Anomaly Detection in Security Logs using Sequence Modeling

Simon Gökstorp, Jakob Nyberg, Yeongwoo Kim, Pontus Johnson and György Dán
Division of Network and Systems Engineering, KTH Royal Institute of Technology
Stockholm, Sweden
Email: {gokstorp, jaknyb, yeongwoo, pontusj, gyuri}@kth.se

***Abstract*—As cyberattacks are becoming more sophisticated, automated activity logging and anomaly detection are becoming important tools for defending computer systems. Recent deep learning-based approaches have demonstrated promising results in cybersecurity contexts, typically using supervised learning combined with large amounts of labeled data. Self-supervised learning has seen growing interest as a method of training models because it does not require labeled training data, which can be difficult and expensive to collect. However, existing self-supervised approaches to anomaly detection in user authentication logs either suffer from low precision or rely on large pre-trained natural language models. This makes them slow and expensive both during training and inference. Building on previous works, we therefore propose an end-to-end trained self-supervised transformer-based sequence model for anomaly detection in user authentication events. Thanks in part to an adapted masked-language modeling (MLM) learning task and domain knowledge-based improvements to the anomaly detection method, our proposed model outperforms previous long short-term memory (LSTM)-based approaches at detecting red-team activity in the "Comprehensive, Multi-Source Cyber-Security Events" authentication event dataset, improving the area under the receiver operating characteristic curve (AUC) from 0.9760 to 0.9989 and achieving an average precision of 0.0410. Our work presents the first application of end-to-end trained self-supervised transformer models to user authentication data in a cybersecurity context, and demonstrates the potential of transformer-based approaches for anomaly detection.**

***Index Terms*—cybersecurity, misuse detection, anomaly detection, sequence modeling, machine learning, transformer, LSTM**

I. Introduction

Activity logging is a crucial part of computer systems security. By collecting logs at hosts and in the network, cybersecurity personnel are able to monitor the state and dynamics of networked systems and processes in real-time. Such real-time monitoring of the system's behavior is essential for situational awareness as cyberattacks become increasingly sophisticated [1], [2].

A recent incident at Microsoft showcases the relevance of monitoring user authentication data in order to detect and prevent advanced persistent threat (APT) attacks. In this attack, state-backed actors used forged authentication keys to gain unauthorized access to e-mail servers and steal government data [3].

In the absence of well-defined malicious behavior to target, anomaly detection-based approaches can be used to identify activity that does not conform to expected activity patterns. Anomaly detection systems work by creating a model of normal system behavior. Broadly, they can be split into two primary categories: statistics-based and machine learning-based [2]. Statistics-based methods use statistical features, typically aggregated over set durations. A significant limitation of these techniques is their heavy reliance on expert knowledge and engineered features. Consequently, they often struggle to adapt to changing system dynamics. In contrast, machine learning (ML)-based approaches aim to improve on these weaknesses by learning patterns in system activity. They are also able to adapt to gradual changes in the activity if trained continuously or iteratively [2].

The most common learning method for machine learning-based anomaly detection in cybersecurity contexts is supervised learning [4]. However, this approach requires large amounts of labeled training data, which can be difficult and expensive to collect in a cybersecurity context. To avoid this limitation, some recent research efforts have investigated self-supervised learning. This learning method uses parts of the existing data as training labels.

A popular implementation of self-supervised learning is sequence modeling. Such models are trained to predict elements in sequences using incomplete contextual clues such as the previous elements of the sequence. This method allows the models to learn the relationships between the elements of sequences comprising normal activity without needing labeled data.

Several recent works have proposed using sequence modeling for anomaly detection in computer logs. Each log entry consists of a series of elements that can be considered as words, and the complete entry is a sentence describing some event. These methods involve deploying a sequence prediction model on the log entries, and using the performance of the sequence model as an anomaly score for the anomaly detection task.

However, existing approaches have various limitations. First, some approaches use older sequence modeling approaches, for example using LSTM-based models [5], [6], leading to sub-optimal performance. Second, more recent approaches rely on large language models pre-trained on natural language [7], [8]. The use of such large models makes these approaches slow and expensive to train and employ as detection models, for example requiring targeted efficiency optimizations to facilitate practical applications [8].

To address these limitations, we propose a novel approach

using the transformer architecture [9] with self-supervised learning for anomaly detection in user authentication data. We train the transformer model end-to-end, including the token embedding and deembedding steps. We thereby avoid the reliance of recent approaches on large language models, enabling smaller and faster models. We also propose improvements to the sequence modeling and training procedures in previous work [5], [6], including an adapted MLM learning task, that help further improve the anomaly detection results.

A recent analysis of security practitioners' view of ML-based methods [10] found that practitioners generally think that ML methods should be used in conjunction with rule-based methods, rather than replacing them. Motivated by this, we evaluate our method for the use-case of a filter in a log processing pipeline. We therefore present results in terms of specificity at 100% recall of each model, in addition to the common metrics of area under curve (AUC) and average precision (AP).

