POLITECNICO DI BARI

DEPARTMENT OF ELECTRICAL AND INFORMATION ENGINEERING
Master in Artificial Intelligence and Data Science

Report of Deep Learning

Cybersecurity Threat Mitigation: Advanced Behavioral Anomaly Detection in User Authentication Logs Using Deep Learning

Students :
Vito Guida, Andrea Natile Martino

Prof :
Vito Walter Anelli

Accademic year 2023-2024

1 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.
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.
In this project we use **LSTM**, **BiLSTM** and **Trasformer** for the task of detecting anomaly logs using the LANL dataset. All model were implemented and trained using PyTorch, leveraging GPU acceleration. Specifically, computations were performed on a dedicated NVIDIA GPU with 12 GB of VRAM, supported by a system with 32 GB of RAM, allowing for efficient training even on large-scale log datasets.

2 Methodologies

2.1 Data Analyses

We used the **LANL** dataset as our initial data source. LANL consists of over one billion log lines collected over 58 consecutive days. The logs include anonymized information on processes, network flows, DNS queries, and authentication events. For our project, we focused exclusively on the authentication (auth) and red team logs. The auth dataset contains authentication logs over the entire 58-day period, while the red team dataset provides logs of malicious activities carried out by simulated adversaries. Each line in the auth log contains the following fields:
"time, source user@domain, destination user@domain, source computer, destination computer, authentication type, logon type, authentication orientation, success/failure".
Given the large volume of the auth dataset, we narrowed our analysis by cross-referencing with the red team data to identify days with the highest number of attacks (based on timestamps). Following this analysis, we selected **day 7** and **day 8** for our study. Day 7 contains no attacks, while day 8 includes 260 out of the 749 total attacks, making it the day with the highest number of recorded attacks.

2.2 Preprocessing

After selecting the relevant days, we applied further preprocessing:

- We removed intermediary authentication logs (i.e., internal system access not directly triggered by a human via the *source@domain* field).
- We split the *user@domain* fields into two separate columns: user and domain.
- Finally, we labeled each row from days 7 and 8 with a binary label: 0 for benign *(non-anomalous)* events, 1 for anomalous events, based on a match with the red team log data.

We emphasize the fully unsupervised nature of our models. Red team labels are utilized exclusively for evaluating performance and facilitating the generation of a balanced test set, in fact during the training of the model, the redteam column is not used.

2.3 Tokenization

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. We employ a feature-level encoding scheme. This means that we tokenize each distinct categorical value within a column and map it to a unique integer index. For instance, if the same user appears in both the "Source User" and "Destination User" fields, they are treated as two distinct tokens and assigned different indices. This approach ensures that the contextual role of a value (e.g., source vs. destination) is preserved in the tokenization process.

2.4 Model architecture

2.4.1 LSTM

One of the models implemented in our study is a custom LSTM-based neural network designed for anomaly detection in sequences of categorical authentication events. The model, named LSTMAnomalyModel, is implemented using PyTorch.
The model processes categorical fields extracted from the authentication logs. Each categorical field is encoded using a dedicated embedding layer. The embeddings are then concatenated along the feature dimension to form the input for the LSTM. For each categorical variable (*e.g., user, domain, computer*), a separate nn.Embedding layer is created.
The concatenated embeddings are passed through a Long Short-Term Memory (LSTM) network, which captures temporal dependencies in the event sequences. The LSTM can operate in two configurations, with dropout or without.
The final hidden state of the LSTM is used as input to a linear output layer, one for each categorical field to be predicted.

2.4.2 BiLSTM

The second model employed in our study is a Bidirectional LSTM (BiLSTM) enhanced with an attention mechanism. As the LSTM model, the embeddings are concatenated to create the input for the model. Unlike a standard LSTM, this model processes the input sequence in both forward and backward directions. This allows the model to capture dependencies from both past and future events. As the previous model we introduce dropout mechanism. On top of the BiLSTM, the model integrates an attention layer. For each categorical variable to be predicted is used a linear output layer.

2.4.3 Transformers

The third model implemented in our project leverages the Transformer architecture. This architecture is well-suited for capturing long-range dependencies and

parallelizing sequence modeling, which is beneficial when working with authentication log data.
We apply and embedding layer as the previous model and, to stabilize and accelerate training, a batch normalization layer is applied to the concatenated embeddings. This normalization is done over the feature dimension and helps standardize the input before feeding it into the transformer.
The core of the model is a Transformer Encoder composed of 2 layers, each consisting of 4 heads, that enables the model to jointly attend to information from different positions in the sequence.
To speed up the convergence we adopted a dropout mechanism also to the trasformer.

