MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

DeepEAD: Explainable Anomaly Detection from System Logs

Wang, Xinda; Kim, Kyeong Jin; Wang, Ye; Koike-Akino, Toshiaki; Parsons, Kieran

TR2023-050 May 31, 2023

Abstract

System logs record rich information for system events. Practical anomaly detection from system logs should be able to address three challenges: 1) understanding complicated attributes in event logs; 2) extracting complex context relations among events; and 3) providing concrete explanations to human analysts. In this paper, we develop an attention-equipped encoder-decoder system to capture context from system logs for explain- able anomaly detection. For each target event, we collect its nearby events in chronological order as its context events. Instead of using a recurrent neural network-based encoder like previous works, we adopt a Transformer-based encoder to extract complex relations among context events and their attributes. Then, a context vector is generated and passed to the decoder, where an attention matrix is learned and used to weigh the context events for detecting the anomalies. Evaluation on the large-scale real-world Los Alamos National Laboratory dataset shows that, compared with existing works, our methods can provide fine- grained one-to-one attention to help explain the importance of each attribute in the context events to the prediction, without sacrificing detection performance.

IEEE International Conference on Communications (ICC) 2023

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

DeepEAD: Explainable Anomaly Detection from System Logs

Xinda Wang*†, Kyeong Jin Kim*, Ye Wang*, Toshiaki Koike-Akino*, Kieran Parsons*
*Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, USA
†Center for Secure Information Systems, George Mason University, Fairfax, VA, USA
xwang44@gmu.edu, {kkim, yewang, koike, parsons}@merl.com

***Abstract*—System logs record rich information for system events. Practical anomaly detection from system logs should be able to address three challenges: 1) understanding complicated attributes in event logs; 2) extracting complex context relations among events; and 3) providing concrete explanations to human analysts. In this paper, we develop an attention-equipped encoder-decoder system to capture context from system logs for explainable anomaly detection. For each target event, we collect its nearby events in chronological order as its context events. Instead of using a recurrent neural network-based encoder like previous works, we adopt a Transformer-based encoder to extract complex relations among context events and their attributes. Then, a context vector is generated and passed to the decoder, where an attention matrix is learned and used to weigh the context events for detecting the anomalies. Evaluation on the large-scale real-world Los Alamos National Laboratory dataset shows that, compared with existing works, our methods can provide fine-grained one-to-one attention to help explain the importance of each attribute in the context events to the prediction, without sacrificing detection performance.**

***Index Terms*—Anomaly detection, Transformer, explainable deep learning, context analysis.**

I. Introduction

Available in almost all computer systems, logs are used to record various events for monitoring, administration, and debugging, which provide a good source of information for analyzing and identifying anomalies. Since modern IT infrastructure systems continuously generate an overwhelming amount of event logs and attacks are evolving and becoming more complex [1], automated anomaly detectors are usually applied to flag potential anomalies. Then, detected events will be handed over to human analysts for further analysis [2]. However, as reported in FireEye M-Trends 2021 [3], the median time for organizations to identify incidents by the help of anomaly detectors is 24 days, yielding too much time to attackers for conducting malicious activities. This is due to two major weaknesses of existing anomaly detectors: *high false-positive rate* and *lack of explanations in detection results*.

Although anomaly detectors aim to filter out unlikely suspicious events to reduce the workload of human analysts, they still generate an excessive number of alerts. Recent surveys conducted by [4] and [5] show that a typical security operations team receives over 11,000 alerts daily, while less than a fifth of them are actual attacks. Given such a huge alert volume, only 7% of alerts are investigated by analysts in time [6]. Failing to effectively consider the context around the event is the main reason for false alarms. To circumvent anomaly detectors, recent attackers tend to organize complex attacks where benign events are interleaved with suspicious ones. Without excluding unrelated benign events, models proposed by [7] and [8] may learn these benign events as anomaly features. To solve this problem, the proposed approach should be able to capture the context where a suspicious event is triggered in event logs and put emphasis on suspicious events.

Although several approaches extract the context from log sequences by adopting a series of recurrent neural network (RNN)-based models [9]–[13], they only predict if there is an anomaly, but cannot explain why, which still necessitates much effort to manually examine neighboring logs and correlate context events. To help understand the context events involved during automated anomaly decisions, attention mechanisms are employed by [14] and [15]. However, their attention vectors are not directly applied to input events. Rather, they are applied over a complex combination of current and historical states, which is less interpretable for understanding the importance of current event [16]. To mitigate this, a one-to-one mapping mechanism between attention and context event should be produced for concrete explanations.