Concretely, the contributions of our work are summarized as follows:

- We propose a self-supervised transformer for anomaly detection in user authentication logs. Unlike previous work, we use self-supervised learning for non pre-trained models that are significantly smaller than existing approaches using BERT.
- We propose an MLM learning task for model training for anomaly detection in user authentication. Unlike previous work, we use the same masking process for training as for inference.
- We consider the use-case of the self-supervised model as a filter in conjunction with other rule-based or ML-based methods in a log processing pipeline, evaluating the ability of each model we compare to discard normal data while retaining the data that corresponds to anomalous activity.

II. Related Work

Anomaly detection for cybersecurity traditionally relies on aggregated statistical features to identify anomalous activity in pre-defined time windows. For example, [11] uses the Isolation Forest algorithm on aggregated statistical features for the goal of insider threat detection. In a similar approach, [12] augments the Isolation Forest algorithm to improve the explainability of its results, and apply this approach to network activity data to identify time intervals containing anomalous activity. These approaches rely on feature engineering to select salient features to be used in the anomaly detection, and are limited to identifying anomalous regions or time intervals rather than specific instances.

Machine learning-based approaches, and particularly deep neural networks, have become the state-of-the-art in numerous fields and are seeing growing interest within anomaly detection [13]. A survey of anomaly-based network intrusion detection carried out in 2022 [4] found that the most common method for training anomaly detection models is supervised learning, with works such as [14] and [15] proposing different approaches using deep learning for intrusion and anomaly detection. However, supervised learning requires large amounts of labeled data for training [13]. Collecting and producing this labeled data can be extremely expensive and time-consuming, especially within the cyber-domain. Thus, the reliance on, and scarcity of, labeled data limits the practical applicability and generalisability of supervised learning for anomaly detection.

Some approaches therefore investigate the use of *self-supervised* learning, avoiding or greatly reducing the need for labeled data. One example of this is [16], who adapted self-supervised feature-extraction approach from computer vision for use for network intrusion detection. They showed that their approach outperformed other self-supervised approaches, while being slightly outperformed by supervised learning methods. Another approach was proposed by [17], who applied a graph neural network for self-supervised intrusion detection on network traffic flows, showcasing significantly improved performance compared to using raw features or previous, baseline approaches. Lastly, [5] proposed an approach using an LSTM model trained with self-supervision for anomaly detection on network authentication data. The work was expanded upon by [6] by introducing an attention mechanism to the LSTM architecture. Both these approaches are based on the idea of using the sequence modeling loss as a metric for anomaly detection. More recently, [18] proposed a two-step process combining supervised with unsupervised learning for intrusion detection, achieving improved results compared to [5] on the Comprehensive, Multi-Source Cyber-Security Events (CMSCSE) dataset [19].

The *transformer* neural network architecture [9] has become a popular choice in other fields such as natural language processing (NLP) [20]. A few works have combined the transformer architecture with self-supervised learning for anomaly detection in a cybersecurity setting. [21] applied a transformer model with self-supervision to Windows event logs. While the model is trained in a self-supervised manner, the model is later tuned using a baseline dataset assumed to be anomaly-free, thereby implicitly reintroducing the requirement for labeled data. The authors also only evaluate the model on a synthetic dataset which limits the reliability of their achieved results for real-world applications. Some works [7], [8] have proposed methods using BERT [22] for sequence modeling of security logs for anomaly detection. BERT is a large language model pre-trained on natural language, primarily from open web sources, that can be finetuned for downstream tasks. Although these approaches are able to harness the language modeling performance of BERT, this use of extremely large neural networks also makes them slow and expensive to fine-tune and to apply during inference.

Finally, we note that these closely related works are unfortunately difficult to compare quantitively due to them using different datasets.

III. Background

In this section we introduce the dataset used and the two neural network model architectures being compared in this work: LSTM and transformer.

A. Dataset

We use the CMSCSE dataset that was collected at the Los Alamos National Laboratory (LANL) [19]. The full dataset contains data from several sources, including Windows authentication event logs, collected over a period of 30 days. At different times during the 30 days, a red team used stolen user credentials to attempt to log in to computers in the network. The authentication events caused by the red team's activity are provided as labels together with the event data. Table I shows the number of authentication events and the number of red team events for days six, seven, and eight. Because this is the only data source in the dataset that contains labels, necessary for model evaluation, we only use the authentication events.

TABLE I
COUNTS OF AUTHENTICATION EVENTS AND RED TEAM EVENTS IN THE CMSCSE DATASET FOR DAYS SIX, SEVEN AND EIGHT.

Day	# Events	# Red Events
6	7 066 526	0
7	7 005 987	1
8	7 022 147	273

Each authentication event is comprised of eleven value fields, which are listed in Table II together with the number of unique occurrences per field. The data is categorical, with each field containing a single string value. Fields that relate to users, domains and computer names have been anonymized by the dataset authors. During the span of days 6–8, 10 163 unique users, 799 unique domains and 12 229 unique computers appear. These three types of fields have "Source" and "Destination" variants. There is a significant overlap between the source and destination fields, with a majority of values appearing in both fields at least once.

TABLE II
NAMES OF FIELDS IN THE CMSCSE DATASET AUTHENTICATION EVENTS, AND THE NUMBER OF UNIQUE VALUES PER FIELD FOR DAYS SIX, SEVEN AND EIGHT.

