The ISC Int'l Journal of Information Security
ISeCure

January 2026, Volume 19, Number 1 (pp. 1–15)
http://www.isecure-journal.org

Risk-Aware Intrusion Detection via Attack Dependency Modeling in Home Robots

Mohammadreza Shahlaei [1], Seyyed Mohsen Hashemi [1,*], and Ali Movaghar [2]
[1] *Department of Computer Engineering, SR.C., Islamic Azad University, Tehran, Iran*
[2] *Department of Computer Engineering, Sharif University of Technology, Tehran, Iran*

ARTICLE INFO.

Article history:
Received:
Revised:
Accepted:
Published Online:

Keywords:
attack dependency, ensemble learning, intrusion detection, home robots, risk-aware, self-attention

Type:

doi:

ABSTRACT

With the growing adoption of home robots, securing them via efficient intrusion detection systems (IDS) is increasingly vital. To address the energy and computational constraints of robotic platforms, we adopt a distributed IDS architecture: a resource-rich remote component performs in-depth detection and analysis of operating system logs and network traffic using deep learning. Meanwhile, a lightweight component on the robot controller enables rapid detection. Remotely estimated attack probabilities are fed into a risk calculator, which then guides a compact ensemble model running on the controller for real-time, risk-based detection. This approach is driven by the need to achieve higher detection performance on the resource-constrained controller. Rather than relying solely on attack frequency, we enhance detection performance by incorporating self-attention mechanisms to capture dependencies and contextual relationships among attacks. Evaluated on a custom home robot dataset, our approach demonstrates that modeling attack interdependencies significantly improves detection performance over frequency-only methods. The resulting system achieves high performance without compromising responsiveness, proving that attention-enhanced, risk-aware detection is both feasible and effective in resource-constrained robotic environments.

1 Introduction

The deployment of home robots is expanding rapidly. This growth highlights the need for security mechanisms that can achieve optimal performance across the critical dimensions of cybersecurity, safety, and privacy simultaneously.

Home robots face a wide range of security threats — including privacy violations, active and passive eavesdropping, network breaches, vandalism, and spying — as well as common robotics threats that affect the integrity, confidentiality, and availability of robotic systems. Vulnerabilities have been identified in interface applications, remote code execution, status checks, internet connections, and webcams. Furthermore, risk assessment in this domain involves evaluating threats — such as user command manipulation, data tampering, denial-of-service attacks, and configuration file corruption — based on various risk-related factors (e.g., likelihood, impact, exploitability). A comprehensive discussion of all these threats, vulnerabilities, and risk-related factors is provided in [1].

* Corresponding author.
Email addresses: m.shahlaei@iau.ac.ir, hashem@srbiau.ac.ir, movaghar@sharif.edu
ISSN: 2008-2045

Given that home robots operate in uncertain environments, they face varying levels of attack and greater safety challenges than those faced by industrial robots [2]. Consequently, ensuring the security of home robots requires different considerations.

One of the most important security mechanisms for home robots is intrusion detection systems (IDSs), which must consider both speed and detection performance. Speed ensures timely detection with minimal latency, while detection performance, measured by metrics such as accuracy, precision, and recall, determines how effectively the IDS distinguishes benign from malicious behavior.

In home robots, we also have limited computing power and energy. Leveraging a home network alongside a distributed system that enables high-performance detection within a more powerful infrastructure is a viable solution for addressing energy and computational constraints in home robots.

The present work specifically addresses the challenge of improving detection performance in distributed IDSs for home robots. One representative implementation of such a distributed architecture is presented in[3]. This architecture explicitly addresses the computational and energy constraints of robot controllers by offloading resource-intensive learning tasks to a remote computer within the home network. The remote computer detection component provides high-performance detection using deep learning. On the robot controller, however, while high-speed intrusion detection is naturally achieved due to its proximity to the core asset, high detection performance is also required. To provide this performance, a detection component based on a risk-guided ensemble model is implemented on the controller itself — all while respecting its computational and energy constraints.

We consider two parameters in the risk calculation formula [4]: the probability and impact of an attack. The impact of an attack depends on several security-related factors, including assets, vulnerabilities, and the security goals associated with those assets. However, the probability of an attack can be estimated based on the number of observed attacks. The attack probability, derived from a deep learning model on a remote computer, is used as an input to the risk model in the IDS presented in [3]. This risk model then serves as the basis for creating an ensemble learning model. Notably, the ensemble model is created on the remote computer, while its evaluation is performed only on the robot controller. This design accommodates the computational and energy constraints of home robots.

Estimating the probability based solely on the number of attacks may not accurately reflect the actual likelihood of an attack. Therefore, incorporating self-attention to capture dependencies among attacks and correlations among records can improve the accuracy of the probabilistic model.

The key challenge addressed in the present works is improving detection performance in both the remote computer and robot controller components. Estimating the attack probability solely from attack frequency fails to account for attack dependencies and limits the performance of both components. To address this challenge, we employ self-attention to model dependencies among attacks. Using self-attention improves detection performance on the remote computer. This leads to a better probability model for attacks and more accurate risk assessment. A more accurate risk assessment, in turn, enables more effective intrusion detection on the robot controller, thereby improving the overall detection performance of the IDS for home robots.

The contributions presented in this paper are as follows:

- Analyzing dependencies among attacks and their impact on intrusion detection accuracy.
- Proposing a self-attention-based model to capture attack dependencies in distributed intrusion detection.
- Demonstrating the feasibility of self-attention on low-energy, computationally constrained devices in a distributed detection framework.

In this introduction, we reviewed risk-based distributed intrusion detection in home robots and discussed how the self-attention mechanism can improve attack probability modeling by capturing inter-attack dependencies. This improvement enhances detection performance, which is the main topic of interest of this paper. We also presented the challenges, proposed solutions, and contributions of this study.

The remainder of this paper is organized as follows. The next section reviews related work and identifies the research gaps that this study aims to address. We then present the IDS architecture and methods. Subsequently, we evaluate the proposed approach. Finally, we present a discussion and conclusion on the findings.

2 Related Work

In the related work section, we first examine security in home robots, focusing on intrusion detection and distributed, prioritized, and risk-based detection methods. The second part focuses specifically on the relationships between attacks and self-attention mech-

anisms.

2.1 Home Robot IDS

Given the expansion of home robots, it is essential to develop an IDS for them. However, the absence of such a system in home robots represents a significant gap. In this context, intrusion detection has been explored in other systems, with several examples found in similar environments such as robots [5] and smart homes [6]. However, these systems do not simultaneously address safety, computational constraints, and privacy concerns, limiting their suitability for home robots.

Subsequently, it is necessary to review intrusion detection methods. Signature-based methods generally focus on attack entities (typically network packets). For home robots, such signatures are generally unavailable, and signature-based methods typically fail to detect new attacks [7]. Another category of methods is anomaly-based approaches, which consider both the system entities and network packets (as attack entities), using anomalous behavior and status as the basis for intrusion detection. Specification-based methods are a subset of anomaly-based approaches that rely on predefined rules or models to describe expected system behavior. They are highly sensitive to deviations, flagging any divergence from the specified operational norms as an anomaly or potential error. The drawback of these methods lies in the complexity of their mathematical models, as well as their limited dynamic capabilities and update potential [8].