In this paper, we propose a deep learning-based explainable anomaly detection system, named DeepEAD, which adopts an attention-equipped encoder-decoder architecture. Specifically, a Transformer-based encoder is used to extract context relations among different attributes in context events. Then, an attention decoder is designed to decode the context information into an attention matrix, which represents the weights for each context attribute as well as enables explainability. After associating context events with attention weights, the event decoder predicts the target event. An anomaly is detected if the actual event is predicted not to happen. Such a process does not require anomaly labels and is fully unsupervised. If anomaly labels are available, we further leverage the transfer learning to transform the event prediction model into the anomaly prediction domain. The transferred model is supervised on the binary prediction task of anomaly detection.

We conduct experimental evaluations on a large-scale real-world dataset collected by the Los Alamos National Laboratory (LANL). Comparison results show that our system enables explainability and achieves comparable detection performance with state-of-the-art works at the same time. Besides,

*Xinda Wang's work was conducted during her internship at MERL.

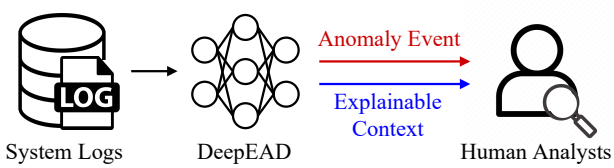

Fig. 1: Workflow of the DeepEAD system.

we perform a case study to show how attention weights work to explain the model behavior in specific cases.

II. Methodology

Assumption. We use system logs as the input to our anomaly detection system. Intuitively, the preceding events can lead to the following event and the postceding events can be affected by the previous event. Further, attackers tend to launch attacks with the help of several different accounts and devices, which causes victim computers become the destination of a series of events. Therefore, for a target event, we assume its *context* to be events happening around the same time and destined at the same device.

Overview. To reduce the workload of human analysts, we aim at detecting suspicious events from system logs, while providing context and explainability for anomaly detection. The workflow of the network security system that employs DeepEAD is shown in Fig. 1, where DeepEAD works as an intermediate step to filter out the overwhelming number of unrelated logs, flag any anomaly event that is a part of an attack, and provide a necessary explanation to the human analysts for further investigation.

As illustrated in Fig. 2 at the top of the next page, DeepEAD is an attention-equipped encoder-decoder system, where DeepEAD takes a sequence of preprocessed context events as inputs. A *Transformer-based context encoder* addresses complex relations among different attributes in context events to generate a context matrix. Then, the *attention decoder* decodes the context vector into an attention matrix, which represents how each attribute in each context event relates to the target event. Finally, the *event decoder* associates context events with an attention matrix through multiplication and uses a neural network to detect the anomaly through unsupervised learning and optionally supervised learning. We detail each module in the following.

A. System Log Preprocessing

Systems logs record various events for monitoring, administration, and debugging. Although logs can be produced from different systems (e.g., operating systems, firewalls, and containers), they are usually well-organized and composed of multiple attributes. For instance, authentication logs of the LANL internal network [17] consist of *timestamp*, *source user*, *destination user*, *source computer*, *destination computer*, *authentication type*, *logon type*, *authentication orientation*, and *success/failure*. Such forms enable us to transform each event log into a vector that can be input into the DeepEAD system. Specifically, for each value in an attribute, we use an individual numeric value to represent it, so that an event e that is composed of m attributes can be represented with a single vector $[attr_0, ..., attr_{m-1}]$. A numeric value is also used to index each different event type. The above event representation makes our method system-independent since it can be easily adapted to any formatted logs generated by other systems. We also discard user-related features (e.g., *user*, *domain*, and *computer ID*) and retain only event-related features to represent events, which allows our system to handle future event logs without considering unseen out-of-vocabulary users, domains, or computer IDs.

Since there are usually an overwhelming number of event logs generated by the computer systems, we only need to consider logs providing context for the target event. Otherwise, unrelated events will introduce a large amount of noise to the detection process. As stated in our assumptions, we sort system logs according to the timestamp and group them by the destination device to determine the context for each event.