Field Name	# Unique Values
Source User	10 159
Source Domain	738
Destination User	10 075
Destination Domain	720
Source Computer	11 450
Destination Computer	11 439
Authentication Type	10
Logon Type	10
Authentication Orientation	7
Success/Failure	2

B. LSTM Networks

LSTM Networks are a form of recurrent neural network (RNN) designed to be able to incorporate long-range dependencies in sequence predictions [23]. RNNs processes elements in the input sequence sequentially, and builds an internal representation that is dependent on both the next element and the previous representation. The internal representation acts as a "memory" of previous elements that have occurred in the sequence. LSTM networks improve upon this process by allowing the model to selectively incorporate elements in the representation, which allows them to process longer sequences than previous RNN architectures.

We include the LSTM network and Tiered LSTM network models from [5], [6] as baseline models, specifically the "EM with Semantic 1 attention" and "TA-BEM" models. The Tiered LSTM model uses a second LSTM network to propagate contextual information from previous authentication events involving the same user and a bidirectional LSTM network to process the features in each authentication event, thereby seeking to improve the prediction accuracy of the model at the cost of slower training and inference due to the added temporal dimension.

C. Transformer

The transformer is a neural network architecture built almost entirely around *attention* mechanisms [9]. Attention mechanisms allow neural network models to assign numerical weights to different elements of an input sequence, and compare elements in the sequence either to themselves or other sequences. Unlike an RNN, which processes each input element sequentially, transformers process the entire input sequence in one operation. This allows for more effective parallel processing compared to RNNs and an improved ability to identify long-range connections between elements, but with the cost of higher computational complexity with increasing input length. We use a decoder-only transformer model as in previous work on sequence modeling with transformers [20].

D. Masked Language Modeling

MLM is a learning task introduced by [22] to train the NLP model BERT. The main principle of MLM is to mask out some tokens in the input sequence, either replacing them with a "[MASK]" token, or with another random token. The model is then tasked with correctly restoring the masked tokens. We use this learning task to train a bidirectional transformer for sequence modeling on the CMSCSE dataset, as it allows the model to consider both the tokens before and after the masked token when making its prediction.

IV. METHOD

This section describes the anomaly detection method and model training schemes used in this work. First, we describe the creation process of a vocabulary corpus from the authentication events. Second, we introduce the method for performing anomaly detection. Next, we detail the self-supervised training and inference methods. Finally, we explain the method of evaluating the anomaly detection models.

A. Vocabulary Creation

We create a vocabulary of tokens present in the authentication events to be used by the anomaly detection models. A token represents a single field-delimited string value, with each such string value being uniquely mapped to an integer index via a dictionary mapping. In order to keep the vocabulary size

bounded, and to handle values unseen in the training set like new usernames, tokens that occur less than a certain frequency threshold are replaced by an "out-of-vocabulary" token.

We compare two different methods of defining the vocabulary. First, we consider the vocabulary proposed by [5], [6], which we refer to as *field-based vocabulary*. In this method, each field included in the authentication events has a separate vocabulary. For example, a user that appears in both the "Source User" and "Destination User" fields will be considered different tokens and therefore map to different indices.

The high number of duplicate tokens in source and destination fields led us to consider a second method of defining the vocabulary, which we refer to as *global vocabulary*. With the global method, all fields in the events share the same vocabulary. Identical tokens will map to the same index regardless of the field it occurs in. This both produces a much smaller vocabulary and reduces the number of string values that are pruned by the out-of-vocabulary frequency threshold.

Vocabularies were generated using word counts from days six, seven and eight, using a token frequency cutoff of 40[1] occurrences across the three days. This cutoff retains $99\,\%$ of all tokens present in the data, while reducing the number of unique tokens by $18\,\%$. The resulting field-based vocabulary had 36 752 tokens, and the global vocabulary 19 859 tokens.

B. Anomaly Detection

We adopt the general approach of self-supervised machine learning for anomaly detection used in [5], [6]. This approach applies the model on the authentication events from one day D at a time, and works as follows:

1) Train model on the authentication events from the previous day D_{i-1}
2) Apply model on the authentication events from day D_i to perform anomaly detection
3) Repeat for the next day D_{i+1}

In this way the anomaly detection model is continually updated, allowing it to adapt to drift in network activity and handle new users introduced to the network (with a 1 day lag-time).

Due to the self-supervised learning method, the models do not explicitly learn to classify events as anomalous. Instead, an anomaly score is computed for each event based on the cross-entropy loss for the predictions of its tokens. When the model produces a loss that is significantly higher than its average performance for a set of authentication events, it means that the model failed at predicting the tokens of that event. This method is based on the principle that higher prediction losses signify events that deviate from the distribution that the model was trained on, and the assumption that this correlates to anomalous activity. A threshold can be set for the anomaly score, where all authentication events that have scores above the threshold are deemed as anomalous.

The anomalous activity we want to identify in the CMSCSE dataset is attempts to log in using stolen credentials. However, the sequence models identify any events that do not conform to the model that has been learned, which are not necessarily the targeted anomalies. Through manual inspection, we identified two patterns in the dataset that were not correlated to the anomalous activity we want to identify:

- The vast majority of authentication events in the dataset have the same value in the "Source User" and "Destination User" fields ($98\,\%$ of events on day six, seven and eight). Events with a different source user and destination user are very uncommon.
- Similarly, nearly all authentication events in the dataset were successful ($99\,\%$ of events on day six, seven and eight), while authentication failures are extremely rare.

