Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Domain knowledge free cloud-IDS with lightweight embedding method

Yongsik Kim[1], Gunho Park[1] and Huy Kang Kim[1*]

Abstract

The expansion of the cloud computing market has provided a breakthrough in efficiently storing and managing data for individuals and companies. As personal and corporate data move to the cloud, diverse attacks targeting the cloud have also increased for heist beneficial information. Therefore, cloud service providers offer protective environments through diverse security solutions. However, security solutions are limited in preventing advanced attacks because it is challenging to reflect the environment of each user. This paper proposes a Cloud Intrusion Detection System (C-IDS) that adapts to each user's cloud environment and performs real-time attack detection using Natural Language Processing (NLP). Notably, the C-IDS learns the deployed client environment logs and detects anomalies using the Seq2Seq model with BI-LSTM and Bahdanau attention. We used multiple domain datasets, Linux, Windows, Hadoop, OpenStack, Apache, OpenSSH, and CICIDS2018 to verify the performance of the C-IDS. C-IDS consists of a 'recognition' that identifies logs in the deployed environment and a 'detection' that discovers anomalies. The recognition results showed an average accuracy of 98.2% for multiple domain datasets. Moreover, the detection results based on the trained model exhibited an average accuracy of 94.2% for the Hadoop, OpenStack, Apache, and CIC-IDS2018 datasets.

Keywords Cloud computing, Cyber security, Natural language processing, Intrusion detection system, Anomaly detection, CICIDS-2018 dataset, System log analysis

Introduction

Technological development allows people to access data efficiently, and diversifying data utilization methods increases the data size [1]. As the size of data increases, databases require fast retrieval, real-time processing, and an understanding of big data. However, the storage and management of large-scale data using traditional data management structure is complicated [2]. Because traditional databases are limited in their ability to support complex data structures, data administrators require better data management solutions.

Therefore, cloud computing has emerged as an efficient data management method [3]. Cloud computing, which provides computer resources at an acceptable price, became an opportunity to transfer locally managed data to virtualized servers. With the advent of cloud computing, users can manage and store data through various cloud computing services without having to understand complex network and server configurations. Moreover, users have moved critical data, such as private customer information to the cloud for convenient utilization. As critical data move to the cloud, cloud-targeted attacks have also increased for stealing critical data [4]. For these reasons, Cloud Service Providers (CSP) provides various security solutions to users based on convenience to protect users' data. Using security solutions, users can configure a security environment that can countermeasure most attacks.

However, cloud-targeted attacks have become more intelligent and advanced, and it is difficult to construct a robust security environment using existing security

*Correspondence:
Huy Kang Kim
cenda@korea.ac.kr
[1] School of Cybersecurity, Korea University, Anam-ro, Seoul, Korea

solutions [5]. Even though the user builds a security policy using existing security solutions, adversaries can break in and deprive critical information owing to some indifference to the components. Internet environment-based cloud services must consider various components of information synchronization, such as processing methods according to each user's request and migration between user-requested hardware [6].

For these reasons, cloud service users want environmentally suitable security solutions that consider each user's environmental components [7]. However, constructing a user-suitable security environment is complex. Designing a user-suitable security environment requires an understanding of each user's environment. That is, a user-suitable security environment should be able to identify the components of the user environment and analyze various large-scale cloud logs to detect acute attacks. Organizing a new security environment requires additional time, resources, and techniques to efficiently process large-scale cloud logs. These requirements can burden users when solving various problems using only their security knowledge.

In this study, we propose a C-IDS that automatically processes various large-scale cloud log datasets in real time and perform attack detection using Natural Language Processing (NLP). C-IDS is automatically deployed using Infrastructure as Code (IaC), understands the user's environment by itself in the deployed environment, and detects anomalies. C-IDS detects anomalies according to the user's environmental components without user intervention; users can check malicious behaviors occurring in various layers without technical security knowledge.

We started with a review of related work related to Non-NLP based IDS and NLP-based IDS ("Related work" section). Our proposed C-IDS framework is detailed in "C-IDS framework" section, and our detection model BI-LSTM based Seq2Seq model is explained in "BI-LSTM based Seq2Seq model" and "Bahdanau attention" sections. We propose the distribution methods to deploy the multiple domains in the cloud in "Multi domain distribution with IaC" section. We present the log recognition and anomaly detection results for multiple domain log datasets in "Log recognition result" and "Anomaly detection result" sections respectively. In "Discussion" section we discuss the model's recognition methods. Finally, we make concluding remarks and show limitations in "Conclusion and future work" section.

Our study's main contributions are as follows:

- We proposed a real-time intrusion detection system for complex cloud environments based on the Seq2Seq model with BI-LSTM and Bahdanau attention. The Seq2Seq model classifies each cloud environment and detects malicious behaviors in cloud logs.
- We converted various raw cloud log information into a machine-readable data format using label encoding. The label encoding comprises a unique word set that identifies and counts the occurrence of each word.
- We used IaC to deploy our proposed framework to multiple cloud environments automatically. With our deployment methods, users can build a user-suitable security framework without additional cloud security knowledge.

Related work

Non-NLP based IDS

As large-scale data move to the cloud, diverse problems and vulnerabilities occur. Therefore, various studies have proposed identifying and classifying problems occurring in the clouds.

Li et al. presented the cloud security vulnerabilities in various cloud environments using multiple security assessment tools [8]. They argued that security accountabilities exist for each service provider and consumer following the service model. To verify the aforementioned results, the authors classified experimental cloud environments into laboratory, On-campus, and Off-campus, and assessed vulnerabilities using the vulnerability assessment tools Nessus, NMAP, and Nikto. The experiments showed vulnerable common ports and HTTP methods, and it was found that various network vulnerabilities exist in the internet environment-based cloud environment. Ali et al. presented cloud security challenges through a diverse classification of cloud computing security challenges [9]. They introduced vulnerabilities not only in internet communication in the cloud but also in the communications with cloud infrastructure components. Furthermore, they noted security problems related to the cloud architecture and Service Level Agreement (SLA). In conclusion, numerous components in various cloud layers must be considered to solve security problems. However, many organizations tend to ignore the essential practices and techniques that need to be implemented when using a cloud system, and attackers steal users' critical data mainly through data control, account hijacking, data sanitization, and malicious insider attacks [10].

Considering the vulnerabilities in the cloud, complex components occurring in various environments must be considered to detect vulnerabilities in the cloud. In the case of a cloud environment, it is possible to apply vulnerability detection methodologies that have already been proposed for each environment. However, it is difficult to apply the existing methodology to various environments because it specifies a certain environment.

Thus, various methodologies have been proposed to detect vulnerabilities in the cloud, supplement the security systems that comprise the cloud environment, and prevent attacks.

Modi et al. presented a cloud intrusion that affected cloud resources, service availability, confidentiality, and integrity [11]. They described diverse methods of integrating and applying IDS or IPS into the cloud and argued that it is challenging to solve the cloud security problem using only a firewall. Faber et al. proposed an autoencoder-based IDS model for a mobile cloud computing environment [12]. They consider internal network traffic and mobile device traffic, creating features based on time windows. To verify the proposed model's performance, they trained the model using only normal packets generated from the CICIDS2018 data; then evaluated by Area Under the Receiver Operating Characteristic Curve (ROC-AUC) [13]. The proposed model exhibited good capabilities against most network attacks without deep packet inspection. Nevertheless, the model exhibited low capabilities for attacks that cannot be easily distinguished from normal packets, such as brute force attacks and cross-site scripting attacks (XSS). Huang et al. proposed a 'Relaxed form of Linear Programming Support Vector Data Description' (RLPSVDD) model that uses time series concept data to detect abnormal performance metrics of cloud services [14]. They argued that Support Vector Data Description (SVDD) one-class classification has weaknesses: an imperfect false alarm rate and computational complexity in time series abnormality detection. The proposed model achieved high accuracy in various datasets using RLPSVDD; however, there was little mention of the data processing methods.

Some IDS for cloud environments showed excellent performance; paradoxically, there were limitations because of the characteristics of the cloud environment [15]. It is challenging to satisfy the security requirements of dynamically added and removed instances. In addition, the virtualization of cloud computing leads to improper access control and requires interaction with security administrators. To overcome the limitations of existing IDS, CSPs build an intrusion detection system by applying a Security Information and Event Management (SIEM) structure to the cloud. SIEM is a monitoring system that detects threats in advance by collecting and analyzing events that occur in established security and IT systems [16]. Through SIEM, the user analyzes widespread log data on the cloud and identifies possible security threats. Lee et al. developed an AI-based SIEM system using a Fully Connected Neural Network (FCNN), Convolutional Neural Network (CNN), and Long Short-Term Memory (LSTM) [17]. Using the TF-IDF mechanism, they processed large-scale security logs generated in various environments as input values. Furthermore, they verified the proposed system's threat detection performance based on public and collected network datasets in the real world. The proposed model showed an average accuracy of 95% for the entire dataset but a low F1-score for the actual network dataset, owing to a low true positive rate. Consequently, SIEM in the cloud helped differentiate between true and false security alerts. However, SIEMs also require interactions with security experts to redeem security based on detected threats. The system is robust only against the attacks adapted, so it has limitations in detecting continuously evolving attacks. Okey et al. proposed a transfer learning IDS based on CNN architecture to detect various attacks occurring in the Cloud IoT environment [18]. The author proposed an ensemble model using five CNN models: VGG16, VGG19, Inception, MobileNet, and EfficientNets. They verified the proposed model using the CICIDS2017 and CICIDS2018 datasets. Notably, the author generated data using the Synthetic Minority Over-sampling technique (SMOTE) to resolve the imbalance in each dataset. As a result, the proposed transfer learning IDS showed high F1-scores of 1.00 and 0.99 in the CICIDS2017 and CICIDS2018 datasets, respectively. Bakro et al. proposed a cloud IDS incorporating SMOTE technology [19]. The author performed feature selection using information Gain (IG), Chi-Square (CS), and Particle Swarm Optimization (PSO), and SMOTE was used to supplement the imbalanced dataset. The training dataset generated through SMOTE was trained on a Random Forest (RF) model, and the proposed methodology was verified using the UNSW-NB15 and Kyoto datasets. As a result, the proposed methodology achieved 98% and 99% accuracy in multi-class classification, respectively.