Considering that both previous and following events can provide context information, we use a bi-directional sliding window to retrieve the context events (after excluding the target event). For an event e_i, we use its n_{pre} preceding events and n_{post} postceding events to form a *context event sequence* $S_i = [e_{i-n_{pre}}, ..., e_{i-1}, e_{i+1}, ..., e_{i+n_{post}}]$. We empirically set both n_{pre} and n_{post} as 10, such that each event has a 20-length context sequence. Also, we limit our search space of context within 24 hours to remove irrelevant events that happen too early or late. Note that these values can be adjusted for time-sensitive applications. A specific event vector is used to pad if there are no sufficient preceding or postceding context events.

B. Transformer-Based Context Encoder

Previous work [18] takes the one-hot encoding for events as the inputs and uses an RNN to extract context information. However, as the LANL event log sample shown in the last subsection, a practical system log is usually composed of multiple attributes, and these attributes may not be independent. Thus, to provide a meaningful context for anomaly detection, correlation among multiple attributes in the same and different context events should be well extracted. To this end, instead of using the one-hot encoding, we first adopt an embedding layer to embed context events and get an *embedded context sequence* $S'_i = [e'_{i-n_{pre}}, ..., e'_{i-1}, e'_{i+1}, ..., e'_{i+n_{post}}]$, where each attribute of each event is encoded into a 128-dimensional vector. To further address complex context relations among multiple attributes from the context sequence, we apply a Transformer-based encoder [19] where the scaled dot product attention allows any attribute in an event can attend to any other events and the multi-head attention mechanism provides multiple different aspects an event attribute can attend to. Then, an embedded context event sequence is transformed into a single fixed-length (e.g., 128 in our case) vector representation *context vector* c_i.

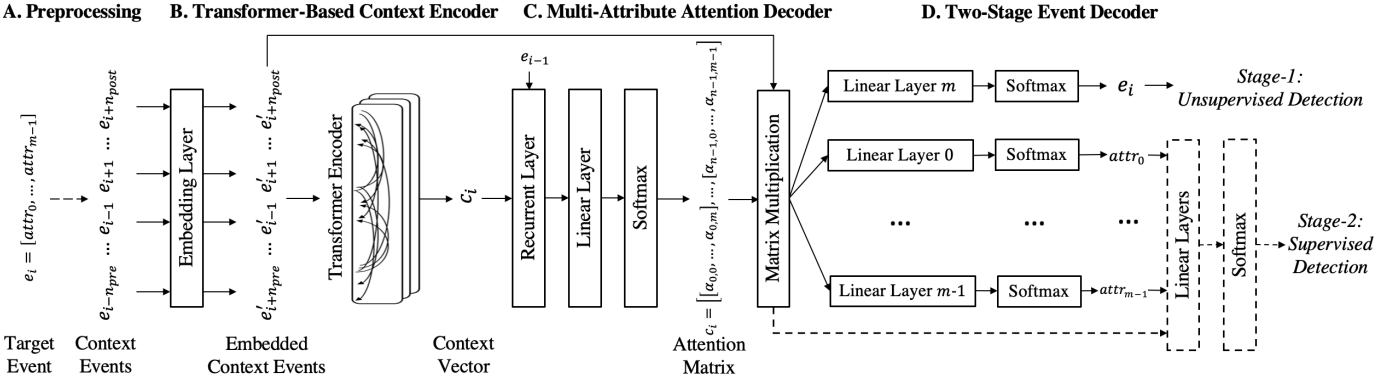

Fig. 2: DeepEAD: An attention-based encoder-decoder architecture.

C. Multi-Attribute Attention Decoder

The design of the attention decoder is to decode the context vector as an *attention matrix* $[[\alpha_{0,0}, ..., \alpha_{0,m-1}], ..., [\alpha_{n-1,0}, ..., \alpha_{n-1,m-1}]]$, where n denotes the number of events in a context event sequence (i.e., n_{pre} + n_{post}) and m represents the number of attributes in a context event. In such a design, each value in the attention matrix corresponds to an attribute in the context event and can be used to explain the correlation with the predicted target event.

To achieve this, we apply the gated recurrent units (GRU) layers [20] to generate the attention matrix. Then, a series of linear layers and a Softmax function are used to normalize the sum of these attention values to 1. Therefore, each value in the attention vector can be regarded as a weight to describe the importance of the corresponding attribute for the target event. In addition, the sum of attention values in each context event represents how it contributes to the target prediction.

D. Two-Stage Event Decoder

The event decoder applies the attention weights of each attribute in the context event sequence to the embedded context events by performing matrix multiplication between the attention matrix and embedded context events, producing a weighted context vector.