In both these cases, the model is likely to have a high performance loss when tasked with predicting the rare case. However, whether the source user and destination user match is not related to whether the event represents an attempt at logging in with stolen credentials, as the attacker can freely control the values of both of these fields. Likewise, an authentication failing is also not indicative of the use of stolen credentials, as the attacker is known to possess valid credentials. We therefore omit the model's prediction loss on the "Source User", "Destination User" and "Success/Failure" fields when computing the anomaly score.

While these fields are excluded for the anomaly score computation, they are still included as input for the sequence modeling task. In doing so, the sequence models are able to learn connections such as a certain source user normally using a certain computer, and can still use these fields when predicting the other fields.

C. Training and Inference

We train and evaluate four model variants: a transformer-based model, a bidirectional transformer-based model, and the two baseline models introduced in Section III. The unidirectional models (the baseline LSTM model and the transformer model) are trained with a causal language modeling task, where each field in the event, except the first, is predicted with the previous fields as context. The sequence order is artificially set as the order fields appear from left to right. This order is also reflected in Table II, where the top field is also the left-most field of the events. The prediction loss is calculated as the average cross-entropy between the output distributions of the model and the input sequence.

The bidirectional transformer model uses the same architecture as the unidirectional transformer model, but is trained with an MLM task. We adapt the approach suggested by [8], based on the original MLM task introduced by [22], in two ways:

- In the training method used by [8], a random number of tokens were masked for each sequence. However, during inference the model is always given exactly one masked token to predict. We therefore randomly choose one token to mask for each event during training, but always mask exactly one token. This provides consistency to the task given to the model during training and during inference.

[1] Cutoff value also used by [5], [6].

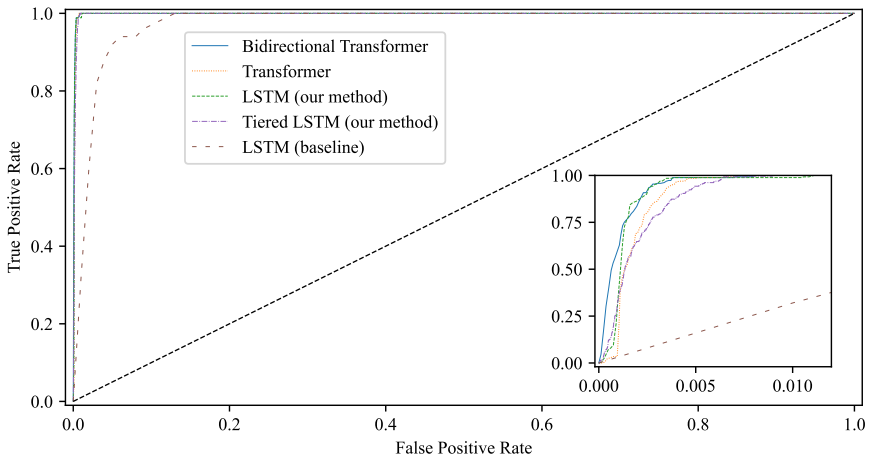

Fig. 1. Receiver operating characteristic curve for the evaluated models with field-based vocabulary. The subplot shows the region of FPR$\leq$0.015 in greater detail. For each model type the version performing the closest to the mean in terms of AUC is shown. The LSTM (baseline) plot is approximated from [5].

- The method used by [8] also includes the "[CLS]" token that was used in the original work to summarize the sentence and subsequently to predict whether a pair of sentences were semantically connected. However, in [8] this next sentence prediction is not carried out during model inference. Thus, including this token only slows down the training and inference processes, with no gain. We therefore remove the "[CLS]" token, instead only asking the model to predict the masked token for each event.

With the MLM task, the masked field is predicted by the model with all other fields of the event as context, meaning the order of the fields is irrelevant. During training the prediction loss is calculated only for the field that was masked. During inference the model is applied to predict each field in each event, one at a time, producing a prediction loss for each field.

Each model is trained for three epochs on the training set. When training the non-tiered LSTM model and the transformer models we shuffle the training set before each epoch of training. We do not shuffle the training set for the Tiered LSTM model as it makes predictions based on earlier events. We also perform validation after each epoch of training in order to determine a choice of hyperparameters based on the training data. At the end of training the version with the best validation score (average sequence modeling loss) was picked for evaluation.

All models shared the same hyperparameter choices to the extent possible. We use a single layer of 128 neurons, attention size 128 for the LSTM models, and token embedding size 128. Each model was trained using the ADAM [24] optimizer with a learning rate of $1e^{-4}$, reduced by a factor of 2 after each epoch, and a dropout rate of 0.25.

This is a more involved training regiment than that suggested in the previous work of [5], [6], which used only a single epoch of training with learning rate scheduling during the epoch, without validation, training data shuffling and dropout. Maintaining a constant learning rate throughout each epoch ensures every training sample is weighted equally during training, and training for several epochs provides more time for the model to fine-tune on the training data. We also use a lower learning rate and add dropout and multi-epoch training to further improve the stability of training. Initial experiments indicated that training for several epochs with a lower learning rate significantly improved the validation loss, while dropout was used to prevent overfitting. We performed brief hyperparameter tuning based on validation performance to find values that achieved good results without excessive resource cost.