Although existing detection methods show excellent ability in a specific domain, it is difficult to detect anomalies in large numbers of cloud logs; thus, extensive research has been proposed to process large-scale cloud logs effectively to detect anomalies [20]. Lu et al. presented a CNN-based model that performs anomaly detection in big data system logs [21]. The proposed model automatically learns the system log event relationship and detects anomalies with high accuracy. To process the collected logs into the CNN model input, the log key in the collected session was vectorized into a dimension vector composed of a 29×128 two-dimensional codebook. By contrast, a linguistic approach that treats and processes each log as a plain text document can extract the grammatical structure and context of log events information [22]. He et al. evaluated an effective log parsing method for large-scale logs by using four existing tools [23]. Analyzing large-scale log messages effectively requires automated log analysis through data

mining, in contrast to existing manual inspections. They compared the performance of existing tools: Simple Logfile Clustering Tool (SLCT), Iterative Partitioning Log Mining (IPLoM), Log Key Extraction (LKE), and LogSig. However, studies analyzing log parsers are limited by the lack of publicly available log parsing tools and benchmark datasets for analysis [24]. Table 1 is a summary of non-NLP based IDS. The previous studies were mainly conducted in specific cloud environments. That is, applying existing detection methods with considering various cloud environment components is complicated. Therefore, an NLP-based IDS has been proposed that can analyze and utilize logs for each characteristic [25].

NLP based IDS

NLP enables machines to understand and interpret natural language for tasks such as text classification, summarization, and translation [26]. NLP, which analyzes and understands the structure of human language based on large amounts of language data, has been applied in various fields, such as data mining and emotion identification. Researchers have used NLP to overcome the limitations of the existing passive security monitoring process with language-based data, such as code or security logs [27]. An NLP-based security process with well-defined formats can detect security flaws more reasonably than existing methods. NLP is also used in digital forensics [28], privacy data preservation [29], and bug reports [30].

Das et al. proposed an intrusion detection system using NLP and ensemble-based machine learning [31]. The proposed model deviates from pattern matching or blacklist-based detection, which depend on attack technology, and identifies unconfirmed attacks by analyzing the characteristics of the log. However, they did not demonstrate whether the proposed model can be used for various logs by testing only HTTP requests. Wang et al. suggested an offline feature extraction model LogEvent2vec, that improves Word to Vector (Word2Vec), a log feature extraction technique [32]. The LogEvent2vec model performs only one transformation by using the log event as an input value, unlike the word2vec model, which uses a list of words in a log. As a result, the LogEvent2vec model performed better than word2vec in terms of computational time and accuracy. Ryciak et al. introduced an anomaly detection model by improving the LogEvent2vec

Table 1 Summary of non-NLP based IDS

Year	Reference	Proposal	Limitations
2010	Li et al.	Analyzed security vulnerabilities occurring in cloud environments using various security assessment tools and present security responsibilities between service providers and consumers through experiments.	-
2013	Modi et al.	Investigated cloud intrusions that impact cloud resources, service availability, confidentiality, and integrity and explain how to apply IDS/IPS integration.	-
2015	Ali et al.	Classified cloud computing security challenges and presents vulnerabilities arising from communication with cloud infrastructure components and Internet communication.	-
2016	He et al.	Evaluated the automation of log analysis through data mining, moving away from traditional manual inspection for large-scale log message analysis, and compare four existing log parser tools.	There is a lack of publicly available log parsing tools and benchmark datasets for analysis.
2017	Huang et al.	Proposed an RLPSVDD model to detect abnormal performance indicators of cloud services using time series concept data.	Little mention of the data processing methods
2018	Lu et al.	Proposed a CNN-based model that performs anomaly detection in big data system logs and detects anomalies with high accuracy by automatically learning log event relationships.	-
2019	Lee et al.	Developed an AI-based SIEM system using FCNN, CNN, and LSTM and verified threat detection performance by processing large-scale security logs generated in various environments with the TF-IDF mechanism.	The system is robust only against the attacks adapted
2021	Faber et al.	Proposed an auto-encoder-based IDS model considering network and mobile device traffic in a mobile cloud computing environment.	The proposed model does not detect attacks sufficiently that are difficult to distinguish from normal packets, such as XSS and brute force attacks.
2023	Okey et al.	Proposed a transfer learning IDS based on CNN architecture to detect various attacks occurring in the Cloud IoT environment.	-
2023	Bakro et al.	Proposed a cloud IDS incorporating SMOTE technology.	-

model [33]. They applied sequence length shortening and fasttext vectorization to the LogEvent2vec model. The proposed model was tested on Blue Gene/L datasets with Gaussian naive Bayes (GNB), RF, and MultiLayer Perceptron (MLP) classifiers, and the MLP model showed the best performance at short sequence lengths. Lv et al. designed an effective system call prediction model using a Sequence-to-Sequence (Seq2Seq) model [34]. They used the Seq2Seq model to solve sequence dependence in short-term system call analysis. Verification of the proposed model using the Australian Defense Force Academy-Linux Dataset (ADFA-LD) dataset showed good performance in CNN, Recurrent Neural Network (RNN), Support Vector Machine (SVM), and RF models. Chaudhari et al. developed a system-call sequence-based IDS to detect attacks related to virtual machines in a cloud infrastructure environment [35]. The author composed the system call sequence as a Bag of N-grams and then composed the related features using TF-IDF. Afterward, an ensemble model consisting of LSTM and RF model was used to detect attacks within the system call sequence. The author used the ADFA-LD dataset to verify the proposed framework. As a result, the proposed framework showed a high accuracy of 97.2% and a false positive rate of 2.4%.

However, presenting a practical solution using only NLP is challenging [36]. Text elements have different relevance to security because the words constituting the security log have an incomplete sentence structure, and there are cases in which they are not expressed in human-readable words. Some studies have emphasized specific text elements to focus on their each relevance; however, then have shown limitations for long sentences. However, there are methodologies to evaluate the relevance between input texts and to consider weights using machine learning; it is difficult to find evidence for the output. Thus, we consider an attention mechanism that partially interprets and explains the neural network behavior. The attention mechanism allows the model to focus on certain parts of the input during processing [37]. In the case of large-scale cloud data, the preprocessing task is highly time-consuming, and there is a problem in that the result value of the trained model depends on the amount of data [38]. The attention mechanism can reduce the computation resource by sub-sequence in a long sequence and improve the model's detection accuracy according to the relevance of the inputs. Among the attention mechanisms, the Bahdanau attention has the advantage of identifying unique elements in each security log. Based on the Bahdanau attention mechanism, users can visually check risk factors in attack logs. Moreover, Bahdanau attention improved the translation performance of the model using a variable length vector, unlike the existing attention mechanism [39]. In this study, Bahdanau's attention was used to process cloud log datasets of different lengths. Table 2 is a summary of NLP based IDS. A recent study used NLP to analyze the variable and complex characteristics of sentences composed in a human-readable format in a cloud environment so that machines can also read them. Notably, the security log was analyzed using linguistic features containing each attack's characteristics so each attack could be classified based on the security log. This paper analyzes and detects attacks using security logs containing each anomaly's characteristics.

Although an NLP based model is adequate for anomaly detection, it is challenging to deploy the model in each environment. Thus, various methods have been proposed to deploy cloud security models effectively. We focused on IaC among the various deployment methods. IaC refers to the management and provisioning of infrastructure through code. With IaC, the operator can organically manage, develop, and build a threat detection infrastructure at an optimal cost. For example, Alonso et al. introduced a method for deploying and monitoring IaC in a cloud continuum environment [40]. The deployment method using IaC has the benefit of supporting broad and multi-stage infrastructure layers in the cloud

Table 2 Summary of NLP based IDS

Year	Reference	Proposal	Limitations
2018	Lv et al.	Designing an effective system call prediction model to resolve sequence dependencies in short-term system call analysis using the Seq2Seq model.	-
2020	Das et al.	Proposed an NLP and ensemble-based machine learning model that identifies unidentified attacks by analyzing log characteristics.	-
2020	Wang et al.	Proposed LogEvent2vec, an offline log feature extraction technique that improved Word2Vec, and performed one transformation using log events as input.	-
2022	Ryciak et al.	Improved the LogEvent2vec model and proposed an anomaly detection model that shortened the sequence length and applied fasttext vectorization.	-
2023	Chaudhari et al.	Developed a system-call sequence-based IDS to detect attacks related to virtual machines in a cloud infrastructure environment.	-

continuum. Cauli et al. proposed a model that validates cloud deployment and the safety of cloud configuration using the proprietary language of Amazon Web Services (AWS) CloudFormation [41]. Based on the above study, we comprise cloud log data in a machine-readable format using NLP and implement an IDS that detects security threats in multiple layers.

Methodology

C-IDS framework

This study uses the C-IDS framework to process cloud log data and detect attacks in real-time. The proposed C-IDS framework focuses on the ability of Neural Machine Translation (NMT) among natural language processing methods. NMT can directly learn mapping from the input text to the related output text in an end-to-end method, breaking away from the limitations of phrase-based machine translation, which translates sentences in word and phrase units [42]. Considering the advantages of NMT, we monitored the extent to which the currently trained model attention feature words were extracted from the incoming logs in real-time, which can use various structures. Moreover, we identified the anonymous log information as heat map images to check for unrecognized words.

In previous log recognition studies, the meaning of words required to analyze and understand security logs was lost [43]. For example, unknown error codes can be treated as abnormal or replaced with words that humans can distinguish [44]. They analyzed the security logs using an existing language model by deleting abnormal error codes. Meaningful words that distinguished normal from abnormal words were also excluded. In this study, we trained the model with all sentences, including 'error codes,' based on the words extracted from the actual operating environment without prior classification and processing. Moreover, we preprocessed log data according to the main characteristics of each environment, making it possible to learn unique error codes in each environment and log sentences that humans cannot read. The advantage of this method is that it can be applied to attack techniques using new patterns or vulnerabilities.

Figure 1 presents an overview of the C-IDS Framework. The C-IDS framework consists of a 'first learning mechanism' that performs initial training on the Seq2Seq model based on log data generated in each environment; a 'detection mechanism' performs actual detection based on the trained Seq2Seq model.

