Received 20 December 2024, accepted 13 January 2025, date of publication 20 January 2025, date of current version 24 January 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3532353

RESEARCH ARTICLE

Reinforcement Learning-Based Generative Security Framework for Host Intrusion Detection

YONGSIK KIM[1], SU-YOUN HONG[2], SUNGJIN PARK[2], AND HUY KANG KIM[1], (Member, IEEE)
[1]School of Cybersecurity, Korea University, Seoul 02841, Republic of Korea
[2]LIG Nex1, Yongin-si 16911, South Korea

Corresponding author: Huy Kang Kim (cenda@korea.ac.kr)

This work was supported in part by Korea Research Institute for Defense Technology Planning and Advancement (KRIT) Grant funded by Korea Government (Defense Acquisition Program Administration—DAPA) (Cyberwar Training Red-Team/Blue-Team Autonomous System, in 2021) under Grant KRIT-CT-21-042; and in part by the National Research Foundation of Korea (NRF) Grant funded by Korea Government [Ministry of Science and ICT (MSIT)] under Grant RS-2024-00359621.

ABSTRACT Protecting users' systems from evolving cybercrime is becoming increasingly challenging. Attackers create more complicated attack patterns and configure attack behavior to resemble normal behavior to evade detection by defenders. Thus, it is indispensable to configure a security system that accurately detects attacks on each user's system. Since the attack does not occur only at a specific point in the network, there is a limitation in identifying computer intrusion simply using network packets. A Host-based Intrusion Detection System (HIDS) is a highly effective tool for monitoring computer systems and detecting unusual or unauthorized activities. HIDS can quickly identify potential security threats by closely monitoring and analyzing system logs, configurations, file integrity, and events specific to a host machine. It helps maintain the security and integrity of individual systems by detecting unauthorized activities or policy violations. With its advanced capabilities and reliable performance, HIDS is essential to any comprehensive host-based security strategy. Although HIDS can detect insider intrusions, the known HIDS detection methods are limited to specific attacks and may be ineffective against new attack patterns. Recently, researchers applied Natural Language Processing (NLP) in HIDS to scrutinize complex attack patterns, but they could have more effectively provided useful outputs for detecting intrusions based on these patterns. In this paper, we use reinforcement learning methodology, Actor-Critic, and NLP to extract keywords that occur on each anomaly system call log and propose a rule generation framework to prevent future intrusion detection using the extracted words. We analyze the anomaly log using NLP and extract the characteristics of each attack log as the 'keyword.' Based on the unique keywords of each attack log, we utilize reinforcement learning to establish a set of rules to protect against attacks. We extracted keywords based on textrank from the system call log sequence and simultaneously provided ground truth data using the extracted keywords. Based on the extracted keywords, the pre-trained Seq2Seq model generate rules according to the reward calculation method in reinforcement learning. When calculating the reward in reinforcement learning, we used the comparison value with the pre-trained Seq2Seq model, the malware log sequence detected by the rule set based on reinforcement learning, and the false positive value generated by the normal data to create its own rule set. We verified the proposed framework using the system call log datasets: ADFA-LD, LID-DS 2021 dataset. The proposed framework demonstrated a high accuracy rate of 96.5% average when faced with different attacks. We compared the accuracy based on the proposed framework detection, textrank, and Seq2Seq model-based keyword extraction methods. As a result, the proposed framework showed relatively high accuracy against various attack logs.

INDEX TERMS Reinforcement learning, natural language processing, host-based intrusion detection system.

The associate editor coordinating the review of this manuscript and approving it for publication was Claudio Zunino.

I. INTRODUCTION

Since overcoming the COVID-19 virus, people have had to adapt to more online circumstances, and services are now available in various domains [1]. However, paradoxically, the ease of access to the online circumstances has made it easier for attackers to access it. Attackers have increased their attacks to steal not just existing attack targets (e.g., companies) but also personal assets and information. Thus, it has become crucial for users to have a defense solution to protect their assets from external attacks. Users identified and reacted to intrusion attacks into their environment by configuring an Intrusion Detection System (IDS). However, it is difficult for individuals to build and manage an intrusion detection solution in response to attacks from attackers that are gradually changing and evolving. The attackers created sophisticated and complex attack patterns to circumvent existing detection methods [2]. Sophisticated and complex attack patterns show that no matter how excellent IDS performance is, it is challenging to perform perfect intrusion prevention. Moreover, attackers invade a user's environment and perform Advanced Persistent Threat (APT) actions after infiltrating to achieve their goals. It is essential to not only adapt to ever-changing attacks but also identify and continuously monitor internal intrusions to ensure the safety of user assets.

The Host-based Intrusion Detection System (HIDS), which detects attacks by utilizing file system, system call, system log, and events, continuously monitors and detects attacks even if they cannot detect an attacker's intrusion [3]. HIDS installed on the host utilizes signature-based technology to detect known attacks or anomaly-based detection technology to detect unknown attacks. In the case of the existent HIDS methodology, there is a disadvantage in that not only must the user set the rules for existing attacks manually, but also, the rules for mutated attacks must be identified and changed. However, there are many difficulties in modifying and applying the rules on their own for users who are not cybersecurity experts. Since HIDS can improve detection performance by analyzing the contextual information of complex attack logs, it is difficult to perform real-time detection based on a massive amount of log information, including attack logs.

This study proposes a Reinforcement Learning Host Intrusion Detection System (RL-HIDS). This generative security framework applies Natural Language Processing (NLP) and Reinforcement Learning (RL) to construct an automated detection method. RL-HIDS has the advantage of creating and applying an optimal rule set for specific attack logs so that users do not have to manually apply defense techniques for specific attacks. To address the limitations of current HIDS, which require manual configuration of rule sets towards specific attacks, RL-HIDS generates rule sets based on specific attack logs. We used extractive summarization-based approaches among the NLP methodologies to extract the essential keywords from each attack log and establish a rule set. Each new keyword is presented when an attack changes or evolves, even if the attack type is the same. We adapted reinforcement learning to create rules suitable for each attack based on keywords representing each attack. The reinforcement learning constructs the optimal rule set corresponding to each attack by combining the extracted keywords. Our study's main contributions are as follows:

- We proposed a reinforcement learning-based intrusion detection system that understands attacks that can occur on the user's system and creates a detection rule set for the attacks. The proposed reinforcement learning model incorporates keywords extracted from each attack log, creating an optimal rule set that best detects attacks.
- We proposed an NLP method to extract the main keywords that distinguish each attack log. This study converted attack log information on the host system into a machine-readable data format. Based on the converted data, essential keywords were extracted for each attack.
- We applied Actor-Critic, a reinforcement learning methodology, to accurately extract keywords for attacks that occur. Based on the extracted keywords, the Actor-Critic model created a rule set; based on the generated rule set, we used the detection metrics in the testbed for each training step to calculate the reinforcement learning as a part of the reward.

II. BACKGROUND

A. INTRUSION DETECTION SYSTEM

IDS means a system that detects and responds to abnormal behavior different from user behavior [4]. IDS mainly detects events that may cause attacks, loggings events information, and reports the logged information to the security operator [5]. IDS can detect specific events more efficiently by comprising various technologies. Depending on the applied technology, IDS is divided into Network-based Intrusion Detection System (NIDS), HIDS, Wireless IDS, and Network Behavioral Analysis.

B. HOST-BASED INTRUSION DETECTION SYSTEM

A Host-based Intrusion Detection System (HIDS) detects and reports unusual events by utilizing models of system behavior. It can effectively identify new attacks by analyzing sequences of system events, including system calls [6]. HIDS detects unusual events in real-time, based on system event sequences such as system calls, when unusual events, such as new or irregular ones, are expressed differently from known ones. NIDS, which performs detection based on network packets, has problems with packet encryption inspection and non-traffic sensitivity attack detection [7]. NIDS must classify attack logs against the massive network traffic generated every time. Identifying attacks through the current detection method is daunting, especially when attackers generate minimal network traffic. Additionally, detecting attacks within the network while passing through the NIDS is highly challenging. Thus, we focused on HIDS, which can detect attacks inside the host system.

C. REINFORCEMENT LEARNING AND ACTOR-CRITIC ALGORITHM

Reinforcement learning is the process of learning what action is optimal when an agent acts in a given environment with a specific state. In the reinforcement learning process, the consequent value of the agent's action is defined as reward. The reward expected value G_t is the sum of the rewards that can be obtained in each state when an episode ends and is defined as follows:

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{1}$$

The discounted return value G_t is calculated using the reward value R obtainable in each state and the discount factor γ, which determines the current worth of each reward. Also, the prediction of the return value ($\mathbb{E}$) is defined as the state-value function $V(s)$, and the state-value function represents the reward for a given state s at each timestep t and is calculated as the sum of all rewards [8]. The state-value function can be defined as follows:

$$V(s) = \mathbb{E}[G_t | S_t = s] \tag{2}$$

The action-value function Q is the expected value for an action in a specific state within a policy π. The Q function is defined as follows:

$$Q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a] \tag{3}$$

In a policy π, the action-value function Q operates based on action a and state s. The Q function is expressed based on G_t. The state s represents the agent's current situation in the environment and provides the information needed to decide the following action. The action a refers to the choices the agent can make in a given state. These choices may be selected from a predefined set or represented as continuous values. Policy is a strategy for an agent deciding an action in a specific state. The policy definition is as follows:

$$\pi(a|s) = \mathbb{P}[A_t = a | S_t = s] \tag{4}$$

Although it is possible to evaluate each state based on a deterministic value function and obtain the optimal result, there are limits to its application when there are countless states and actions. Thus, we applied policy-based reinforcement learning to select the optimal action in the state based on a policy. In policy-based reinforcement learning, the agent directly learns policy π to select the optimal action in each state. The agent probabilistically selects action an in state s and determines the optimal action by optimizing the policy to maximize the expected cumulative reward. In order to maximize the reward in policy-based reinforcement learning, we must evaluate policy π and optimize the policy by increasing the evaluation score of each policy. Notably, policy gradient is a representative example of a method for optimizing policy. Policy gradient is defined as $J(\theta)$, and θ means a parameter for approximating the policy. The equation 5 defines the policy gradient as $J(\theta)$ using the stationary distribution $d(s)$, which is the probability of starting in each state.

$$J(\theta) = \sum_{s \in S} d(s) * V_{\pi_\theta}(s_0) \tag{5}$$

Policy gradient can be expressed as the expected value of the aggregate of rewards, and it is the same as the expected value of return; it can be expressed as $V_{\pi_\theta}(s_0)$, which has the same format as the value function value in the initial state s_0 do. The policy gradient uses gradients taken on both sides because of a gradient ascent that maximizes the policy value. The policy gradient for Time T is defined as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau[\sum_{t=0}^{T-1} \nabla_\theta log\pi_\theta(a_t|s_t) G_t] \tag{6}$$

This type of policy gradient is defined as the Monte-Carlo policy gradient. Monte-Carlo policy gradient collects random samples and updates policy parameters. However, the Monte-Carlo policy gradient uses the return G_t as is, so the trajectory can change significantly during training, which causes a high variance problem. Therefore, the Actor-Critic algorithm has proposed, approximating the action-value function using the new parameter w.

