

Adversarial Asymmetric Agents for Network Intrusion Detection Using Deep Reinforcement Learning

Curtis Rookard

Department of Computer Science and Technology, SUNY Empire State University, Saratoga Springs, NY 12866, USA

Abstract—Machine learning-based network intrusion detection systems (NIDS) increasingly operate in environments where adversaries can adapt their behavior in response to deployed defenses. However, most empirical IDS studies evaluate defenders against static, dataset-defined attack distributions and do not explicitly model an adaptive attacker with a distinct learning strategy or action space. This study proposes an adversarial asymmetric learning framework for intrusion detection in which an attacker agent dynamically selects network attacks while a defender agent classifies observed network flows. We evaluate three attacker–defender pairings derived from supervised deep neural network (DNN) and Deep Q-Network (DQN) agent types. We report accuracy, precision, recall, F1-score, and G-Mean under stratified 10-fold cross validation across traditional local area network, industrial control system (ICS)/supervisory control and data acquisition (SCADA), and internet-of-things (IoT) networking environments. Empirically, the network-flow classifier performs best when paired with an intelligent attacker using reinforcement learning, even when the defender itself is not reinforcement-learning based. Overall, the results show that learning asymmetry and multi-agent interaction can substantially alter measured detection outcomes and that supervised defenders remain highly competitive in the presence of intelligent adversarial agents.

Keywords—Network intrusion detection; cybersecurity; deep reinforcement learning; deep q-networks; adversarial machine learning; multi-agent learning

I. INTRODUCTION

The continued growth of information technology and information processing systems worldwide have seen an increase of computing related resources. Underpinning the development and expansion of these information systems is the growth and proliferation of computer networks which now have to accommodate more than just traditional computing platforms. New technologies, such as internet-of-things, embedded systems, and industrial control systems and supervisory control and data acquisition platforms now comprise a significant share of all computing systems [1]. Application of these systems necessitate the need for reliable, trustworthy network connectivity. Unfortunately, the continued expansion and adoption of these computer systems have introduced additional attack vectors for adversaries to exploit.

Pinnacle to the protection of these computing systems is a need for robust cybersecurity technologies to ensure the threats do not comprise the aforementioned information systems. In addition to traditional cybersecurity techniques such as virtual

local area networks/network segmentation, access controls, and intrusion prevention and detection systems play a crucial role [2]. Most intrusion detection systems today rely on robust machine learning mechanisms to ascertain whether to permit or to deny traffic from entering a computer network or system [3].

With the advent of more sophisticated artificial intelligence techniques, such as large language models [4], and generative adversarial networks (GANs) [5], there has been a renewed interest in applying other methods for cybersecurity, specifically for intrusion detection. As such, we have seen an increase in the utilization of more advanced machine learning algorithms, extending into the reinforcement learning domain. In reinforcement learning, an intelligent agent interacts with environmental stimuli in order to obtain the highest reward possible. In cases relating to cybersecurity, this could be presented in the sense of a cybersecurity agent interacting with the networking environment.

Interestingly, with the increased utilization of deep reinforcement learning agents for defense, there has also been a rise in the usage of deep reinforcement learning agents for offensive security purposes [6]. The combination of defender and attacker learning introduces a methodological challenge for intrusion detection evaluation: the defender is no longer tested only against a static distribution of attacks, but against an opponent whose behavior can change the mixture of observed traffic. This motivates a framework that can compare symmetric and asymmetric attacker–defender learning configurations under a common experimental protocol.

In this work, we make the following contributions to the intrusion detection literature:

- We propose an adversarial asymmetric learning framework for network intrusion detection in which the attacker and defender are modeled as interacting agents with potentially different learning mechanisms and action spaces.
- We examine the role-dependent effect of learning architecture by comparing a symmetric DNN–DNN baseline against two role-reversed asymmetric configurations: a DQN attacker with a DNN defender and a DNN attacker with a DQN defender. This controlled design allows us to determine whether the effect of reinforcement learning depends on the role in which it is deployed.

- We evaluate these scenarios on three widely used intrusion detection datasets spanning distinct network types: local area networking represented by the CIC-IDS 2017 dataset [7], industrial control system (ICS) and supervisory control and data acquisition (SCADA) environments represented by the DNP3 dataset [8], and IoT networking represented by the ACI-IoT dataset [9].
- We conduct a comprehensive comparative analysis across scenarios and datasets using standard intrusion detection metrics of accuracy, precision, recall, F1-score, and G-Mean to quantify how asymmetric learning changes measured detection performance under adaptive adversarial interaction.

Based on the literature reviewed in Section II, we identify a gap in the systematic evaluation of the directional placement of supervised and reinforcement learning mechanisms across opposing attacker and defender roles. The novelty of our work therefore lies not in the use of DNN or DQN models individually, but in the controlled comparison of their role asymmetry within an explicit attacker–defender interaction in which the attacker can influence the mixture of network traffic encountered during learning. The rest of our study is described in the following sections. Section II provides a greater overview of the related literature with both intrusion detection and reinforcement learning. Section III covers the methodology presented in our manuscript, while section IV provides the numerical analysis of our results. Section V provides the discussion of our work, and finally section VI concludes our manuscript.

II. RELATED LITERATURE

After a careful review of literature, we find a lack of prior research examining deep reinforcement learning for intrusion detection under the same premises we adopt in our work. We aim to most notably present a competitive formulation with asymmetric agents using a combination of both deep reinforcement learning agents and traditional deep learning agents for both offensive and defensive intrusion detection. While numerous studies evaluate conventional machine learning and deep learning classification, and a growing body of work applies reinforcement learning to intrusion detection and traffic classification, these approaches typically assume a single decision-maker, that is, the defender, and do not explicitly model an adaptive adversary with a different learning strategy, information advantage, or action space. Our proposed framework is based upon this asymmetry, enabling a more realistic characterization of attacker–defender dynamics across heterogeneous network scenarios.

Accordingly, this section summarizes the most representative related works that inform our experimental design and baselines. We first review established intrusion detection results on widely used datasets to situate performance expectations under common supervised-learning setups, then discuss the most relevant intelligent intrusion detection studies utilizing reinforcement learning in the broader literature. Finally, we highlight similar works applying adversarial attacks for cybersecurity to thus better position our work against the current literature.

A. Traditional Intrusion Detection

Intrusion detection systems (IDSs) are commonly categorized by both deployment scope, that is either host-based vs. network-based, and detection philosophy, which are divided into signature-based vs. anomaly-based. Signature-based systems match traffic to known patterns and typically offer low false-alarm rates for previously observed attacks, whereas anomaly-based systems aim to detect deviations from a learned notion of “normal” behavior and are therefore better positioned to identify novel or low-footprint threats. As operational networks have diversified through the additional of new technologies and software, anomaly-based IDS systems have increasingly relied on statistical learning models that map flow-based or packet-derived features to either benign or attack labels. Broad surveys in the cyber analytics literature document this progression and show that classical methods such as decision trees, support vector machines, k-nearest neighbors, and shallow ensembles remain competitive in some environments, but also that their performance is often sensitive to feature engineering choices, dataset artifacts, and class imbalance [10].