The First learning mechanism trains the model based on existing norm log data in each environment when the model is deployed. Norm log data are normal logs that can occur in each environment, including error codes. The First learning mechanism consists of sentence preprocessing that refines the log data into input values and a training process that trains a Seq2Seq model with Bi-directional Long Short-Term Memory (BI-LSTM) and attention methods. In sentence preprocessing, the log data collected from each environment are converted into a numeric array through label encoding for use as input values for the model. The label encoding technique assigns unique integer values to each word. Label encoding is an effective way to convert a security log with many unique words into numeric form [45]. Figure 2 shows an example of the label encoding used to convert a raw cloud log into a numerical array. Label encoding requires a rule that assigns unique integers to words. Thus, we organized a unique word set extracted from the log datasets collected from each environment. We then used the index numbers of the words included in the unique word set as integer values. To obtain unique words in each log, we tokenize the input log sentences based on the spaces and extracted words. Notably, we used all the words present in the log data, including error codes, to reduce the loss of meaning. The unique word set consists of words that can appear in all domains within a cloud environment. Creating a unique word set for a user's multi-domain cloud environment can be utilized across all composed environments. Subsequently, we trained the Seq2Seq model using the input values generated during the training process.

In the detection mechanism, the trained Seq2Seq model performed real-time log recognition and anomaly detection in the deployed environment. Through the Seq2Seq model, we checked whether the log is in a suitable format for spawning in each environment and a common log that usually occurs. The detection mechanism consists of a 'sentence preprocess' that preprocesses logs as input values in real time and a 'recognition and detection process' that recognizes and detects new logs based on input values. First, the new log sentence is converted into a numeric array by performing the same sentence preprocessing as in the first learning process. At this time, it is checked whether a new word exists in each log sentence, and if a new unique word exists, it is added to the unique word set for label encoding.

The recognition and detection process recognizes and detects input values in real-time using a pre-trained Seq2Seq model. First, the trained Seq2Seq model in the first learning mechanism outputs the feature words of the log sentence based on the input values. The trained Seq2Seq model uses words representing each environment's characteristics and finds and outputs feature words with each environment's characteristics. If a feature word does not exist in the log sentences, it is recognized as another log because it contains words that cannot occur in the learning environment. We expressed

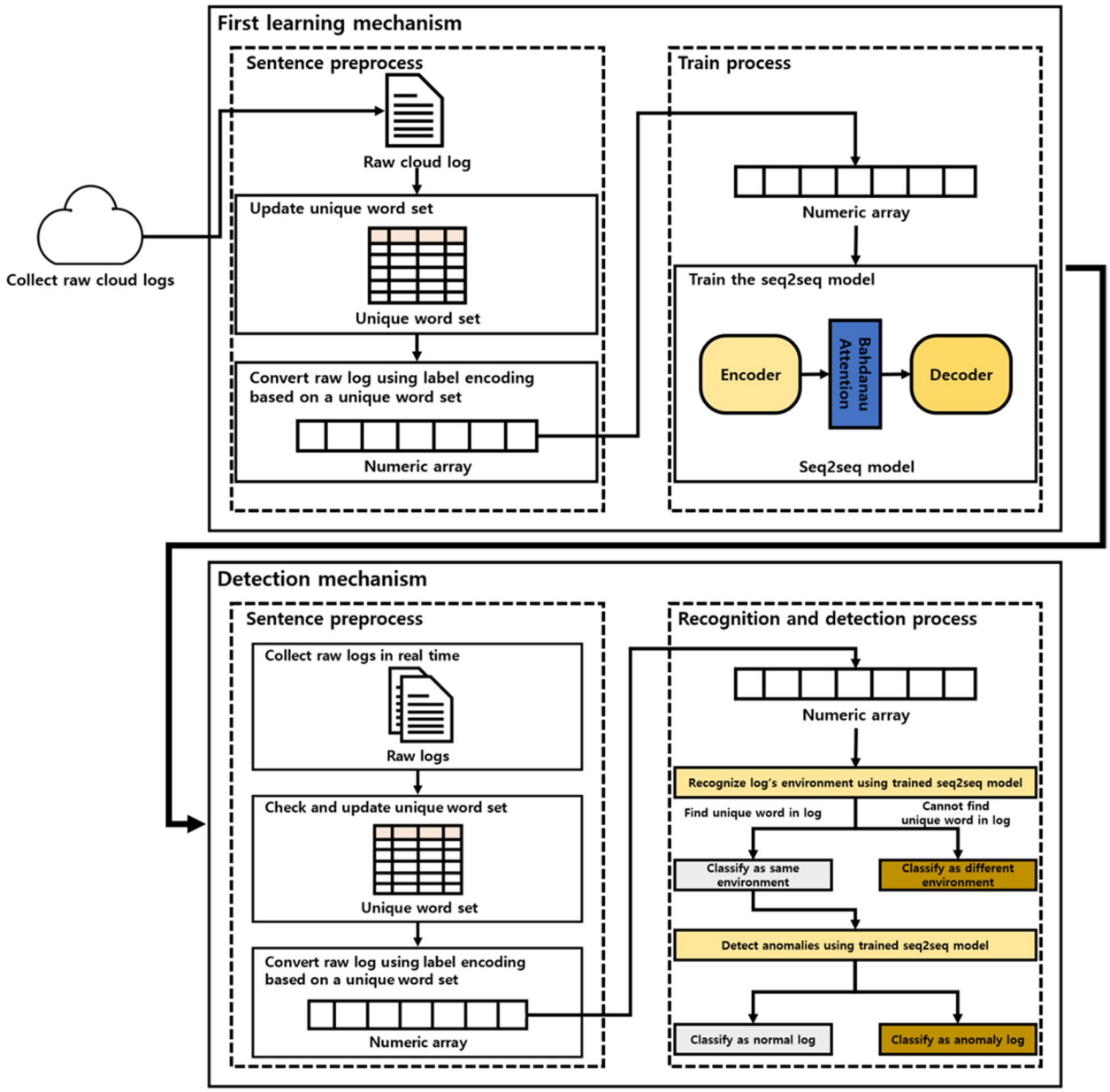

Fig. 1 The overall structure of the C-IDS framework. With cloud system's log data, the first learning mechanism performs initial training on the Seq2Seq model. With the trained Seq2Seq model, the detection mechanism performs anomaly detection

the results visually to confirm that the model recognized them precisely. Based on this methodology, new attacks can be detected without any model modification using only normal data generated in the existing environment. In this experiment, the trained Seq2Seq model uses normal log printed out feature word for new log sentences. The process then checks whether the output feature words are composed of common words that are mainly used in each environment. If a feature word is uncommon, it is classified as an anomaly.

BI-LSTM based Seq2Seq model

This study predicts and outputs feature words with environmental characteristics from log sentences to determine whether log data occur appropriately in each environment. However, these methodologies must verify that the trained model can understand all log sentences and extract words. Therefore, we created a sequence model that represents each sentence thoroughly. However, the simple RNN model has limitations in that the lengths of the input and output vectors must be the same,

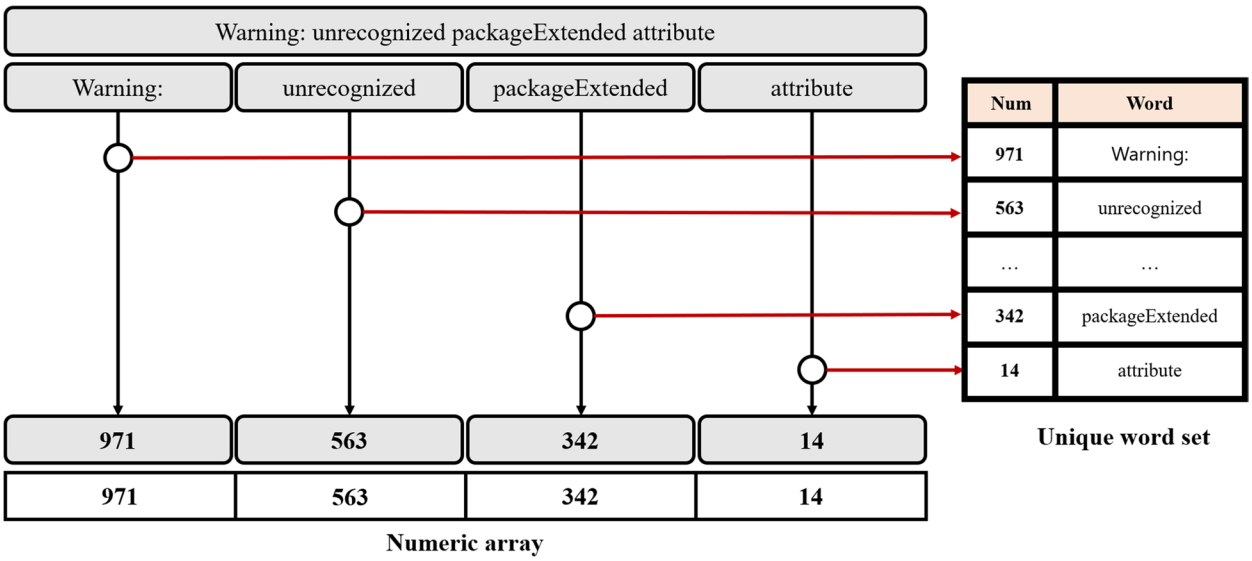

Fig. 2 The structure of label encoding. We tokenized log sentences based on white-space to convert words to a numeric array. In label encoding, we used all words presented in the log data (e.g., error codes) to reduce the loss of meaning of words

which is a long-term dependency problem. We implemented a Seq2Seq model using the LSTM structure to overcome the limitations of the RNN.

The Seq2Seq model generated output sentences for each sentence [46]. The Seq2Seq model is an encoder-decoder model composed of an encoder and a decoder. The encoder compresses the sequential input data and expresses them as a fixed context vector, whereas the decoder creates a new sequence from the context vector. The Seq2Seq model has the benefit of understanding input sentences, but information loss may occur in log sentences because they do not have the intact form of a sentence. To use log sentences composed of a series of intentional words without losing information, all words must be trained uniformly. Therefore, we construct the encoder as bi-directional structure to understand log words in a balanced manner. The bi-directional structure can consider past and future information of the input value [47].

Figure 3 shows the implemented BI-LSTM structure. A backward LSTM layer is added to the encoder composed of forward LSTM and concatenate the hidden state value output from forward and backward LSTMs. The concatenated hidden state computes a context vector, and the decoder predicts and outputs a feature word for each log sentence based on the context vector.

Bahdanau attention

The BI-LSTM Seq2Seq model is suitable for standing log sentences composed of diverse words however, problems arise when the log sentence length is long and the layers are deep. Significantly, there is a bottleneck problem in that the encoder has too much information to compress, and the decoder uses only part of the information compressed by the encoder for prediction. Consequently, the quality of the output value is degraded owing to information loss when the input sentence is long.