We design the rest of this module as a two-stage learning process. Since the number of anomalies may be few for a system at the very beginning and anomaly labels require excessive human work, we design the first stage to be unsupervised which does not require any anomaly labels. In this stage, we train a model to predict the event given the context, so that the anomaly is detected if the actual event is predicted not to happen. In the second stage, we transform the context knowledge learned from the first stage into a supervised model. With the availability of anomaly labels from the prior detection and human verification, we fine-tune the transferred model with a small number of labeled data. As a result, it can directly predict if there is an anomaly or not.

1) Stage-1: Unsupervised Detection

In the first stage, we train an event prediction model whose output is a probability distribution over all types of events. After generating the weighted context vector, a series of linear layers and Softmax layers are used to predict not only the event type but also the values for each attribute. Here, our goal is to enable the model to learn more from each attribute of context events by considering all of the losses in each attribute branch. Note that we use only the outputs of the main branch (i.e., event type) to determine the anomaly.

The main branch will assign a probability for each event type. By sorting the probability, we will get the top k events, i.e., the most possible k events given the current context. Then, if the actual event falls into the top k events, it will be regarded as a normal event. Otherwise, it means that the actual event should be unlikely to happen, which indicates that an anomaly is detected. This stage is fully unsupervised since no anomaly label is required for training.

2) Stage-2: Supervised Detection

After the cold start, some anomaly events may be detected and further confirmed by human analysts so that anomaly labels become available. Considering the lack of anomaly samples and huge manual efforts during the anomaly investigation, we adopt transfer learning by transforming the context knowledge learned from event prediction to anomaly prediction. The rationales of such a design are due to two-fold similarities. First, some types of events are highly likely to be anomalies. Second, no matter what the prediction goal is, given a series of preceding and postceding events, the specific events composing an activity are determined. For example, given a set of events: *i. input the username; ii. check the privacy policy; iii. follow a Twitter account; iv. enter the password; and v. enter the one-time verification code*, it is easy to identify that *i*, *iv*, and *v* belong to a logon activity.

To retain such similarity and transfer from event prediction to anomaly prediction, we use a small amount of data labeled with anomaly or normality to fine-tune the model. Specifically, we freeze the context encoder and attention decoder module (i.e., retain the parameters) to keep the context information among attributes as well as events and fine-tune the model using labeled data for anomaly prediction.

More detailedly, as shown in Fig. 2 with dotted lines, a set of linear layers are added to concatenate the pre-confidence features of each branch as well as the weighted context vector after the first linear layer as the inputs. The adoption of the latter is inspired by the ResNet [21] to retain more original information. After the Softmax, the final prediction results will be expressed in binary, i.e., whether there is an anomaly or not. Note that this stage is optional if no anomaly labels are available. However, our experimental results show that the additional supervised learning stage can improve the performance of anomaly detection.

III. Evaluation

A. Implementation

Inspired by DeepCASE [18], we develop DeepEAD with extensive extensions to address complex correlation among multiple attributes of different context events, generate fine-grained attention, and integrate a two-stage multi-branch classifier. In total, we construct DeepEAD with 2K new LoC in Python and PyTorch.

B. Experimental Setup

Dataset. We evaluate our DeepEAD on a real-world dataset collected from LANL's internal computer network collected during 58 consecutive days [17]. The LANL dataset includes large-scale logs from multiple sources, i.e., authentication, process, network flow, and DNS lookup events, and each log represents a single event. In the first 30 days of data, some authentication events have been labeled as redteam compromise events. Therefore, we perform our experiments on the authentication logs and use these redteam events as the ground truth labels of anomaly behavior.

Runtime Environments. All the experiments are conducted on a Ubuntu 20.04 server with an Intel i7-7700K CPU running at 4.20GHz, 64 GB RAM, and an NVIDIA TITAN Xp GPU. The deep learning architecture is built on the NVIDIA CUDA Toolkit 11.6 and cuDNN v7.6.0.

Evaluation Metrics. We adopt the area under the receiver operating characteristic (AUROC) to show the ability of an anomaly detection system. The ROC curve plots the true positive (TP) rate against the false positive (FP) rate at various threshold settings. For both unsupervised learning and supervised learning (i.e., binary classification), TPs are the predictions that are actual redteam events, whereas FPs are predictions that are actual non-redteam events.

