.66%	74.51%

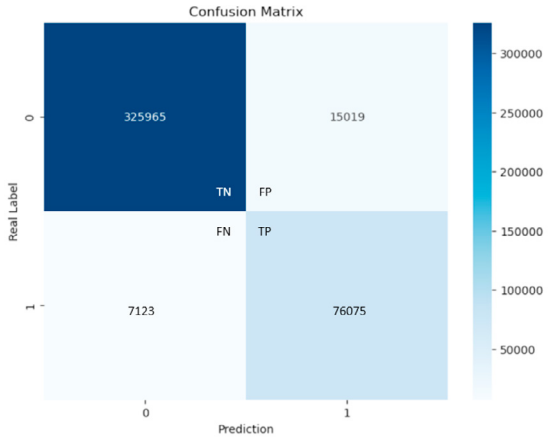

Fig. 3. DQN confusion matrix for CSE-CICIDS2017

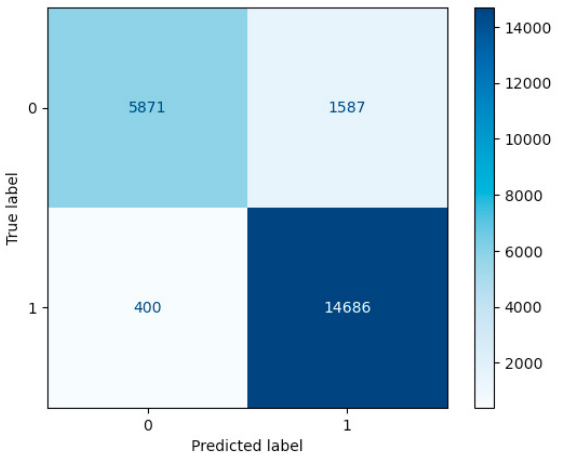

Fig. 4. DQN confusion matrix for NSL-KDD

Table 3. Comparison of IDS approaches

Work	ML Techniques	Adaptive-Learning	Dataset	Accuracy
[2]	DQN	✓	NSL-KDD	78%
[17]	DQN + RF	✓	NSL-KDD	77%
[5]	DQN	✓	NSL-KDD	74%
[13]	DQN	✓	CICIDS2017	97%
[19]	DQN	✓	CICIDS2017	99%
Our approach	DQN + RF	✓	NSL-KDD	91%
			CICIDS2017	95%

improves significantly, achieving 98.8% accuracy on CICIDS2017, while the lower performance of NSL-KDD reflects its limited and outdated features.

The ablation study confirms this complementarity: DQN alone is effective for real-time detection (94.2% on CICIDS2017) but cannot differentiate attack types, while RF alone can classify multiple classes (98.8% on CICIDS2017) but suffers from imbalance and reduced precision, especially on NSL-KDD. The hybrid NEMESIS design takes advantage of both strengths: DQN filters benign traffic, and RF focuses on harder multiclass cases, achieving the best overall F1 score and accuracy across benchmarks.

Confusion matrices (Figures 3–4) show that most benign and attack samples are correctly detected, with errors mainly from rare attack types. Comparative results (Table 3) further demonstrate that NEMESIS outperforms classical models (SVM, Naive Bayes, CNN-BiLSTM) and prior DQN approaches on NSL-KDD (91% accuracy) [2, 9], and achieves competitive performance on CICIDS2017 (94.2%), offering better multiclass differentiation and adaptability than RL-only or supervised methods [19, 13].

6. Conclusion

In this paper, we proposed NEMESIS, a hybrid intrusion detection approach that combines deep Q-network reinforcement learning with a supervised random forest classifier. Our method efficiently detects and classifies network

intrusions, including unseen attack types, leveraging DQN for initial binary filtering and RF for multiclass attack identification. We detailed the core architectural components, training strategies, and hyperparameter selection. The experimental results show that NEMESIS outperforms most existing methods in benchmark data sets. Future work will focus on extending NEMESIS to counter adversarial and distributed attack scenarios.

References

[1] Adawadkar, A.M.K., Kulkarni, N., 2022. Cyber-security and reinforcement learning—a brief survey. Engineering Applications of Artificial Intelligence 114, 105116.

[2] Alavizadeh, H., Alavizadeh, H., Jang-Jaccard, J., 2022. Deep q-learning based reinforcement learning approach for network intrusion detection. Computers 11, 41.

[3] Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A., 2017. Deep reinforcement learning: A brief survey. IEEE Signal Processing Magazine 34, 26–38.

[4] Breiman, L., 2001. Random forests. Machine learning 45, 5–32.

[5] Caminero, G., Lopez-Martin, M., Carro, B., 2019. Adversarial environment reinforcement learning algorithm for intrusion detection. Computer Networks 159, 96–109.

[6] Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., 2002. Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research 16, 321–357.

[7] Dhanabal, L., Shantharajah, S., 2015. A study on nsl-kdd dataset for intrusion detection system based on classification algorithms. International journal of advanced research in computer and communication engineering 4, 446–452.

[8] Haroon, M., Ali, H., 2023. Ensemble adversarial training based defense against adversarial attacks for machine learning-based intrusion detection system. Neural Network World 317, 336.

[9] Jiang, K., Wang, W., Wang, A., Wu, H., 2020. La détection d'intrusion dans le réseau combine l'échantillonnage hybride avec un réseau hiérarchique profond. IEEE access 8, 32464–32476.

[10] Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J., 2019. Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity 2, 1–22.

[11] Kilincer, I.F., Ertam, F., Sengur, A., 2021. Machine learning methods for cyber security intrusion detection: Datasets and comparative study. Computer Networks 188, 107840.

[12] Liao, H.J., Lin, C.H.R., Lin, Y.C., Tung, K.Y., 2013. Intrusion detection system: A comprehensive review. Journal of Network and Computer Applications 36, 16–24.

[13] Malik, M., Saini, K.S., 2023. Network intrusion detection system using reinforcement learning techniques, in: 2023 International Conference on Circuit Power and Computing Technologies (ICCPCT), IEEE. pp. 1642–1649.

[14] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al., 2015. Human-level control through deep reinforcement learning. nature 518, 529–533.

[15] Nguyen, T.T., Reddi, V.J., 2021. Deep reinforcement learning for cyber security. IEEE Transactions on Neural Networks and Learning Systems 34, 3779–3795.

[16] Phulre, A.K., Jain, S., Jain, G., 2024. Evaluating security enhancement through machine learning approaches for anomaly based intrusion detection systems, in: 2024 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS), IEEE. pp. 1–5.

[17] Sethi, K., Sai Rupesh, E., Kumar, R., Bera, P., Venu Madhav, Y., 2020. A context-aware robust intrusion detection system: a reinforcement learning-based approach. International Journal of Information Security 19, 657–678.

[18] Tan, X., Su, S., Huang, Z., Guo, X., Zuo, Z., Sun, X., Li, L., 2019. Wireless sensor networks intrusion detection based on smote and the random forest algorithm. Sensors 19, 203.

[19] Tellache, A., Mokhtari, A., Korba, A.A., Ghamri-Doudane, Y., 2024. Multi-agent reinforcement learning-based network intrusion detection system, in: NOMS 2024-2024 IEEE Network Operations and Management Symposium, IEEE. pp. 1–9.

[20] Venter, H., Eloff, J.H., 2003. A taxonomy for information security technologies. Computers & Security 22, 299–307.