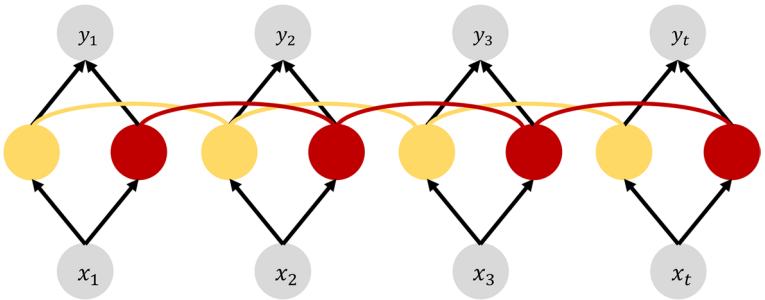

Fig. 3 The structure of BI-LSTM. The odd level of nodes represents forward LSTM, even level of nodes represents backward LSTM

In this study, we used the 'Bahdanau attention' among the attention methods to focus on feature words that reflect each environment simultaneously. The Bahdanau attention mechanism is more accurate in detecting logs of various types of attacks than the simple model structure [48]. We applied an attention mechanism that keeps a hidden state vector so that the decoder can refer to the context vector when predicting a word.

Figure 4 shows the structure of the Bahdanau attention mechanism applied to the Seq2Seq model. Bahdanau attention 'pay attention' to certain parts of the log input while processing it rather than using the entire input equally. Bahdanau attention is helpful when the input has a variable length, and the model needs to focus on certain parts of the input.

Figure 5 shows the Bahdanau's attention obtained by applying a security log. We used security logs. The length of each log is variable and requires attention to a specific word to train the model accurately. During training, the Bahdanau attention mechanism is implemented using an additional set of parameters called 'attention weights'. These attention weights are used to compute the

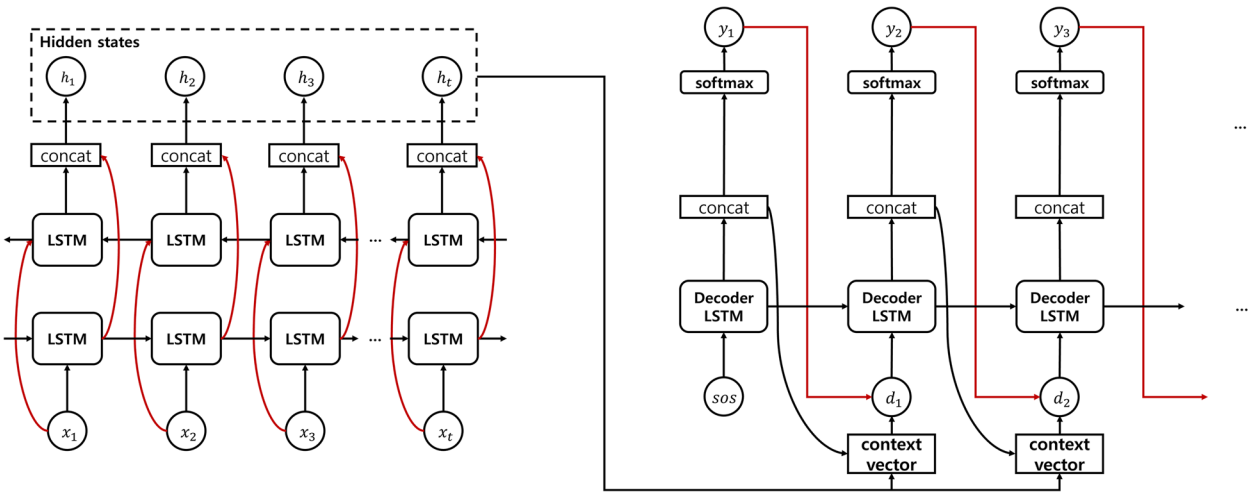

Fig. 4 The structure of the Seq2Seq model with Bahdanau attention mechanism. Bahdanau attention stores the encoder's hidden states so the decoder can refer to a specific part of the input sentence. The Seq2Seq model's decoder refers to the previous predicted word, the previous hidden states, and the encoder's hidden states when predicting the next word

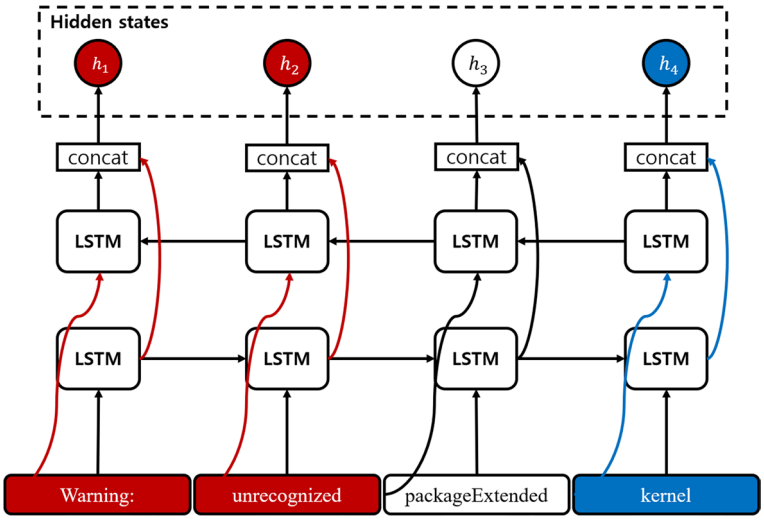

Fig. 5 Illustration of applying Bahdanau attention mechanism to security log. In this experiment, we use attention weight to focus on the extracted keywords from security log

weighted sum of the input, which is then used as part of the model's prediction. The attention weights are computed using a set of equations that involve the input and hidden states of the model at each time step.

The Bahdanau attention mechanism is implemented using the following equations. First, the attention weights are computed using the current hidden state of the model, h_t, and the input, X. The attention weights are computed using the dot product between h_t and each element in X, followed by a softmax function:

$$a_t = \text{softmax}(h_t^T X) \tag{1}$$

The attention context vector c_t is then computed as a weighted sum of the inputs, using the attention weights as weights:

$$c_t = \sum_{i=1}^{T_x} a_{t,i} x_i \tag{2}$$

The final hidden state of the model, h'_t, is then computed using the previous hidden state, h_{t-1}, the attention context vector, c_t, and the input at the current time step, x_t. The hidden state is expressed as an equation through the LSTM.

$$h'_t = \text{LSTM}(h_{t-1}, c_t, x_t) \tag{3}$$

however, it is challenging to deploy the framework separately. Therefore, we propose a managing server that automatically identifies the cloud environment and then applies and manages the framework to each environment. However, applying framework automation is difficult even when using a managing server, because each user's environment configuration method can differ. Therefore, we use IaC to deploy a managing server in the user environment. IaC can provide infrastructure using high-level codes and standardize the application development environment. We deployed a managing server using Terraform among the IaC tools and distributed the framework to each environment using the managing server. Terraform can be applied to various cloud services, such as AWS, Google Cloud Platform, and Microsoft Azure without being limited to specific cloud services. Moreover, Terraform can declare and manage resources in the cloud using code.

We used the managing server to automate the entire analysis, learning, and deployment processes. Figure 6 shows the distribution of the proposed framework in each environment using the IaC and managing server. First, the distributed management server scans the user's cloud environment before deploying the framework in each environment. Subsequently, the managing server deploys the framework in each identified environment. The deployed framework performed training based on the normal log that existed in each environment. Finally, the framework performs log recognition and anomaly

Algorithm 1 Mechanism of managing server

Multi domain distribution with IaC

```
procedure Initialize framework deployment with IaC
    for each environment E in the deployed cloud do
        Scan the current state S of E
        if S is exists then
            Deploy the framework in E using the managing server
        Train the framework on normal logs existing in E
        for Log event L in E do
            Use the trained framework to recognize and detect anomalies in L
        while The framework is active in E do
            Send S to the managing server in real-time, ensuring its proper functioning
            if The managing server receives S as a failure then
                Check deployment as unsuccessful
                Retry deployment using IaC until successful
                if Retry is successful then
                    Continue normal framework operations
                else
                    Alert the managing server and log the failure for further analysis
```

This framework performs recognition and detection based on logs generated in various environments detection in each environment based on the trained model. Significantly, the managing server checks the

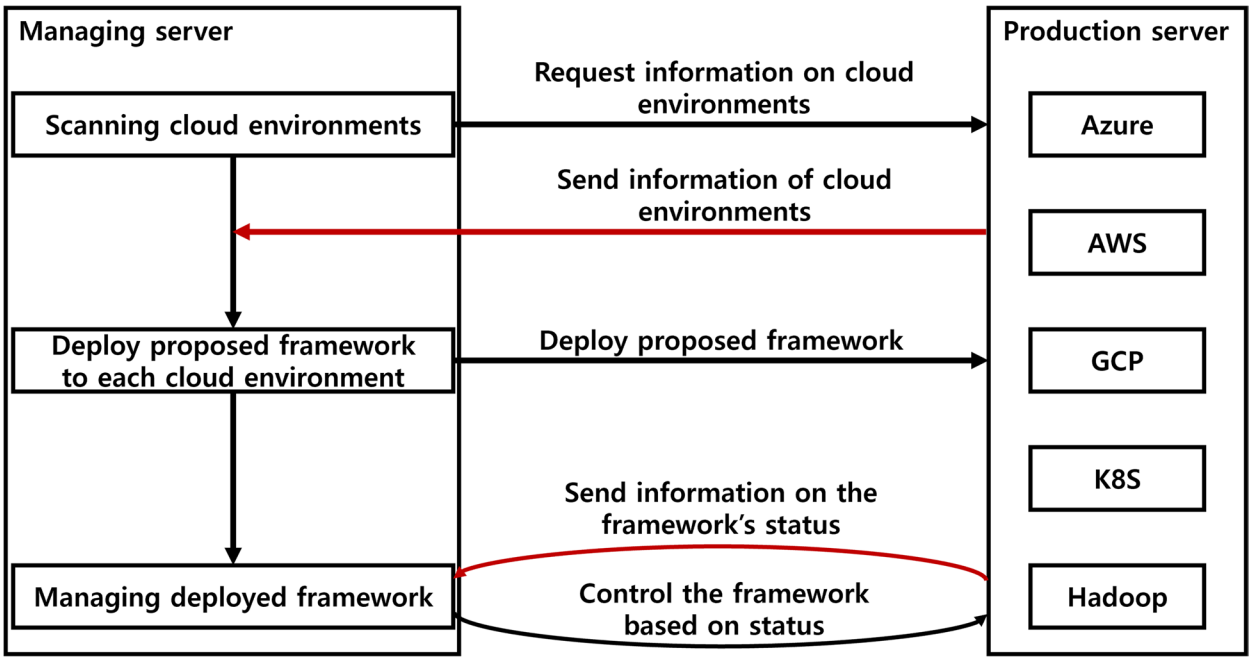

Fig. 6 Operation process of the deployed a managing server. The deployed managing server deploys the proposed framework to each identified environment. The model in the deployed framework trains the normal log that exists in each environment. The trained model performs log recognition and anomaly detection in each environment, and the framework transfers status information to the managing server in real time. Managing server controls each deployed framework based on received status information

real-time status and controls such that the framework can perform training and detection correctly.