More recent work has accelerated the shift from manual feature engineering toward deep learning (DL) approaches that can learn nonlinear decision boundaries and capture higher-order feature interactions. Deep neural network (DNN)-based IDS models are typically trained in a supervised manner on labeled flow records, and they have been extended with temporal architectures, including recurrent neural networks (RNNs), gated recurrent units (GRUs), and long-short term memory (LSTM) networks alongside representation learning to better model sequential dependencies or reduce reliance on handcrafted preprocessing. For example, Vinayakumar et al. evaluate DNN-based intrusion detection across multiple benchmark datasets and highlight that modern network behavior and rapidly evolving attack patterns motivate continual dataset refresh and careful cross-dataset benchmarking [11]. At the same time, end-to-end NIDS research has explored learned representations that incorporate contrastive objectives and hierarchical feature extraction to improve generalization under distribution shift, which is especially relevant when models trained on one environment are deployed in another [12].

Because IDS evaluation is tightly coupled to dataset construction, a substantial body of work focuses on the limitations of legacy benchmarks and on the design of more realistic datasets. In the work of Tavallaee et al. these researchers present inherent issues in KDD Cup 1999 dataset. These issues include redundancy and biased difficulty distributions, and they thus propose the NSL-KDD dataset to mitigate those artifacts, which partly explains why many “high-accuracy” results on older benchmarks fail to translate to operational settings [13]. Dataset-centric surveys further show that modern datasets attempt to better reflect contemporary traffic mixtures, feature sets, and attack diversity, while still exhibiting challenges such as severe imbalance and dataset-specific leakage [14], [15]. For instance, the UNSW-NB15 was created to include a hybrid of modern normal activity and synthesized contemporary attacks, thus improving upon the previous NSL-KDD dataset [16]. Similarly, the CIC-IDS 2017 dataset provides labeled flow records derived from network packet captures and has become

a standard evaluation dataset for supervised NIDS baselines [7]. These findings motivate our use of multiple datasets spanning heterogeneous network types (LAN, ICS/SCADA, and IoT) and allows us to better ascertain IDS performance.

B. Reinforcement Learning for Intrusion Detection

Reinforcement learning has been increasingly applied to cybersecurity because it often mathematically models adaptive decision-making under uncertainty, and partial observability. In many cyber defense formulations, a reinforcement learning agent learns a policy for sequential actions such as dynamic access control, moving target defense, deception, or response orchestration. In contrast to static supervised learning, reinforcement learning can optimize long-run objectives by explicitly learning from interaction feedback rather than relying solely on fixed labels.

Within the intrusion detection literature specifically, reinforcement learning appears in two primary strands. The first treats intrusion detection as a sequential classification problem where the agent's action is the predicted class and the reward encodes correctness or utility; Deep Q-Learning variants have been proposed to learn intrusion detection policies under this framing [17]. The second strand uses reinforcement learning as an optimization layer for IDS pipelines. These pipelines aim to select features, tune thresholds, or adapt detection policies online in non-stationary environments. Ren et al. propose a deep reinforcement learning-based intrusion detection method that emphasizes feature selection which provides performance benefits on modern traffic datasets [18]. More recent applied studies continue to report that deep reinforcement learning can be competitive with supervised baselines under certain conditions, especially in situations wherein the environment is dynamic or when a defender must trade off sensitivity with operational costs [19], [20].

However, a consistent limitation in much of the intrusion detection literature applying reinforcement learning is that the learning environment is often derived from a fixed dataset stream, and the "adversary" is modeled implicitly through pre-labeled attack samples rather than as an adaptive opponent. Surveys of deep reinforcement learning for intrusion detection, such as those seen in IoT contexts, note that many published approaches assume a single decision-maker, often the defender, and do not explicitly study the stability issues that arise when both attacker and defender learn simultaneously [21]–[24]. Instances incorporating both attacker and defender interactions are more naturally captured by stochastic or Markov games to better learn robust strategies against adaptive opponents [25]–[27]. Our work directly targets this gap for intrusion detection by evaluating asymmetric and symmetric pairings for both attacker and defender across heterogeneous network settings, thereby empirically characterizing how learning asymmetry alters detection outcomes.

C. Adversarial Methods for Cybersecurity

Adversarial machine learning has become a central concern for ML-based IDS because attackers may attempt to evade detection through evasion attacks, corrupt training data through data poisoning, or extract model behavior such as through model inference attacks, to enable more effective intrusions.

In NIDS, these threats are complicated by domain constraints; feature perturbations must correspond to realizable network traffic, preserve protocol validity, and often maintain malicious functionality. Recent surveys propose taxonomies tailored to NIDS and synthesize both attack and defense mechanisms, emphasizing that robustness claims in cybersecurity must account for traffic realism and threat-model fidelity [28]–[30].

A growing set of works operationalizes these concerns by designing attacks that respect practical constraints. Sheatsley et al. show that even with a constrained attack surface, NIDS models can remain vulnerable to adversarial examples and that designing realistic networks alone does not guarantee robustness against adversarial attacks [31]. GAN-based approaches have also been explored to generate adversarial traffic flows. Zolbayer et al. introduce NIDSGAN to craft practical adversarial flows and evaluate their ability to circumvent ML-based NIDS under realistic feature constraints [32]. Similarly, Debicha et al. propose Adv-Bot, a black-box evasion method that aims to produce botnet traffic capable of evading detection while still preserving intended malicious behavior, and they also discuss mitigation strategies [33]. Rookard shows that even deep learning-based NIDS can still be heavily affected by adversarial attacks on training data [34]. Complementary lines of work study defender strategies under limited query access and partial feedback, highlighting the challenges defenders face when they cannot directly observe model outputs or when probing risks triggering detection [35].

This adversarial literature motivates two design choices in our work. First, it reinforces the need to evaluate IDS methods under conditions that reflect adaptivity and strategic behavior rather than assuming static attack distributions. Second, it suggests that robustness should be assessed not only by aggregate accuracy but by sensitivity-focused metrics and by stability across datasets. Rather than generating adversarial perturbations in feature space, our work models adaptivity through an explicit attacker agent whose learning strategy may differ from the defender's learning, providing a complementary, game-aware perspective on the adversarial presence in network intrusion detection.

III. METHODOLOGY

In this section, we cover the primary methodologies used in our research, specifically, we cover Markov decision processes, and their usage for deep Q-networks. We then discuss our algorithm and datasets applied in our work.

A. Markov Decision Process

We formulate our intrusion detection problem as a Markov decision process (MDP), a standard framework for modeling sequential decision-making in stochastic environments [36], [37]. An MDP is typically defined by the tuple $\langle S, A, T, R, \gamma \rangle$, where S is the set of states, A is the set of actions, $T(s' | s, a)$ denotes transition dynamics of moving to new state s' given action a and current state s , $R(s, a)$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor that controls the weighting of future rewards.

Accordingly, we model network intrusion detection as follows: let S denote the set of network observations in the intrusion detection dataset, where each state corresponds

to a single network flow represented by a feature vector; these features include flow duration, source bytes, destination bytes, protocol type, and related statistics. The action space A corresponds to the defender agent's classification decision for a given flow, either being benign or an attack. The transition function T captures how traffic evolves from one observed flow to the next, reflecting the underlying mixture of benign and adversarial behavior present in the network stream. Finally, the reward signal is defined as a binary outcome: the agent receives $r = 1$ for a correct classification and $r = 0$ otherwise, i.e., $r \in \{0, 1\}$. The MDP formulation is therefore used as a simplified, dataset-driven model of attacker-defender interaction using the network flow traffic data.