2.5 The role of the sequence length

In the context of anomaly detection on log data, the sequence length represents the number of past events that the model observes in order to predict the next event. This parameter is particularly critical in sequential models such as LSTM, BiLSTM, and Transformer, as it determines the temporal depth of the context available during learning. In this project, the sequence length was implemented using a sliding window approach: for each event to be predicted, the model considers the *sequence length* preceding events. For instance, given a *sequence length* value of 10, the model receives as input the ten preceding rows from the log (converted into numerical indices) and attempts to predict the subsequent event. This logic is implemented in the `is_event_anomalous` function, where the input sequence is constructed before computing the anomaly score.

The sliding window used in this implementation has a fixed width equal to the sequence length and advances by a step of sequence length + 1. This approach ensures that the sequences used for evaluation do not overlap, and that each sequence corresponds to a unique prediction target, simplifying the computation and interpretation of anomaly scores.

2.6 Training and evaluation

For the training we splitted the day 7 in 70% for the actual training of each model and the 30% for finding the best threshold (*validation set*). We have utilized Adam as optimizer, for all the models we used an embedding dimension of 32, an hidden dimension of 64 and 0.3 as dropout parameters. For the final evaluation we used all the redteam events of day 8 (i.e. 260 events) and the same amount of events taken from the validation dataset to maintain a balance between redteam and regular entries.

To assess the performance of the proposed models (LSTM, BiLSTM, Transformer), several training runs were carried out under varying experimental conditions. The models were trained on different dataset sizes, varying the number of log lines to simulate different volumes of system activity and to evaluate scalability. Additionally, experiments were performed with different numbers of training epochs , in order to observe the convergence behavior and risk of overfitting over time.

During experimentation, we also explored the impact of dropout regularization on model training. When dropout was enabled (*DROP=True*), we observed that the training loss remained higher compared to the configuration with dropout disabled. This is expected behavior, as dropout randomly deactivates neurons during training, which limits the model's capacity and prevents overfitting. Conversely, when dropout was set to *False*, the models achieved a lower training loss, indicating a greater fit on the training data, despite this, we obtained valid results in both cases that will be shown in the chapter of the results.

During evaluation, the trained models were tested on a held-out portion of the dataset using an anomaly detection strategy that is explained in the next section 2.6.1

2.6.1 Anomaly Detection

Due to the unsupervised 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.

We employ two methods to determine the anomaly score threshold. The first approach involves analyzing the distribution of scores from non-anomalous events within the validation set (30% of day 7). We identify the average and maximum scores in this distribution and then iterate through values within this range to pinpoint the optimal threshold.

Alternatively, we approximate the validation set score distribution(2.1) as a normal distribution. By calculating its mean and standard deviation, we can set the threshold as the upper limit of the 68% confidence interval (i.e., one standard deviation above the mean).

But we found that the first approach gives better result, so we adopted it for setting the threshold because we get better results.

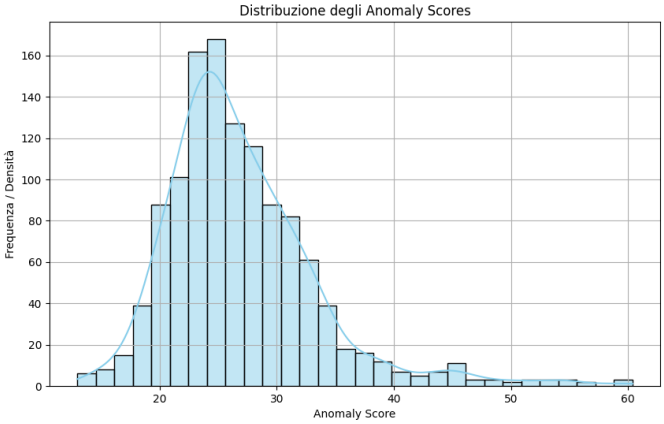


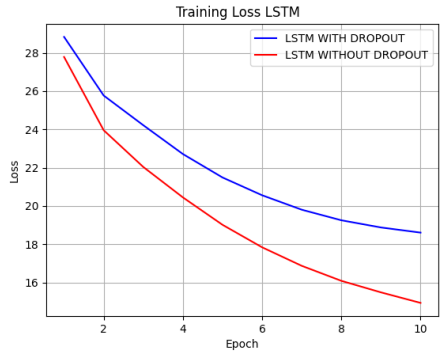
Figure 2.1: Distribution of scores of validation set