An Algorithm 1 presented a pseudo-code that utilizes a managing server and IaC to automatically deploy the framework for every environment. The log event L occurring in the deployed environment is a log sentence describing the state of the system and network operation. The trained framework can quickly identify and respond to threats or issues by continuously analyzing L. The environment E in which the trained framework operates represents a specific component or service within the deployed cloud.

Experiments

We verified whether the proposed framework recognizes and detects logs correctly; experiments were conducted in two aspects. First, we verified the performance of the log recognition capability using operating system environment logs, distributed system environment logs generated in the cloud environment, server system logs, and network system logs from the AWS environment. Specifically, the deployed model must check whether it precisely recognizes the logs generated in each environment. To verify that each deployed model recognized logs that occurred only in its environment, we created a test dataset mixed with other environment logs. When constructing the test dataset, we incorporated the logs of all datasets in equal proportions, including the deployed environment. For example, when testing the model deployed in a Windows environment, we extracted an equal number of logs from all datasets, including the Windows dataset. The trained model deployed in the Windows environment recognizes Windows environment logs among the various environment logs: operating system, distributed system, server system, and network system. Significantly, we use random selection to prevent data bias when extracting the logs from each dataset. We configured the ratio of the training dataset and test dataset as 80:20.

In addition, we validated the performance of the model's anomaly detection capability using the labeled log datasets of Hadoop, OpenStack, Apache, and network logs. In the detection experiment, we tested whether the model trained with the normal log dataset correctly performed anomaly detection without model modifications. Consequently, we demonstrated that the proposed framework can efficiently handle various cloud environment logs.

Dataset

We used a public dataset comprising seven environmental logs to confirm that the proposed model can be applied to various domains in the cloud [49]. In this experiment, we used Linux and Windows logs to verify

the operating system environment, Hadoop and OpenStack logs to verify the distributed environment in the cloud, Apache and OpenSSH to verify the server system environment, and CICIDS2018 logs to verify the network environment. Notably, Hadoop, OpenStack, and Apache data were labeled in the public dataset. Table 3 shows information on the overall datasets.

The Linux dataset consists of 25,567 Linux internal system log messages. The Linux log dataset was collected from /var/log/message in the Linux server. The Windows dataset consists of 114,608,388 Windows event logs messages. However, neither the Linux nor Windows log datasets were labeled. The Hadoop log dataset consists of 11,175,629 distributed file system log messages. Hadoop log data are organized into log sequences with Block IDs, and abnormal labels are assigned according to specific Block IDs. The OpenStack dataset comprises 207,820 randomly performed task logs that include creating, pausing, and deleting virtual machines. The OpenStack dataset was classified as abnormal by injecting an error at a specific time point. The Apache dataset, composed of Apache web server logs, consists of 56,481 log messages. The original dataset was unlabeled; however, we labeled 16,803 error logs in the Apache dataset for anomaly detection. The OpenSSH dataset composed of OpenSSH server logs, consists of 655,146 server logs. The CICIDS2018 dataset consists of 63,175 dhclient log data that perform Distribution Denial of Service (DDoS), Denial of Service (DoS), bot attacks, brute force, infiltration, and web attacks.

In this experiment, it was necessary to balance the dataset with an imbalanced number of messages to ensure that each model recognizes and detects them appropriately. Thus, we newly created an anomaly message using the SMOTE method. The used datasets used contained more normal messages than anomalies, and a simple random generation method could cause overfitting problems. The K-Nearest Neighbor (KNN) algorithm-based SMOTE can redeem a small number of attack messages and is particularly effective for network datasets. We adjusted the training and test datasets based on the SMOTE technique to maintain an 80:20 ratio because it balances the need for a substantial training dataset with the requirement for a robust test set.

Log recognition metrics

A previous NMT experiment used the Bilingual Evaluation Understudy (BLEU) score to evaluate the accuracy of predicted sentences against actual sentences [50]. However, we focused on how competently the model recognizes the logs occurring in each environment. Thus, we verified the recognition results based on the feature word outcomes of the model and evaluated the model's performance in terms of accuracy, precision, recall, and F1-score. This experiment was conducted based on the two feature words that yielded optimal recognition results in each environment. We compared the feature words with the original log to check whether the log data were created in the same environment.

In this case, True Positive (TP) classifies existing logs as identical environment logs, and True Negative (TN) classifies different environment logs as different environments. False Positive (FP) classifies different environment logs as the existing environment, and False Negative (FN) classifies existing environment logs as different environments. These metrics are expressed as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

$$F1\text{- }Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{7}$$

Table 3 Description of multiple domain log datasets in cloud

System	Dataset	Messages	Anomalies sequences	Data size	Description	Labeled
Operating system	Linux	25567	-	2.25MB	Linux system logs	No
	Windows	114608388	-	26.09GB	Windows event logs	No
Distributed system	Hadoop	11175629	16838	1.47GB	Hadoop distributed file system logs	Yes
	OpenStack	207820	18434	58.61MB	OpenStack infrastructure logs	Yes
Server system	Apache	56481	16803 (error logs)	4.90MB	Apache web server error log	Yes
	OpenSSH	655146	-	70.02MB	OpenSSH server log	No
Network system	CICIDS2018	63175	12635	6.43MB	Network logs from the AWS environment	Yes

In addition, we demonstrated the proposed methodology by showing the recognition results of different environment logs as a heat map.

Experiment results

Log recognition result

Table 4 shows the results of log recognition based on the proposed model. The proposed recognition model exhibits a high average accuracy of 98.2% in diverse environments. Remarkably, the operating and network system results exhibited a high F1-score. In the case of distributed systems, the OpenStack dataset showed high recognition results but relatively poor results for the Hadoop dataset. Consequently, we confirmed that the proposed model could recognize each log in various environments. Moreover, we used heatmaps to ensure that the model correctly recognized the logs.

Table 4 Log recognition results on multiple domain log datasets

System	Dataset	Accuracy	Precision	Recall	F1-score
Operating system	Linux	0.987	1.000	0.935	0.966
	Windows	0.998	0.995	0.996	0.995
Distributed system	Hadoop	0.917	0.883	0.999	0.938
	OpenStack	0.982	0.993	0.988	0.988
Server system	Apache	0.999	1.000	0.998	0.999
	OpenSSH	0.999	1.000	0.996	0.998
Network system	CICIDS2018	0.996	0.995	0.995	0.995

Figures 7, 8, 9, 10 show the model's recognition results as a heatmap for on each system log dataset. Considering the heatmap, the proposed model trains the characteristics of the deployed environment and satisfies the logs according to these characteristics. Most importantly, the proposed model can extract trained environmental features, even in a dataset with a mixture of various environmental logs.

Figure 7 shows the heatmap result from the operating system. The proposed model prints a heatmap based on logs generated in the Linux and Windows systems among the operating systems. Consequently, the proposed model utilizes feature words generated by the operating system when recognizing logs. Figure 8 shows the heatmap results for the distributed system. The model mostly utilizes special feature words composed of numbers and English letters in the distributed system. In the cases of Hadoop and OpenStack, the model recognizes the logs generated in the environment based on various unique words. Figure 9 shows the heatmap results from the Server system. This model utilizes special feature words composed of numbers and characters in the server system. For example, in the case of Apache, the internal environment log was recognized using the IP address. Figure 10 shows the heatmap result for the network system. The proposed model uses diverse words that can occur in a network environment. In summary, the proposed model shows the capability that it can recognize the log of the same environment in various environments without modification.

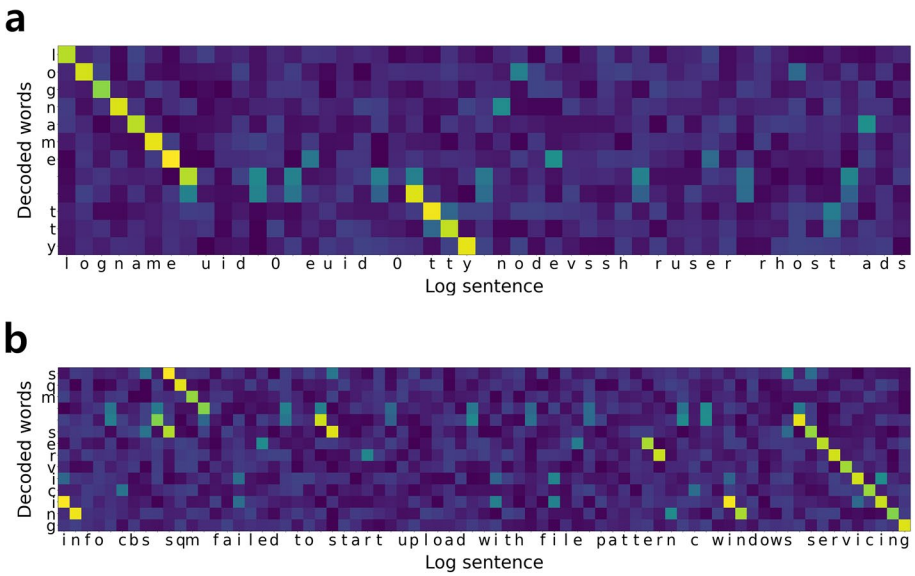

Fig. 7 Operating system heatmap between the log sentence and the feature words predicted by the model. If feature words are output from the original log sentence, the results are expressed through high values in the heatmap-the proposed model utilized words generated by the operating system. **a** Example of heatmap on Linux log dataset. **b** Example of heatmap on Windows log dataset