The actor-critic algorithm is a method that helps find the best action in a given environment. It comprises two networks: an Actor Network that decides on actions based on the current state and a Critic Network that assesses the value of each state [9]. In the Actor-Critic algorithm, the Actor network learns the policy θ and selects the optimal action in each state. In this process, the policy is continuously improved to maximize cumulative rewards. The critic network provides feedback on the actions chosen by the actor, and the actor can efficiently update the policy based on the feedback provided by the critic network. The Actor-Critic algorithm can learn efficiently even in complex environments; it can use feedback from the critic network to reduce the policy gradient's variance and increase the actor network's learning speed and stability. In summary, the critic network in the Actor-Critic algorithm updates the action-value parameter w, and the actor network updates the policy parameter θ based on the critic's feedback. The Actor-Critic formula can be expressed as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau[\sum_{t=0}^{T-1} \nabla_\theta log\pi_\theta(a_t|s_t) Q_w(s_t, a_t)] \tag{7}$$

The Actor-Critic method that parameterizes the Q function value is called Q Actor-Critic. Neural network-based Q Actor-Critic has the advantage of being able to train without the end of an episode, but it is unstable because the value of the error function changes significantly depending on the Q value. Therefore, Advantage Actor-Critic (A2C) uses the advantage value proposed. A2C determines how good the state-value function $V(s)$ and the corresponding action value $Q_w(s_t, a_t)$ are in a given state. The equation 8 shows the advantage value that equals the action-value minus the state-value function.

$$A(s_t, a_t) = Q_w(s_t, a_t) - V_v(s_t) \quad (8)$$

v is the parameter of another state-value function. At this time, if $Q(s_t, a_t) = \mathbb{E}[r_{t+1} + \gamma V(s_{t+1})]$ is used according to the Bellman optimization equation, the advantage value is calculated using the equation 9. It can be expressed as:

$$A(s_t, a_t) = r_{t+1} + \gamma V_v(s_{t+1}) - V_v(s_t) \quad (9)$$

The policy gradient equation can be expressed as equation 10 using the advantage value defined in equation 9.

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau[\sum_{t=0}^{T-1} \nabla_\theta log\pi_\theta(a_t|s_t)A(s_t, a_t)] \quad (10)$$

In this paper, we propose a security framework based on the A2C algorithm to extract optimal keywords from each attack log and create rules to prevent each attack based on the keywords.

III. RELATED WORK

A. MACHINE LEARNING-BASED HIDS

HIDS can identify potential attacks by analyzing system behaviors, including user behavior, known attacks, and statistical attributes [10]. HIDS also provides detailed information about attacks and verifies the system's integrity [11]. HIDS effectively detects individual attacks, but analyzing system calls from multiple hosts is challenging [12]. In addition, there is a disadvantage in that it is challenging to build a HIDS suitable for the host environment. Therefore, various studies proposed detecting progressively intelligent attacks in host environments based on HIDS.

Besharati et al. proposed the HIDS model Logistic Regression HIDS (LR-HIDS) to protect virtual machines in a cloud environment [13]. The proposed model combines decision tree, Linear Discriminant Analysis (LDA), and Multi-Layer Perceptron (MLP) models through a bagging algorithm. The author performed feature selection using a logistic regression algorithm and verified the proposed model using Cloudsim software and the Network Security Laboratory-Knowledge Discovery in Database (NSL-KDD) dataset. They classified various attacks in NSL-KDD into four types: Probe, DoS, R2L, and U2R, and showed how feature selection was performed and how it was not performed. The proposed model showed an accuracy of 94.96% and 97.51% when feature selection was not performed and when it was performed, respectively. This accuracy was higher than what was achieved by applying basic machine learning methodology on the same dataset.

Park et al. presented Siamese convolutional neural network (Siamese-CNN), one of the few-shot learning techniques, to improve the disadvantage of relearning every time a new attack occurs [14]. The author imaged each cyberattack in 64 × 64 size and classified similar types of attacks using Siamese-CNN. The author used attack types such as Brute-force, SQL injection, and ZIP Slip in the LID-DS dataset to verify the proposed model. Comparing the proposed model with Vanila-CNN, Naive Bayes, Decision tree, logistic regression, and MLP, the Siamese-CNN showed the best performance with an F1-score of 90.0%. Moreover, testing the proposed model using the NSL-KDD dataset, similar to NID-DS, showed higher accuracy than other machine learning-based methodologies tested similarly, with an f1-score of 82.0%.

Wang et al. proposed a framework that uses GraphSAGE to detect host-based threats using system entities without prior knowledge of attack patterns [15]. The author mentioned that it was challenging to select a rule set to detect attacks and detect anomalies by analyzing nodes on the graph. The author used an external tool called Camflow to create a system provenance graph showing the attack behavior in time order [16]. The system provenance graph expresses the intrusion behavior that occurred on the host. The author used the StreamSpot, Unicorn SC-1, SC-2, and DARPA TC datasets to verify the proposed framework. The training data was limited to attack behavior, and as a result, SteamSpot, Unicorn SC-1, and SC-2 datasets showed good accuracy of 99.0%, 95.0%, and 93.0%, respectively. However, the DARPA TC dataset showed relatively lower accuracy than previous datasets due to the high imbalance problem between normal and abnormal nodes in the dataset. In particular, the 'fivedirections' scene in the DARPA dataset, which has the most severe imbalance in the DARPA TC dataset, showed an average result of 73.5% but better performance than known log analysis tools such as Log2Vec.

Hwang et al. proposed an ensemble security framework that detected attacks using a Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM)-based sequence model, and Deep Neural Network (DNN) ensemble framework [17]. The proposed framework pre-trains host-based data: network traffic, system logs, and host statistics into each model CNN, LSTM-based sequence model, and DNN model, and then derives prediction results for data sources that co-occurred. After pre-training the models, each model's prediction results are combined to infer benign and malignancy. The author collected seven attacks, including ransomware and Mirai, using the CREME testbed to verify the proposed framework [18]. As a result of the verification, the proposed framework reached an F1-score of 1.0, but the log-based detection rate was low because the assumed malicious point was not clearly revealed in the simple log data set.

Joraviya et al. proposed an IDS using system call log-based images to solve the low detection rate problem that can occur when learning by integrating redundant metadata and system call parameters [19]. The author preprocessed the system call log generated in a containerized cloud environment by label encoding to use it as an input and then constructed the selected features into images with sizes of 64 × 64 and 128 × 128, respectively. The author evaluated the proposed model's performance using the LID-DS 2019. As a result, images with sizes of 64 × 64 and 128 × 128 showed F1-scores of 95% and 90%, respectively.

Table 1 is a summary of machine learning-based HIDS. Existing studies mainly evaluated the accuracy and F1-score of the proposed models based on the confusion matrix. Furthermore, they proposed a methodology to solve the imbalance problem of the used datasets. However, some of the datasets used were old, which had the disadvantage of making it difficult to evaluate whether the proposed model could detect various attack types.

B. NLP-BASED HIDS

The previous HIDS showed excellent performance in detecting various attacks, but the gradual complexity of attack patterns and the resulting overwhelming amount of system call traces caused high false alarm rates in existing HIDS [20]. Notably, security operators have limitations in analyzing the overwhelming amount of system call logs. Thus, HIDS methodologies using NLP have been proposed to analyze recent complex attack pattern problems and the overwhelming number of system call logs. NLP-based HIDS can detect attacks by quickly understanding the contextual data of system call logs due to the similarity between natural language and system call sequences. Therefore, diverse NLP-based studies have been conducted to detect attacks in the host environment.

Chawla et al. proposed a combined CNN and Gated Recurrent Units (GRU) model to perform intrusion detection based on system call sequences [21]. Using a one-dimensional CNN, the author pre-processed the system call sequence as input to GRU. The GRU layer classified the received input value as 'normal' or 'attack' with the softmax layer. The author verified the proposed model through the Australian Defence Force Academy Linux Dataset (ADFA-LD) dataset and showed an accuracy of 81.0%. Tahir and Qadir demonstrated a model capable of detecting advanced malware targeting MIPS, ARM, and x86 IoT devices using a Random Forest (RF) model [22]. First, they collected 524 malware and 524 benign samples from IoT architectures such as x86, MIPS, and ARM. The author created a system call log from the samples using the ELF Digest tool and converted it for machine learning. Initially, a technique called Bag of Words (BoW) was utilized to extract features. This involved expressing all distinct words as feature vectors, without considering their order in the text. The proposed method performs data cleaning in the feature extraction step and ranks each feature using the Gain Ratio Attribute Evaluator. The author compared the accuracy of the processed system call sequence using Soft Vector Machine (SVM), Logistic Regression, RF, MLP, and Bootstrap Aggregating method. As a result, the RF model showed the best performance at 99.0%; the author mentioned that the model effectively performed cross-architectural analysis and detection.

Kim et al. proposed an anomaly-based HIDS method based on the ensemble method [23]. The proposed model utilizes Back-Propagation Through Time (BPTT), an LSTM based system call language model, and proposes a new ensemble method to avoid the overfitting problem. The new ensemble method lowered the false alarm rate by constructing a thresholding classifier corresponding to each model. The author verified the proposed model using the ADFA-LD, KDD98, and UNM-lpr datasets and showed The area under curve (AUC) values of 92.8%, 99,4%, and 96.9%, respectively.

LV et al. demonstrated the sequence prediction model by applying the Sequence-to-Sequence (Seq2Seq) model to solve the sequence dependency in intrusion prediction [24]. The author utilized the GRU model to solve the problem of a long-term dependence on system call sequences with long input values. The author used a question-answering system to construct train and test sets. They set the first part of the system call sequence as the question of the question-answering system and the second part as the answer. The author used the bilingual evaluation understudy (BLEU) score to verify whether the proposed model appropriately predicts the target sequence. As a result of verifying the proposed model with the ADFA-LD dataset, the maximum BLEU value was 40.4, showing better performance as the input sequence length increases. Notably, they mentioned that the longer the sequence, the more practical information the prediction model can obtain and make accurate predictions. However, as a limitation of the GRU model, the author mentioned that performance was not good when the input sequence exceeded a certain length.