3 Result and discussion

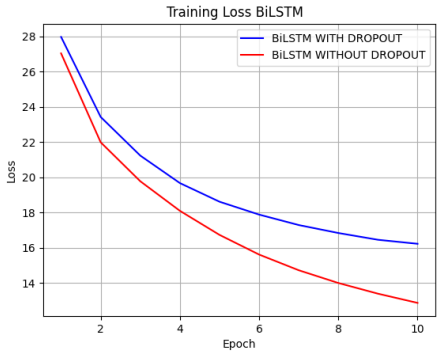
During the training and evaluation phase, we tested three models (LSTM, BiLSTM, and Transformer) under various configurations, changing the number of epochs (from 1 to 10), enabling or disabling dropout, and varying the amount of data used for training and testing coming from the day 7 of the dataset, the results are showed in Figure 3.2 and 3.3. For performance evaluation, we adopted an anomaly detection strategy based on the model's ability to predict the next event given the previous `seq-len` events. For each experiment, we built a confusion matrix and used the **F1-score** as the metric to compare between the models, an example is the Table 3.1.

	Predicted	
Actual	**Positive**	**Negative**
Positive	TP : 255	FN : 5
Negative	FP : 1	TN : 259

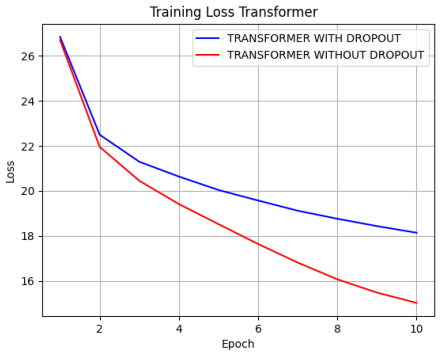
Table 3.1: Example of confusion matrix of a BiLSTM with 5 epochs and Dropout



(a) LSTM



(b) BiLSTM + Attention



(c) Transformer

Figure 3.1: Training loss difference with and without Dropout for each model.

The results were highly satisfactory, with F1-scores consistently above **0.95** across most configurations. Differences in performance between models were minimal, as were variations in training loss. We observed that enabling dropout slightly increased the training loss (Figure 3.1), suggesting a regularization effect. However, in terms of F1-score, the impact of dropout was negligible.

Model	Dropout	Epochs	TrainLoss	TP	FP	TN	FN	Accuracy	Precision	Recall	f1-score
LSTM	True	1	28,84	260	0	260	0	1	1	1	1
LSTM	True	3	23,79	260	1	259	0	0,9981	0,9962	1	0,9981
LSTM	True	5	21,59	260	1	259	0	0,9981	0,9962	1	0,9981
LSTM	True	8	19,31	257	4	256	3	0,9865	0,9847	0,9885	0,9866
LSTM	True	10	18,47	250	14	246	10	0,9538	0,947	0,9615	0,9542
LSTM	False	1	27,86	259	1	259	1	0,9962	0,9962	0,9962	0,9962
LSTM	False	3	21,89	256	10	250	4	0,9731	0,9624	0,9846	0,9734
LSTM	False	5	18,98	255	7	253	5	0,9769	0,9733	0,9808	0,977
LSTM	False	8	16,18	247	14	246	13	0,9481	0,9464	0,95	0,9482
LSTM	False	10	15,03	257	34	226	3	0,9288	0,8832	0,9885	0,9328
BiLSTM	True	1	27,88	260	1	259	0	0,9981	0,9962	1	0,9981
BiLSTM	True	3	22,20	260	0	260	0	1	1	1	1
BiLSTM	True	5	18,84	255	1	259	5	0,9885	0,9961	0,9808	0,9884
BiLSTM	True	8	16,94	259	6	254	1	0,9865	0,9774	0,9962	0,9867
BiLSTM	True	10	16,20	260	2	258	0	0,9962	0,9924	1	0,9962
BiLSTM	False	1	27,35	260	3	257	0	0,9942	0,9886	1	0,9943
BiLSTM	False	3	20,25	256	8	252	4	0,9769	0,9697	0,9846	0,9771
BiLSTM	False	5	16,75	257	5	255	3	0,9846	0,9809	0,9885	0,9847
BiLSTM	False	8	14,18	258	37	223	2	0,925	0,8746	0,9923	0,9297
BiLSTM	False	10	13,76	250	9	251	10	0,9635	0,9653	0,9615	0,9634
Transformer	True	1	26,99	260	0	260	0	1	1	1	1
Transformer	True	3	20,25	260	0	260	0	1	1	1	1
Transformer	True	5	16,80	260	0	260	0	1	1	1	1
Transformer	True	8	15,09	260	3	257	0	0,9942	0,9886	1	0,9943
Transformer	True	10	13,99	260	1	259	0	0,9981	0,9962	1	0,9981
Transformer	False	1	26,18	260	0	260	0	1	1	1	1
Transformer	False	3	17,49	260	0	260	0	1	1	1	1
Transformer	False	5	15,53	260	0	260	0	1	1	1	1
Transformer	False	8	12,37	260	0	260	0	1	1	1	1
Transformer	False	10	11,91	260	2	258	0	0,9962	0,9924	1	0,9962