Shuffling the training data is a particularly important change due to the format of the dataset being split into days. The characteristics of the events change significantly between daytime and nighttime, for example there is less traffic at night and most users are most active during the day. When training on the data in sequential order the model first trains on the hours just after midnight, and ends training with the hours just before midnight at the end of the day. This causes the model to be tuned to nighttime activity at the end of training at cost of performance on daytime activity. This is avoided by shuffling the training set, resulting in better performance overall and importantly a lower performance discrepancy between events created during the day and those created during the night.

During inference, the models are fed all authentication events from the evaluation set and compute an anomaly score for each event. All events with a score above the anomaly threshold are classified as anomalous.

Model training and evaluation was implemented using Python and the PyTorch framework [25]. All training and evaluation was done using virtual machines on the Google Cloud platform, equipped with 8 vCPUs, 30 GB of RAM and 1 V100 GPU.

D. Evaluation

The CMSCSE dataset comes with a set of labels for events constituting known anomalous activity, caused by a red team

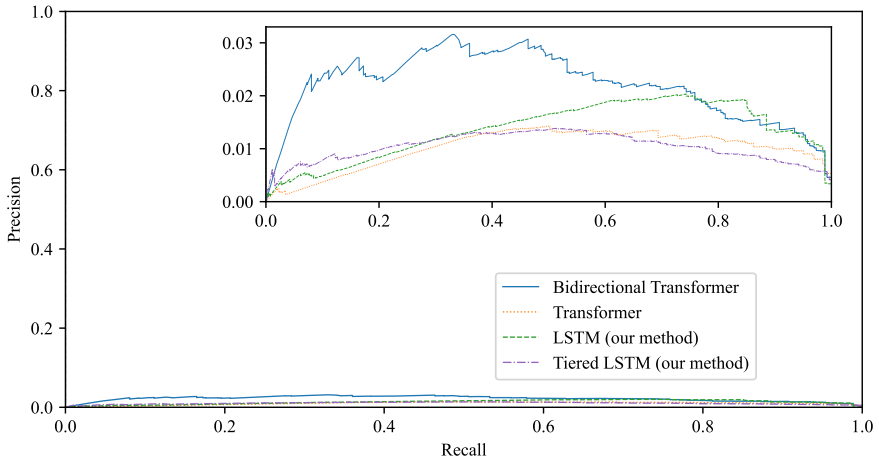

Fig. 2. Precision-recall curve for the baseline LSTM and tiered LSTM models and the transformer model with field-based vocabulary. The subplot shows the region of precision$\leq$0.033 in greater detail. For each model type the version performing the closest to the mean in terms of AUC is shown.

attempting to log in using stolen credentials. We use these labels to calculate the detection accuracy of the models on our evaluation set.

We use the receiver operating characteristic (ROC) curve to consider the possible combinations of true positive rate (TPR) and FPR achievable by the models at different threshold values, and compute the area under curve (AUC) of this curve as a performance metric, as done by [5], [6]. It is important to note that the dataset is heavily unbalanced between the amounts of anomalous and benign activity, with only 0.004% of events in day eight being labeled as red-team activity. As such, the AUC values will be skewed towards high values and may give a false impression of model performance if presented alone. Therefore, in addition to the AUC, we report the precision and recall of the models. We provide precision-recall curves which show the possible trade-offs between precision and recall achievable by the models depending on the prediction threshold.

Finally, we consider the use-case of our models as an initial filter in a log processing pipeline or as part of an ensemble model. In such a use-case the model's ability to filter out the normal activity while retaining all anomalous activity is more relevant than, for example, its precision. For this reason we also report the specificity of each model when achieving 100% recall. This represents what proportion of the non-anomalous data can be filtered out by the model without discarding any anomalous events, in an ideal case.

V. Numerical Results

Each model variant, except the tiered LSTM, was trained and evaluated five times with different random number generator seeds. The tiered LSTM models were only trained and evaluated on three different seeds, as they are slow and resource-intensive due to the tiered structure. The seeds affect the model parameter initialization and the sequence in which data is loaded during training (if shuffling is used), and thus lead to variations in the final result. For each model we average each performance metric across all seeds and compute the standard deviation.

We used day seven for training and day eight for evaluation to be able to directly compare our results with [5] and [6]. A subset of 100 000 events from day six was used for validation during training.

We report results for the two baseline models as reported by [5], [6], the results achieved by these models with our training and anomaly detection method outlined in section IV, and the results of our two transformer models. Table III shows the AUC, average precision and specificity at 100% recall for each model, averaged across five different random number generator seeds (three for the tiered LSTM models), along with one unit of standard deviation.

Figure 1 shows ROC curves for all models except the baseline tiered LSTM model, as the authors of [6] did not publish this data. The figure shows that the bidirectional transformer model and the LSTM model outperform the other models, with the bidirectional transformer achieving a slightly higher AUC than the LSTM model. Both these models reach a higher TPR at lower FPR than the transformer and tiered LSTM models, resulting in a higher AUC. All models significantly outperform the baseline LSTM model. Figure 2 shows the precision-recall curve for all non-baseline models, while the baseline models could not be included as precision-recall data was not published. In this figure the difference between the bidirectional transformer and the other models is more significant, with the bidirectional transformer achieving higher precision at lower recall when compared to the unidirectional transformer, LSTM and tiered LSTM.