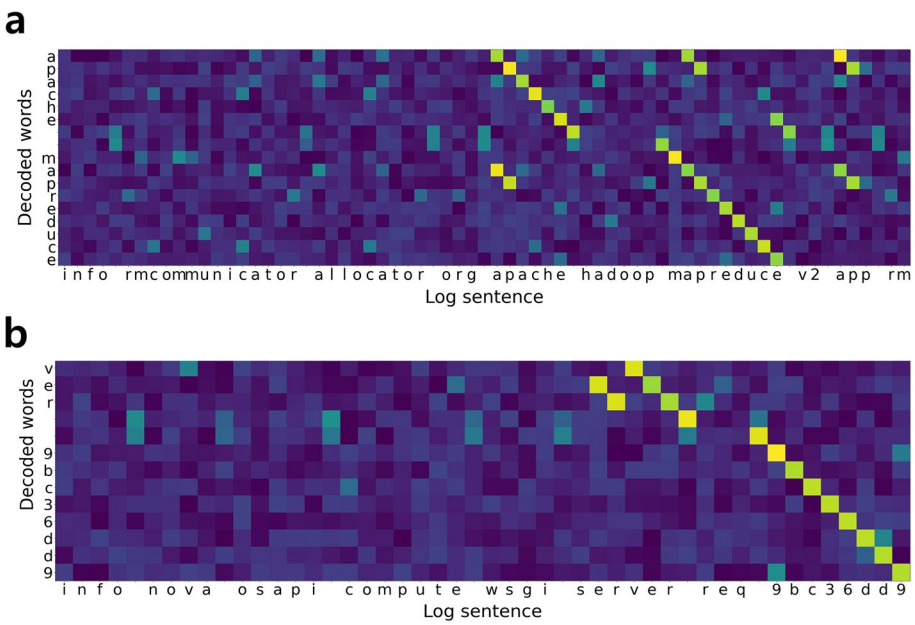

Fig. 8 Distributed system heatmap between log sentence and the feature words printed by the proposed model. The model uses special feature words composed of numbers and English letters that occurred in the distributed system. **a** Example of heatmap on Hadoop log dataset. **b** Example of heatmap on OpenStack log dataset

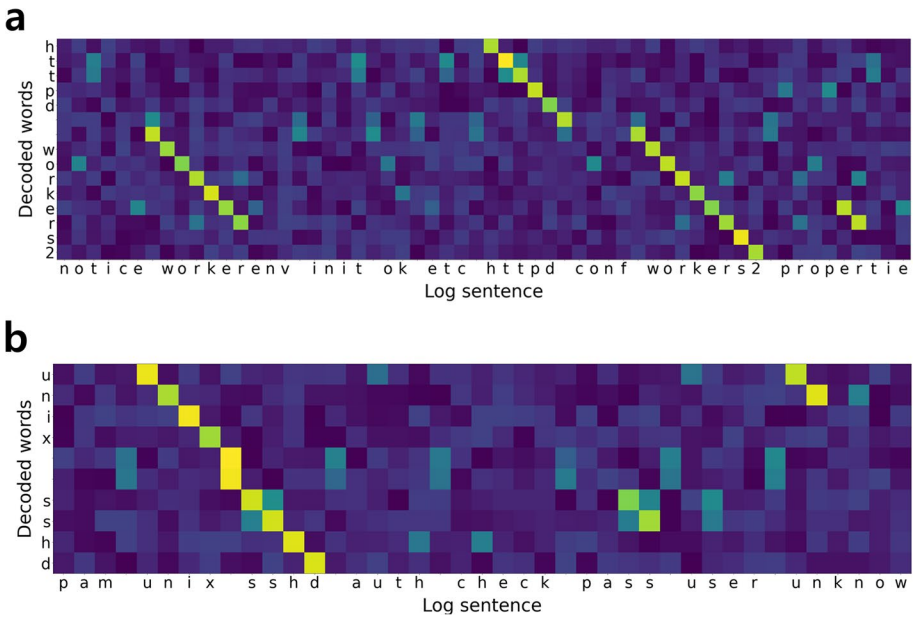

Fig. 9 Server system heatmap between the log sentence and the feature words predicted by the model. The proposed model utilized words composed of numbers and special characters. For instance, the model recognized the internal Apache environment log using the IP address. **a** Example of heatmap on Apache log dataset. **b** Example of heatmap on OpenSSH log dataset

Anomaly detection result

We used the labeled dataset to verify that the model trained only on normal data performed anomaly detection. In this experiment, we used the distributed system datasets Hadoop and OpenStack log, the server system dataset Apache, and the network system dataset: CIC-IDS2018. The metric calculation method for anomaly detection is the same as that used in the recognition experiment. However, TP classifies normal logs as normal logs, and TN classifies anomaly logs as anomaly log.

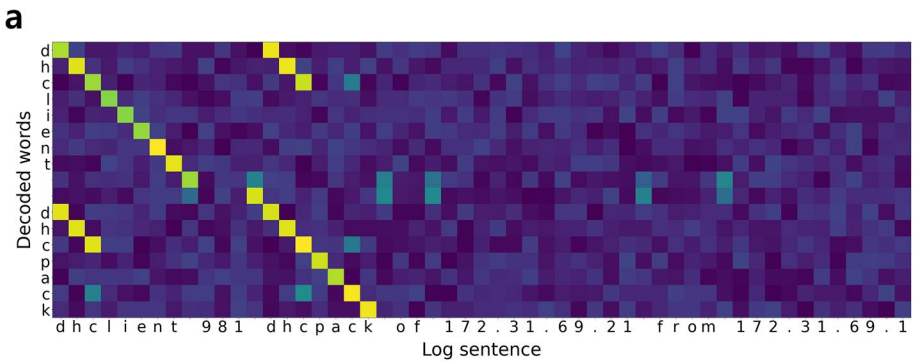

Fig. 10 Network system heatmap between the log sentence and feature words outputted by the proposed model. The model uses diverse words that can occur in the network system. **a** Example of heatmap on CICIDS2018 log dataset

Table 5 Anomaly detection results on multiple domain log datasets

System	Dataset	Accuracy	Precision	Recall	F1-score
Distributed system	Hadoop	0.996	0.949	0.996	0.971
	OpenStack	0.801	0.872	0.994	0.929
Server system	Apache	0.995	0.999	0.998	0.999
Network system	CICIDS2018	0.976	0.999	0.909	0.952

An FP classifies a normal log as an anomaly log, and an FN classifies an anomaly log as a normal log.

Table 5 shows the results of log recognition based on the proposed model. The proposed model exhibits an average accuracy of 94.2% for anomaly detection. The Hadoop and Apache datasets exhibit accuracies of 99.6% and 98.8%, respectively. The CICIDS2018 dataset, showed an accuracy of 97.6%, whereas the OpenStack dataset showed a relatively low accuracy of 80.1% because the unique ID of each log could not be determined from the context. As a result, the proposed model is effective for logs that maintain sentence structure and partially effective for logs that do not maintain and properly detect attacks sentence structure.

Discussion

We mentioned that the deployed anomaly detection model should adapt to each environment to detect attacks in each cloud environment. In summary, the distributed attack detection model should recognize logs for each environment and detect attacks based on the classified logs. Through experiments, we verified that the deployed model could distinguish only the logs generated in each environment well among the classified logs. Notably, the model recognizes that each cloud environment based on the logs is comparable; however, there are some variations.

Table 6 shows examples of what the model generally recognized and did not recognize as logs of each environment. The proposed model preferentially outputs feature words that reflect environmental characteristics among the words in the log generated in each environment. The model prints feature words in the Linux log dataset specific to the Linux environment, such as logname and rhost; similarly, in the Windows log dataset, the model prints feature words specific to the Windows environment, such as windowsupdateagent. In addition, the model can recognize numbers related to each environment, including memory addresses composed of hexadecimal numbers and port numbers. In the case of OpenStack, the model recognizes the log of the OpenStack environment using hexadecimal numbers.

Although the model reflected the characteristics of the deployed environment log, some misrecognitions still occurred during the recognition process. First, the model prints similar words, rather than the original words, as feature words because of mis-decoding. For instance, in the Apache false negatives example, the original log sentence mentioned "323 children", but the model printed "32391 children" with a high frequency of appearance. Our experiments revealed that words with higher weights are more likely to be printed during the feature word prediction process. Moreover, there was a case of a model outputting a longer sentence by combining the same words. In the case of the OpenStack false negatives example, another word, "ae3c556f73e", was printed instead of "ae3c" included in the original sentence. From these examples, we verified that the model uses word aggregation to print feature words that reflect each environment.

Previous research was conducted on the importance of words in system call sequences or security logs and the relationships between words. Previous research has proven that good results can be obtained when the