B. Deep Q-Networks

We solve the above MDP using value-based reinforcement learning. In particular, we build on the Q-learning algorithm [37], which learns an action-value function $Q(s, a)$ representing the expected discounted return gained by our intelligent agent taking action a in state s through following an optimal policy. This optimal action-value function thus satisfies the Bellman optimality relationship,

$$Q^*(s, a) = \mathbb{E} \left[r + \gamma \max_{a' \in A} Q^*(s', a') \mid s, a \right], \quad (1)$$

and Q-learning iteratively approximates Q^* via temporal-difference updates of the form

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left(r + \gamma \max_{a' \in A} Q(s', a') - Q(s, a) \right), \quad (2)$$

where, α is the learning rate.

We therefore employ a Deep Q-Network (DQN), which approximates $Q(s, a)$ using a neural network parameterized by θ , denoted $Q(s, a; \theta)$ [38] as network-flow feature spaces are high-dimensional and continuous and maintaining a tabular Q-function is infeasible. Given a state feature vector, represented as a network flow, as input, the network outputs a vector of Q-values, one per defender action which thus allows the agent to select actions using an ϵ -greedy policy (i.e., choosing $\arg \max_a Q(s, a; \theta)$ with probability $1 - \epsilon$ and exploring otherwise). To stabilize learning, we use experience replay and a target network: transitions (s, a, r, s') are stored in a replay buffer and sampled uniformly to break temporal correlations, while a separate set of parameters θ^- is periodically updated to provide a slowly moving target. The DQN is trained using the Huber temporal-difference loss between the predicted action value, $Q(s, a; \theta)$ and the target $y = r + \gamma \max_{a' \in A} Q(s', a'; \theta^-)$. The Huber loss reduces sensitivity to large temporal-difference errors while retaining a quadratic penalty for smaller errors.

C. Adversarial Asymmetric Agents for Reinforcement Learning

To capture adaptive attacker-defender dynamics that are not represented in standard supervised IDS evaluation, we extend the single-agent MDP formulation in Sections III-A–III-B to a two-agent competitive learning environment. Rather than assuming that malicious traffic arrives according to a fixed

distribution, as in offline classification, we model an attacker agent that actively influences which type of traffic is encountered by the defender. This design allows us to study how detection performance changes when learning is symmetric, wherein both agents use the same class of learner, versus asymmetric wherein agents use different learning paradigms, which is central to the comparative scenarios evaluated in this work.

1) *Two-agent game formulation:* We model the interaction as a two-player stochastic game, akin to a Markov game), defined by the tuple $\langle S, A_D, A_A, T, R_D, R_A, \gamma \rangle$, where S is the shared state space, A_D and A_A are the defender and attacker action spaces, T defines the transition dynamics, R_D and R_A are the respective reward functions, and $\gamma \in [0, 1)$ is the discount factor. The state $s_t \in S$ corresponds to the observed feature vector of a network flow at time step t . The defender action $a_t^D \in A_D$ is the predicted class label, while the attacker action $a_t^A \in A_A$ represents the attacker's selection of an attack mode that determines the type of traffic the defender will face next. This reflects a realistic adversarial capability: while a defender may only observe a flow and decide how to label it, an attacker can adaptively choose which tactics/techniques to employ over time.

2) *Reward structure and competitive objective:* We use a competitive (zero-sum) reward design that directly encodes detection success. Let $\mathbb{I}[\cdot]$ denote the indicator function and let y_t be the ground-truth label (mapped to the defender's label space when coarse categories are used). The defender receives:

$$r_t^D = \mathbb{I}[a_t^D = y_t], \quad (3)$$

and the attacker receives the complementary reward

$$r_t^A = 1 - r_t^D. \quad (4)$$

Thus, the defender maximizes correct classification over time, while the attacker maximizes defender misclassification. Thus, attacker reward reflects defender error induced through adaptive traffic-mixture selection. Because the attacker neither perturbs flow features nor synthesizes adversarial examples, this advantage represents a mixture-selection effect rather than conventional feature-space evasion

3) *Asymmetry in action spaces and information:* Our framework is intentionally asymmetric in two ways. First, the attacker and defender may have different action spaces: the defender acts in a coarse IDS label space, e.g. denial-of-service, while the attacker acts in a fine-grained space that captures specific attack types, e.g. GoldenEye and Slowloris attacks as seen in the CIC-IDS 2017 dataset. Second, the attacker controls the traffic mixture by selecting which class of flows is drawn next, whereas the defender reacts to whatever flow is observed. These design choices jointly model a realistic attacker advantage: even if the defender's classifier is strong on average, an adaptive attacker can concentrate effort on classes wherein the defender is weakest. Before selecting an action, the attacker observes the current normalized flow-state representation. The attacker does not directly observe the defender's model parameters, class probabilities, or an explicit history of defender decisions, instead adapting from the scalar

reward and replayed interaction history, whereas the supervised DNN attacker is trained using labeled data and subsequently acts as a fixed selector. The two attackers, the DNN attacker and the DQN attacker, share the same action space but do not optimize equivalent learning objectives.

4) Agent instantiations and the three evaluated pairings:

We implement each agent as either 1) a standard deep neural network (DNN) classifier trained via supervised learning, or 2) a Deep Q-Network (DQN) trained via reinforcement learning. This yields the three scenarios evaluated in Section I: 1) *Both DNNs*, where the attacker follows a fixed DNN-based policy for selecting attack modes and the defender follows a fixed DNN classifier; 2) *DNN Defender*, where the defender is supervised and fixed while the attacker learns a DQN policy; 3) *DQN Defender*, where the attacker is supervised and fixed while the defender learns a DQN. For DNN agents, the policy is deterministic at inference time, whereas DQN agents act using an ϵ -greedy strategy during training to balance exploration and exploitation.

5) *Learning procedure for DQN agents*: When an agent is instantiated as a DQN, it learns an action-value approximation $Q(s, a; \theta)$ for its respective action space (attacker or defender). Each agent maintains its own replay buffer and target network and minimizes the temporal-difference loss described in Section III-B. To improve stability, we train with standard DQN stabilizers (experience replay and target networks) and update each agent from its own buffer using mini-batches drawn from recent interaction experience. Empirically, this setup is sufficient for stable learning in our dataset-driven environments and enables direct comparison to supervised baselines under identical state representations and evaluation metrics.

6) Training and testing process within cross-validation:

Within each cross-validation fold, supervised DNN agents are trained only on the training split. DQN agents interact only with flows drawn from the training split while learning through the aforementioned sampling mechanism. After training, we evaluate the defender's detection performance on the held-out fold using the same metrics reported in Section IV-A. This ensures that the reported results measure generalization to unseen flows while still allowing attacker adaptivity during training to shape the defender's learning experience.

D. Datasets

In this section, we cover the intrusion detection datasets used across our three intrusion detection environments.