Looking at AUC, all models significantly outperform the baseline LSTM and tiered LSTM models. The highest AUC is achieved by the bidirectional transformer model with field-based vocabulary, although the LSTM model achieves nearly as high AUC when using our training and anomaly detection

TABLE III
AREA UNDER CURVE, AVERAGE PRECISION, AND SPECIFICITY AT 100% RECALL. MODELS WERE TRAINED WITH TWO TYPES OF TOKEN VOCABULARIES. AVERAGES ARE PRESENTED TOGETHER WITH ONE UNIT OF STANDARD DEVIATION. "−" DENOTES THAT THIS VALUE WAS NOT REPORTED.

Model Name	Vocab.	AUC	Average Precision	Specificity at 100% recall
LSTM (baseline) [5]	Field	0.9760 ± 0.0030	−	$0.8800 \pm -$
Tiered LSTM (baseline) [6]	Field	0.9880 ± 0.0030	−	−
LSTM (our method) [5]	Field	0.9986 ± 0.0002	0.0200 ± 0.0013	0.9888 ± 0.0012
LSTM (our method) [5]	Global	0.9980 ± 0.0001	0.0125 ± 0.0010	0.9909 ± 0.0016
Tiered LSTM (our method) [6]	Field	0.9982 ± 0.0004	0.0153 ± 0.0017	0.9890 ± 0.0049
Tiered LSTM (our method) [6]	Global	0.9975 ± 0.0004	0.0093 ± 0.0014	0.9884 ± 0.0025
Transformer	Field	0.9981 ± 0.0006	0.0166 ± 0.0047	0.9920 ± 0.0024
Transformer	Global	0.9984 ± 0.0005	0.0193 ± 0.0131	0.9923 ± 0.0010
Bidir Transformer	Field	$\mathbf{0.9989 \pm 0.0003}$	$\mathbf{0.0410 \pm 0.0153}$	$\mathbf{0.9927 \pm 0.0014}$
Bidir Transformer	Global	0.9988 ± 0.0004	0.0244 ± 0.0082	0.9908 ± 0.0014

method. Considering a global vocabulary the bidirectional transformer still achieves the highest average AUC, followed by the unidirectional transformer model.

In terms of average precision the best performing model by far is the bidirectional transformer with field vocabulary, followed by the same model with global vocabulary. The bidirectional transformer with field-based vocabulary also achieves the highest specificity at 100% recall. Due to the MLM learning task the bidirectional transformer is able to use the full context of all other tokens in each event for each token prediction. This capability allows it to learn the distribution of tokens in normal activity better than the unidirectional transformer, leading to more accurate identification of anomalous authentication events.

TABLE IV
AVERAGE TIME TAKEN FOR TRAINING AND EVALUATION OF MODELS, IN MINUTES.

Model Name	Vocab.	Training	Evaluation
LSTM (baseline) [5]	Field	−	−
Tiered LSTM (baseline) [6]	Field	−	−
LSTM (our method) [5]	Field	26.98	8.78
LSTM (our method) [5]	Global	16.00	7.47
Tiered LSTM (our method) [6]	Field	348.73	115.24
Tiered LSTM (our method) [6]	Global	243.43	83.78
Transformer	Field	24.42	8.19
Transformer	Global	15.30	7.09
Bidir Transformer	Field	25.70	19.30
Bidir Transformer	Global	16.62	14.57

Table IV shows the average time taken for training and evaluation for each model and vocabulary type. The authors of [5], [6] did not publish the time taken for training and evaluating their models, we are therefore unable to report these values. Notably, the Tiered LSTM model took significantly longer for training and evaluation than the LSTM and transformer models. The unidirectional transformer and the LSTM models using our method show very similar training and evaluation times, with the transformer being slightly faster in all cases. Due to the MLM task, the bidirectional transformer is slightly slower in training than the unidirectional transformer, and significantly slower during evaluation as each event must be processed several times by the bidirectional model. Comparing the two types of vocabulary, models using a global vocabulary took on average 36% less time to train and 20% less time for evaluation. This vocabulary is significantly smaller than the field-based one, resulting in smaller memory footprint and faster processing.

VI. DISCUSSION

Self-supervised versus Supervised learning: As the most popular learning method for machine learning-based anomaly detection in cybersecurity is supervised learning, it is important to compare our self-supervised approach to a supervised method. The related work of [18] proposes a method combining a supervised learning module named LightGBM with an unsupervised module using K-means clustering. In their paper they report results achieved on the CMSCSE dataset, comparing these to the performance of the bidirectional LSTM model from [5]. Specifically, they achieve an AUC of 0.9832, while [5] achieved 0.9760, and our best model achieved an average AUC of 0.9989. It must be noted that the results presented by [18] are computed using a larger training and evaluation set than we have used in this work, and therefore despite using the same original dataset these results are not directly comparable as the data is split in different ways. The model proposed by [18] is also faster than our self-supervised models during inference. However, it provides a useful indication of the performance improvements we achieve with our method, and that methods using self-supervised learning are capable of matching the performance of those using supervised learning on this dataset without needing labeled data for training.