Table 6 Examples of log recognition on multiple domain log datasets

Dataset	Type	Example of log sentence (After preprocessing)	Printed feature words
Linux	Examples of true positives	sshd pam unix 31201 authentication failure logname uid 0 euid 0 tty nodevssh ruser rhost adsl 70.242.75.179 dsl ksc2mo swbell net	logname, tty
		sshd pam unix 17129 authentication failure logname uid 0 euid 0 tty nodevssh ruser rhost ip 216.69.169.168 ip secureserver net	rhost, net
	Examples of false negatives	gpm 2094 imps2 auto detected intellimouse ps 2	mtrring, mtrr
		gdm binary 2803 couldn't authenticate user	lof, core
Windows	Examples of true positives	info cbs sqm failed to start upload with file pattern c windows servicing sqm std sqm flags 0x2 hresult 0x80004005 e fail	sqm, servicing
		info cbs session 30546173 4281313522 initialized by client windowsupdateagent	cbs, windowsupdateagent
	Examples of false negatives	info csi 00000001 2016 9 27 20 30 31 455 wcpinitialize wcp dll version 0 0 0 6 called stack 0x7fed806eb5d 0x7fef9fb9b6d 0x7fef9f8358f 0xff83e97c 0xff83d799 0xff83db2f	0x7fef9d987866, 0x7fef9d987866
		info cbs sqm cleaning up report files older than 10 days	cbs, for
Hadoop	Examples of true positives	info main org apache hadoop yarn webapp webapps web app mapreduce started at 6226	hadoop, msra
		info rmcommunicator allocator org apache hadoop mapreduce v2 app rm rmcontainerallocator before scheduling pendingreds 1 scheduledmaps 10 scheduledreds 0 assignedmaps 0 assignedreds 0 completedreds 0 contalloc 0 contrel 0 hostlocal 0 racklocal	apache, mapreduce
	Examples of false negatives	info asyncdispatcher event handler org apache hadoop yarn util rackresolver resolved msra sa 41 fareast corp microsoft com to default rac	recalculating, rmcommunicator
		info rmcommunicator allocator org apache hadoop ipc client retrying connect to server msra sa 41 8030 already tried 0 time s retry policy is retryuptomaximumcountwithfixedsleep sleeptime 1 millisecond	rmcommunicator, milliseconds
OpenStack	Examples of true positives	info nova osapi compute wsgi server req 9bc36dd9 91c5 4314 898a 47625eb93b09 113d3a99c3da401fbd62cc2caa5b96d2 54fadb412c4e40cdbaed9335e4c35a9e 10 11 10 1 get v2 54fadb412c4e40cdbaed9335e4c35a9e servers detail http 1 1 status 200 len 1893 time 0	server, 9bc36dd9
		info nova osapi compute wsgi server req 1d647fe1 d879 4988 889e d860ef5b8338 113d3a99c3da401fbd62cc2caa5b96d2 54fadb412c4e40cdbaed9335e4c35a9e 10 11 10 1 get v2 54fadb412c4e40cdbaed9335e4c35a9e servers detail http 1 1 status 200 len 1893 time 0	d879, d860ef5b8338
	Examples of false negatives	info nova osapi compute wsgi server req 405a1c42 ae3c 45ec abaf eac55b56f73e 113d3a99c3da401fbd62cc2caa5b96d2 54fadb412c4e40cdbaed9335e4c35a9e 10 11 10 1 get v2 54fadb412c4e40cdbaed9335e4c35a9e servers detail http 1 1 status 200 len 1893 time 0	ae3c556f73e, eac55b56f73e
		info nova compute claims req beb938db df6e 4611 8113 1a148a0224bc 113d3a99c3da401fbd62cc2caa5b96d2 54fadb412c4e40cdbaed9335e4c35a9e instance d6b7bd36 2943 4363 9235 fffdd89ea40e total disk 15 gb used 0 00 gb	complete, driver
Apache	Examples of true positives	notice jk2 init found child 1984 in scoreboard slot 1	init, child
		notice workerenv init ok etc httpd conf workers2 propertie	httpd, workers2
	Examples of false negatives	notice jk2 init found child 32657 in scoreboard slot 2	init, 32650
		notice jk2 init found child 323 in scoreboard slot 1	notice, 32391
OpenSSH	Examples of true positives	pam unix sshd auth check pass user unknow	unix, sshd
		failed password for root from 183.62.140.253 port 46880 ssh	failed, ssh
	Examples of false negatives	pam service sshd ignoring max retries 6	from, invalid
		message repeated 5 times failed password for root from 5.36.59.76 port 42393 ssh	getaddrinfo, preauth
CICIDS2018	Examples of true positives	kernel 0 000000 tsc detected 2400 054 mhz processo	kernel, pci
		dhclient 981 dhcpack of 172.31.69.21 from 172.31.69.1	dhclient, dhcpack
	Examples of false negatives	kernel 0 204005 smpboot total of 1 processors activated 4800 15 bogomip	sooters, socket
		sh 873 internet systems consortium dhcp client 4 3	cloud, cless

model is judged to predict the words that make up each sentence appropriately. However, we showed that the predicted words expected by the researcher and the predicted words judged by the model are partially different through experiments. Through experiments, we showed that the words predicted by the researcher and those predicted by the model are partially different. Our research can prove that a model using natural language may show high accuracy even though it does not perform well when classified based on logs in a specific environment. Also, we verified whether the characteristics observed in the recognition experiment were also evident in the anomaly detection phase.

Table 7 presents an example of anomaly detection based on a model that was trained only on the normal logs of each environment. The model detected anomalies using words that reflected the environmental characteristics of the words in the log sentence. The Hadoop example shows an anomaly detection based on feature words that could occur in the Hadoop environment, such as "taskattemptlistenerimp". In addition, we demonstrate that the detection model performs attack detection by utilizing specific words, including numbers. In the OpenStack true positives example, attacks were identified using hexadecimal numbers that are not comprehensible to humans. The Apache true positives

Table 7 Examples of anomaly detections on multiple domain log datasets

Dataset	Type	Example of log sentence (After preprocessing)	Predicted feature words
Hadoop	Examples of true positives	info containerlauncher 6 org apache hadoop yarn client api impl containermanagementprotocolproxy opening proxy msra sa 41 fareast corp microsoft com 30535	apache, containermanagementprotocolproxy
		info ipc server handler 22 on 22927 org apache hadoop mapred taskattemptlistenerimpl progress of taskattempt attempt 1445087491445 0001 r 000000 0 is 0 30769232	taskattemptlistenerimpl, ipc
	Examples of false negatives	info main org apache hadoop mapred maptask kvstart 14562216 58248864 kvend 12516336 50065344 length 2045881 6553600	app, rmcontainerallocator
		info main org apache hadoop mapred maptask spilling map output	apache, msra
OpenStack	Examples of true positives	info nova osapi compute wsgi server req 5a2050e7 b381 4ae9 92d2 8b08e9f9f4c0 113d3a99c3da401fbd62cc-2caa5b96d2 54fadb412c4e40cdbaed9335e4c35a9e 10 11 10 1 get v2 54fadb412c4e40cdbaed9335e4c35a9e servers detail http 1 1 status 200 len 1583 time 0 1919448	nova, 54fadb412c4e40cdbaed9335e4c35a9e
		info nova virt libvirt imagecache req addc1839 2ed5 4778 b57e 5854eb7b8b09 active base files var lib nova instances base a489c868f0c37da93b76227c91bb03908ac0e742	imagecache, 5854eb7b8b09
	Examples of false negatives	info nova osapi compute wsgi server req 551d57c1 f0bb 4ebe 9845 018f0742b197 113d3a99c3da401fbd62cc-2caa5b96d2 54fadb412c4e40cdbaed9335e4c35a9e 10 11 10 1 get v2 54fadb412c4e40cdbaed9335e4c35a9e servers detail http 1 1 status 200 len 1893 time 0 2654262	server, 551d57c1197
		info nova osapi compute wsgi server req 346d44ff f4e7 4477 8efe 3a5c36ce8a63 113d3a99c3da401fbd62cc2caa5b96d2 54fadb412c4e40cdbaed9335e4c35a9e 10 11 10 1 get v2 54fadb412c4e40cdbaed9335e4c35a9e servers detail http 1 1 status 200 len 1893 time 0 4282429	346d4fff, http
Apache	Examples of true positives	notice jk2 init found child 32528 in scoreboard slot 1	init, 32528
		error client 65.68.235.27 directory index forbidden by rule var www htm	65.68.235.27, htm
	Examples of false negatives	error client 65.68.235.27 directory index forbidden by rule var www htm	65.68.235.1, htm
		error client 4.245.93.87 directory index forbidden by rule var www htm	client, html
CICIDS2018	Examples of true positives	systemd 1 mysql service service hold off time over scheduling restart	systemd, scheduling
		kernel 0 854727 allocating ima mok and blacklist keyring	allocating, blacklist
	Examples of false negatives	kernel 0 410462 libata version 3 00 load	kernel, loaded
		kernel 0 000000 hpet clockevent register	host, hot

example demonstrates that the model identifies anomalies based on a specific IP address. Through this experiment, we verified that words composed of numbers and special characters are important factors for distinguishing anomalies. However, some misdetections occurred during the detection experiment. First, false positives occurred owing to erroneous decoding for reasons similar to those in the recognition experiments. In particular, the model did not distinguish between normal logs and anomalies when detection was performed using only human-readable words.

Because some existing detection methods treat numbers and special characters as stopwords, the detection results were improved. However, as a result of experiments based on actual logs generated in each environment, the numbers and special characters significantly impacted the accuracy of the recognition and detection experiments. Moreover, it was possible to utilize words that humans cannot read when applying the NLP method to a security log. In summary, various factors must be reflected to perform accurate anomaly detection of logs generated in each environment.

Conclusion and future work

This paper proposed a C-IDS framework that recognizes the logs of each environment and performs anomaly detection by applying logs generated in multiple-domains in the cloud without modification. The C-IDS framework can be applied to various environments through the IaC tool terraform and deployed in each environment without any individual configuration. The deployed framework trains logs in the environment and performs recognition and anomaly detection. As a result of the experiment, the average recognition result was 98.2%, and the average anomaly detection results were 98.2% and 94.2%, respectively.

In summary, the proposed model can be used as general purpose and multi-role IDPS for cloud environments. It can also be easily applied to any cloud platform because the proposed system is light-weight. The traditional IDPS mainly analyzes network traffic or local system logs, and frequently increases the overall CPU or disk I/O usage. However, when the IDPS performs statistical analysis based on the long term dataset, a large amount of log storage to handle cumulative log events. However, our proposed system supports stream-lined (or in-line) log analytics, as shown in Fig. 7. These two aspects (applicability to most applications or OSes, and light-weight implementation) benefit most service providers based on cloud systems.

However, the proposed framework has several limitations. First, as the operating system dataset used in this study is not labeled, it is unknown whether the system accurately detects anomalies. Furthermore, only normal log data must be available in the deployed environment. We visualized the recognition results through a heatmap but could not determine which features in the log sentence had an impact. In future studies, we will determine how each word affects a log sentence based on the expressions of each word.

Abbreviations

C-IDS	Cloud intrusion detection system
IDS	Intrusion detection system
NLP	Natural language processing
CSP	Cloud service providers
IaC	Infrastructure as code
IG	Information gain
CS	Chi-square
PSO	Particle swarm optimization
SLA	Service level agreements
ROC-AUC	Area under the receiver operating characteristic curve
XSS	Cross-site scripting attacks
RLPSVDD	Relaxed form of linear programming support vector data description
SVDD	Support vector data description
SIEM	Security information and event management
FCNN	Fully connected neural network
CNN	Convolutional neural network
LSTM	Long short-term memory
SLCT	Simple logfile clustering tool
IPLoM	Iterative partitioning log mining
LKE	Log key extraction
Word2Vec	Word to vector
GNB	Gaussian naive Bayes
RF	Random forest
MLP	MultiLayer perceptrons
Seq2Seq	Sequence-to-sequence
RNN	Recurrent neural networks
SVM	Support vector machine
AWS	Amazon web services
NMT	Neural machine translation
BI-LSTM	Bi-directional long short-term memory
DDoS	Distribution denial of service
DoS	Denial of service
SMOTE	Synthetic minority over-sampling technique
KNN	K-nearest neighbor
BLEU	Bilingual evaluation understudy
TP	True positive
TN	True negative
FP	False positive
FN	False negative
HTTP	HyperText transfer protocol

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2024-00359621).