C. Performance Evaluation

For a fair comparison, we adopt the same training and test datasets as those of existing works. Specifically, to compare with existing RNN-based works [11], we train our model on the first 12 days of data with 133M normal events and 316 redteam events and conduct the test on the following 18 days of data, composed of 210M normal events and 385 redteam events. We also compare our DeepEAD with a state-of-the-art (SOTA) work with attention mechanisms [14] and the baseline work named DeepCASE [18]), where the training phase is performed on the Day 7 data with 9M normal events and 1 redteam event and the test is on the Day 8 data with 9M normal events and 261 redteam events. Note that, since the SOTA work is unsupervised and its adopted training set only contains 1 anomaly sample that is insufficient for supervised training, we use only the unsupervised stage of DeepEAD for comparison.

TABLE I: Comparison results with existing anomaly detection methods [11].

Method	AUROC	Attention	Explainability
EM	0.932	×	×
BiEM	0.895	×	×
Tiered-EM	0.948	×	×
Tiered-BiEM	0.902	×	×
DeepEAD (stage-1)	0.939	✓	✓
DeepEAD (stage-2)	**0.958**	✓	✓

TABLE II: Comparison results with existing attention-based anomaly detection methods [14], [18].

Method	AUROC	Explainability	Fine Granularity
EM-fixed	0.976	✓̸	×
EM-syntactic	0.975	✓̸	×
EM-semantic1	0.980	✓̸	×
EM-semantic2	0.976	✓̸	×
DeepCASE	0.920	✓	×
DeepEAD	<u>0.971</u>	✓	✓

Table I shows the comparison results with a series of existing RNN-based anomaly detection methods [11] including simple Long Short-Term Memory (LSTM)-based Event Model (EM), Bidirectional EM (BiEM), Tiered-EM, and Tiered-BiEM. The AUROC of stage-1 in DeepEAD (unsupervised detection) is 0.939, which outperforms all other methods except Tiered-EM (0.948). Stage-2 further increases the AUROC to 0.958. Further, these other methods do not incorporate attention mechanisms so they cannot provide explainability. By contrast, in DeepEAD, the attention matrix multiplied with context events enables DeepEAD to explain the importance of each context event for the detection results.

Furthermore, we compare the proposed DeepEAD with SOTA attention-based approaches including EM with fixed, syntactic, and two different semantic attention. As shown in Table II, the AUROC of DeepEAD is comparable with attention-based EM models. More importantly, the attention in EM models cannot guarantee concrete explainability. The reason is that the attention in EM models (e.g., LSTM) is applied to a complex combination of multiple inputs (e.g., previous hidden states). Instead, the attention in DeepEAD and DeepCASE is directly applied to each simple attribute in the context event as introduced in Section II-C. In addition, although DeepEAD adopts a similar attention mechanism, it fails to address the complex information between attributes and cannot provide any insights on the importance at attribute granularity. Therefore, compared with these works, DeepEAD provides superior one-to-one fine-grained explainability without sacrificing too much detection performance.

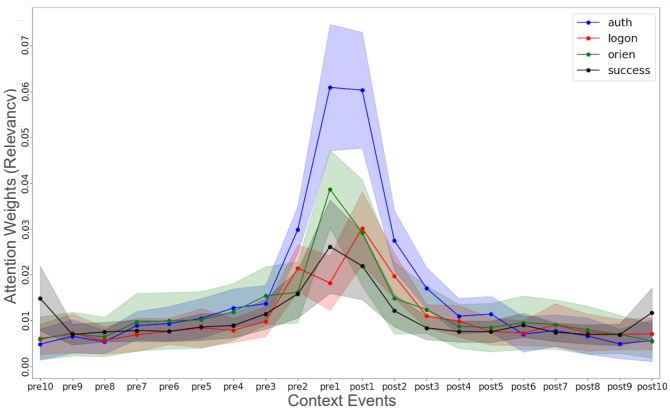

Fig. 3: Average attention weights over different attributes in each context event.

D. Explainability Analysis

Since each value in the attention vector of DeepEAD is directly applied to the context event, it enables us to explain the relevancy of each context event and attribute with the prediction. In this subsection, we analyze the explainability of our models from two perspectives: i) the general model behavior by analyzing the attention weights for all testing logs; and ii) the specific decision of the model by studying the attention weights in individual cases.