Ablation study: Table V presents an ablation study covering the four main changes we propose and detail in section IV-C, compared to the baseline LSTM method proposed by [5]:

1) Improvements to the model training method outlined in section IV-C, including training for three epochs instead of one, shuffling the training set before each epoch of training, reducing learning rate after each epoch, and adding dropout.
2) Applying a transformer-based model instead of an LSTM-based one
3) Computing the anomaly score for each event based on a subset of the fields in the dataset, described in section IV-B.
4) Applying a bidirectional transformer-based model with our adapted MLM task, as described in section IV-C

TABLE V
ABLATION STUDY OF OUR PROPOSED METHOD AND MODEL. FOR BREVITY PERFORMANCE IS ONLY PRESENTED FOR MODELS USING A FIELD-BASED VOCABULARY.

Model Name	AUC	Average Precision	Specificity at 100% recall
LSTM (baseline) [5]	0.9760 ± 0.0030	$-$	$0.8800 \pm -$
LSTM with our training method [5]	0.9939 ± 0.0002	0.0028 ± 0.0001	0.9838 ± 0.0009
Transformer with our training method	0.9947 ± 0.0015	0.0040 ± 0.0018	0.9892 ± 0.0019
Transformer with our training and anomaly detection method	0.9981 ± 0.0006	0.0107 ± 0.0029	0.9920 ± 0.0024
Bidirectional transformer with our training and anomaly detection method	$\mathbf{0.9989 \pm 0.0003}$	$\mathbf{0.0248 \pm 0.0084}$	$\mathbf{0.9927 \pm 0.0014}$

The table shows that each of these additions contribute to improving the results achieved across all three metrics. In terms of AUC and specificity at 100% recall, the largest improvements compared to the baseline are gained by using our training method, allowing the same LSTM model to achieve significantly better results. Interestingly, each metric sees the second largest improvement brought by different additions. The second-largest gain in specificity at 100% recall is seen in the change from an LSTM-based model to a transformer-based model, the second-largest change in terms of AUC is due to our anomaly detection method, and lastly the second-largest gain in AP is seen with the addition of the MLM task and the bidirectional transformer. Overall, while each improvement contributes to better performance, our training method stand out as most impactful.

Suitability of the approach: The sequence modeling performed for the unidirectional transformer and the baseline LSTM model is autoregressive, predicting the next element in each event based on the previous elements. However, the elements of the authentication events considered do not have a natural ordering. The order used in this work is the order in which the fields appear from left to right in the data. This puts the "Source User" field first, meaning that all predictions of the following fields are conditioned on that particular field. Any order could be used, however, which calls into question the suitability of an ordered sequence-modeling approach.

We train a bidirectional transformer using our modified MLM learning task. This is a non-sequential model that predicts each token in each authentication event conditioned on all other tokens in the event. Together with the self-attention step of the transformer model, which directly compares all tokens with each other, this makes the approach entirely unreliant on the ordering of the input values for each sample.

Base rate fallacy: While the transformer model achieves a high AUC and reaches high TPR at low FPR, the base rate of anomalies in the dataset must be taken into account when evaluating these results. Looking instead at precision as a more representative performance measures for this unbalanced dataset, the bidirectional transformer model achieves an average precision of 4.1%.

In this case, the performance of our best model is not high enough to allow it to be used as a detector on its own, as it would overwhelm a potential operator with false positives and cause alert fatigue. However, we believe that the model can be used together with other tools as an initial filter in a processing pipeline or model ensemble. By using the model in this way the data amount to be processed in the next step of automated processing, or considered by a human analyst, can be drastically reduced. To evaluate the models in this use-case we present the specificity achieved by each model when that model reaches 100% recall. These results show that the bidirectional transformer model with field-based vocabulary can filter out an average of 99.27% of normal activity, in an ideal scenario, without any false negatives, while the best LSTM-based model can only filter out on average 99.09%. The best baseline model from [5] achieved a specificity of 88%.

The CMSCSE dataset: In the CMSCSE dataset, the red team's only reported method of attack is to use stolen credentials [19]. The lack of variety in attacks in the dataset limits discussion about our approach's and the models' ability to generalize across different attack methods. In principle the approach can be transferred to different datasets, log sources and user activity profiles and attack behavior, with the choice of which fields to use for computing the anomaly scores needing special attention depending on the target attack behavior. However, investigating this is left for future work involving other datasets.

Adversary threat model: Since the approach involves training a model directly on network log data, an adversary could potentially perform a data poisoning attack by using their own activity to mold the training data. Assuming no other systems or safeguards are in place, an attacker could outnumber benign activity with their own malicious activity to the point where it is no longer considered anomalous by the model.