1) *CIC-IDS 2017 dataset*: The University of New Brunswick's Canadian Institute for Cybersecurity crafted the CIC-IDS 2017 dataset to address limitations in present intrusion detection datasets. Improving upon the previously released NSL-KDD dataset [13] and UNSW-NB15 datasets [16], this dataset better resembles real world network traffic present in a local area network. Traffic used to create the CIC-IDS 2017 dataset originated from network packet capture (.pcap) files taken over a five-day period.

In this dataset, there are over 2.8 million instances of traffic, with several instances being more reflective of normal, benign traffic. There are 78 features present in the dataset which include network flow characteristics such as the duration of the

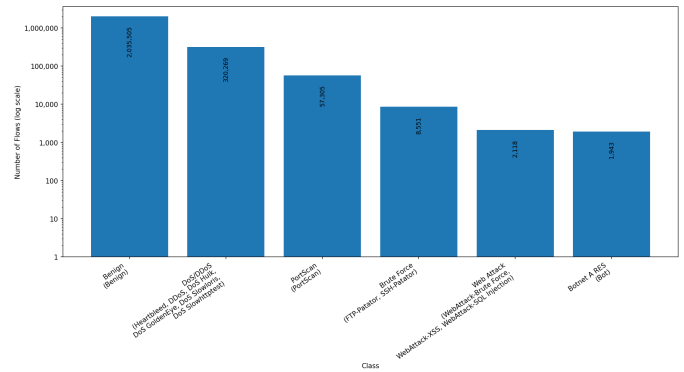


Fig. 1. Distribution of traffic present in the CIC-IDS 2017 dataset using a logarithmic scale for the number of network flows. Traffic class raw counts are indicated on each bar. Specific attacks for each traffic class are indicated in parentheses.

network flow, the total length of the packet in bytes, ports used in communication, transmission control protocol flags (TCP) present. An overview of the distribution of the network traffic is provided in Fig. 1.

2) *DNP3 dataset*: The DNP3 intrusion detection dataset, created by the Internet of Things and Applications (ITHACA) lab at the University of Western Macedonia, provides a means to determine the efficacy of machine learning and deep learning models on industrial control systems and supervisory control and data acquisition (SCADA) platforms. The dataset is centered upon the Distributed Networking Protocol version 3, which is a common protocol used in many SCADA environments [8].

Across both the training and testing datasets on the DNP3 intrusion detection dataset, there are over seven thousand unique flows with 84 different features reflective of netflow traffic. We opt to utilize the balanced training and testing datasets as provided by the ITHACA lab for our work.

3) *ACI-IoT dataset*: In order to more accurately assess threats, especially within IoT networking environments, the Army Cyber Institute at the United States Military Academy developed a novel dataset, the ACI-IoT Network Traffic dataset [9] which captures the complexities of traffic originating from various IoT devices with both wired and wireless connections to the network.

There are over 1.2 million instances of unique traffic present in the ACI-IoT dataset, with over 84 different features, akin to the features captured in the netflow traffic of the CIC-IDS 2017 dataset. An overview of the distribution of the network traffic is provided in Fig. 2.

E. Proposed Experimental Design

This subsection specifies the complete experimental protocol used to evaluate adversarial asymmetric learning in intrusion detection. Our design is driven by two requirements: 1) the comparison across the three attacker-defender pairings must be fair (identical state representation, identical train/test splits, and consistent evaluation metrics), and 2) the interaction between attacker and defender must be *adaptive* for

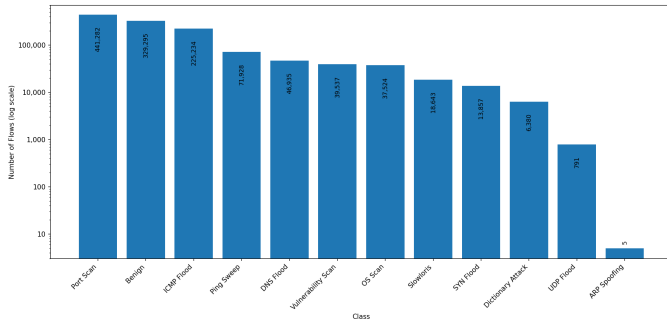


Fig. 2. Distribution of traffic present in the ACI-IoT 2023 dataset using a logarithmic scale for the number of network flows. Traffic class raw counts are indicated on each bar.

reinforcement learning agents while still permitting out-of-sample generalization measurement using cross validation.

1) Preprocessing and state representation: For each dataset, we construct a tabular feature matrix where each row corresponds to a single network flow and each column corresponds to a numeric flow feature. Categorical fields are converted to numeric form using standard encodings, and missing/infinite values are removed from the dataset. To ensure comparable optimization dynamics across datasets and to stabilize both DNN and DQN training, features are scaled using statistics computed on the training split only, and the same transformation is applied to the corresponding test fold. The resulting normalized feature vector constitutes the environment state $s_t \in S$ for both agents.

2) Label spaces and coarse-fine mapping: Each dataset includes a potentially large set of specific attack labels. For the defender, we define a *coarse* action space A_D consisting of benign traffic plus a small set of broad attack categories appropriate for the dataset. For the attacker, we define a *fine* action space A_A that corresponds to the available attack subclass labels in the dataset. A deterministic mapping $g(\cdot)$ maps each fine label to the defender's coarse label space, and correctness is computed using the coarse label $y_t^{(coarse)} = g(y_t^{(fine)})$.

a) Cross-validation process: All reported results are obtained using stratified 10-fold cross validation. In each fold, we split data into a training portion and a held-out test portion. Any preprocessing parameters and any supervised model fitting are performed strictly on the training split. Reinforcement learning agents interact only with the training split during learning, and the learned defender policy is then evaluated on the held-out test fold. This protocol ensures that attacker adaptivity can shape training-time exposure, while the final metrics quantify generalization to unseen flows.

3) Dataset-driven interaction environment: Within each fold, we construct an episodic environment from the training split. At each time step t , the attacker selects an action $a_t^A \in A_A$, which indexes a subset (pool) of training flows whose fine-grained label corresponds to a_t^A . The environment then samples the next observation x_{t+1} from that pool (uniformly at random). The defender observes the resulting state $s_{t+1} = x_{t+1}$ and selects an action $a_{t+1}^D \in A_D$ corresponding to its predicted coarse label. The defender reward is binary,

$$r_{t+1}^D = \mathbb{I}[a_{t+1}^D = g(y_{t+1}^{(fine)})], \quad (5)$$

and the attacker reward is complementary,

$$r_{t+1}^A = 1 - r_{t+1}^D. \quad (6)$$

Episodes terminate after a fixed horizon or when insufficient samples remain in one or more pools, depending on the dataset. To reduce sampling artifacts, we attempt to avoid immediate reuse of the same flow within an episode, while permitting reuse across episodes to ensure adequate training exposure.

4) Agent configurations and the three scenarios: We evaluate the following attacker-defender pairings under identical folds and preprocessing:

- Both DNNs: Both attacker and defender are trained as supervised DNN classifiers and then fixed during interaction/evaluation.
- DNN Defender: The defender is a supervised DNN classifier (fixed after training) and the attacker learns an adaptive policy with a DQN.
- DQN Defender: The attacker is a supervised DNN selector (fixed after training) and the defender learns a DQN policy for classification.