Machine learning-based methods have emerged as probabilistic approaches for detecting conditions associated with attack states or behaviors. These methods have received considerable attention in this area. Compared to signature-based methods, machine learning-based approaches can detect unknown attacks. Moreover, they offer greater flexibility than specification-based approaches.

However, not all machine learning approaches perform equally well; recent studies show a‘ clear performance gap between classical machine learning and deep learning methods. Recent internet of thing (IoT) intrusion detection research demonstrates that deep learning models significantly outperform traditional machine learning approaches, particularly on large and complex datasets. While classical methods such as Support Vector Machine (SVM), Random Forest, and K-Nearest Neighbors (KNN) show a sharp decline in performance on more challenging IoT datasets, deep learning models — especially Long Short-Term Memory (LSTM) and biLSTM — consistently achieve high accuracy across diverse benchmark datasets, highlighting their superior capability to capture complex and temporal features in IoT intrusion detection [9].

Another key aspect of the related work is the implementation of a distributed system to address computational limitations. For example, one IDS for vehicle robots has been reported to employ a distributed architecture consisting of two components: one embedded within the robot's controller and another hosted on a remote Personal Computer (PC) [5]. However, this referenced work does not actually involve two cooperative components; it simply offloads the entire detection task to the remote side. Nevertheless, it shows that shifting the detection component to a remote computer, despite a slight increase in detection latency, enables deep learning methods to achieve higher detection accuracy. However, in this distributed system, considerations for risk and detection performance improvements within the robot's controller were not addressed.

The intrusion detection component on the robot controller still faces limitations, making risk-based methods a more suitable approach. This issue has also been addressed in the context of system distribution [10]. Furthermore, adjusting this risk is highly complex and cannot be easily applied by users of home systems. In this regard, the use of recommender systems has also been considered, as seen in certain studies [11].

2.2 Attack Dependency Analysis

The attack probability model provided on the remote PC is used to calculate the risk. This model is subsequently fed as input to our ensemble model for attack detection on the robot controller. However, when a deep learning model is used, our model considers only the number of attacks observed during the training and testing periods. This limitation becomes important when dependencies exist among attacks, as attack scenarios and graphs can be used to model these dependencies and predict the probability of subsequent attacks. We need to account for these dependencies among attacks when developing the probabilistic model. Therefore, the remainder of this section reviews the related work through the following key dimensions:

- What are the relationships between various types of attacks, and what solutions are available for analyzing them?
- Does accounting for the dependency between attacks improve intrusion detection?
- Is there any evidence of the use of self-attention mechanisms in IDSs?
- Do any existing studies combine attack dependency modeling, risk assessment, and prediction

or recommendation in a single framework?

There are various relationships among different types of attacks, including logical, temporal, spatial, and protocol-based relationships. To analyze and identify these different attack relationships, methods such as graph modeling and temporal correlation analysis can be useful. The work presented in [12] demonstrates various types of modeling among attacks.

Various approaches have been proposed to analyze attack relationships. Al Musawi et al. [13] investigate how topological indicators such as modularity and assortativity influence network vulnerability under different attack scenarios, revealing structural dependencies between attacks from a connectivity perspective. Lyu et al. [14] propose a graph aggregation-based method (AGCM) that correlates multi-stage attacks by modeling abnormal flows and capturing both temporal and logical dependencies across attack steps without relying on prior knowledge.

Regarding the impact of considering attack relations on intrusion detection, two studies are noteworthy. The first study, presented in the survey by Momand et al. [15], demonstrates that recurrent architectures—specifically, Recurrent Neural Networks (RNNs), Gated Recurrent Units (GRUs), and LSTMs—can successfully model multi-stage attack scenarios by capturing temporal correlations. The second, by Oliveira et al. [16], provides a deeper sequential analysis showing that LSTM effectively learns temporal dependencies between consecutive network flows, offering a more granular perspective on sequential attack detection.

Another work [17] examines the impact of attack dependency on intrusion detection, but within a CPS environment. In this study, the authors introduced two types of neural networks: one that captures time-based dependencies between events, and another that generates rare attack samples to handle imbalanced data. Their model was tested on water treatment and water distribution datasets and achieved good detection performance. Although their work focuses on industrial water systems rather than home robots, the key idea of accounting for the sequence and timing of events to improve attack detection aligns with our approach.

While these studies focus mainly on temporal dependencies — such as sequential correlations between flows or events — we argue that intrusion detection should consider all forms of attack relations simultaneously, including temporal, logical, spatial, and protocol-based dependencies.

The use of self-attention mechanisms in IDSs has a precedent. In one study [18], self-attention was combined with convolutional neural networks (CNN) and LSTM for IDS in a cloud environment. This work aimed to overcome the one-dimensional nature of deep learning methods. Therefore, self-attention mechanisms were employed to enhance feature extraction from the CNN output. The combination of these methods achieved better performance than traditional deep learning methods.

Another significant contribution is the IDS for the Industrial Internet of Things (IIoT) that combines self-attention and CNN [19]. Another significant contribution is an IDS for the Industrial Internet of Things (IIoT) that combines self-attention and CNN [19]. This study emphasizes that, due to repetitive patterns and imbalanced training data, identifying the most informative features using self-attention mechanisms is important for deep learning methods. Subsequently, these parameters are processed by a CNN for intrusion detection.

In both studies, self-attention is used for preprocessing and feature extraction to enhance the input representation before it is passed to the primary classifier, rather than being integrated into the detection model itself.

The use of the same dataset for training and testing, along with the poor performance of deep learning-based IDSs in detecting certain attacks, is another issue highlighted in [20]. The study suggests that employing self-attention can improve intrusion detection performance. However, this paper does not discuss dependencies among attacks or the use of self-attention to model such relationships, distinguishing it from our approach.

Aswathy and Annalakshmi (2025) propose an Adversarial Resilient Intrusion Detection System (AR-IDS) for general network security [21]. Their work integrates a transformer with multi-head self-attention, adversarial training, and contrastive learning. However, while AR-IDS mentions modeling dependencies between attacks, it does not detail how self-attention captures these relationships. This distinguishes their approach from ours, which explicitly models attack dependencies using a multilayer deep learning model and dependency representation formulas.

Regarding the use of attack dependencies for attack prediction, one can mention the recommendation system proposed in [22]. In this system, an attack graph is constructed to represent dependencies among attacks. The recommendation system then uses collaborative filtering, the attack graph, and risk-based approaches to predict subsequent attacks. Although this article does not explicitly address IDSs or use machine learning for attack prediction, the sequence

of activities undertaken in this work closely resembles our approach.

Based on the questions and answers discussed, we conclude that there is a dependency between attacks, and considering these dependencies in IDS enhances detection performance. The self-attention mechanism can capture these dependencies among attacks, and examples of its application in IDSs have been reported. In contrast to prior approaches that either overlook attack dependency modeling or employ self-attention merely as a preprocessing step for input enhancement, our method embeds self-attention as a core component of the detection model. Furthermore, while previous studies fail to specify how self-attention captures attack relationships, our work explicitly formulates and learns attack dependencies through self-attention. Our approach also differs from most existing work by encompassing all forms of attack relations, rather than only temporal ones, through the use of self-attention.