Authors' contributions

Each author made a significant contribution to the research and preparation of the manuscript. Yongsik Kim mainly conducted manuscript experiments, and Gunho Park and Huy Kang Kim prepared and conducted experiments. All authors read and approved the final version of the manuscript.

Availability of data and materials

Some or all of the data, models, or codes used in this study to support this work are available from the authors upon reasonable request.

Declarations

Ethics approval and consent to participate
This article does not contain any studies with human participants or animals performed by any of the authors.

Competing interests
The authors declare no competing interests.

Received: 13 November 2023 Accepted: 10 September 2024
Published online: 27 September 2024

References

1. Nazareth DL, Choi J (2021) Market share strategies for cloud computing providers. J Comput Inf Syst 61(2):182–192
2. Siddiqa A, Karim A, Gani A (2017) Big data storage technologies: a survey. Front Inform Technol Electron Eng 18(8):1040–1070
3. Yang HL, Lin SL (2015) User continuance intention to use cloud storage service. Comput Hum Behav 52:219–232
4. Singh S, Jeong YS, Park JH (2016) A survey on cloud computing security: Issues, threats, and solutions. J Netw Comput Appl 75:200–222
5. Ren K, Wang C, Wang Q (2012) Security challenges for the public cloud. IEEE Internet Comput 16(1):69–73
6. Jangjou M, Sohrabi MK (2022) A comprehensive survey on security challenges in different network layers in cloud computing. Arch Comput Methods Eng 29(6):3587–3608
7. Kumar R, Goyal R (2019) On cloud security requirements, threats, vulnerabilities and countermeasures: A survey. Comput Sci Rev 33:1–48
8. Li HC, Liang PH, Yang JM, Chen SJ (2010) Analysis on cloud-based security vulnerability assessment. In: 2010 IEEE 7th International Conference on E-Business Engineering. IEEE, Shanghai, p 490–494
9. Ali M, Khan SU, Vasilakos AV (2015) Security in cloud computing: Opportunities and challenges. Inform Sci 305:357–383
10. Al Nafea R, Almaiah MA (2021) Cyber security threats in cloud: Literature review. In: 2021 International Conference on Information Technology (ICIT). IEEE, Amman, p 779–786
11. Modi C, Patel D, Borisaniya B, Patel H, Patel A, Rajarajan M (2013) A survey of intrusion detection techniques in cloud. J Netw Comput Appl 36(1):42–57
12. Faber K, Faber L, Sniezynski B (2021) Autoencoder-based ids for cloud and mobile devices. In: 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid). IEEE, Melbourne, p 728–736
13. Sharafaldin I, Lashkari AH, Ghorbani AA (2018) Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSp 1:108–116
14. Huang C, Min G, Wu Y, Ying Y, Pei K, Xiang Z (2017) Time series anomaly detection for trustworthy services in cloud computing systems. IEEE Trans Big Data 8(1):60–72
15. Patel A, Taghavi M, Bakhtiyari K, Júnior JC (2013) An intrusion detection and prevention system in cloud computing: A systematic review. J Netw Comput Appl 36(1):25–41
16. Lee JH, Kim YS, Kim JH, Kim IK (2017) Toward the siem architecture for cloud-based security services. In: 2017 IEEE Conference on Communications and Network Security (CNS). IEEE, Las Vegas, p 398–399
17. Lee J, Kim J, Kim I, Han K (2019) Cyber threat detection based on artificial neural networks using event profiles. IEEE Access 7:165607–165626
18. Okey OD, Melgarejo DC, Saadi M, Rosa RL, Kleinschmidt JH, Rodríguez DZ (2023) Transfer learning approach to ids on cloud iot devices using optimized cnn. IEEE Access 11:1023–1038
19. Bakro M, Kumar RR, Alabrah A, Ashraf Z, Ahmed MN, Shameem M, Abdelsalam A (2023) An improved design for a cloud intrusion detection system using hybrid features selection approach with ml classifier. IEEE Access 11:64228–64247
20. Habeeb RAA, Nasaruddin F, Gani A, Hashem IAT, Ahmed E, Imran M (2019) Real-time big data processing for anomaly detection: A survey. Int J Inf Manag 45:289–307
21. Lu S, Wei X, Li Y, Wang L (2018) Detecting anomaly in big data system logs using convolutional neural network. In: 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech). IEEE, Athens, p 151–158
22. Bertero C, Roy M, Sauvanaud C, Trédan G (2017) Experience report: Log mining using natural language processing and application to anomaly detection. In: 2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE). IEEE, Toulouse, p 351–360
23. He P, Zhu J, He S, Li J, Lyu MR (2016) An evaluation study on log parsing and its use in log mining. In: 2016 46th annual IEEE/IFIP international conference on dependable systems and networks (DSN). IEEE, Toulouse, p 654–661
24. Zhu J, He S, Liu J, He P, Xie Q, Zheng Z, Lyu MR (2019) Tools and benchmarks for automated log parsing. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP). IEEE, pp 121–130
25. Sworna ZT, Mousavi Z, Babar MA (2023) Nlp methods in host-based intrusion detection systems: A systematic review and future directions. J Netw Comput Appl 220:103761
26. Hirschberg J, Manning CD (2015) Advances in natural language processing. Science 349(6245):261–266
27. Singh K, Grover SS, Kumar RK (2022) Cyber security vulnerability detection using natural language processing. In: 2022 IEEE World AI IoT Congress (AIIoT). IEEE, Seattle, p 174–178
28. Ukwen DO, Karabatak M (2021) Review of nlp-based systems in digital forensics and cybersecurity. In: 2021 9th International Symposium on Digital Forensics and Security (ISDFS). IEEE, Elazig, p 1–9
29. Mahendran D, Luo C, Mcinnes BT (2021) Privacy-preservation in the context of natural language processing. IEEE Access 9:147600–147612
30. Peters F, Tun TT, Yu Y, Nuseibeh B (2017) Text filtering and ranking for security bug report prediction. IEEE Trans Softw Eng 45(6):615–631
31. Das S, Ashrafuzzaman M, Sheldon FT, Shiva S (2020) Network intrusion detection using natural language processing and ensemble machine learning. In: 2020 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, Canberra, p 829–835
32. Wang J, Tang Y, He S, Zhao C, Sharma PK, Alfarraj O, Tolba A (2020) Logevent2vec: Logevent-to-vector based anomaly detection for large-scale logs in internet of things. Sensors 20(9):2451
33. Ryciak P, Wasielewska K, Janicki A (2022) Anomaly detection in log files using selected natural language processing methods. Appl Sci 12(10):5089
34. Lv S, Wang J, Yang Y, Liu J (2018) Intrusion prediction with system-call sequence-to-sequence model. IEEE Access 6:71413–71421
35. Chaudhari A, Gohil B, Rao UP (2024) A novel hybrid framework for cloud intrusion detection system using system call sequence analysis. Clust Comput 27(3):3753–3769
36. Galassi A, Lippi M, Torroni P (2020) Attention in natural language processing. IEEE Trans Neural Netw Learn Syst 32(10):4291–4308
37. Varol Arısoy M (2022) Lzw-cie: a high-capacity linguistic steganography based on lzw char index encoding. Neural Comput Appl 34(21):19117–19145
38. Khan S, Alam M (2023) Preprocessing framework for scholarly big data management. Multimed Tools Appl 82(25):39719–39743
39. Bahdanau D, Cho K, Bengio Y (2015) Neural machine translation by jointly learning to align and translate. In: Bengio Y, LeCun Y (eds) 3rd International conference on learning representations. ICLR 2015, San Diego, 7–9 May 2015. Conference Track Proceedings
40. Alonso J, Orue-Echevarria L, Osaba E, López Lobo J, Martinez I, Diaz de Arcaya J, Etxaniz I (2021) Optimization and prediction techniques for self-healing and self-learning applications in a trustworthy cloud continuum. Information 12(8):308
41. Cauli C, Li M, Piterman N, Tkachuk O (2021) Pre-deployment security assessment for cloud services through semantic reasoning. In: International Conference on Computer Aided Verification. Springer, Cham, p 767–780
42. Wu Y, Schuster M, Chen Z, Le QV, Norouzi M, Macherey W, Krikun M, Cao Y, Gao Q, Macherey K, Klingner J, Shah A, Johnson M, Liu X, Kaiser L, Gouws S, Kato Y, Kudo T, Kazawa H, Stevens K, Kurian G, Patil N, Wang W, Young C, Smith J, Riesa J, Rudnick A, Vinyals O, Corrado G, Hughes M, Dean J (2016)

Google's neural machine translation system: Bridging the gap between human and machine translation. CoRR abs/1609.08144
43. Studiawan H, Sohel F, Payne C (2020) Anomaly detection in operating system logs with deep learning-based sentiment analysis. IEEE Trans Dependable Secure Comput 18(5):2136–2148
44. Nedelkoski S, Bogatinovski J, Acker A, Cardoso J, Kao O (2020) Self-attentive classification-based anomaly detection in unstructured logs. In: 2020 IEEE International Conference on Data Mining (ICDM). IEEE, Sorrento, p 1196–1201
45. Jackson E, Agrawal R (2019) Performance evaluation of different feature encoding schemes on cybersecurity logs. In: 2019 SoutheastCon. IEEE, Huntsville, p 1–9
46. Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks. Adv Neural Inform Process Syst 27:3104–3112
47. Schuster M, Paliwal KK (1997) Bidirectional recurrent neural networks. IEEE Trans Signal Process 45(11):2673–2681
48. Dey A (2020) Deep ids: A deep learning approach for intrusion detection based on ids 2018. In: 2020 2nd International Conference on Sustainable Technologies for Industry 4.0 (STI). IEEE, Dhaka, p 1–5
49. He S, Zhu J, He P, Lyu MR (2020) Loghub: a large collection of system log datasets towards automated log analytics. arXiv preprint arXiv:200806448
50. Papineni K, Roukos S, Ward T, Zhu WJ (2002) Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting of the Association for Computational Linguistics. Association for Computational Linguistics, Philadelphia, p 311–318

Publisher's Note