In supervised configurations, the attacker's DNN acts as a selector over A_A , choosing the specific attack, and the defender's DNN acts as a classifier over A_D , choosing the type of attack. In reinforcement learning configurations, the DQN output dimension matches the corresponding agent action space. Our experimental design is described in Fig. 3.

5) Supervised learning baselines (DNN): For DNN agents, we train a feed-forward multilayer perceptron on the training split using categorical cross-entropy loss and the Adam optimizer. We employ standard regularization and early stopping based on a validation subset carved from the training fold to prevent overfitting. After training, the resulting model is used as a deterministic policy, with actions selected using argmax over predicted class probabilities, during interaction and test-fold evaluation.

6) Reinforcement learning agents (DQN): Reinforcement learning agents (DQN). For DQN agents, each agent maintains an online Q-network, a target network, and an experience replay buffer. The DQN uses the Adam optimizer with a learning rate of 1×10^{-4} , with the discount factor, $\gamma = 0.001$, a replay-buffer capacity of 5,000 transitions, and replay mini-batches of 256 samples. The Q-network uses 128 hidden units with ReLU activation, and the target network is synchronized every 100 optimization updates. Exploration follows an ϵ -greedy policy initialized at $\epsilon = 1.0$, with a minimum value of 0.10 and decay coefficient of 0.995 applied at 100-action intervals. Each episode contains 100 interaction steps. The attacker-DQN configuration was trained for 50 episodes (5,000 interaction steps), and the defender-DQN configuration was trained for 50 episodes (5,000 interaction steps).

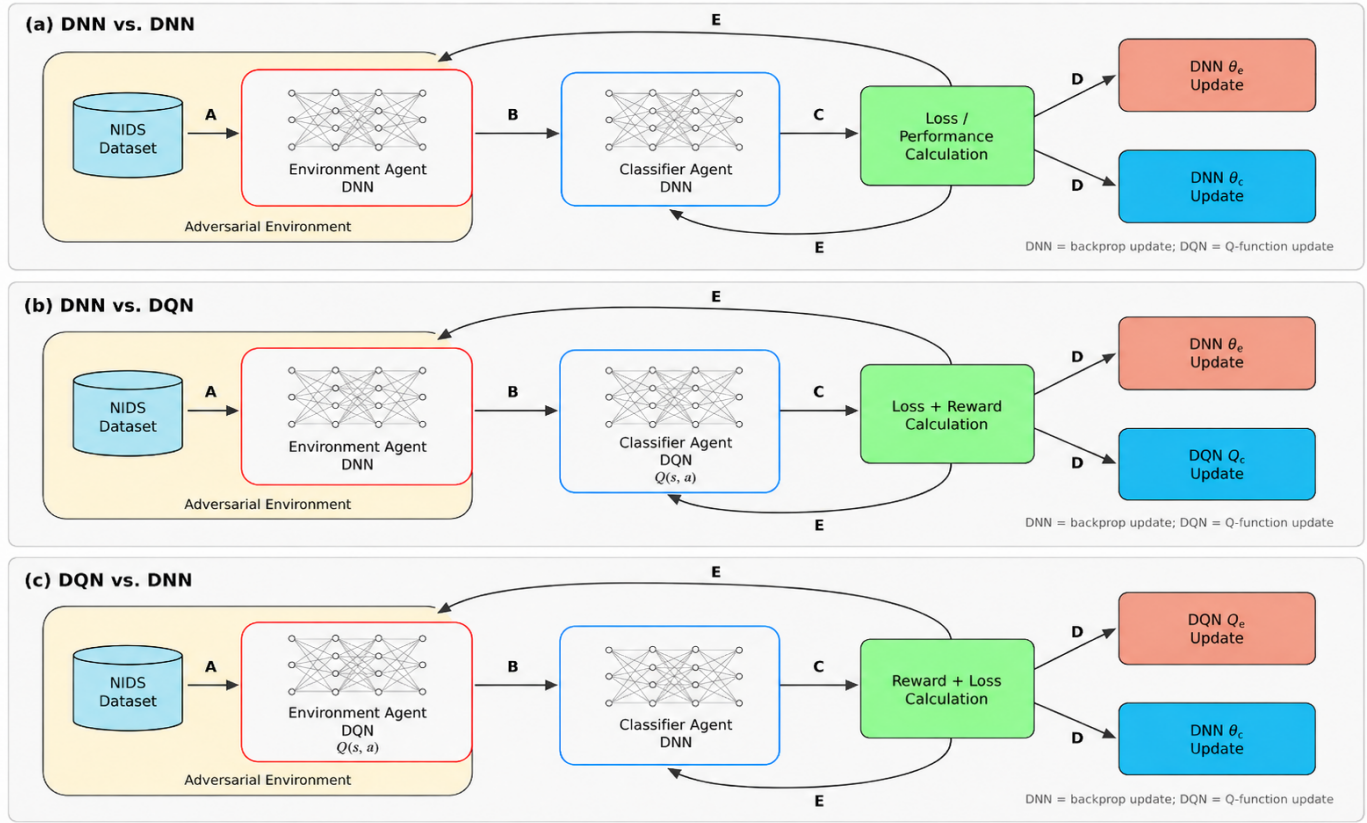


Fig. 3. Overview of the proposed adversarial-learning framework under three model pairings: (a) DNN versus DNN, (b) DNN versus DQN, and (c) DQN versus DNN. In each configuration, the adversarial agent samples network traffic from the networking dataset in step A and forwards the selected or modified traffic instance to the classifier agent in step B. The classifier agent then performs traffic classification or detection, and the resulting performance feedback is used to compute the relevant loss, reward, or combined feedback signal in step C. In step D, the feedback is routed to the appropriate update mechanism: DNN-based agents are updated through parameter optimization, while DQN-based agents are updated through their respective Q-function. The updated parameters or policies are then returned to the corresponding agents in step E, allowing the adversarial and classifier agent to continue learning across recurrent iterations. Figure adapted from [39].

7) *Evaluation metrics and statistical testing:* After training, we evaluate the defender on the held-out test fold and compute accuracy, precision, recall, F1-score, and G-Mean as defined in Section IV-A. Metrics are averaged across folds and reported with standard deviations. To assess whether observed differences between pairings are robust, we additionally apply paired statistical t-tests at an $\alpha = 0.05$ comparing the per-fold metric values of each method against alternatives, and we annotate statistically significant improvements in our result tables.

IV. NUMERICAL ANALYSIS

A. Performance Metrics

To evaluate the effectiveness of our approach, we report accuracy, precision, recall, F1-score, and geometric mean (G-Mean). Accuracy is computed as $\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$, while the F1-score is defined as $F1 = \frac{TP}{TP+\frac{1}{2}(FP+FN)}$. Precision and recall are given by $\text{Precision} = \frac{TP}{TP+FP}$ and $\text{Recall} = \frac{TP}{TP+FN}$, respectively, and G-Mean is calculated as $\text{G-Mean} = \sqrt{\frac{TP}{TP+FN} \cdot \frac{TN}{FP+TN}}$. For multiclass experiments, precision, recall, F1-score, and G-Mean

are computed using macro averaging across the defender's coarse classes, while accuracy is computed globally across all observations. All experiments are executed on a workstation equipped with an Intel Core i7 CPU and 16 GB of RAM.