Zhang et al. proposed a behavioral semantics enhancement method for system call sequences to overcome evolutionary intrusion attacks such as obfuscation techniques [25]. The author abstracted the system call sequence and created the extended system call sequence by differential encoding. Furthermore, the token sequence of the extended system call sequence was acquired through the utilization of NLP's word segmentation and pre-training technology. Based on the tokens of each sequence, vectorization was performed through 'Syscall2Vec', 'SyscallType2Vec', and 'Syscall-Code2Vec'. Afterward, the text-CNN model performed anomaly classification for each vectorized sequence. They used the ADFA-LD dataset to verify the performance of the proposed model and recorded an accuracy of 98.73%.

Sakarkar et al. detected and compared DNS Tunneling attacks using Simple RNN, 1-D CNN, LSTM, and GRU models [26]. The author performed DNS Tunneling using the dnscat2 tool and then obtained attack data using the Wireshark tool. Since the network packet data composed of text cannot be directly used in the model learning process, the text is converted into numbers by tokenizing with the NLP model. In addition, since the lengths of the tokens are different, the lengths were fixed by padding after tokenization. For cross-validation, the author split the train and test dataset ratios into 60:40 and 80:20,

TABLE 1. Summary of machine learning-based HIDS.

Reference	Proposed Algorithm	Dataset	Best Performance	Year
Besharati *et al.*	LR-HIDS (decision tree, LDA, and MLP through a bagging algorithm)	NSL-KDD	97.51% (Accuracy)	2019
Park *et al.*	Siamese-CNN	LID-DS 2019	89.0% (Accuracy)	2021
Wang *et al.*	GraphSAGE-based framework	treamSpot, Unicorn SC-1, SC-2, and DARPA TC	99.0% (F1-Score in SteamSpot)	2022
Hwang *et al.*	CNN, LSTM-based model, DNN ensemble	Collected from the CREME testbed	100% (F1-Score)	2023
Joraviya *et al.*	deep CNN	LID-DS 2019	95% (F1-score in 64 x 64 images)	2024

TABLE 2. Summary of NLP-based HIDS.

Reference	Proposed Algorithm	Dataset	Best Performance	Year
Kim *et al.*	BPTT model	ADFA-LD, KDD98, UNM-lpr	92.8%, 99,4%, and 96.9% (AUC value)	2016
LV *et al.*	GRU model	ADFA-LD	40.4% (BLEU)	2018
Chawla *et al.*	CNN-GRU model	ADFA-LD	81.0% (Accuracy)	2019
Zhang *et al.*	behavioral semantics enhancement method	ADFA-LD	98.73% (Accuracy)	2021
Sakarkar *et al.*	Simple RNN, 1-D CNN, LSTM, and GRU model	Collected using the dnscat2 tool	98% (F1-score, Simple RNN)	2021
Tahir and Qadir	RF model	They collected data directly from IoT architecture	99.0% (F1-score)	2022
Zaboli *et al.*	LLM based methods	Collected GOOSE, SV packet from HIL testbed	98.3% (F1-score in SV)	2023

TABLE 3. Summary of RL-based IDS.

Reference	Proposed Algorithm	Dataset	Best Performance	Year
Hsu and Matsuoka	DQN	NSL-KDD, UNSW-NB15, and collected PANW datasets	97.95% (Accuracy in PANW)	2020
Lopez *et al.*	DQN, DDQN, Policy Gradient, and Actor-Critic	-	91.20%, 93.72% (F1-score, DDQN)	2020
Ren *et al.*	DT and RFE based methods	CSE-CIC-IDS 2018	93.54% (F1-score with RFE)	2022
Malik and Saini	DQN algorithm, distributed agents, and attention mechanism	NSL-KDD and CIC-IDS 2017	98.7% (Accuracy in CIC-IDS 2017)	2023
Liang *et al.*	MDP	Generated a payload using a policy network	NSL-KDD and AWID	2023

respectively. After pre-processing the dataset, the author compared accuracy using 1-D CNN, Simple RNN, LSTM, and GRU models. As a result, among the proposed models, Simple RNN showed a good F1-score of 98% on both datasets with different ratios, and when the train and test dataset ratios were split at 80:20, most models showed good performance.

Zaboli et al. constructed and compared various Large Language Model (LLM) based IDS for IEC 61850-based digital substation communication security [27]. The author utilized human-in the-loop (HITL) to alleviate problems that may arise when LLM is applied to high-risk fields. In the proposed framework, HITL checks outputs with low confidence during the LLM model learning process and adjusts the output. The author used the Generic Object Oriented Substation Event (GOOSE) and sampled value (SV) generated by the hardware-in-the-loop (HIL) testbed to verify whether the proposed framework performs abnormality detection properly. The author compared three LLM models: ChatGPT 4.0, Anthropic's Claude 2, and Google Bard/PaLM 2 according to the level of human recommendation. As a result of the verification, among the LLM models, ChatGPT 4.0 showed the best performance with 98.18% and 98.3% for GOOSE and SV, respectively. The more the human recommendation process was applied, the better the performance.

Table 2 is summary of NLP-based HIDS. The proposed NLP-based HIDSs appropriately detected attacks within the system call sequence log. Based on the aforementioned studies, we can confirm that the NLP model effectively comprehends the contextual data of the system call sequence because the NLP model processes the system call sequence like natural language. However, system call sequence logs have variable lengths and require a unique approach for processing and application.

C. SYSTEM CALL SEQUENCE SUMMARIZATION

Since long system call sequences help predict attacks, they require significant computational resources to be detected. Additionally, it becomes a factor that reduces model performance as the sequence length becomes too long. Therefore, various studies proposed to summarize system call sequences and extract keywords, which are important words.

Dijkman and Wilbik summarized the number of statements using clustering and pruning techniques to improve the processing speed of long event log sequences [28]. The proposed model generates subsequences in logs of a particular length and clusters the subsequences using a clustering algorithm or similarity metric simulation: Markov similarity. Once the medoid of each clustering is determined, it is used as a quantifier and summarizer for every statement. At this time, the author performs pruning to reduce the possible number.

Meng et al. showed a new log summarization method, breaking away from manual or rule-based log summarization methods [29]. The author extracted information using template matching, regular expression, and a set consisting of a predicate and two arguments; select the rank for the generated sets using cosine similarity and textRank.

Locke et al. proposed LogAssist, a log analysis framework for efficient log analysis [30]. The proposed model parses the input logs, groups them by ID, forms a workflow, and summarizes log events using n-grams. The author tested with Hadoop Distributed File System (HDFS) and Zookeeper provided by the open source dataset and mentioned that it showed a high log compression rate. However, since the verification method for the summarized log is subjective, it is difficult to verify whether the summarized log contains vital information.

Mvula et al. compare system call trace log datasets with word embedding methods: Word to Vector (Word2Vec) and Global Vectors for Word Representation (GloVe) to extract semantic relationships from system call logs [31]. The author utilized the ADFA-DS, NGIDS-DS, WWW 2019, and LID-DS 2021 datasets and classified normals and attacks through Extremely Randomized Trees (ERT), a binary classification algorithm. As a result, accuracy differed depending on the number of duplicate samples in each dataset; the LID-DS 2021 dataset achieved stable results due to its relatively low number of duplicates. Moreover, the author divided it into four feature sets based on the application of consecutive calls and the expression of the number of calls in each call trace. After conducting the experiment, the datasets ADFA-LD, The Next Generation Intrusion Detection System Dataset (NGIDS-DS), WWW 2019, and LID-DS 2021 showed an accuracy of 95.85%, 95.64%, 95.52%, and 99.96%, respectively, when there were no consecutive calls. The author mentioned that it improves the performance of Machine Learning (ML) models when there are no consecutive calls. Based on the above, the author mentioned that the proposed method helps reduce the number of duplicate samples and maintain data diversity.

D. REINFORCEMENT LEARNING-BASED IDS

Performing extractive summarization on system call sequences using the existing NLP method showed good performance on NLP-based HIDS. Notably, performing an anomaly detection model based on a LLM showed excellent performance. However, there is still a limitation in that human interaction is required. In the case of existing text summary methods, only short, general, and simple summaries are output for specific sentences [32]. Also, some existing methods do not rank sentences according to importance and do not consider the overall summary. This limitation arises when summarizing specific logs, which may result in using general words, such as time. Text summarization with reinforcement learning can create a more suitable model because the model receives feedback from the real environment based on reward and is immediately modified and updated [33]. Also, it can replace humans by fine-tuning it based on predefined rewards. Therefore, various reinforcement learning-based methodologies were proposed to summarize, generate, and apply them to intrusion detection.

Liang et al. proposed GPTFuzzer, a reinforcement learning-based WAF test framework, to solve the existing learning-based approach's efficiency and limited performance problems [34]. The proposed model pre-trains the transformer model based on the pre-defined attack grammar and performs reinforcement learning based on the pre-trained transformer. The agent conducts black box tests WAF by Markov Decision Process (MDP). The proposed model generates a payload using a policy network and calculates a reward depending on whether the payload passes through the WAF. Then, the agent is updated based on the reward. The authors noted that the proposed model performed better than the existing mutation-based and search-based approaches.

Ren et al. proposed recursive feature elimination (RFE) feature extraction and deep reinforcement learning-based model ID-RDRL [35]. The author performs feature selection through decision tree (DT) and RFE and processes the data through a mini-batch module. The processed data is input to CNN for additional feature extraction; then anomaly detection is performed through a Deep Q-network (DQN) based deep reinforcement learning (DRL) model. To verify the proposed model, the author used the CSE-CIC-IDS 2018 dataset. As a result of the verification, the F1-score when using only simple DT was 92.23%, but when using RFE, the result was 93.54%, claimed to have been obtained. Moreover, they mentioned that combining the DRL model and RFE obtained a result of 94.89 The author asserted that the execution time was reduced, and good performance was achieved through the RFE process in the proposed ID-RDRL model.

Hsu and Matsuoka proposed a DQN-based anomaly network intrusion detection system [36]. The proposed RL agent is set up as an intrusion detection engine, and whether it was detected correctly is set as the reward value. The proposed RL model consists of a learning mode and a detection mode. The same model with the reward function paralyzed was used for the detection mode. To operate the proposed DQN model, the author converted features such as IP address based on one-hot encoding. The author used NSL-KDD and UNSW-NB15 and directly collected Palo Alto Networks (PANW) datasets to verify the proposed model. As a result, the proposed model showed accuracies of 91.4%, 91.8%, and 97.95% on the NSL-KDD, UNSW-NB15, and PANW datasets, respectively.