Figure 3.2: Result with 100000 rows for the training

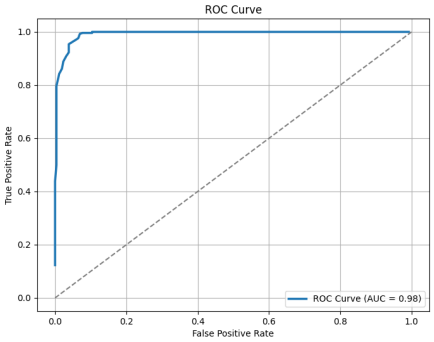
Another relevant finding concerns the number of training epochs: increasing it excessively (beyond 6–7 epochs) sometimes led to a slight decrease in F1-score, indicating overfitting, especially when training was performed on a dataset of 100,000 log lines. To validate this hypothesis, we repeated the experiments using a larger dataset of 2 million log lines and in this case, as shown in Figure 3.3, the F1-scores stabilized close to 1, demonstrating the models' improved generalization capabilities

when trained on a significantly larger volume of data.

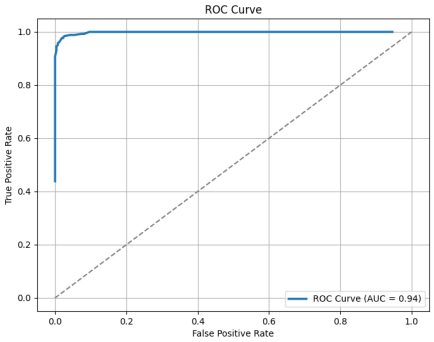
Model	Dropout	Epochs	TrainLoss	TP	FP	TN	FN	Accuracy	Precision	Recall	f1-score
LSTM	False	10	15,7536	260	1	258	0	0,9981	0,9962	1	0,9981
BiLSTM	False	10	14,786	260	2	257	0	0,9961	0,9924	1	0,9962

Figure 3.3: Result with 2 million rows for the training

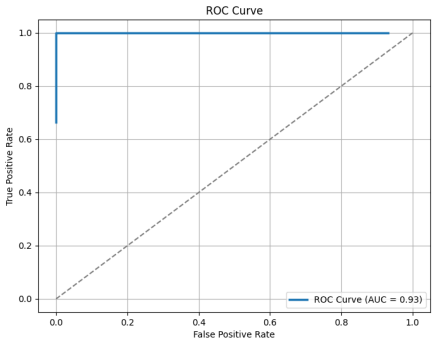
The three figures below in Figure 3.4 illustrate the ROC (*Receiver Operating Characteristic*) curves obtained for the LSTM, BiLSTM, and Transformer models trained with 10 epochs, Dropout enabled , and 100000 rows. In all cases, the curves show excellent performance, with areas under the curve (AUC) close to 1, confirming the models' strong ability to distinguish between normal and anomalous log events.



(a) LSTM



(b) BiLSTM + Attention



(c) Transformer

Figure 3.4: AUC roc curve difference between each model.

4 Future Works And Conclusions

Overall, the results obtained from the anomaly detection experiments were highly satisfactory, with F1 scores consistently exceeding 0.95 across different configurations and model architectures. This confirms the effectiveness of the chosen sequential approaches (LSTM, BiLSTM, Transformer) in modeling log data for anomaly detection tasks. During the project, several challenges were encountered—particularly related to the large scale of the datasets and the need to manage them efficiently, both in terms of data preprocessing and computational resources. Processing millions of log lines required careful batching, memory management, and leveraging GPU acceleration to maintain reasonable training times.

For future developments, it would be valuable to explore more flexible modeling strategies, such as using adaptive sequence lengths, which could allow the model to dynamically adjust its temporal context based on the nature of the data.