The novelty lies in integrating self-attention, which can capture dependencies among attacks, into this architecture, thereby improving detection accuracy over dependency-blind baselines. Specifically, the novelty of this work lies in two key aspects: the application of the self-attention mechanism within a distributed IDS and the integration of attack dependency modeling into the self-attention mechanism, which improves the overall efficiency and performance of the IDS. These innovations differentiate our work from previous studies.

3 Methodology and Architecture

In this section, we examine the IDS architecture and analyze the algorithms it employs. The home robot IDS is deployed within a home network and designed with a distributed architecture to address the energy and computational limitations of home robots. The architecture of the home robot IDS is depicted in Figure 1.

In this architecture, the intrusion detection components are located both on a remote computer and within the home robot's controller. The integration of self-attention mechanisms into various algorithms within the intrusion detection components has significantly enhanced performance, as highlighted in Figure 1 using a distinct color for clarity.

3.1 Interaction Between Remote Computer and Robot Controller

To clarify the interaction between the Remote Computer and the Robot Controller in the proposed distributed IDS, it is important to note that intrusion detection systems generally perform three core functions: data collection, data processing, and intrusion detection. In the proposed architecture, these functions are explicitly distributed between the two components to address the computational and energy constraints of the home robot.

Data collection is performed on both the robot controller and the remote computer and includes network traffic and operating-system-level features. All collected data are transmitted to the remote computer, where the primary data processing is performed. This processing includes deep learning-based analysis, self-attention-based modeling, and risk calculation, all of which require substantial computational resources.

The self-attention mechanism is employed exclusively on the remote computer to model dependencies among features, records, and attack stages, thereby improving the estimation of attack probabilities. The risk calculator then uses these refined probabilities to construct a risk-aware ensemble model.

The resulting ensemble model is subsequently transferred to the robot controller, where the intrusion detection function is performed in real time using a lightweight evaluation process. Therefore, detection decisions at the robot controller are not independent; rather, they are directly informed by the models and risk assessments generated on the remote computer. This cooperative design enables high detection performance while respecting the robot controller's energy and computational limitations.

3.2 Remote Computer Components

Both the high-performance intrusion detection component and the risk calculator component are located on a remote computer connected to the robot controller. This remote computer performs high-performance intrusion detection, provides recommendations to the risk calculator component, and supplies the required ensemble model for the intrusion detection component within the robot controller. Additionally, it can collect data from the robot's operating system and computer network to support intrusion detection activities.

Employing deep learning techniques with multilayer perceptrons (MLPs) can achieve relatively high performance in intrusion detection. However, as explained in the introduction, different attacks are interdependent, and attackers typically execute attacks according to specific behavioral patterns, resulting in the formation of attack paths and scenarios.

A typical attack scenario follows stages such as reconnaissance, initial access, persistence, privilege escalation, lateral movement, and exfiltration. For ex-

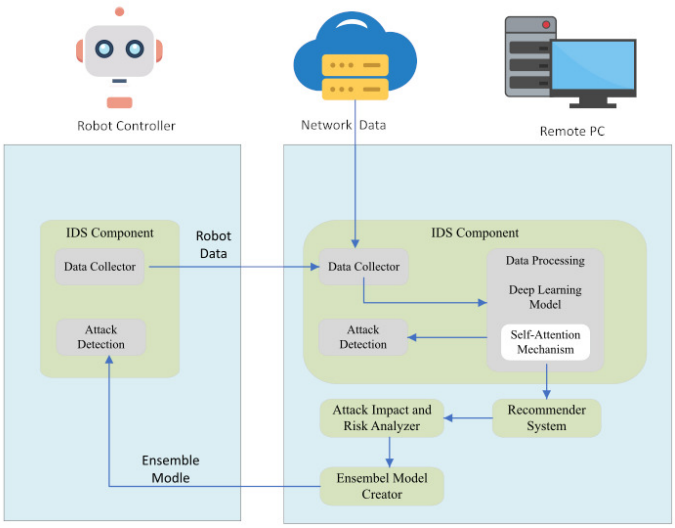


Figure 1. Architecture of the home robot IDS

ample, a scanning attack can identify vulnerabilities that are subsequently exploited through injection attacks to gain access to a system. Man-in-the-Middle (MITM) captures sensitive data (e.g., passwords), enabling brute-force access. Injection attacks embed malicious code to escalate privileges or maintain persistence. Denial-of-Service (DoS) attacks can also act as diversions by masking covert activities such as credential theft. These interwoven steps demonstrate how attackers construct effective pathways while complicating defensive efforts.

As discussed in the related work, considering data dependencies can improve intrusion detection performance. On the other hand, self-attention mechanisms account for this issue in their classification. We employ this approach to achieve higher-performance intrusion detection and to create better probabilistic models for the risk calculator component.

The self-attention mechanism possesses several advantageous features, such as the ability to process all records in parallel, uncover multiple feature correlations, and assign weights to important features. It is particularly useful when relationships between records are complex, and the data involve long sequences. This mechanism is both scalable and fast, making it suitable for spatial and temporal analyses.

To explain self-attention, let us assume the input X consists of n samples, each with d features, as follows:

$$X = \{x_1, x_2, \ldots, x_n\}, \quad x_i \in \mathbb{R}^d \tag{1}$$

Each vector x_i contains contains features extracted from network traffic or robot controller data. The goal is to identify relationships among these features and samples. To model these dependencies, we use weight matrices W_Q, W_K, W_V:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V \tag{2}$$

Here, Q represents the significance of a sample x_i in detecting anomalies, K reflects the sensitivity of each feature or sample to others, and V encodes the actual information associated with features, such as raw data. The matrix QK^T captures intra-sample relationships, which are the interactions between features within a specific data sample, and inter-sample relationships, which are the interactions between different samples in the dataset.

Implementing this mechanism requires sufficient data volume and entails high computational costs. However, with more advanced processing infrastructure than the robot controller, this mechanism can be employed for intrusion detection. Given the absence of well-documented attacks on home robots and the presence of feature correlations in the dataset, this mechanism proves applicable. In our experiment, the data was neither excessively large to hinder speed nor too small to lead to inaccurate conclusions.

The neural network architecture of the self-attention model and its comparison with deep MLP learning are shown in Figure 2. Below is a summary of the key components:

- **Dense Layers:**
  - Fully connected layers with a trainable *kernel* (weight matrix) and *bias* (offset vector).
  - Kernel dimensions reduce the feature space in each layer, e.g., 648 → 64 (self-attention MLP) or 648 → 32 (standard MLP).
- **ReLU Activation:**
  - Introduces non-linearity by mapping negative values to 0 while keeping positive values unchanged.
  - Prevents vanishing gradients and enables learning of complex patterns.
- **Self-Attention Mechanism:**
  - Computes attention weights to identify dependencies between features.
  - Includes a weight matrix and bias vector for better global context modeling.
- **Softmax Activation:**
  - Applied in the final layer to convert logits into probabilities for classification tasks.
- **Kernel and Bias:**
  - Kernel ($n \times m$): Weights connecting n input features to m output features.
  - Bias (m): Added for flexibility in fitting data.

The self-attention-enhanced MLP introduces a self-attention layer to better capture dependencies among features, while both architectures effectively combine dense layers, ReLU, and softmax for robust classification.

Algorithm 1 illustrates this process. In this algorithm, we first define a custom Keras class to implement the self-attention layer. This class is then used to build the learning model.