B. Traditional Local Area Network Scenario

TABLE I. PERFORMANCE COMPARISON ACROSS DEFENDER/ATTACKER PAIRINGS USING CROSS-VALIDATION FOR THE TRADITIONAL LAN SCENARIO USING THE CIC-IDS 2017 DATASET. THE BEST RESULTS PER METRIC ARE BOLDED. STATISTICALLY SIGNIFICANT IMPROVEMENTS AT $\alpha = 0.05$ (PAIRED T-TEST ACROSS FOLDS VERSUS THE OTHER TWO MODELS) ARE DENOTED WITH AN ASTERISK (*)

Model	Accuracy	Recall	Precision	F1-Score	G-Mean
Both DNNs	0.9436 (0.0165)	0.4272 (0.0629)	0.4149 (0.0995)	0.4065 (0.0604)	0.6336 (0.0744)
DQN Defender	0.9668 (0.0037)	0.5287 (0.0105)	0.4915 (0.0601)	0.4865 (0.0211)	0.6961 (0.0425)
DNN Defender	0.9828* (0.0090)	0.8168* (0.0762)	0.6539* (0.0696)	0.6775* (0.0682)	0.8048* (0.0456)

Table I provides an overview of the performance metrics achieved when both defender and attacker utilize a DNN, when the defender uses a DQN, and attacker uses a DNN,

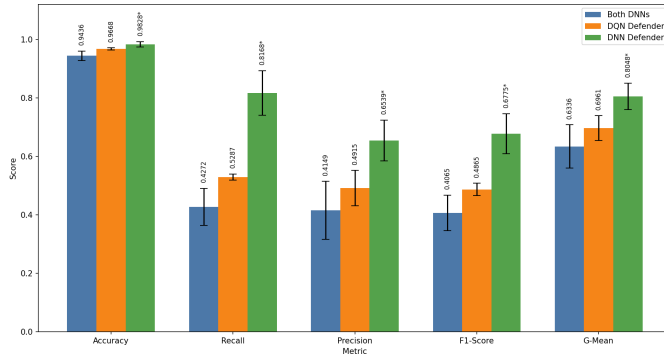


Fig. 4. Bar chart of the results of the model pairings in the LAN scenario when using the CIC-IDS 2017 dataset. Statistically significant results are indicated with an asterisk (*).

and when the defender uses a DNN and the attacker uses a DQN for the traditional LAN scenario using the CIC-IDS 2017 dataset. Across all metrics in the CIC-IDS 2017 LAN scenario, the **DNN Defender** pairing achieved the strongest mean performance, statistically outperforming other models with respect to all of the performance metrics. Notably, we observe that the DNN defender statistically outperformed the other attacker-defender pairings by over 10% in metrics other than accuracy. The largest differences were seen in the recall class; this observation suggests having a DNN defender and DQN attacker yielded a lower false negative rate compared to the other models. Our results are depicted in Fig. 4.

C. ICS/SCADA Scenario

TABLE II. PERFORMANCE COMPARISON ACROSS DEFENDER/ATTACKER PAIRINGS USING CROSS-VALIDATION FOR THE ICS/SCADA SCENARIO USING THE DNP3 DATASET. THE BEST RESULTS PER METRIC ARE BOLDED. STATISTICALLY SIGNIFICANT IMPROVEMENTS AT $\alpha = 0.05$ (PAIRED T-TEST ACROSS FOLDS VERSUS THE OTHER TWO MODELS) ARE DENOTED WITH AN ASTERISK (*)

Model	Accuracy	Recall	Precision	F1-Score	G-Mean
Both DNNs	0.9481 (0.0000)	0.9693 (0.0007)	0.9052 (0.0001)	0.9195 (0.0001)	0.9452 (0.0001)
DQN Defender	0.9474 (0.0021)	0.9656 (0.0049)	0.9039 (0.0036)	0.9175 (0.0029)	0.9446 (0.0022)
DNN Defender	0.9477 (0.0017)	0.9635 (0.0024)	0.9044 (0.0031)	0.9172 (0.0011)	0.9450 (0.0020)

Table II provides an overview of the performance metrics achieved when both defender and attacker utilize a DNN, when the defender uses a DQN, and attacker uses a DNN, and when the defender uses a DNN and the attacker uses a DQN for the ICS/SCADA scenario using the DNP3 dataset. In the ICS/SCADA scenario using the DNP3 dataset, having **both DNNs** for the attacker and defender produced the best mean performance across all reported metrics. Importantly, all three configurations exhibit extremely low fold-to-fold variability; standard deviations ≤ 0.0049 across metrics, indicating that DNP3 is a comparatively stable dataset in which supervised decision boundaries already capture most of the separability available in the features. While having **both DNNs** provided the best performance; the performance was not statistically significant. Our results are depicted in Fig. 5.

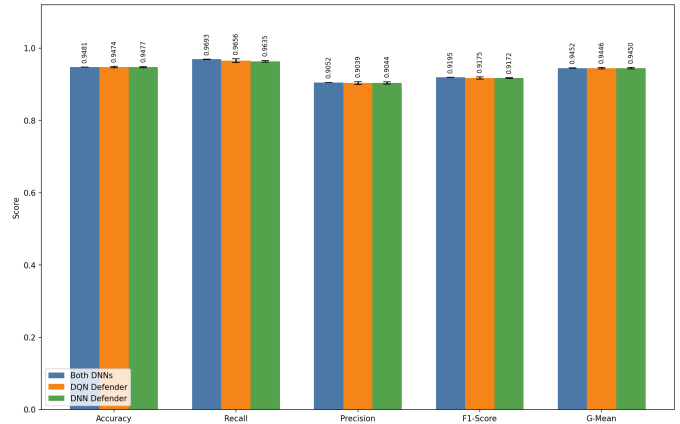


Fig. 5. Bar chart of the results of the model pairings in the ICS/SCADA scenario when using the DNP3 dataset. Statistically significant results are indicated with an asterisk (*).

D. IoT Network Scenario

TABLE III. PERFORMANCE COMPARISON ACROSS DEFENDER/ATTACKER PAIRINGS USING CROSS-VALIDATION FOR THE IOT ENVIRONMENT SCENARIO USING THE ACI-IoT DATASET. THE BEST RESULTS PER METRIC ARE BOLDED. STATISTICALLY SIGNIFICANT IMPROVEMENTS AT $\alpha = 0.05$ (PAIRED T-TEST ACROSS FOLDS VERSUS THE OTHER TWO MODELS) ARE DENOTED WITH AN ASTERISK (*)

Model	Accuracy	Recall	Precision	F1-Score	G-Mean
Both DNNs	0.9494 (0.0154)	0.5662 (0.0101)	0.5665 (0.0126)	0.5659 (0.0122)	0.7470 (0.0100)
DQN Defender	0.8360 (0.3185)	0.6298 (0.3211)	0.6621 (0.1605)	0.6067 (0.2958)	0.7928 (0.1326)
DNN Defender	0.9906 (0.0066)	0.7898 (0.0061)	0.7896 (0.0100)	0.7896 (0.0080)	0.8872 (0.0065)

Table III provides an overview of the performance metrics achieved when both defender and attacker utilize a DNN, when the defender uses a DQN, and attacker uses a DNN, and when the defender uses a DNN and the attacker uses a DQN for the IoT environment scenario using the ACI-IoT 2023 dataset. For the ACI-IoT network scenario, having a **DNN defender** with a DQN attacker achieved the strongest mean performance across all metrics with respect to traffic classification for the defender. The results of the **DNN defender** remain highly stable across folds, with the standard deviations ≤ 0.0100 for all metrics, suggesting more reliable generalization in this heterogeneous and imbalanced IoT setting. Results were not statistically significant, however. Our results are depicted in Fig. 6.