VII. CONCLUSION

We presented a transformer-based approach for anomaly detection in authentication event data. This method leverages self-supervised learning and thus does not require data labels for training. Compared to previous LSTM-based models, our transformer-based method shows superior performance. Specifically, we improve the AUC achieved from 0.9760 to 0.9989 and specificity at 100% recall from 0.8800 to 0.9927. We also achieve an average precision of 0.0410. We find the approach of sequence modeling on authentication log data promising; however, the dataset used in this work prevents us from drawing wider conclusions about transferability and performance on other log sources and attack behavior. Other than investigating the performance of our approach on different datasets, a potential direction for further improving our approach in future work is to incorporate context, possibly in the form of user or computer activity history.

References

[1] H.-J. Liao, C.-H. Richard Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 16–24, 2013.

[2] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, 2019.

[3] Z. Whittaker, *Microsoft lost its keys, and the government got hacked*, Jul. 2023. [Online]. Available: https://techcrunch.com/2023/07/17/microsoft-lost-keys-government-hacked/.

[4] Z. Yang, X. Liu, T. Li, *et al.*, "A systematic literature review of methods and datasets for anomaly-based network intrusion detection," *Computers & Security*, vol. 116, p. 102 675, May 2022, ISSN: 0167-4048. DOI: 10.1016/j.cose.2022.102675.

[5] A. Tuor, R. Baerwolf, N. Knowles, B. Hutchinson, N. Nichols, and R. J. Jasper, "Recurrent neural network language models for open vocabulary event-level cyber anomaly detection," in *AAAI Workshops*, 2018.

[6] A. Brown, A. Tuor, B. Hutchinson, and N. Nichols, "Recurrent neural network attention mechanisms for interpretable system log anomaly detection," in *Proceedings of the First Workshop on Machine Learning for Computing Systems*, 2018.

[7] H. Guo, S. Yuan, and X. Wu, "LogBERT: Log anomaly detection via BERT," in *2021 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2021, pp. 1–8.

[8] Y. Lee, J. Kim, and P. Kang, "LAnoBERT: System log anomaly detection based on BERT masked language model," *Applied Soft Computing*, vol. 146, p. 110 689, Oct. 2023. DOI: 10.1016/j.asoc.2023.110689.

[9] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017, pp. 5998–6008.

[10] J. Mink, H. Benkraouda, L. Yang, *et al.*, "Everybody's Got ML, Tell Me What Else You Have: Practitioners' Perception of ML-Based Security Tools and Explanations," in *2023 IEEE Symposium on Security and Privacy (SP)*, ISSN: 2375-1207, May 2023, pp. 2068–2085. DOI: 10.1109/SP46215.2023.10179321.

[11] G. Gavai, K. Sricharan, D. Gunning, R. Rolleston, J. Hanley, and M. Singhal, "Detecting Insider Threat from Enterprise Social and Online Activity Data," in *Proceedings of the 7th ACM CCS International Workshop on Managing Insider Security Threats*, 2015.

[12] M. A. Siddiqui, J. W. Stokes, C. Seifert, *et al.*, "Detecting cyber attacks using anomaly detection with explanations and expert feedback," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 2872–2876.

[13] N. Elmrabit, F. Zhou, F. Li, and H. Zhou, "Evaluation of machine learning algorithms for anomaly detection," in *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, 2020.

[14] C. Xu, J. Shen, X. Du, and F. Zhang, "An intrusion detection system using a deep neural network with gated recurrent units," *IEEE access : practical innovations, open solutions*, vol. 6, pp. 48 697–48 707, 2018. DOI: 10.1109/ACCESS.2018.2867564.

[15] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41 525–41 550, 2019. DOI: 10.1109/ACCESS.2019.2895334.

[16] Z. Wang, Z. Li, J. Wang, and D. Li, "Network Intrusion Detection Model Based on Improved BYOL Self-Supervised Learning," *Security and Communication Networks*, vol. 2021, pp. 1–23, Oct. 2021, ISSN: 1939-0122, 1939-0114. DOI: 10.1155/2021/9486949. [Online]. Available: https://www.hindawi.com/journals/scn/2021/9486949/.

[17] E. Caville, W. W. Lo, S. Layeghy, and M. Portmann, "Anomal-E: A self-supervised network intrusion detection system based on graph neural networks," *Knowledge-Based Systems*, vol. 258, p. 110 030, Dec. 2022, ISSN: 0950-7051. DOI: 10.1016/j.knosys.2022.110030.

[18] Z. Zhang, L. Wang, G. Chen, *et al.*, "STG2P: A two-stage pipeline model for intrusion detection based on improved LightGBM and K-means," *Simulation Modelling Practice and Theory*, vol. 120, p. 102 614, Nov. 2022, ISSN: 1569-190X. DOI: 10.1016/j.simpat.2022.102614. (visited on 10/10/2023).

[19] A. D. Kent, "Cybersecurity data sources for dynamic network research," in *Dynamic Networks in Cybersecurity*, 2015.

[20] T. B. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.

[21] K. Steverson, C. Carlin, J. Mullin, and M. Ahiskali, "Cyber intrusion detection using natural language processing on windows event logs," in *2021 International Conference on Military Communication and Information Systems (ICMCIS)*, May 2021, pp. 1–7.

[22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," May 2019. DOI: 10.48550/arXiv.1810.04805.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, 1997.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017. arXiv: 1412.6980 [cs.LG].

[25] A. Paszke, S. Gross, F. Massa, *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Adv. Neur. In.*, vol. 32, 2019.