1) Explainability in General: To get an overview of the model behavior during the prediction, we consider the statistics of all attention weights over the test samples in 18 days of data. In Fig. 3, the solid lines show the attention weights for each context event (including 10 preceding and 10 postceding events listed in chronological order) and the shadows denote the standard deviation. In particular, each attribute (i.e., *authentication type*, *logon type*, *authentication orientation*, and *success/failure*) is represented with different colors.

In general, the context events that happen near the target event in time (e.g., the nearest preceding/postceding events *pre1* and *post1*) are shown to provide more attention than others, which is consistent with our assumption that nearest events offer more context information and are more relevant to the target event. Among different attributes, authentication types in neighboring context events are more important than others. We can also see that the attention weights in authentication orientation and success/failure of the neighboring preceding events (e.g., *pre1*) are higher than neighboring postceding events (e.g., *post1*) and those in logon type are quite the contrary. This indicates that the orientation and success/failure of most preceding events contribute more to the target events and the logon type of the next postceding events are more affected by the target event.

2) Explainability in Individual Cases: Attention weights can help explain the importance of corresponding attributes and context during the model prediction. We use the heatmap to visualize the attention weights over attributes of different context events. Fig. 4 shows a heatmap for a real-world normal case in our test dataset where lighter colors denote higher attention weights. Each column represents a context event from left to right in time order. For instance, the first column illustrates the earliest context event (*pre1*): `Unknown Network LogOff Success`. The first four rows are for four attributes in the system logs. Each cell exhibits the attention weight on the current attribute value and the sum of attention weights in the first four rows for each column is shown in the last row to suggest the importance of each context event. We can find that nearest context events in this case get higher attention weights and the authentication type of nearest preceding and postceding events have the most attention, which aligns with the trend of attention statistics depicted in Fig. 3. An `Unknown Network LogOff Success` (*pre1*) after three `Kerberos Network Logon Success` brings valuable context to the target event. Also, although both the nearest preceding and postceding events are `Unknown Network LogOff Success` (*pre2*, *pre3*, *pre4*), different attention weights are given: *authentication type* in *post1*, *logon type* in *pre1*, and *orientation type* in *post1* are more important than the corresponding one in *post1*, *pre1*, and *post1*, and the whole *post1* event contributes more than *pre1* event during the decision making.

Even though more attention weights are more likely given to nearest context events, they learn to differ for different cases. Fig. 5 presents an anomaly case where the fifth/sixth postceding event and fourth/fifth preceding event are most relevant with the predicted events while the nearest preceding event is not that important like general ones. This may be because there are repeated `NTLM Network LogOn Success` and `Unknown Network LogOff Success` around the target event. When the occurrence of such a pattern exceeds a threshold (e.g., twice in this case), it is brought to the attention of the model. Among all attributes in this case, orientation types usually gain more attention weights but some specific values (e.g., *logon type* `Network` in *post1*) can also receive high attention, which indicates that they contribute more to the prediction of the target event.

IV. Related Works

Machine learning has been widely used for detecting anomalies to deal with the huge amount of log data generated by modern systems and complicated contexts among them. A series of RNN-based systems [9]–[11] are proposed to predict future events from the previous log sequence. Similar to our stage-1, they determine anomaly if a low probability is assigned to the ground truth event. The authors in [12] and [13] conclude the context relation among different logs with authentication or heterogeneous graphs and then apply a traditional logistic regression or clustering algorithm. Their methods rely on pre-defined rules to construct the graph, which is hard to adapt to other system logs.

ALEAP [22] is one of the earlier works that incorporate attention mechanism into LSTM-based event prediction for anomaly detection, but it does not leverage attention for the explanation. Further, the work of [14] and [15] tries to use attention weights to explain the prediction behaviors. However,

Fig. 4: One example of the heatmap showing attention weights on a normal case.

Fig. 5: One example of the heatmap showing attention weights on an anomaly case.

these attentions are applied over a complex combination of multiple inputs, e.g., hidden states, not each attribute token, whose explainability is controversial [16], [23]. While DeepCASE [18] mitigates this by applying a 1-to-1 map between attention and context event, it assumes inputs as well-represented events and does not consider complex relations among multiple attributes of system logs.

V. Conclusion

In this work, we have presented DeepEAD, an attention-equipped encoder-decoder architecture for explainable anomaly detection from system logs. A Transformer-based encoder is adopted to address complex relations among attributes in multiple context events. A multi-attribute attention decoder is designed to generate fine-grained attention weights so as to enable concrete explainability for each context event attribute. During the cold start, we have applied an unsupervised learning-based event decoder for event prediction. An anomaly will be detected if the predicted event does not happen. When anomaly labels are available, we apply transfer learning to fine-tune a binary classifier for anomaly prediction.