The increased computational complexity, extended training time, and the need for tuning specific parameters in self-attention layers can raise concerns about their use. However, since understanding attack dependencies is crucial for effective intrusion detection and improving the performance of IDS is of great importance, the use of Self-Attention layers is justified despite the associated costs.

This component collects intrusion detection information from the network as well as operating-system-level data from the home robot. Additionally, using its self-attention layers, it provides attack probability estimates to the risk calculator. These attack probabilities incorporate dependencies among attacks, and the risk assessment module can use the recommendations generated from them to evaluate the impact of the attacks.

Algorithm 1 Self-Attention Layer Algorithm

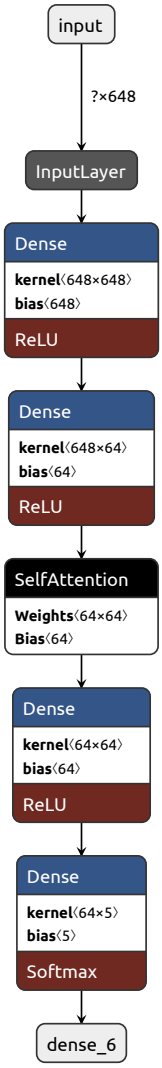
1: **1. Create Self-Attention Class with the following methods:**
2: **1.1. Set Output Dimension:**
3: `self.output_dim` ← `output_dim`
4: **1.2. Initialize Weights and Biases:**
5: `self.W` ← Weights with shape (feature_dim, output_dim)
6: `self.b` ← Biases with shape (output_dim,)
7: **1.3. Define Callable Function for Attention Calculation and Output Generation:**
8: $e \leftarrow \text{Tanh}(inputs \cdot self.W + self.b)$
9: $a \leftarrow \text{Softmax}(e, \text{axis} = 1)$
10: `output` ← $inputs \times a$
11: **1.4. Define Function to Compute Output Shape:**
12: **Return** $(input\_shape[0], input\_shape[1], self.output\_dim)$
13: **2. Integrate Class in Model Definition:**
14: Combine Self-Attention Layer with other layers during model creation

The risk calculator component considers the importance of attacks in IDSs and calculates risk by multiplying the probability of an attack by its impact [4]. In this formula, because calculating attack impact is complex and no list of assets, vulnerabilities, or attack consequences is available, we set the impact to a constant value of 1.

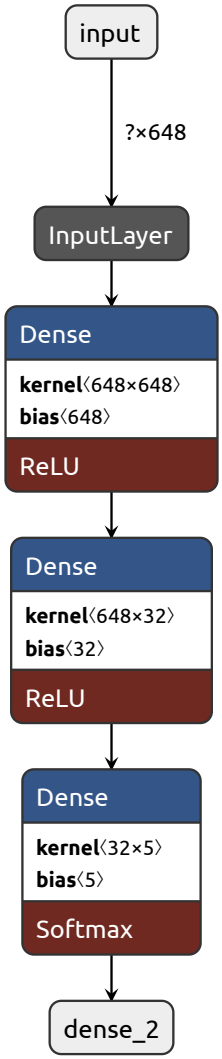
3.3 Robot Controller Component

The component integrated into the robot's controller performs rapid, risk-based intrusion detection, thereby enhancing the overall detection performance. We employ ensemble learning methods to achieve risk-aware intrusion detection. In the risk calculator component on the remote PC, the ensemble model provides the infrastructure required for implementing risk-based real-time intrusion detection in the robot controller. Initially, P_m^i represents the probability assigned to each attack type by the trained model based on a file from the dataset. Subsequently, we apply the True Positive Rate (TPR) metric. By multiplying the model's output probability by the TPR, we obtain a more realistic probability for each class. This adjusted value reflects both the probability estimated by the model and its class-specific ability to correctly identify attacks, particularly for classes with lower TPR values. We then define the probability of each attack class by averaging across all files (indexed by m) and comparing it with other attack classes (each indexed by i). Finally, the risk associated with each attack class is determined according to Formula 3.

$$r^i = e^i \times \frac{\sum_m TPR_m^i P_m^i}{\sum_j \sum_m TPR_m^j P_m^j} \tag{3}$$



(a) Neural network architecture of self-attention based learning model



(b) Neural network architecture of deep MLP learning model

Figure 2. Neural network architectures

As noted earlier, we use a constant value of one to represent the impact of each attack (e^i). This value can be adjusted when precise impact estimates are available. Moreover, the impact of the attack is considered in real-time intrusion detection, and unlike the attack probability, it is not pre-multiplied within the ensemble model created in the risk calculator component.

The self-attention-based ensemble model is constructed from multiple dataset files containing disk, Random Access Memory (RAM), Central Processing Unit (CPU), and network data. The creation of deep learning models (MLP or self-attention-based) for each of these files occurs on a remote computer. The ensemble model is then generated in the risk calculator component and transmitted to the robot controller for evaluation. Consequently, the evaluation within the robot controller is lightweight, accounting for energy and computational constraints.

4 Results and Evaluation

Through the distributed design, we overcome computational and energy constraints. Low latency is ensured by deploying a detection component directly on the robot controller. Consequently, the primary metric we focus on in this context is detection performance.

To evaluate the performance of the proposed method in this paper, we first need to define the baseline methods against which it will be compared. As discussed in the related work, IDSs from other domains are not well-suited to home robots.

Moreover, signature-based methods have limited

ability to detect unknown attacks, while specification-based methods rely on predefined rules and may lack flexibility. Classical machine learning methods have limited ability to capture complex attack patterns, whereas deep learning methods can model such complexities more effectively but typically do not explicitly consider dependencies among attacks.

This study focuses on addressing the dependency between attacks and the potential of self-attention mechanisms. Accordingly, we compare deep learning performance both without and with the self-attention mechanism, based on the same methodology and experimental setup presented in this paper. Additionally, we compare ensemble models using risk estimates derived from deep learning, with and without self-attention, to evaluate the improvement in detection performance achieved by the self-attention mechanism.

To support the evaluation of the proposed IDS, we employ the dataset introduced in [3],which was specifically developed to address the lack of realistic datasets for intrusion detection in home robotic environments. This dataset was constructed using a realistic home network and robot controller configuration and contains both normal operation and diverse attack scenarios. The data collection process captures network-level and system-level behaviors relevant to robotic platforms. Standard preprocessing steps—such as data cleaning, addressing class imbalance, and removing outliers—were applied to ensure the dataset's suitability for intrusion detection evaluation.

This dataset initially contains approximately one million records. However, the total number is reduced after preprocessing steps such as outlier removal and class balancing. The dataset includes six defined classes: DoS, MITM, Password, Injection, Scanning, and Normal traffic. The collected features are grouped into four main categories:

- Disk-related features: including disk read/write volume, disk response time, and other performance metrics.

- RAM-related features: such as the number of memory pages, required physical and virtual memory, and related memory usage indicators.

- CPU-related features: including process status, execution time, CPU utilization, and similar system-level metrics.

- Network-related features (Net): such as source and destination IP addresses and ports, communication protocol, connection state, and other network traffic characteristics.

To implement the self-attention mechanism, we used Python with the TensorFlow and Keras libraries. Additionally, we added a custom class to define the self-attention layer.

4.1 Self-Attention-Based Learning on a Remote PC

In this section, we evaluate the dataset using the architecture shown in Figure 2 along with Algorithm 1. We then compare the results with those from the same deep MLP architecture. This comparison includes both the version without self-attention and the version with self-attention.