Malik and Saini proposed a network intrusion detection system using the DQN algorithm, distributed agents, and attention mechanism [37]. The author configures a reliable router as an agent and predicts Q-value from packet feature vectors. They implemented a packet class-based reward system that appropriately predicts harmful packets in imbalanced datasets with intrusion and non-intrusion. The author compared it with other models using the NSL-KDD and CIC-IDS 2017 datasets to verify the proposed model. As a result, the accuracy of the proposed model applying the attention mechanism was 97.4%, an increase of about 16% compared to the simple DQN model based on the NSL-KDD dataset. Also, the CIC-IDS 2017 dataset, which has a data imbalancing problem, showed an accuracy of 98.7%, proving that the attention mechanism and applied reward system are adequate. We referred to the paper and applied an attention mechanism to analyze the attack log occurring on the host effectively.

Lopez-Martin et al. implemented and compared various deep reinforcement learning methodologies on a network dataset [38]. The author implemented and compared various deep reinforcement learning models: DQN, Double Deep Q-Network (DDQN), Policy Gradient, and Actor-Critic using labeled network datasets: NSL-KDD and AWID datasets. As a result, among various deep reinforcement learning models, the DDQN model showed the highest F1-score of 91.20% and 93.72% in the two datasets, respectively. Notably, the AC model showed more robust accuracy than other models regardless of changes in the discount factor. Referring to the paper, we confirmed that reinforcement learning can perform similarly or superior to artificial intelligence using a general neural network on datasets with correct answers.

Table 3 is summary of RL-based IDS. As proposed in various studies, RL-based IDS trained on itself based on applied data and showed similar or better accuracy than existing neural network-based models. In particular, RL-based IDS performed well even when applied to various datasets by adapting to input values. Based on the above study, we will propose a HIDS framework that configures an optimal rule set to detect existing attacks by utilizing system log sequences occurring in the host environment.

IV. METHODOLOGY

A. RL BASED HIDS FRAMEWORK

This paper uses the RL-based HIDS framework to create an optimal rule set corresponding to each attack. The proposed framework focuses on the extractive summarization method. The extractive summarization method demands that significant sentences or phrases from the original text are chosen to create a summary. In a study of the previous log sequence, the meaning of essential words in the log sequence was lost due to data cleaning [39]. Consequently, essential keywords that could be derived from a specific attack log sequence may be forfeited, leading to a significant setback. When error codes that cannot be analyzed are treated as abnormal or replaced with words humans can recognize, there is a limitation in that worthy unique words can be deleted [40]. The extractive summarization methodology can sufficiently extract the main keywords from security sentences [41]. Based on the extractive summarization methodology, we supplemented the method of extracting keywords from each attack sentence in an existing rule-based format. We extracted the keyword of each attack log sequence and constructed a rule set based on the extracted keyword to check how efficiently it detects the attack that occurs. Also, we utilized the log tokenizer proposed to learn from every word extracted from the attack log. This method has the advantage of learning partially different attack techniques for each log sequence.

The Seq2Seq-based model performed well in summarizing text or extracting keywords from documents composed of various sentences [42]. However, when applying a Seq2Seq-based model to attack sequences, insignificant words were designated as keywords, such as pid and timestamp. To improve for these limitations, we used an Actor-Critic methodology to apply Seq2Seq-based extractive summarization methodology to security sentences. When a simple Seq2Seq model is applied to security sentences, the model shows a high bias toward data of low importance. Therefore, we used the Actor-Critic algorithm to extract important keywords in each attack. Also, Actor-Critic showed excellent performance even in security log datasets where normal and attack data are unbalanced [43]. We adopted a ground-truth answer to the critic network on which parts to 'focus' in each attack log sequence. Providing a ground-truth answer as an input in the Actor-Critic allows the agent to perform the desired action, thereby obtaining a high return. However, if we extract specific keywords from each security log sentence using the general Q Actor-Critic algorithm, the importance of keywords in the security sentence can be ambiguous at each step [44]. The Q function-based Actor-Critic is unable to accurately determine the value of the chosen action by the agent. This inability makes it difficult to apply the same framework to classify similarly structured security attack sentences. Therefore, we used A2C using advantage value to set the weight when the agent searches for keywords in each security log sentence.

Figure 1 shows an overview of the RL-based HIDS Framework. Following the figure 1, the HIDS framework has two main parts. First, there is the pre-training process to train an LSTM-based Seq2Seq model using the raw attack log. Second, there is the rule generation process to train reinforcement learning based on the pre-trained Seq2Seq model to create rules.

The pre-train Seq2Seq model process trains a Seq2Seq model based on generated keywords and label-encoded attack sequences. The pre-train process consists of a log tokenization process to refine specific attack logs as input values for learning and a pre-train Seq2Seq model process to train a Seq2Seq model based on the generated input values. The pre-train Seq2Seq model is behavior cloning of the Actor to perform the A2C algorithm based on Seq2Seq, which performs keyword extraction in the A2C algorithm. Behavior cloning is an imitation learning algorithm that learns the behavior to be applied based on supervised learning to improve the policy. We configured the optimal policy based on fewer samples by pre-setting the direction of the actor to apply the proposed framework to various attack log sequences.

In the log tokenization process, a label encoding method converts a specific attack log into an input value. Label encoding is a technique that converts words in a sentence into unique integer values, and is one of the effective methods of embedding without changing the dimension of security log data [45]. In order to carry out label encoding, it is necessary to allocate a unique integer value to every word. We constructed a word set using a specific attack log set to label encoding the system call sequences. The words in the

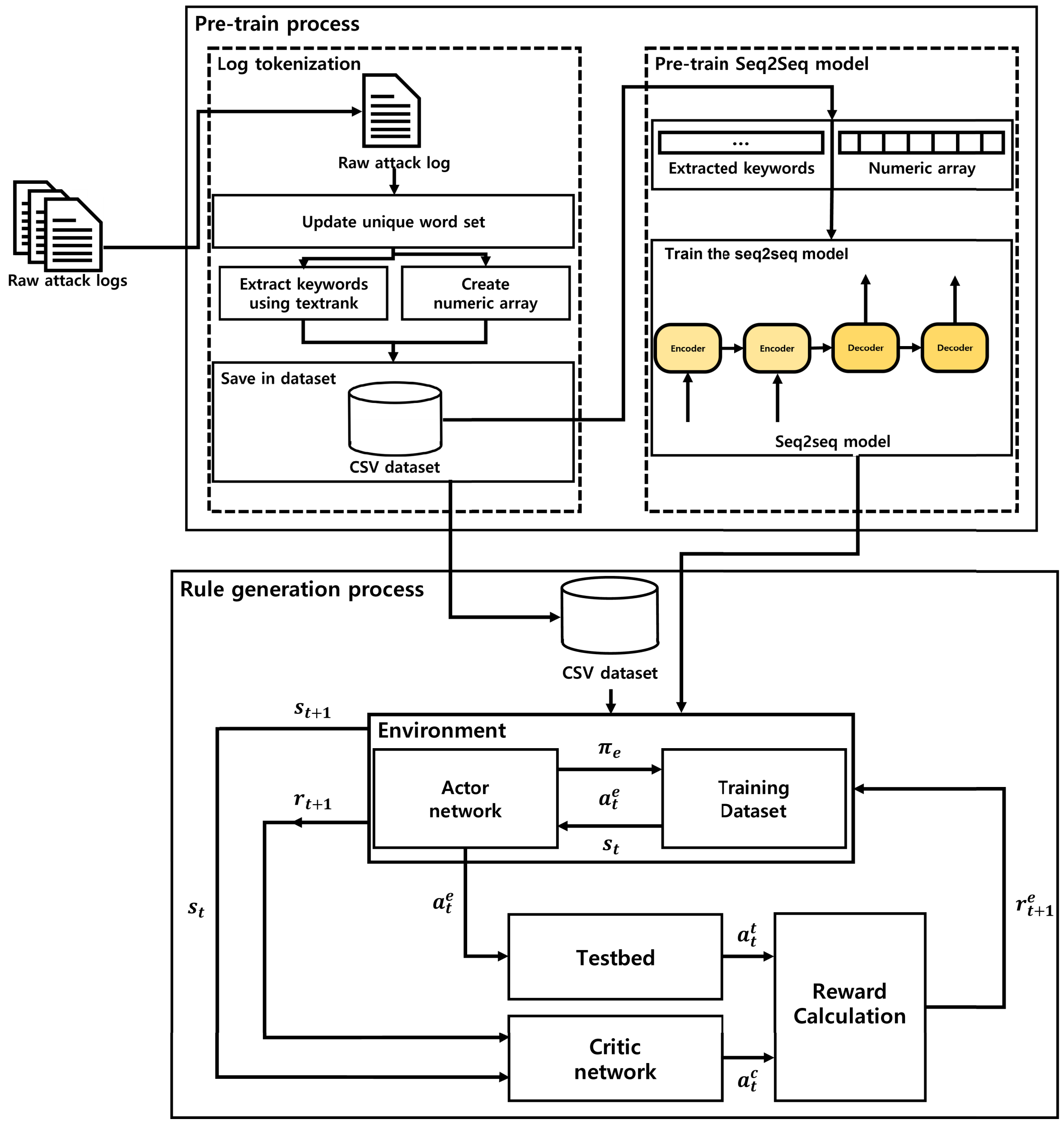

FIGURE 1. The overall structure of the RL-based HIDS framework.

word set were assigned unique integer values in the system call sequence based on their index numbers. To extract the words from the system call sequence, we must tokenize the sequence using spaces and then extract the following words. RL-HIDS performs pre-training based on attack logs in a specific host environment, either from previously occurring attacks or published attack datasets. We collected the unique words extracted from each system call sequence into one word set, and if a new unique word exists, the word set is updated.

To properly train Seq2Seq, it's necessary to provide both the input value and the correct answer. Therefore, we extracted the keywords of each log sequence based on the textrank algorithm. The textrank algorithm is a tool that uses the PageRank algorithm to analyze natural language. It determines the significance of words in a sentence [46].