Experimental evaluation on a large-scale real-world dataset has shown that the DeepEAD achieves comparable performance with state-of-the-art works. Additionally, the explainability analysis on context events has demonstrated the effectiveness of the DeepEAD to facilitate human investigation.

References

[1] M. Du *et al.*, "Lifelong anomaly detection through unlearning," in *Proc. of the ACM SIGSAC Conf. on Computer and Commun. Security*, pp. 1283–1297, 2019.

[2] F. B. Kokulu *et al.*, "Matched and mismatched socs: A qualitative study on security operations center issues," in *Proc. of the ACM SIGSAC Conf. on Computer and Commun. Security*, pp. 1955–1970, 2019.

[3] FireEye, "M-Trends 2021: Cyber Security Insights." https://vision.fireeye.com/editions/11/11-m-trends.html.

[4] D3 Security, "The Time for SOAR is Now." https://d3security.com/blog/the-time-for-soar-is-now/.

[5] Redscan, "Overcoming cyber security alert fatigue." https://www.redscan.com/news/overcoming-cyber-security-alert-fatigue/.

[6] DEMISTO, "The State of SOAR Report, 2018." https://start.paloaltonetworks.com/the-state-of-soar-report-2018.

[7] L. Bilge, Y. Han, and M. Dell'Amico, "Riskteller: Predicting the risk of cyber incidents," in *Proc. of the ACM SIGSAC conf. on computer and commun. security*, pp. 1299–1311, 2017.

[8] Y. Liu *et al.*, "Cloudy with a chance of breach: Forecasting cyber security incidents," in *24th USENIX Security Symposium (USENIX Security 15)*, pp. 1009–1024, 2015.

[9] M. Du *et al.*, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proc. of ACM SIGSAC conf. on computer and commun. security*, pp. 1285–1298, 2017.

[10] Y. Shen, E. Mariconti, P. A. Vervier, and G. Stringhini, "Tiresias: Predicting security events through deep learning," in *Proc. of ACM SIGSAC Conf. on Computer and Commun. Security*, pp. 592–605, 2018.

[11] A. R. Tuor *et al.*, "Recurrent neural network language models for open vocabulary event-level cyber anomaly detection," in *Workshops at the thirty-second AAAI conf. on artificial intelligence*, 2018.

[12] F. Liu *et al.*, "Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within enterprise," in *Proc. of the ACM SIGSAC Conf. on Computer and Commun. Security*, pp. 1777–1794, 2019.

[13] B. Bowman, C. Laprade, Y. Ji, and H. H. Huang, "Detecting lateral movement in enterprise computer networks with unsupervised graph AI," in *23rd Int. Symp. on Research in Attacks, Intrusions and Defenses (RAID 2020)*, pp. 257–268, 2020.

[14] A. Brown *et al.*, "Recurrent neural network attention mechanisms for interpretable system log anomaly detection," in *Proc. of the First Workshop on Machine Learning for Computing Systems*, pp. 1–8, 2018.

[15] A. Patil *et al.*, "Explainable LSTM model for anomaly detection in HDFS log file using layerwise relevance propagation," in *2019 IEEE Bombay Section Signature Conf. (IBSSC)*, pp. 1–6, IEEE, 2019.

[16] S. Jain and B. C. Wallace, "Attention is not explanation," *arXiv preprint arXiv:1902.10186*, 2019.

[17] A. D. Kent, "Cybersecurity Data Sources for Dynamic Network Research," in *Dynamic Networks in Cybersecurity*, Imperial College Press, 2015.

[18] T. van Ede *et al.*, "Deepcase: Semi-supervised contextual analysis of security events," *IEEE Security and Privacy*, 2022.

[19] A. Vaswani *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[20] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of the IEEE conf. on computer vision and pattern recognition*, pp. 770–778, 2016.

[22] S. Fan *et al.*, "Aleap: Attention-based lstm with event embedding for attack projection," in *Int. Performance Computing and Commun. Conf. (IPCCC)*, pp. 1–8, IEEE, 2019.

[23] S. Wiegreffe and Y. Pinter, "Attention is not not explanation," *arXiv preprint arXiv:1908.04626*, 2019.