To optimize the self-attention-based model, we experimented with different output feature vector dimensions and numbers of training epochs. Setting the dimension to 64 and the epoch count to 30 led to a notable improvement in model performance.

We used cross-entropy as the loss function to measure prediction errors. The Adam optimizer was employed for parameter adjustment, and accuracy was selected as the performance metric. ReLU was selected for its effectiveness in capturing nonlinear relationships, while Softmax was selected for its ability to convert outputs into probability distributions. Both functions were used as activation functions in the learning model.

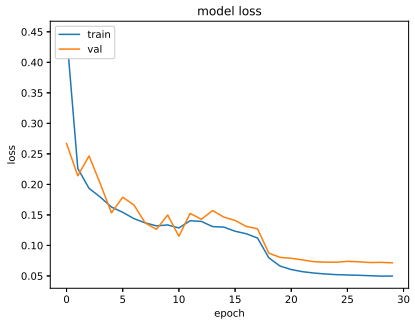
Additionally, early stopping was employed as a regularization technique to prevent overfitting, while a learning rate decay of 0.1 was introduced to improve performance when accuracy stopped improving. These configurations are summarized in Table 1.

Table 1. Configuration of the Self-Attention based Model

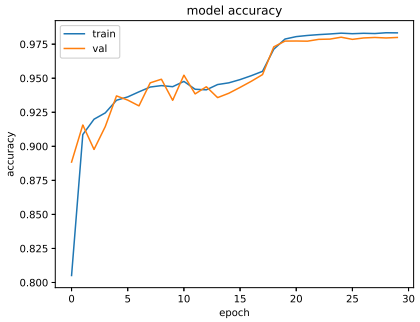
Configuration	Value
Loss Function	Cross Entropy
Performance Metric	Accuracy
Activation Functions	ReLU, Softmax
Optimizer	Adam
Epoch Size	30
Learning Rate	0.001
Learning Rate Decay	ReduceLROnPlateau (Factor = 0.1, Patience = 7, Epsilon = 1e-4)
Early Stopping	Enabled (Patience = 10, Mode = 'min')

Figure 3 shows the accuracy and loss of the self-attention-based learning model. The reduction and stabilization of training and validation loss at low values, together with the increase and stabilization of training and validation accuracy, indicate that the model was effectively trained.

The confusion matrix, a standard tool for evaluating classification performance, is also presented in Figure 4. According to this figure, the self-attention-based model generally exhibits high detection perfor-



(a) Loss of the self attention based model



(b) Accuracy of the self attention based model

Figure 3. Loss and accuracy of the deep neural network

mance, with rare misclassifications occurring primarily in the password attack class.

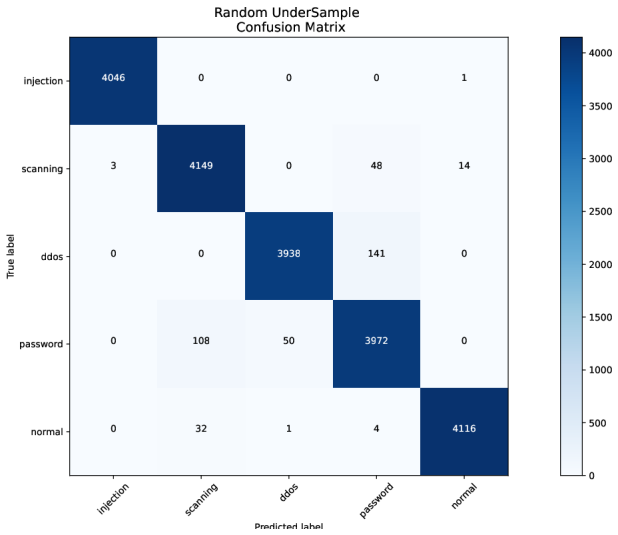


Figure 4. Self attention learning confusion matrix

Table 2 compares the baseline deep MLP intrusion detection model with an enhanced variant incorporating a self-attention mechanism. Both models are evaluated across multiple dataset files using multiple performance metrics to assess the impact of the self-attention module on detection accuracy and robustness.

For easier comparison and visual analysis, these values are also presented in Figure 5.

Table 2 presents various metrics for evaluating model performance. We used loss to measure the deviation between the predicted and actual values and accuracy to assess the proportion of correctly classified samples. To facilitate model tuning and optimization, both metrics were monitored during the training phase. To ensure generalization capability, we also evaluated their results during the evaluation phase. We focused on reducing false alarms and minimizing missed positive cases, as reflected by precision and recall, respectively. We also used the F1-score to capture the balance between these two metrics. Given the priority of minimizing missed positive cases, we employed the F-beta score, which allows us to assign greater weight to recall. Finally, we used the Jaccard similarity coefficient to assess similarity, a metric commonly used in cases of multi-class classification.

In this comparison, we aimed to demonstrate the consistent detection performance of the self-attention-based model across different files. To avoid inconclusive and imprecise conclusions, problematic datasets such as net7 were excluded. The analysis shows that on all remaining datasets, self-attention generally outperforms deep MLP learning, achieving near-optimal accuracy in most cases.

The results are summarized using average and aggregated metrics in Table 3. Based on the results in this table, the Self-Attention method achieves an average improvement of 2.9 percent in accuracy and 2.8 percent in F1-score compared to Deep MLP. The largest improvement is observed on the net8 dataset, with a 33.5 percent increase in accuracy, while on the RAM dataset, no significant improvement is achieved (less than 0.4 percent).

As shown by the results, performance improvement of self-attention over deep MLP in F1-score is positive for 11 out of 13 datasets, representing 84 percent of all cases. The largest improvement in the performance metric is observed for net8, with a 33.5 percent increase.

Only two datasets, namely RAM and net6, show negligible degradation of less than 0.5 percent. Overall, the self-attention model achieves better detection performance than MLP across most datasets, with substantial gains in several cases.

In the RAM Dataset, both methods showed similar performance with an F1-score of 0.876 for MLP versus 0.873 for self-attention. The lack of significant improvement can be attributed to the fact that RAM-related features are primarily instantaneous and do not exhibit reliable temporal dependencies. Although