Figure 2 is an example of performing textrank to extract keywords from the attack sequence. We vectorized numerous attack log sequences used as input values using word2vec. Word2vec is an artificial neural network-based model that can obtain vector representations of numerous words with a low computational effort [47]. However, there is a limitation in that keywords that appear less frequently in the context can be omitted when simply using the textrank. In this study, we used word2vec to utilize semantic similarity between words [48]. Based on the Continuous Bag of Words (CBOW) learning

Algorithm 1 Pre-Train the Seq2Seq Model

```
    Input: Input sequences X and ground-truth output sequences Y.
    Output: Pre-trained Seq2Seq model
0: procedure Training Steps($X$, $Y$)
0:     for Input sequences ($X$, $Y$) do
0:         Get $N$ from $X$
0:         Conduct encoding in Seq2Seq model on N and get the last state of encoder $h_t$
0:         Conduct decoding in Seq2Seq model with $h_t$ and obtain the output sequence $\hat{Y}$
0:         Calculate the loss according to cross-entropy and update the parameters of model
0:     end for
0: end procedure
0: procedure Testing Steps($X$, $Y$)
0:     for Input sequences ($X$, $Y$) do
0:         Get $N$ from $X$
0:         Generates output $\hat{Y}$ for input value N based on the trained model
0:         Evaluate the model with keywords from textrank and ground-truth answer
0:     end for
0: end procedure
```

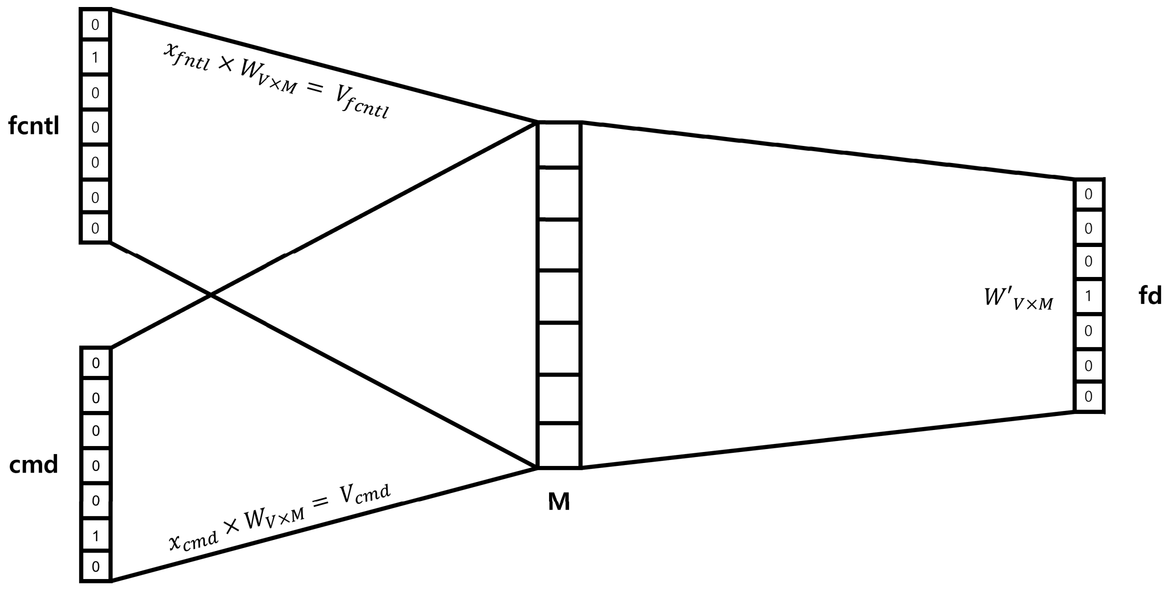
fcntl
0
1
0
0
0
0
0
$x_{fntl} \times W_{V\times M} = V_{fcntl}$
cmd
0
0
0
0
0
1
0
$x_{cmd} \times W_{V\times M} = V_{cmd}$
M
$W'_{V\times M}$
0
0
0
1
0
0
0
fd

FIGURE 2. The example of textrank using system call logs.

method, we constructed word2vec so that textrank can find important words from surrounding words in each attack log.

The formula 11 expressed the graph-based ranking model, textrank.

$$S(V_i) = (1-d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j) \tag{11}$$

Based on the cosine similarity algorithm, the textrank proposed a similarity measure between sentences. The formula beta represents the similarity measure between sentences in textrank.

$$sim(s_1, s_2) = \frac{|\{w_k | w_k \in S_1 \& w_k \in S_2\}|}{log|S_1| + log|S_2|} \tag{12}$$

In summary, the formula shows dividing the number of common words w_k between two sentences S by the log sum of the number of words in each sentence. Based on the vectorized attack sequences, we construct a sentence similarity matrix using a cosine similarity algorithm on textrank. Afterward, we extracted keywords for each attack sequence based on the similarity matrix. The extracted keyword is stored in the dataset as a pair with the label-encoded attack sequence. Textrank performs well when extracting keywords based on the similarity between each sentence, but in the case of security sentences that do not consist of complete sentences, the importance is low, but the essential words that make up the log sentence can be designated as keywords. Therefore, using the keyword extraction value

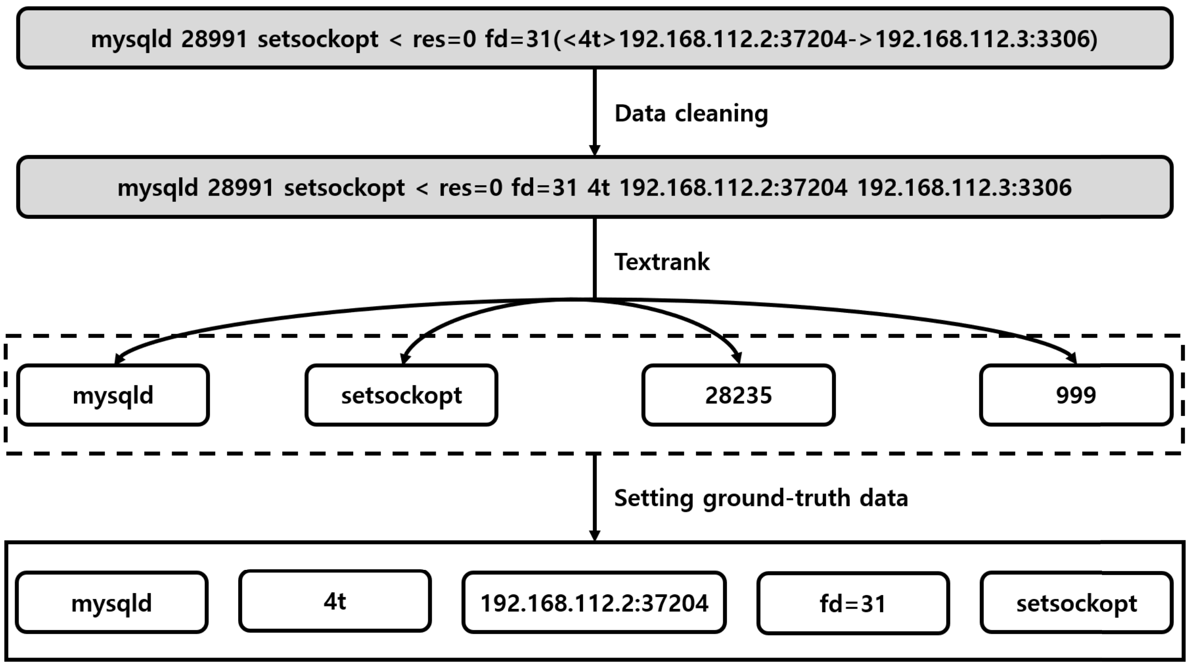
mysqld 28991 setsockopt < res=0 fd=31(<4t>192.168.112.2:37204->192.168.112.3:3306)
Data cleaning
mysqld 28991 setsockopt < res=0 fd=31 4t 192.168.112.2:37204 192.168.112.3:3306
Textrank
mysqld
setsockopt
28235
999
Setting ground-truth data
mysqld
4t
192.168.112.2:37204
fd=31
setsockopt

FIGURE 3. Example of applying textrank and ground-truth data to attack logs.

Algorithm 2 The Custom Cross-Entropy Loss Function

```
    Input: The prediction distribution $y\_pred$ and actual answer $y\_true$
    Output: Adjusted loss
0:  procedure Custom cross-entropy loss function steps
        Calculate base loss using cross-entropy using $y\_pred$ and $y\_true$
        Get rewards computed during reinforcement learning and normalize rewards to [0-1].
        Weight of the reward $\alpha = 0.5$
        Calculate the Adjust loss = base_loss * (1 - $\alpha$ * normalized_rewards)
        Return the adjusted loss
0:  end procedure
```

obtained simply by using textrank as an answer for training is challenging. To solve this problem, we provided ground-truth data that answers the extracted keywords when pre-training the model. We provide answers for un-important keywords to extract more important keywords from each attack log sentence based on the ground-truth data.

Figure 3 shows the procedure for applying ground-truth data based on the keywords that extracted the attack log using textrank. First, we extract keywords in specific attack logs based on textrank. We performed data cleaning locally only for specific characters, such as parentheses, to ensure that meaningful words were not damaged during the data cleaning process. Since there may be high-importance words among the keywords that were not extracted later that reflect the characteristics of the attack log, we found and added keywords that reflected the characteristics of the attack log but were not extracted after keyword extraction.

The algorithm 1 represents the process of pre-training the Seq2Seq model. First, the Seq2Seq model encoding converted attack log data as input based on the log tokenization process. In this process, the final state h_t is obtained, and the decoding process of the Seq2Seq model is performed based on h_t. The loss value is calculated based on $\hat{Y}$ obtained during the decoding process and Y, the ground-truth value. We used cross-entropy as a loss function to ensure that the keyword predicted by the Seq2Seq model will most likely be the keyword obtained or set in each attack log [49]. The cross-entropy loss function is generally used to minimize the output word sequence and the correct word sequence generated by the model. We used the cross-entropy loss function to make the Seq2Seq model minimize the difference in probability distribution between the actual correct answer and the predicted value and predict more accurate and significant keywords in attack logs. The trained model outputs a keyword set $\hat{Y}$ for the input value N in the test step and compares $\hat{Y}$ with the expected keyword set to evaluate the model's performance.

In the rule generation process, reinforcement learning is performed based on the configured dataset and the pre-trained Seq2Seq model. Before performing reinforcement learning, the pre-trained Seq2Seq model provides initial model parameters to the reinforcement learning agent. Based on the constructed dataset, the pre-trained Seq2Seq model generate keywords.