V. DISCUSSION

In the traditional LAN setting (CIC-IDS 2017), the DNN defender configuration achieved the best overall performance and produced statistically significant improvements across all reported metrics. The largest gains occurred for recall and F1-score, indicating that this pairing was most effective at detecting attacks rather than merely achieving high overall accuracy in an imbalanced setting. This suggests that, in LAN traffic where the feature space is rich but the attack classes can be sparse, a strong supervised decision boundary for the

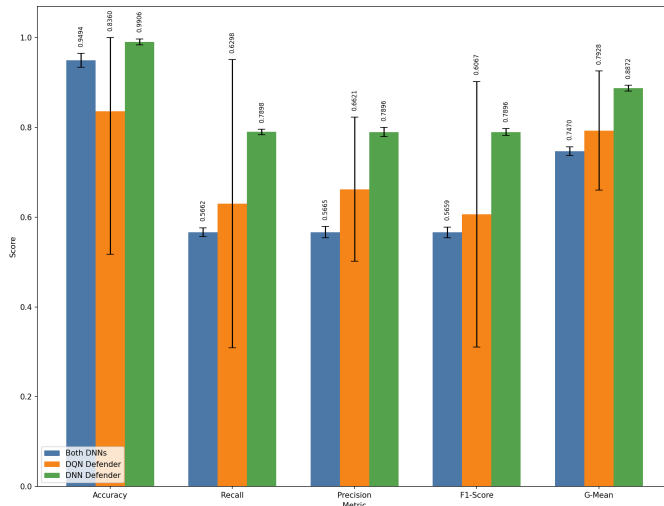


Fig. 6. Bar chart of the results of the model pairings in the IoT environment scenario when using the ACI-IoT dataset. Statistically significant results are indicated with an asterisk (*).

defender can remain robust even when the attacker adaptively influences the mixture of encountered samples.

In the ICS/SCADA setting (DNP3), having both defender and attacker utilize a normal DNN configuration produced the strongest and most stable results across every metric. Unlike the LAN and IoT datasets, DNP3 is substantially smaller and more controlled, and we used the balanced splits provided with the dataset. Under these conditions, supervised learning appears sufficient to capture the discriminative structure of the data, while reinforcement learning-based interaction provides limited additional benefit.

In the IoT setting (ACI-IoT), having a defender DNN in conjunction with an attacker DQN again achieved the best results and remained consistently stable across folds. In contrast, the reversal of the configuration in which the defender utilized a DQN and the attacker used a DNN displayed very high fold-to-fold variance. Our DQN defender failed to consistently learn an optimal policy for detecting network threats, resulting in substantial fold-to-fold variability.

A central finding across these experiments is therefore that reinforcement learning is not uniformly advantageous simply because it provides an adaptive learning mechanism. Instead, its effect depends on the role in which it is deployed and on the characteristics of the underlying network environment. In the CIC-IDS 2017 and ACI-IoT scenarios, the strongest mean performance was obtained when reinforcement learning was assigned to the attacker while the defender retained a supervised DNN, whereas the symmetric DNN configuration performed best in the more controlled DNP3 environment. These results suggest that the interaction among attacker capability, defender learning mechanism, and traffic heterogeneity can be as important as the selection of the classifier itself when evaluating intelligent NIDS.

We do acknowledge certain limitations with our work. The attacker controls the mixture of existing labeled traffic rather than generating new feature-space perturbations, and the evaluation does not include random, uniform, empirical-

frequency, or hardest-class selection baselines. Consequently, some observed attacker advantage may reflect repeated selection of naturally difficult attack classes. Stratified flow-level cross-validation does not guarantee separation of records originating from the same sessions, hosts, or capture periods; the results therefore demonstrate flow-level rather than full temporal generalization. Despite these limitations, the proposed framework provides a useful basis for evaluating how asymmetric attacker-defender learning strategies influence intrusion-detection performance across heterogeneous network environments.

VI. CONCLUSION

Our work introduced an adversarial asymmetric learning framework for network intrusion detection that explicitly models an adaptive attacker alongside a defender, enabling controlled comparison of symmetric and asymmetric learning configurations. We formulated the interaction as a two-agent stochastic game in which the attacker influences the traffic mixture through fine-grained attack selection while the defender performs coarse-grained flow classification, and we evaluated DNN- and DQN-based agents under stratified 10-fold cross validation across three representative network environments: CIC-IDS 2017 (LAN), DNP3 (ICS/SCADA), and ACI-IoT (IoT).

Across experiments, we found that the best-performing pairing depends on the dataset regime. In the LAN and IoT settings, the DNN Defender configuration achieved the strongest detection performance and remained stable across folds, yielding substantial gains in sensitivity-focused metrics (recall, F1-score, and G-Mean). In the ICS/SCADA setting, the symmetric Both DNNs configuration produced the best and most consistent results, suggesting that supervised learning can be sufficient in smaller, more controlled and balanced environments. Conversely, reinforcement learning-based defenders exhibited elevated variance in heterogeneous and imbalanced settings, highlighting the importance of stability considerations when applying value-based deep reinforcement learning to IDS classification under adversarial pressure, as in the ACI-IoT dataset. We thus demonstrate that the effectiveness of adaptive learning in NIDS depends on both the role of the learning agent and the characteristics of the network environment.

Overall, the results provide two primary takeaways. First, explicitly modeling an adaptive attacker can materially change measured IDS performance relative to static evaluation, particularly for minority-class detection. Second, supervised defenders remain highly competitive under adversarial traffic-mixture control, while reinforcement learning-based defenders may require additional stabilization and richer reward models to realize consistent benefits. Several extensions can strengthen the realism and utility of this framework. A first direction is to incorporate cost-sensitive or asymmetric utilities such as including heavier penalties for false positives in high-volume IoT traffic or higher rewards for detecting high-impact ICS attacks. A second direction is to model temporality and partial observability more explicitly so that reinforcement learning can exploit genuine sequential structure. A third direction is to expand the adversary model beyond mixture control to include constrained feature-space perturbations or traffic

synthesis, bridging this framework with adversarial machine learning methods.

DECLARATION OF GENERATIVE AI AND AI-ASSISTED TECHNOLOGIES IN THE MANUSCRIPT PREPARATION PROCESS

During the preparation of this work, the author utilized the generative AI service ChatGPT version 5.5 in order to improve the language and readability of the content. After using this service, the author reviewed and edited the content as needed and takes full responsibility for the content of the published article.