Table 2. Comparison of Baseline Deep MLP and Self-Attention-Augmented Deep MLP Models

file_name	method	loss	accuracy	val_loss	val_accuracy	precision	recall	f1_score	fbeta	jaccard_score
CPU	Deep MLP	0.221	0.8841	0.2393	0.8775	0.89070172	0.875691184	0.874691956	0.884129453	0.778870691
	MLP + Self	0.2215	0.8849	0.2306	0.8808	0.98077253	0.98050720	0.98053169	0.98069341	0.96175980
DISK	Deep MLP	0.0804	0.9692	0.0936	0.9687	0.96879679	0.968433303	0.968344315	0.968614638	0.938798533
	MLP + Self	0.0498	0.9833	0.0715	0.9799	0.98467367	0.98236451	0.97987321	0.98387431	0.94268431
RAM	Deep MLP	0.2438	0.8752	0.2508	0.8808	0.891709862	0.877280859	0.87613122	0.884327388	0.78138942
	MLP + Self	0.2443	0.8738	0.2446	0.8772	0.89055002	0.87584973	0.87282399	0.88159486	0.77912157
net1	Deep MLP	0.0224	0.9989	0.0287	0.9992	0.998329588	0.99833333	0.998333339	099830979	0.99667221
	MLP + Self	0.2215	0.8849	0.2306	0.8808	0.98077253	0.98050720	0.98053169	0.980693416	0.96175980
net2	Deep MLP	0.0721	0.9851	0.0608	0.9867	0.987231622	0.986545682	0.986594247	0.986960875	0.973448595
	MLP + Self	0.0193	0.9947	0.0171	0.9949	0.99416004	0.99397684	0.99398745	0.99408850	0.98802581
net3	Deep MLP	2.6592	0.9874	0.3611	0.9909	0.990406735	0.99019806	0.990196871	0.990300711	0.98058641
	MLP + Self	1.6332	0.9773	0.0751	0.9967	0.99769354	0.99767582	0.99767579	0.99768513	0.99536243
net4	Deep MLP	2.1325	0.9847	0.0116	0.9957	0.996065574	0.996000333	0.996000132	0.996035397	0.992032534
	MLP + Self	0.2819	0.9894	0.0744	0.9901	0.98979149	0.98958420	0.98958271	0.98969052	0.97938314
net5	Deep MLP	12.633	0.9903	82.3193	0.9795	0.979627062	0.978200085	0.978234578	0.978953442	0.957330365
	MLP + Self	0.0082	0.9958	0.0103	0.9961	0.9956819528	0.99564001	0.99564001	0.99566412	0.99131788
net6	Deep MLP	1.1562	0.9951	0.0096	0.9972	0.996533638	0.996544189	0.996543686	0.996533522	0.99311218
	MLP + Self	0.2279	0.9211	0.1983	0.9276	0.93068087	0.92356558	0.92252839	0.92608609	0.85798593
net7	Deep MLP	1.93E-09	1	4.10E-09	1	1.00E+00	1	1.00E+00	1	1.00E+00
	MLP + Self	High resource consumption, solution not achievable.								
net8	Deep MLP	0.6801	0.6224	0.726	0.6338	0.767198926	0.632052882	0.531427855	0.517310907	0.462044821
	MLP + Self	0.1214	0.9573	0.0935	0.9650	0.96805591	0.96812923	0.96804016	0.96803399	0.93822721
net9	Deep MLP	0.3501	0.9393	0.1685	0.9492	0.946118526	0.945121951	0.945079951	0.945594318	0.895953757
	MLP + Self	0.2774	0.8998	0.1379	0.9740	0.98190555	0.98170731	0.98170788	0.98177146	0.96407185
net10	Deep MLP	125.1748	0.9069	91.4425	0.905	0.905562517	0.892405063	0.890762065	0.895052498	0.805714286
	MLP + Self	0.8175	0.8965	0.4580	0.9149	0.89973251	0.9034810	0.90342794	0.89957453	0.82395382

self-attention can capture various attack relationships, the temporal aspect of RAM-based attacks is not sufficiently prominent to benefit from this capability. In the net6 dataset, the self-attention model achieved lower accuracy than MLP (0.921 vs. 0.995). Further error analysis reveals that most of self-attention errors occur in the Password Attack class, suggesting that the sequential patterns for this attack type in the net6 dataset are likely highly complex and noisy.

Table 3. Summary of model performance across all datasets (excluding net7)

Metric	Deep MLP	Self-Attention	Difference
Accuracy (Mean ± Std)	0.915 ± 0.112	0.944 ± 0.048	+0.029
F1-score (Mean ± Std)	0.914 ± 0.071	0.942 ± 0.039	+0.028
Precision (Mean)	0.918	0.950	+0.032
Recall (Mean)	0.911	0.946	+0.035

To support the claim of real-time feasibility, we report quantitative runtime and memory metrics for both the baseline Deep MLP and the proposed Self-Attention model on the CPU dataset (2454 training samples). Table 4 summarizes the computational overhead.

Table 4. Computational overhead comparison between Deep MLP and Self-Attention.

Metric	Deep MLP	Self-Attention
Total parameters	137,937	155,569
Model size (KB)	538.82	607.69
Inference time (ms/step)	2	2–3
Training time (sec/epoch)	5–6	6
Best validation accuracy	0.8775	0.8808

The Self-Attention model introduces only 12.8 percent additional parameters (+17,632 params, +68.87 KB) compared to the MLP baseline. The inference

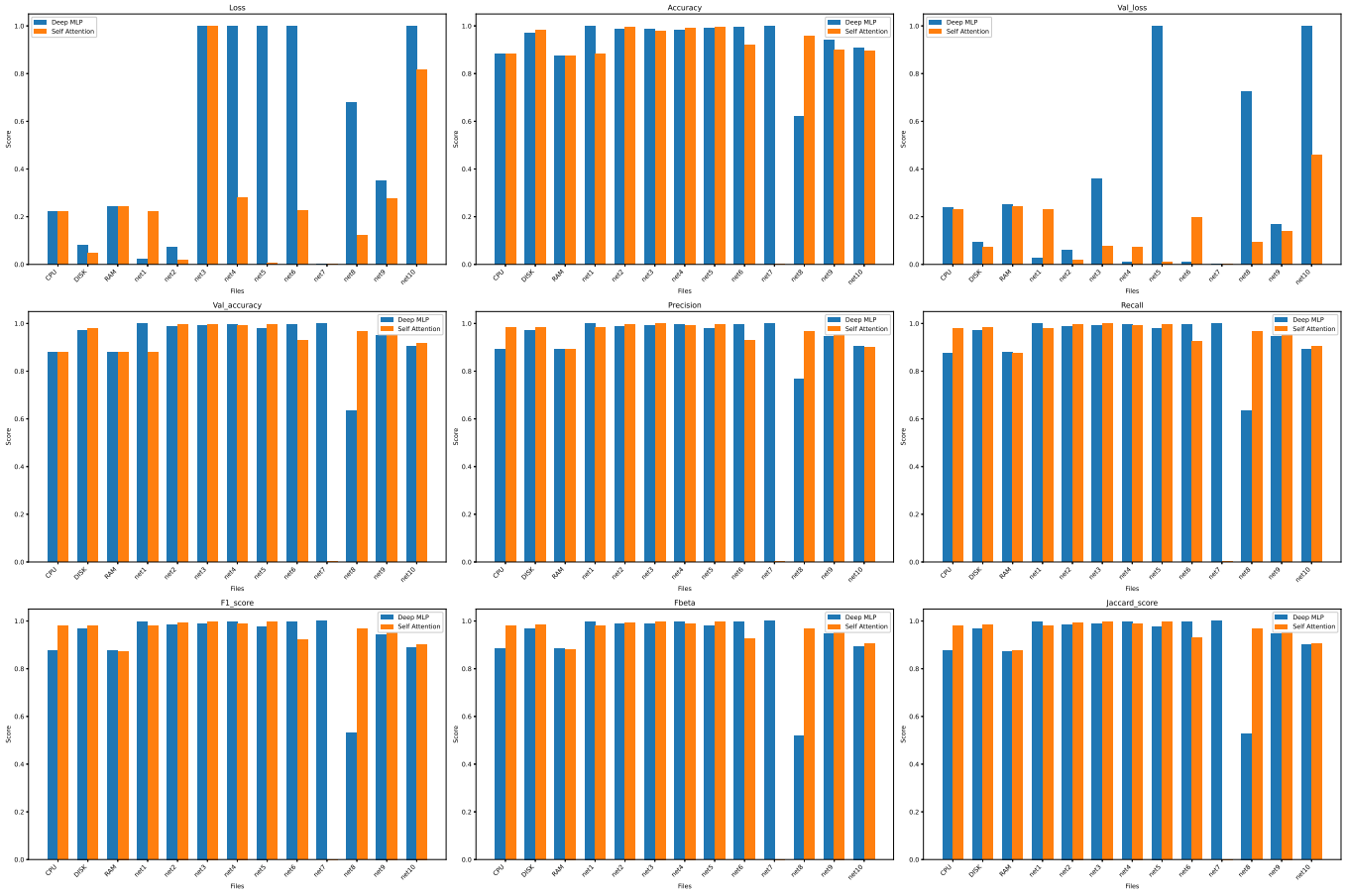


Figure 5. Comparison of Deep MLP and Self Attention mechanism based on various metrics