The process utilizes cross-entropy to determine whether reinforcement learning keywords' output deviates from the pre-trained Seq2Seq model. This was applied to avoid deviating too much from the pre-trained model when reinforcement learning outputs keywords. We integrated the reward from reinforcement learning into the cross-entropy used during pre-training. We modified the approach to ensure the model predicts the correct answer and maximizes the reward. Algorithm 2 is expressing the custom cross-entropy loss function. We applied reward values to existing cross-entropy loss values and evaluated the effectiveness of the keywords predicted by the model in attack logs. We were concerned that the reward values would be too large or unbalanced, so we normalized them to a value between 0 and 1. Based on reward value, we optimized the Seq2Seq model to acquire higher rewards in the environment while adjusting keyword prediction to produce more effective results in attack log detection. We set the reward weight value to 0.5 to reflect the cross-entropy loss and compensation information equally. A detection rule is established using the keyword output from reinforcement learning to improve accuracy. The detection rule is then tested in the testbed to measure its effectiveness in identifying attack logs. We created a firewall rule based on a logical expression to see if each log sequence included the keyword and checked whether it was detected appropriately to check whether each attack log was detected based on the output keywords. Rules simply applied based on the AND or OR operator are inaccurate in attack logs because they are similar to normal logs. So, we configured the rule to be created using keywords in the testbed using a combination of AND and OR operators. In this study, a rule combining AND and OR operators was applied, resulting in the most optimal accuracy and its combination. The testbed consists of new attack sequences of the same type and provides rewards based on the attack detection rate. This paper calculates the reward value based on the detection results and the pre-trained Seq2Seq model. Finally, the rule generation process provides an optimal rule set for a specific attack sequence.

B. LSTM BASED SEQ2SEQ MODEL

This study's reinforcement learning outputs predicted keywords to generate an appropriate rule set for a specific attack. Extracting keywords using simple reinforcement learning can be time-consuming when optimizing the model. Moreover, it's important to verify that the trained model comprehends the entire sentence and accurately extracts words. Therefore, we utilized a sequence model to comprehensively learn and analyze each sentence. Among sequential models, Recurrent Neural Network (RNN) has the disadvantage that the length of input and output values must be the same, is effective only for relatively short sequences, and has a loss of meaning due to long-term dependency problems. Therefore, we constructed the Seq2Seq model using the LSTM.

Fig 4 shows the LSTM-based Seq2Seq model applied in this study. The Seq2Seq model is a model that outputs a target sequence for a source sequence [50]. The Seq2Seq model consists of an encoder and a decoder. The Seq2Seq encoder expresses sequence data as one fixed context vector, and the Seq2Seq decoder outputs a new sentence based on the expressed context vector.

$$h_t = \mathrm{sigm}(W^{hx}x_t + W^{hh}h_{t-1}) \tag{13}$$

Following the equation 13, the Seq2Seq model calculates the hidden states according to sequence x.

$$y_t = W^{yh}h_t \tag{14}$$

As shown in equation 14, the decoder predicts the output sequence y based on the hidden state.

$$p(y_1, \ldots, y_{T'}|x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t|v, y_1, \ldots, y_{t-1}) \tag{15}$$

The Seq2Seq model uses Equation 15 to predict an output sequence by applying the same formula used for the input sequence. In this equation, v is the context vector. Since the Seq2Seq model constitutes a sentence as a unit, there is an advantage in that the number of tokens in the input sentence and the number of tokens in the output sentence may be different. If a log sequence has extended sentence information, the encoder of the Seq2Seq model may experience information loss during compression; a bottleneck problem may occur in which only part of the decoder's sentence is used for prediction. Therefore, we applied bahdanau attention to focus on keywords in the log sequence for attacks. The bahdanau attention helps to detect the attack log better than the simple Seq2Seq model by focusing on a specific part of the sentence [51]. Bahdanau attention is implemented based on a parameter called 'attention weight.' The model utilizes attention weights to assess the significance of inputs, which are then leveraged for making predictions. Attention weights are calculated through the following equation:

First, using the current hidden state of the model compute the attention weights h_t, and the input X. To calculate the attention weights, a dot product is performed between h_t and every element in X. Afterward, a softmax function is applied.

$$a_t = \mathrm{softmax}(h_t^T X) \tag{16}$$

To calculate the attention context vector, we use the input and the attention weights to find a weighted sum, which gives us the value of c_t.

$$c_t = \sum_{i=1}^{T_x} a_{t,i}x_i \tag{17}$$

To calculate the final hidden state of the model, we use the previous hidden state (h_{t-1}), the attention context vector (c_t),

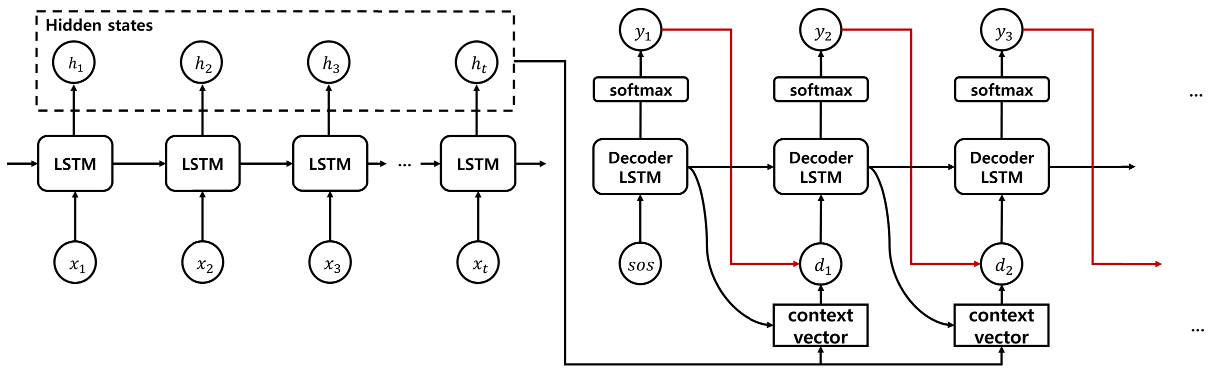

FIGURE 4. The structure of LSTM-based Seq2Seq model.

Algorithm 3 Reinforcement Learning

```
    Input: Input sequences $X$ and ground-truth output sequences $Y$ and pre-trained Seq2Seq model ($\pi_\theta$)
    Output: Trained Actor-Critic model
0:  procedure Training Steps
            Initialize the pre-trained Seq2Seq model $V_{\pi_\theta}(s)$ as actor network
            Initialize the pre-trained Seq2Seq model $Q_{\pi_\theta}(s, a)$ as critic network
0:    for Input sequences $(X, Y)$ do
0:        Get specific attack log $N$ from $X$
0:        for each time t ∈ T do
0:            Select action $a_t$ according to state $s_t$
0:            Execute action $a_t$, and receive reward $r_t$ and next state $s_{t+1}$
0:            Calculate the normalized rewards: $r\_normalized$ = ($r_t$ - min($r_t$)) / (max($r_t$) - min($r_t$) + $\epsilon$)
0:            Calculate $A(s_t, a_t)$ using the critic network following the equation 9
0:            Calculate detection rate $d_t$ using the testbed
0:            Get the base loss value $base\_loss = cross\_entropy(Y, \pi_\theta(X))$
0:            Calculate the loss value using custom loss function $custom\_loss = base\_loss * (1 - \alpha * r\_normalized)$
0:            Calculate policy gradient in actor using advantage function following the equation 20
0:            Update the model using $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
0:            Update the state $s_t = s_{t+1}$
0:        end for
0:    end for
0:  end procedure
```

TABLE 4. Description of ADFA-LD dataset.

Attack type	Number of system call sequences	Description
Adduser	91	Attackers tried to add new super user
Java-meterpreter	124	Java based Meterpreter
Hydra-FTP	162	Password brute force
Hydra-SSH	176	Password brute force
Meterpreter	75	Linux meterpreter payload
Web-shell	118	PHP file based webshell

and the current input (x_t). This results in the computation of h'_t. The LSTM expresses the hidden state as an equation.

$$h'_t = \mathrm{LSTM}(h_{t-1}, c_t, x_t) \tag{18}$$

C. REINFORCEMENT LEARNING

We used reinforcement learning to extract keywords for specific attacks and develop an optimal set of rules. The process is based on A2C, configuring the optimal policy based on critic feedback. The actor and critic networks in A2C are composed of pre-trained Seq2Seq models. In the case of an actor network, the probability θ for the action is calculated based on the pre-trained Seq2Seq for extracting a specific keyword is obtained. The critic network obtains value in state s_t, s_{t+1} based on the pre-trained Seq2Seq.

TABLE 5. Description of LID-DS 2021 dataset.

Attack type	Number of system call sequences	Description
CVE-2012-2122	20.7 (million)	Attackers repeatedly try to bypass authentication with the same incorrect password
CVE-2014-0160	1.9 (million)	Attackers obtained sensitive information from process memory
CVE-2017-7529	1.3 (million)	Vulnerable integer overflow vulnerability
CVE-2018-3760	115.1 (million)	Information leak vulnerability in Sprockets
CVE-2019-5418	400.9 (million)	File content disclosure vulnerability in action view
CVE-2020-9484	223.6 (million)	Remote code execution in apache tomcat
Bruteforce	9.5 (million)	Excessive authentication attempts
PHP_CWE-434	97.5 (million)	Upload file with malicious type
SQL injection	96.2 (million)	Improper SQL command

TABLE 6. Overall detection results on each log datasets.

Dataset	Accuracy	Precision	Recall	F1-score
ADFA-LD	0.971	0.959	0.968	0.964
LID-DS 2021	0.958	0.999	0.939	0.965

TABLE 7. Anomaly detection results on LID-DS log datasets.

Attack type	Accuracy	Precision	Recall	F1-score
CVE-2012-2122	0.999	0.999	0.998	0.999
CVE-2014-0160	0.840	1.000	0.760	0.864
CVE-2017-7529	0.990	1.000	0.989	0.994
CVE-2018-3760	0.992	1.000	0.986	0.993
CVE-2019-5418	0.978	1.000	0.966	0.983
CVE-2020-9484	0.832	1.000	0.750	0.857
Bruteforce	1.000	1.000	1.000	1.000
PHP_CWE-434	0.999	0.999	0.999	0.999
SQL injection	0.999	1.000	0.999	0.999

TABLE 8. Anomaly detection results on ADFA-LD log datasets.