REFERENCES

- [1] P. Sharikov, "Contemporary cybersecurity challenges," in *The Implications of Emerging Technologies in the Euro-Atlantic Space: Views from the Younger Generation Leaders Network*. Springer, 2023, pp. 143–157.
- [2] R. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, 3rd ed. Wiley, 2020.
- [3] O. H. Abdulganiyu, T. Ait Tchakoucht, and Y. K. Saheed, "A systematic literature review for network intrusion detection system (ids)," *International journal of information security*, vol. 22, no. 5, pp. 1125–1162, 2023.
- [4] H. Xu, S. Wang, N. Li, K. Wang, Y. Zhao, K. Chen, T. Yu, Y. Liu, and H. Wang, "Large language models for cyber security: A systematic literature review," *ACM Trans. Softw. Eng. Methodol.*, Sep. 2025. [Online]. Available: <https://doi.org/10.1145/3769676>
- [5] H. Alqahtani and G. Kumar, "A comprehensive review of generative AI techniques and their impact on cybersecurity," *Soft Computing*, vol. 29, no. 13–14, pp. 4945–4982, 2025.
- [6] A. M. K. Adawadkar and N. Kulkarni, "Cyber-security and reinforcement learning—a brief survey," *Engineering Applications of Artificial Intelligence*, vol. 114, p. 105116, 2022.
- [7] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani *et al.*, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *ICISSp*, vol. 1, no. 2018, pp. 108–116, 2018.
- [8] V. Kelli, P. Radoglou-Grammatikis, A. Sesis, T. Lagkas, E. Fountoukidis, E. Kafetzakis, I. Giannoulakis, and P. Sarigiannidis, "Attacking and defending dnp3 ics/scada systems," in *2022 18th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 2022, pp. 183–190.
- [9] E. A. Nack, M. C. McKenzie, and N. D. Bastian, "Aci-iot-2023: A robust dataset for internet of things network security analysis," in *MILCOM 2024-2024 IEEE Military Communications Conference (MILCOM)*. IEEE, 2024, pp. 1–6.
- [10] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.
- [11] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41 525–41 550, 2019.
- [12] L. Li, Y. Lu, G. Yang, and X. Yan, "End-to-end network intrusion detection based on contrastive learning," *Sensors*, vol. 24, no. 7, p. 2122, 2024.
- [13] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*. Ieee, 2009, pp. 1–6.
- [14] A. Thakkar and R. Lohiya, "A review of the advancement in intrusion detection datasets," *Procedia Computer Science*, vol. 167, pp. 636–645, 2020.
- [15] H. Hindy, D. Brosset, E. Bayne, A. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, "A taxonomy of network threats and the effect of current datasets on intrusion detection systems," *IEEE Access*, vol. 8, pp. 104 650–104 675, 2020.
- [16] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 military communications and information systems conference (MilCIS)*. IEEE, 2015, pp. 1–6.
- [17] H. Alavizadeh, H. Alavizadeh, and J. Jang-Jaccard, "Deep Q-learning based reinforcement learning approach for network intrusion detection," *Computers*, vol. 11, no. 3, p. 41, 2022.
- [18] K. Ren, Y. Zeng, Z. Cao, and Y. Zhang, "ID-RDRL: A deep reinforcement learning-based feature selection intrusion detection model," *Scientific Reports*, vol. 12, p. 15370, 2022.
- [19] M. A. Hossain, "Deep Q-learning intrusion detection system (DQ-IDS): A novel reinforcement learning approach for adaptive and self-learning cybersecurity," *ICT Express*, vol. 11, no. 5, pp. 875–880, 2025.
- [20] R. Kanimozhi and P. S. Ramesh, "Deep reinforcement learning-based intrusion detection scheme for software-defined networking," *Scientific Reports*, vol. 15, p. 38827, 2025.
- [21] C. Rookard, "Spider: Space satellite intrusion detection using explainable reinforcement learning for next-generation satellite communication networks," in *Telecom*, vol. 7, no. 1. MDPI, 2026, p. 3.
- [22] J. F. Cevallos Moreno, A. Rizzardi, S. Sicari, and A. Coen-Porisini, "Deep reinforcement learning for intrusion detection in internet of things: Best practices, lessons learnt, and open challenges," *Computer Networks*, vol. 236, p. 110016, 2023.
- [23] A. Gueriani, H. Kheddar, and A. C. Mazari, "Deep reinforcement learning for intrusion detection in IoT: A survey," arXiv preprint, 2024.
- [24] W. Yang, A. Acuto, Y. Zhou, and D. Wojtczak, "A survey for deep reinforcement learning based network intrusion detection," arXiv preprint, 2024.
- [25] G. Caminero, M. Lopez-Martin, and B. Carro, "Adversarial environment reinforcement learning algorithm for intrusion detection," *Computer Networks*, vol. 159, pp. 96–109, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1389128618311216>
- [26] K. Hammar and R. Stadler, "Finding effective security strategies through reinforcement learning and self-play," arXiv preprint, 2020.
- [27] R. Elderman, L. J. J. Pater, A. S. Thie, M. M. Drugan, and M. M. Wiering, "Adversarial reinforcement learning in a cyber security simulation," in *Proceedings of the 9th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART, INSTICC*. SciTePress, 2017, pp. 559–566.
- [28] K. He, D. Kim, and M. R. Asghar, "Adversarial machine learning for network intrusion detection systems: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 538–566, 2023.
- [29] H. A. Alatwi and C. Morisset, "Adversarial machine learning in network intrusion detection domain: A systematic review," arXiv preprint, 2021.
- [30] J. Vitorino, I. Praça, and E. Maia, "SoK: Realistic adversarial attacks and defenses for intelligent network intrusion detection," *Computers & Security*, vol. 134, p. 103433, 2023.
- [31] R. Sheatsley, N. Papernot, M. J. Weisman, G. Verma, and P. McDaniel, "Adversarial examples for network intrusion detection systems," *Journal of Computer Security*, vol. 30, no. 5, pp. 727–752, 2022.
- [32] B.-E. Zolbayar, R. Sheatsley, P. McDaniel, M. J. Weisman, S. Zhu, S. Zhu, and S. Krishnamurthy, "Generating practical adversarial network traffic flows using NIDSGAN," arXiv preprint, 2022.
- [33] I. Debicha, B. Cochez, T. Kenaza, T. Debatty, J.-M. Dricot, and W. Mees, "Adv-bot: Realistic adversarial botnet attacks against network intrusion detection systems," *Computers & Security*, vol. 129, p. 103176, 2023.
- [34] C. Rookard, "Evaluating adversarial resilience of deep reinforcement learning algorithms for network intrusion detection," in *2026 IEEE 5th International Conference on AI in Cybersecurity (ICAIC)*, 2026, pp. 1–6.
- [35] A. McCarthy, E. Ghadafi, P. Andriotis, and P. Legg, "Defending against adversarial machine learning attacks using hierarchical learning: A case study on network traffic attack classification," *Journal of Information Security and Applications*, vol. 72, p. 103398, 2023.
- [36] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, ser. Wiley Series in Probability and Statistics. Wiley, 1994.

- [37] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA: MIT Press, 2018. [Online]. Available: <http://incompleteideas.net/book/the-book-2nd.html>
- [38] 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.
- [39] C. Rookard and A. Khojandi, "Rriot: Recurrent reinforcement learning for cyber threat detection on iot devices," *Computers & Security*, vol. 140, p. 103786, 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167404824000877>