time ranges from 2 to 3 milliseconds per step, while the training time per epoch is approximately 6 seconds on a standard CPU. Both models operate within a memory footprint of less than 1 MB and inference latency under 10 ms. These results confirm that the proposed self-attention mechanism adds no prohibitive computational or memory overhead, and the model remains suitable for real-time applications without requiring specialized hardware.

4.2 An Ensemble Model Based on Self-Attention in Robot Controllers

In this section, we aim to demonstrate that the self-attention-based ensemble model achieves better intrusion detection performance. To this end, we compare the new ensemble model with two scenarios.

The first scenario assumes equal weight for all attacks, without considering any probabilities or risks as factors for intrusion detection. The second scenario involves an ensemble model constructed from deep MLP models that incorporates attack probabilities into the risk calculation, with the impact of each attack assigned a fixed weight of one.

The scenario considered in this paper involves assessing the risk of various attacks using a self-attention-based ensemble model. The methodology section provides details of this approach.

Table 5 compares the performance of the proposed method with that of the two initial scenarios. The column definitions in this table are similar to those in Table 2.

As As shown in Table 5, the self-attention-based ensemble model consistently outperforms both the deep MLP-based risk model and the risk-agnostic model across all evaluation metrics. The self-attention approach achieves the highest F1-score (0.9997040), compared to 0.9996003 for the deep MLP and 0.9991208 for the model without risk. While the absolute differences appear small due to the already high baseline of deep learning-based intrusion detection, the improvement is consistent and not merely statistical. As the number and diversity of attacks and features increase, the performance gap may widen further in favor of the self-attention ensemble. Overall, the results confirm that incorporating risk through a self-attention mechanism leads to more accurate and robust intrusion detection in robot controllers.

Table 5. Comparison of Ensemble Models: Risk-Agnostic, Risk-Aware Deep MLP, and Risk-Aware Self-Attention Based Approaches

Models	Precision	Recall	F1 score	F_β score	Jaccard score
Ensemble model by self-attention based risk	0.9997041	0.9997039	0.9997040	0.9997043	0.9993421
Ensemble model by deep MLP based risk	0.9996007	0.9996003	0.9996003	0.9996005	0.9992010
Ensemble model without risk	0.9991223	0.9991208	0.9991208	0.9991215	0.9982432

5 Discussion

The machine learning model with self-attention achieved better performance on both the remote PC and the robot controller. This consequently enhanced intrusion detection on the remote PC and risk-based detection on the robot controller. Overall, our approach improves the performance of the distributed IDS for home robots in terms of detection accuracy and speed while addressing energy and computational constraints. Despite the performance improvements achieved by employing the self-attention mechanism, it seems necessary to address issues such as temporal dependencies using methods like LSTM, which cannot currently be explored due to dataset limitations. Additionally, there are known causal dependencies between attacks that differ from dependencies learned automatically by machine learning methods, and these have not been investigated in this work.

The use of single-board computers in home robot controllers and the ability to install the Linux operating system and ROS middleware have become prevalent in robotics production. Running Python code and self-attention mechanism libraries on this infrastructure indicates that the implementation of the desired IDS is feasible.

Regarding scalability, cloud, fog, and edge computing architectures can enable the proposed approach to scale effectively. However, due to their unique security challenges, this aspect must be addressed separately.

Ultimately, employing the self-attention mechanism in the home robot's IDS significantly enhances its performance by capturing attack dependencies that conventional methods overlook. This dependency-aware capability improves detection performance across both the remote PC and the robot controller components, leading to a more robust and reliable distributed IDS. As a result, the overall security of the home robot is meaningfully increased, highlighting the key contribution and distinct advantage of the proposed approach over dependency-blind baselines.

6 Conclusion and Future Work

The distributed IDS of the home robot overcomes its energy and computational limitations. The component on the remote PC runs high-performance deep learning for intrusion detection and also provides attack probability recommendations. These attack probabilities are used in the IDS on the robot controller for rapid, risk-based intrusion detection employing an ensemble model. However, deep learning methods do not account for the dependencies between attacks when calculating attack probabilities. The results show that the self-attention mechanism used in this paper can model attack dependencies, potentially improving detection performance.

Despite the performance improvements achieved by self-attention, several limitations remain and should be addressed in future work. First, temporal dependencies could be addressed using LSTM methods, but this requires datasets with sufficient temporal coverage. Second, known causal dependencies between attacks were not explored here and could be addressed in future work, for example, by combining existing attack scenarios with patterns discovered by self-attention. Finally, given that more powerful computational infrastructure facilitates dependency-aware intrusion detection, our approach can be implemented on EFC infrastructure — edge, fog, and cloud — and likewise extended to other domains.

Declaration Statements

- **Use of AI:** During the preparation of this work, the author used ChatGPT (OpenAI), Claude (Anthropic), DeepSeek, and Grok (xAI) to refine the text and enhance readability. After using these tools, the author reviewed and edited the content as needed and takes full responsibility for the publication's content.
- **Conflict of Interest:** All authors declare that they have no conflicts of interest.
- **Data Availability Statement:** The source code and sample processed datasets used in this study are publicly accessible at: https://github.com/fsh211/home-robot-IDS/.

References

[1] Seyyed Mohsen Hashemi and Patrick C. K. Hung. Comprehensive guide to home robot security: From system analysis to solution recommenda-

tion. *Journal of Information Assurance and Security*, 2025.

[2] M. Yani, A. R. A. Besari, N. Yamada, and N. Kubota. Ecological-inspired system design for safety manipulation strategy in home-care robot. In *2020 International Symposium on Community-Centric Systems, CcS 2020*. Institute of Electrical and Electronics Engineers Inc., 2020. ISBN 9781728187419 (ISBN). . URL https://ieeexplore.ieee.org/document/9231354/.

[3] Mohammadreza Shahlaei and Seyyed Mohsen Hashemi. A risk-aware and recommender distributed intrusion detection system for home robots. *Journal of Information Security and Applications*, 83:103777, 2024. ISSN 2214-2126. . URL https://www.sciencedirect.com/science/article/pii/S2214212624000802.

[4] Maksim Goman. Towards unambiguous it risk definition. In *Proceedings of the Central European Cybersecurity Conference 2018*, pages 1–6, 2018.