Attack type	Accuracy	Precision	Recall	F1-score
Adduser	0.959	0.917	0.926	0.921
Java-meterpreter	0.973	0.947	0.973	0.960
Hydra-FTP	0.995	0.995	0.989	0.994
Hydra-SSH	0.967	0.957	0.962	0.959
Meterpreter	0.976	0.978	0.980	0.979
Web-shell	0.960	0.959	0.980	0.969

Algorithm 3 represents the process by which reinforcement learning creates a rule set. First, the actor and critic networks are initialized based on the pre-trained Seq2Seq model. Then, action a_t is performed at each time step t based on the attack log N that constitutes the input set get We use reinforcement learning to learn each keyword based on the reward and create a rule set based on the keyword. The reward is compared to the previously trained Seq2Seq model, r_1, and the rule set generated based on reinforcement learning is The reward was calculated based on the detected malware log sequence r_2 and the false positive value r_3 generated by the rule set. The equation 19 represents the reward calculation method in A2C.

$$R = ((0.3 * r_1) + (0.5 * r_2) - (0.2 * r_3)) + R * \gamma \tag{19}$$

Each reward value was normalized to prevent one reward from having too significant an impact. Additionally, we followed the previously learned Seq2Seq guide but adjusted it to avoid too much impact. This study calculated ratios of 30%, 50%, and 20%, respectively. Finally, a discount factor was applied to the existing reward to weight the recent reward further. In the critic network, calculate the advantage value based on equation 9. The critic network is performed based on the keywords and expected keyword values output through the pre-trained Seq2Seq model. We configured the output keyword based on the actor network into a detection rule and calculated the detection rate d_t of each attack payload in the testbed. Then, we calculate the policy gradient based on the calculated $A(s_t, a_t)$ and d_t. In order to use the detection rate of how many attacks were caught using extracted keywords when calculating policy gradient, we modified the equation 10 and composed it as follows. The equation 20 is a policy gradient calculation formula that applies the detection rate.

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau[\sum_{t=0}^{T-1} \nabla_\theta log\pi_\theta(a_t|s_t)d_t A(s_t, a_t)] \tag{20}$$

d_t indicates whether the output keyword helps detect an attack when calculating the policy gradient at each step. It is composed of a value between 0 and 1. Finally, the model is updated to create an optimal rule set from each attack log, and the state is updated.

V. EXPERIMENTS

We verified that the proposed HIDS framework precisely extracts keywords for attacks and creates an optimal rule set based on the keywords. Based on the ADFA-LD, and LID-DS 2021 datasets with various attack logs, we demonstrated that the proposed framework creates an optimal rule set for each attack and effectively detects the attack. In addition, we confirmed that the rule set actually created from various attack datasets detects the same type of attack.

A. DATASET

We utilized various datasets (ADFA-LD, LID-DS 2021) to check whether the proposed framework creates an optimal rule set for each attack.

1) ADFA-LD

ADFA-LD is a dataset composed of system call traces through various scenarios on Ubuntu [52]. ADFA-LD dataset

TABLE 9. Examples of rule set derived from CVE 2017-7529, LID-DS 2021 dataset.

Example of log sentence	Example of generated rule set

fd=20(<4t>172.24.0.5:38748->172.24.0.6:80)
res=342 data=Server: nginx/1.6.3...
sendfile > out_fd=20(<4t>172.24.0.5:38748->172.24.0.6:80)
in_fd=21(<f>/etc/nginx/html/index.html) offset=-623 size=21703
...
write < res=204 data= [alert] 110: *94 sendfile() failed
	$\{172.17.0.5 \wedge 172.17.0.6\} \bigvee \{\mathit{sendfile}\}$

used an auditing utility called Auditd to generate system call traces from specific processes.

Table 4 shows the information on the ADFA-LD dataset. ADFA-LD dataset includes system call traces of different types of attacks. Some of the attacks composed of the dataset derived from new zero-day malware.

2) LID-DS 2021

The LID-DS 2021 dataset for host-based intrusion detection systems consists of recently performed cyber attack methods and scenarios [53]. The LID-DS 2021 dataset consists of system call sequences generated on the host, and is labeled as normal logs and malicious logs.

Table 5 shows the information on the LID-DS 2021 dataset. The LID-DS 2021 dataset comprises 15 scenarios, 11 simple attacks, and four complex attacks. This study performs rule generation for specific attacks, so complex attacks are excluded. We excluded overlapping CWE-434 attacks and directory traversial attacks in dataset. Each dataset represents a specific attack, and normal logs and malicious logs are separated inside the dataset. To make the dataset smaller and improve keyword extraction, we removed redundant log sequences and eliminated unnecessary information such as event time, event number, and CPU values. Since the LID-DS dataset is less imbalanced than the existing one, we adjusted the training: test ratio to 8:2 using an internal log sequence.

B. EVALUATION METRICS AND RESULTS

The Recall-Oriented Understudy for Gisting Evaluation (ROUGE) method was used for summarization research using existing reinforcement learning [54]. The ROUGE method uses the n-gram technique to compare human-generated summaries with model-generated summaries and display them as numerical values. Identifying an attack through a rule set can prove to be a daunting task for an individual. Moreover, expressing it numerically can be highly subjective. Therefore, we verified the detection results based on the framework's rule set and evaluated the framework's performance using precision, recall, and F1-score. At this time, True Positive (TP) means that the rule set detects an attack, and True Negative (TN) does not detect a normal log. False Positive (FP) means that the rule set detects normal logs as attack logs, and False Negative (FN) means that it does not detect attack logs. Metrics is expressed as follows.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{21}$$

$$Precision = \frac{TP}{TP + FP} \tag{22}$$

$$Recall = \frac{TP}{TP + FN} \tag{23}$$

$$F1\text{-}Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{24}$$

Table 6 demonstrates the accuracy for each dataset. Our proposed framework showed sufficient performance in various attack logs. In particular, it showed excellent performance in the ADFA-LD dataset, which has a repetitive structure.

Table 7 shows the detection result table when it is performed based on each attack in the LID-DS dataset. The proposed framework achieved an excellent accuracy of over 99% in brute force, SQL injection, password bypass attack, and integer overflow. However, it showed relatively low accuracy in memory leak or remote code execution attacks, with an average of 83%. Notably, the proposed framework showed confusion when the constructed log sequence was similar to the normal or completely different. In summary, the proposed framework showed an average detection accuracy of 95.8% in LID-DS 2021. As a result, it was shown that most attack logs can be detected based on the rule set generated by the proposed framework.

Table 8 represents the detection result table for each attack on the ADFA-LD dataset. The proposed framework showed acceptable results when discovering repetitive word sets within the log sequence. However, in the case of the ADFA-LD dataset, repetitive word sets occurring in some attack log sequences were similar to normal log sequences, affecting accuracy.

VI. CONCLUSION AND FUTURE WORK

This paper utilizes reinforcement learning to develop an optimal set of rules tailored to a specific attack. We used a pre-trained LSTM based Seq2Seq model to initialize reinforcement learning efficiently and performed loss calculation using cross-entropy and testbed so that reinforcement learning does not deviate.

Notably, the rule set generated by the proposed framework based on each attack included important keywords such as the victim or IP address that each attack specifies. Table 9 shows a specific attack log and a rule set created based on the attack log. The example shows that a rule set was created based on the attacker's IP address and the victim's IP address. The experiment shows that the proposed framework extracts key keywords from attack logs and specifies attack targets.

For this reason, it was confirmed that new attacks performed by specific attackers or new attacks derived from existing attacks can also be detected. However, despite applying a ground-truth answer, when the attacker's attack fails and cannot be distinguished from normal logs, the accuracy is lowered, so there is a limitation that it is difficult to distinguish some using only the combination of AND and OR operators. Furthermore, the proposed framework showed good performance, but there are some limitations. First, a large amount of computation is required to process and train based on the tremendous amount of system call log sequences. In addition, considerable time is required to output a rule set based on the proposed framework. The proposed framework outputs an optimal rule set for a specific attack, but an update is required if a new keyword occurs despite the same attack type. Moreover, there is a problem with the dataset gradually becoming biased depending on the number of reinforcement learning updates. This study applied reinforcement learning to each dataset and created a rule set for each dataset. On the contrary, there was a problem: the provided dataset became biased as the amount of learning performed based on fixed learning data increased. In future work, we will apply a lightweight and real-time update method to the proposed framework.

REFERENCES

[1] H. Saleous, M. Ismail, S. H. AlDaajeh, N. Madathil, S. Alrabaee, K.-K.-R. Choo, and N. Al-Qirim, "COVID-19 pandemic and the cyberthreat landscape: Research challenges and opportunities," *Digit. Commun. Netw.*, vol. 9, no. 1, pp. 211–222, Feb. 2023.

[2] W. Tounsi and H. Rais, "A survey on technical threat intelligence in the age of sophisticated cyber attacks," *Comput. Secur.*, vol. 72, pp. 212–233, Jan. 2018.

[3] L. Vokorokos and A. Baláž, "Host-based intrusion detection system," in *Proc. IEEE 14th Int. Conf. Intell. Eng. Syst.*, May 2010, pp. 43–47.

[4] K. A. Jackson, *Intrusion Detection System (IDS) Product Survey*. Los Alamos, NM, USA: Los Alamos National Laboratory, 1999.

[5] M. Ozkan-Okay, R. Samet, Ö. Aslan, and D. Gupta, "A comprehensive systematic literature review on intrusion detection systems," *IEEE Access*, vol. 9, pp. 157727–157760, 2021.

[6] J. H. Ring, C. M. Van Oort, S. Durst, V. White, J. P. Near, and C. Skalka, "Methods for host-based intrusion detection with deep learning," *Digit. Threats, Res. Pract.*, vol. 2, no. 4, pp. 1–29, Dec. 2021.

[7] J. Hu, "Host-based anomaly intrusion detection," *Handbook Inf. Commun. Secur.*, vol. 2, pp. 235–255, Jan. 2010.

[8] J. Yu, W. Guo, Q. Qin, G. Wang, T. Wang, and X. Xing, "$*AIRS*$: Explanation for deep reinforcement learning based security applications," in *Proc. 32nd USENIX Secur. Symp.*, 2023, pp. 7375–7392.

[9] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Trans. Syst. Man, Cybern.*, vols. SMC-13, no. 5, pp. 834–846, Sep. 1983.

[10] A. Pal Singh and M. Deep Singh, "Analysis of host-based and network-based intrusion detection system," *Int. J. Comput. Netw. Inf. Secur.*, vol. 6, no. 8, pp. 41–47, Jul. 2014.

[11] M. R. Ayyagari, N. Kesswani, M. Kumar, and K. Kumar, "Intrusion detection techniques in network environment: A systematic review," *Wireless Netw.*, vol. 27, no. 2, pp. 1269–1285, Feb. 2021.

[12] M. Liu, Z. Xue, X. Xu, C. Zhong, and J. Chen, "Host-based intrusion detection system with system calls: Review and future trends," *ACM Comput. Surveys*, vol. 51, no. 5, pp. 1–36, Sep. 2019.

[13] E. Besharati, M. Naderan, and E. Namjoo, "LR-HIDS: Logistic regression host-based intrusion detection system for cloud environments," *J. Ambient Intell. Humanized Comput.*, vol. 10, no. 9, pp. 3669–3692, Sep. 2019.

[14] D. Park, S. Kim, H. Kwon, D. Shin, and D. Shin, "Host-based intrusion detection model using Siamese network," *IEEE Access*, vol. 9, pp. 76614–76623, 2021.

[15] S. Wang, Z. Wang, T. Zhou, H. Sun, X. Yin, D. Han, H. Zhang, X. Shi, and J. Yang, "THREATRACE: Detecting and tracing host-based threats in node level through provenance graph learning," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 3972–3987, 2022.

[16] T. Pasquier, X. Han, M. Goldstein, T. Moyer, D. Eyers, M. Seltzer, and J. Bacon, "Practical whole-system provenance capture," in *Proc. Symp. Cloud Comput.*, Sep. 2017, pp. 405–418.

[17] R.-H. Hwang, C.-L. Lee, Y.-D. Lin, P.-C. Lin, H.-K. Wu, Y.-C. Lai, and C. K. Chen, "Host-based intrusion detection with multi-datasource and deep learning," *J. Inf. Secur. Appl.*, vol. 78, Nov. 2023, Art. no. 103625.

[18] H.-K. Bui, Y.-D. Lin, R.-H. Hwang, P.-C. Lin, V.-L. Nguyen, and Y.-C. Lai, "CREME: A toolchain of automatic dataset collection for machine learning in intrusion detection," *J. Netw. Comput. Appl.*, vol. 193, Nov. 2021, Art. no. 103212.

[19] N. Joraviya, B. N. Gohil, and U. P. Rao, "DL-HIDS: Deep learning-based host intrusion detection system using system calls-to-image for containerized cloud environment," *J. Supercomput.*, vol. 80, no. 9, pp. 12218–12246, Jun. 2024.

[20] Z. T. Sworna, Z. Mousavi, and M. A. Babar, "NLP methods in host-based intrusion detection systems: A systematic review and future directions," *J. Netw. Comput. Appl.*, vol. 220, Nov. 2023, Art. no. 103761.

[21] A. Chawla, B. Lee, S. Fallon, and P. Jacob, "Host based intrusion detection system with combined CNN/RNN model," in *Proc. ECML PKDD Workshops*, Jan. 2019, pp. 149–158.

[22] I. Tahir and S. Qadir, "Machine learning-based detection of IoT malware using system call data," in *Proc. 4th Int. Conf. Digit. Futures Transformative Technol. (ICoDT2)*, Oct. 2024, pp. 1–8.

[23] G. Kim, H. Yi, J. Lee, Y. Paek, and S. Yoon, "LSTM-based system-call language modeling and robust ensemble method for designing host-based intrusion detection systems," 2016, *arXiv:1611.01726*.

[24] S. Lv, J. Wang, Y. Yang, and J. Liu, "Intrusion prediction with system-call Sequence-to-Sequence model," *IEEE Access*, vol. 6, pp. 71413–71421, 2018.

[25] Y. Zhang, S. Luo, L. Pan, and H. Zhang, "Syscall-BSEM: Behavioral semantics enhancement method of system call sequence for high accurate and robust host intrusion detection," *Future Gener. Comput. Syst.*, vol. 125, pp. 112–126, Dec. 2021.

[26] G. Sakarkar. (2021). *Advance Approach for Detection of Dns Tunneling Attack From Network Packets Using Deep Learning Algorithms*. [Online]. Available: https://gredos.usal.es/xmlui/bitstream/handle/10366/147246/Advance_Approach_for_Detection_of_DNS_Tu.pdf?sequence=1

[27] A. Zaboli, S. L. Choi, T.-J. Song, and J. Hong, "ChatGPT and other large language models for cybersecurity of smart grid applications," in *Proc. IEEE Power Energy Soc. Gen. Meeting (PESGM)*, Jul. 2024, pp. 1–5.

[28] R. Dijkman and A. Wilbik, "Linguistic summarization of event logs—A practical approach," *Inf. Syst.*, vol. 67, pp. 114–125, Jul. 2017.

[29] W. Meng, F. Zaiter, Y. Huang, Y. Liu, S. Zhang, Y. Zhang, Y. Zhu, T. Zhang, E. Wang, Z. Ren, F. Wang, S. Tao, and D. Pei, "Summarizing unstructured logs in online services," 2020, *arXiv:2012.08938*.

[30] S. Locke, H. Li, T. P. Chen, W. Shang, and W. Liu, "LogAssist: Assisting log analysis through log summarization," *IEEE Trans. Softw. Eng.*, vol. 48, no. 9, pp. 3227–3241, Sep. 2022.

[31] P. K. Mvula, P. Branco, G.-V. Jourdan, and H. L. Viktor, "Evaluating word embedding feature extraction techniques for host-based intrusion detection systems," *Discover Data*, vol. 1, no. 1, p. 2, Mar. 2023.

[32] S. Ghodratnama, A. Behehsti, and M. Zakershahrak, "A personalized reinforcement learning summarization service for learning structure from unstructured data," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jul. 2023, pp. 206–213.

[33] J. D. Chang, K. Brantley, R. Ramamurthy, D. Misra, and W. Sun, "Learning to generate better than your LLM," 2023, *arXiv:2306.11816*.

[34] H. Liang, X. Li, D. Xiao, J. Liu, Y. Zhou, A. Wang, and J. Li, "Generative pre-trained transformer-based reinforcement learning for testing web application firewalls," *IEEE Trans. Dependable Secur. Comput.*, vol. 21, no. 1, pp. 1–15, Jun. 2023.

[35] K. Ren, Y. Zeng, Z. Cao, and Y. Zhang, "ID-RDRL: A deep reinforcement learning-based feature selection intrusion detection model," *Sci. Rep.*, vol. 12, no. 1, p. 15370, Sep. 2022.

[36] Y.-F. Hsu and M. Matsuoka, "A deep reinforcement learning approach for anomaly network intrusion detection system," in *Proc. IEEE 9th Int. Conf. Cloud Netw. (CloudNet)*, Nov. 2020, pp. 1–6.
[37] M. Malik and K. S. Saini, "Network intrusion detection system using reinforcement learning techniques," in *Proc. Int. Conf. Circuit Power Comput. Technol. (ICCPCT)*, Aug. 2023, pp. 1642–1649.
[38] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, "Application of deep reinforcement learning to intrusion detection for supervised problems," *Expert Syst. Appl.*, vol. 141, Mar. 2020, Art. no. 112963.
[39] H. Studiawan, F. Sohel, and C. Payne, "Anomaly detection in operating system logs with deep learning-based sentiment analysis," *IEEE Trans. Dependable Secur. Comput.*, vol. 18, no. 5, pp. 2136–2148, Sep. 2021.
[40] Y. Kim, G. Park, and H. K. Kim, "Domain knowledge free cloud-IDS with lightweight embedding method," *J. Cloud Comput.*, vol. 13, no. 1, p. 143, Sep. 2024.
[41] J. Sun, Z. Xing, H. Guo, D. Ye, X. Li, X. Xu, and L. Zhu, "Generating informative CVE description from ExploitDB posts by extractive summarization," 2021, *arXiv:2101.01431*.
[42] P. Radhakrishnan and G. Senthil Kumar, "Machine learning-based automatic text summarization techniques," *Social Netw. Comput. Sci.*, vol. 4, no. 6, p. 855, Nov. 2023.
[43] Z. Li, C. Huang, S. Deng, W. Qiu, and X. Gao, "A soft actor-critic reinforcement learning algorithm for network intrusion detection," *Comput. Secur.*, vol. 135, Dec. 2023, Art. no. 103502.
[44] S. Yang, X. Duan, X. Wang, D. Tang, Z. Xiao, and Y. Guo, "Extractive text summarization model based on advantage actor-critic and graph matrix methodology," *Math. Biosciences Eng.*, vol. 20, no. 1, pp. 1488–1504, 2022.
[45] J.-R. Jiang and Y.-T. Chen, "Industrial control system anomaly detection and classification based on network traffic," *IEEE Access*, vol. 10, pp. 41874–41888, 2022.
[46] R. Mihalcea and P. Tarau, "TextRank: Bringing order into text," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Jul. 2004, pp. 404–411.
[47] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*.
[48] X. Zuo, S. Zhang, and J. Xia, "The enhancement of TextRank algorithm by using word2vec and its application on topic extraction," *J. Phys., Conf. Ser.*, vol. 887, Aug. 2017, Art. no. 012028.
[49] Y. Keneshloo, T. Shi, N. Ramakrishnan, and C. K. Reddy, "Deep reinforcement learning for Sequence-to-Sequence models," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 7, pp. 2469–2489, Jul. 2020.
[50] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2014, pp. 1–10.
[51] A. Dey, "Deep IDS: A deep learning approach for intrusion detection based on IDS 2018," in *Proc. 2nd Int. Conf. Sustain. Technol. Ind. 4.0 (STI)*, Dec. 2020, pp. 1–5.
[52] G. Creech and J. Hu, "Generation of a new IDS test dataset: Time to retire the KDD collection," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2013, pp. 4487–4492.
[53] M. Grimmer, T. Kaelble, F. Nirsberger, E. Schulze, T. Rucks, J. Hoffmann, and E. Rahm. (2021). *Lid-DS 2021*. [Online]. Available: https://dbs.uni-leipzig.de/files/research/publications/2022-9/pdf/CRITIS_2022_Extended_Abstract_LID-DS-2021.pdf
[54] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, 2004, pp. 74–81.

YONGSIK KIM received the M.S. degree from the School of Cybersecurity, Korea University, in 2024. His research interests include artificial intelligence and anomaly detection.

SU-YOUN HONG received the Ph.D. degree in applied electronic engineering from Korea Advanced Institute of Science and Technology, in 2013. She is currently affiliated with LIG Nex1 for cyber security. Her current research interest includes the automated action of cyber-threat/defense.

SUNGJIN PARK received the B.S. degree in cybersecurity from Ajou University, in 2020. He is currently a Research Engineer with LIG Nex1. His research interests include cyber threat intelligence, AI security, and offensive.

HUY KANG KIM (Member, IEEE) received the B.S. degree in industrial management, the M.S. degree in industrial engineering, and the Ph.D. degree in industrial and systems engineering from Korea Advanced Institute of Science and Technology (KAIST), in 1998, 2000, and 2009, respectively. He founded A3 Security Consulting, the first information security consulting company in South Korea, in 1999. Also, he was a member and the Last Leader of the KAIST UNIX Society (KUS), the Legendary Hacking Group, South Korea. Currently, he is a Professor with the School of Cybersecurity, Korea University. Before joining Korea University, he was the Technical Director (TD) and the Head of the Information Security Department, NCSOFT, from 2004 to 2010, one of the most famous MMORPG companies in the world. His recent research interest includes solving many security problems in online games based on the user behavior analysis.