[5] George Loukas, Yongpil Yoon, Georgia Sakellari, Tuan Vuong, and Ryan Heartfield. Computation offloading of a vehicle's continuous intrusion detection workload for energy efficiency and performance. *Simulation Modelling Practice and Theory*, 73:83–94, 2017.

[6] Amjad Rehman Khan, Muhammad Kashif, Rutvij H Jhaveri, Roshani Raut, Tanzila Saba, Saeed Ali Bahaj, et al. Deep learning for intrusion detection and security of internet of things (iot): current analysis, challenges, and possible solutions. *Security and communication networks*, 2022, 2022.

[7] Krishna Keerthi Chennam, Fahmina Taranum, and Maniza Hijab. An Overview of Cyber Physical System (CPS) Security, Threats, and Solutions. In Rajdeep Chakraborty, Anupam Ghosh, Jyotsna Kumar Mandal, and S. Balamurugan, editors, *Convergence of Deep Learning In Cyber-IoT Systems and Security*, pages 415–433. Wiley, 1 edition, December 2022. ISBN 978-1-119-85721-1 978-1-119-85768-6. . URL https://onlinelibrary.wiley.com/doi/10.1002/9781119857686.ch20.

[8] Khurum Nazir Junejo and Junaidi Goh. Behaviour-based attack detection and classification in cyber physical systems using machine learning. In *Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security*. ACM, 2016.

[9] Abdullah Waqas. Performance evaluation of deep learning models on diverse iot datasets for intrusion detection. *The ISC International Journal of Information Security*, 18(1):19–33, jan 2026. .

[10] Sheng Zhou, Amjad Ali, Ala Al-Fuqaha, Marwan Omar, and Li Feng. Robust Risk-Sensitive Task Offloading for Edge-Enabled Industrial Internet of Things. *IEEE Transactions on Consumer Electronics*, pages 1–1, 2023. ISSN 1558-4127. . URL https://ieeexplore.ieee.org/abstract/document/10285434. Conference Name: IEEE Transactions on Consumer Electronics.

[11] Nikolaos Polatidis, Elias Pimenidis, Michalis Pavlidis, Spyridon Papastergiou, and Haralambos Mouratidis. From product recommendation to cyber-attack prediction: generating attack graphs and predicting future attacks. *Evolving Systems*, 11(3):479–490, September 2020. ISSN 1868-6486. . URL https://doi.org/10.1007/s12530-018-9234-z.

[12] Harjinder Singh Lallie, Kurt Debattista, and Jay Bal. A review of attack graph and attack tree visual syntax in cyber security - ScienceDirect. *Computer Science Review*, 2020. . URL https://www.sciencedirect.com/science/article/abs/pii/S1574013719300772.

[13] Ahmad F Al Musawi, Satyaki Roy, and Preetam Ghosh. Examining indicators of complex network vulnerability across diverse attack scenarios. *Scientific reports*, 13(1), 2023. ISSN 2045-2322. . URL https://europepmc.org/articles/PMC10598276.

[14] Hongshuo Lyu, Jing Liu, Yingxu Lai, Beifeng Mao, and Xianting Huang. Agcm: A multi-stage attack correlation and scenario reconstruction method based on graph aggregation. *Computer Communications*, 2024.

[15] Nuno Oliveira, Isabel Praça, Eva Maia, and Orlando Sousa. Intelligent Cyber Attack Detection and Classification for Network-Based Intrusion Detection Systems. *Applied Sciences*, 11(4):1674, January 2021. ISSN 2076-3417. . URL https://www.mdpi.com/2076-3417/11/4/1674.

[16] A. Momand, S. U. Jan, and N. Ramzan. A systematic and comprehensive survey of recent advances in intrusion detection systems using machine learning: Deep learning, datasets, and attack taxonomy. *Journal of Sensors*, 2023(1): 6048087, 2023. .

[17] Md Shakil Siddique, Md. Ashikur Rahman Khan, Ishtiaq Ahammad, Nishu Nath, Joysri Rani Das, and Fardowsi Rahman. An intelligent intrusion detection system for cyber-physical systems using gan-lstm networks. *Franklin Open*, 2025. .

[18] Chao Lu, Hanhua Dai, Jie Zhou, and Haixia Wang. Exploring self-attention mechanism of deep learning in cloud intrusion detection. In *Cloud Computing: 10th EAI International Conference, CloudComp 2020, Qufu, China, De-*

cember 11-12, 2020, Proceedings, pages 57–73. Springer International Publishing, 2021.

[19] Mohammed S. Alshehri, Othmane Saidani, Faisal S. Alrayes, Saad F. Abbasi, and Junaid Ahmad. A self-attention-based deep convolutional neural networks for iiot networks intrusion detection. *IEEE Access*, Mar 2024. .

[20] S. Seo, S. Han, J. Park, S. Shim, H. E. Ryu, B. Cho, and S. Lee. Hunt for unseen intrusion: Multi-head self-attention neural detector. *IEEE Access*, 9:129635–129647, September 2021. .

[21] M. P. Aswathy and T. Annalakshmi. Adversarial resilient intrusion detection: Enhancing cybersecurity with transformer-based deep learning architectures. In *Proceedings of the 6th International Conference on Data Intelligence and Cognitive Informatics (ICDICI-2025)*. IEEE Xplore, 2025. IEEE Xplore Part Number: (to be added if available).

[22] Nikolaos Polatidis, Elias Pimenidis, Michalis Pavlidis, Spyros Papastergiou, and Haralambos Mouratidis. From product recommendation to cyber-attack prediction: Generating attack graphs and predicting future attacks. *Evolving Systems*, 11:479–490, Sep 2020.

Mohammadreza Shahlaei received his M.S. degree in Software Computer Engineering in 2012. He is currently a Ph.D. candidate at the Science and Research Branch of Islamic Azad University (IAU) in Tehran, Iran. His research interests include cybersecurity, big data, and software architecture.

Seyyed Mohsen Hashemi is an Associate Professor of Computer Science at the Science and Research Branch of Azad University and formerly served as Dean of the Department of Software Engineering and Artificial Intelligence. He has also been a research scholar at the Human-Machine Lab within the Software and Informatics Research Centre at the University of Ontario Institute of Technology. His research focuses on software-intensive systems for electronic services and e-business. A recognized contributor to service computing, he is the author of *Global Village Services as the Future of Electronic Services*. In industry, his experience encompasses IT project management, IT auditing, and cybersecurity. With 25 years of academic experience, he has supervised more than 120 PhD and MSc theses.

Ali Movaghar is Professor Emeritus of Computer Engineering at Sharif University of Technology in Tehran, Iran, where he has been a faculty member since 1993. He earned his B.S. in Electrical Engineering from the University of Tehran in 1977, followed by an M.S. and Ph.D. in Computer, Information, and Control Engineering from the University of Michigan in 1979 and 1985, respectively. During his academic career, Dr. Movaghar has held visiting positions at INRIA in Paris (1984), the University of California, Irvine (2011), and the University of Michigan, Ann Arbor (2023–2025). His professional experience also includes work at AT&T Information Systems in Naperville, Illinois (1985–1986), and service as a lecturer at the University of Michigan (1987–1989). Dr. Movaghar's research interests include formal methods, model checking, performance analysis, distributed real-time systems, and dependable AI. He is a Senior Member of both IEEE and ACM.