Detecting Network Anomalies Using Machine Learning

University of New Haven

Matthew Grubelic, Chris Saliby, Anthony Saldana, Luke Turbert, Andrew Rittenhouse
Advisors: Vahid Behzadan, Ph.D; Liberty Page

SAIL

Abstract

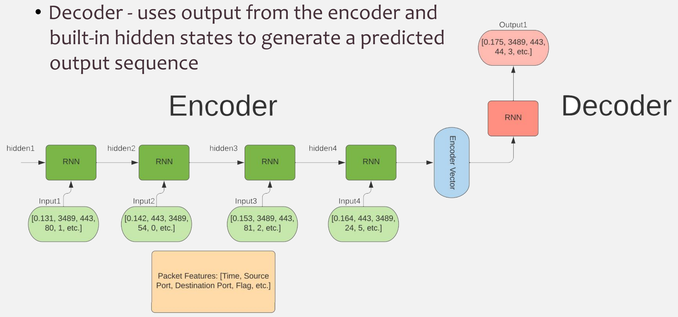
The risk of crippling cyber attacks on computer systems is increasing rapidly each day. Current software and techniques used to defend against these malicious attacks are showing their limitations and are being completely overwhelmed in some cases. As a result, a more modern and forward-thinking solution is becoming increasingly necessary. The team is implementing one such software solution using a sequence to sequence neural network in Python3 with the PyTorch library to observe malicious events and predict when the next attack might happen. In this project, a deep learning sequence-to-sequence model, modeled after the behavior of predictive typing technologies, was implemented on the DARPA 1999 dataset to detect malicious traffic and determine the probability of another attack in the future. The model functions by observing sequences of network packets, using them to predict upcoming sequences of packets, and comparing the actual observed data to the prediction. Since the amount of notable research and experimentation done in this area so far is lacking, the results yielded by this network traffic anomaly detection approach further demonstrate that the use of a sequence-to-sequence model is a viable, though still emerging, solution for modern intrusion detection systems.

Overview

The sequence to sequence model is a deep learning model which maps a sequence of inputs to a sequence of outputs. It consists of two recurrent neural networks (RNNs) - an encoder and a decoder. Introduced in 2014, seq2seq models are widely used in applications like language translation, even being used by Google Translate, but their security applications have yet to be fully explored.

How It Works:

- Encoder - converts input sequence into hidden vector
- Decoder - uses output from the encoder and built-in hidden states to generate a predicted output sequence



Anomaly Detection Plan:

- Train the model to predict sequences with no attacks accurately
- After training, sequences it predicts poorly could be attacks
- Define a "threshold" for what is considered poorly predicted

Datasets:

- Used "transfer" protocols: TCP, SMTP, HTTP, TELNET, SSHv1, IRC, POP
- **Test set**: From week 5 of DARPA 1999, contains 64,902 sequences and 841,083 packets with various attacks
- **Training set**: From week 1 of DARPA 1999, contains 81,896 sequences and 826,566 packets with no attacks

Technologies

Methodology & Workflow

Training function

Step 1: We made a script to parse the DARPA 1999 Dataset into sequences of 4 to 320 packets. These sequences were saved into a ".csv" file and then inputted into the training script. We encoded each numeric column into z-score and each categorical one into a one-hot encoded vector so they can be used by the model as a feature.

↓

Step 2: We parsed the sequences of packets into pairs of "input" and "target". The "input" is the first 3 packets of the sequence and the "target" is the rest of the packets in the sequence. The "input" sequence will be used by the model to predict the "target" sequence.

↓

Step 3: The input sequence is then fed into the Encoder, so that each input can be condensed to a single vector representation with the use of a GRU. The encoder outputs this vector and a "hidden state" vector that is then inputted into the Decoder. The Decoder produces a prediction of the target packet sequence.

↓

Step 4: The predicted output is then compared to the actual target packet sequence and the loss between the two values is calculated. The model then learns from this value and uses it to change its parameters to make the loss value lower, increasing accurracy of predictions.

↓

Steps 3 and 4 happen continuously until all iterations of the sequences end. We then save the parameter values from the last iteration into a file for use in the test function.

Testing function

Steps 1 to 3 from the training function are repeated in the testing function. However, the saved parameter values from the training function are used as the model parameters for the testing function.

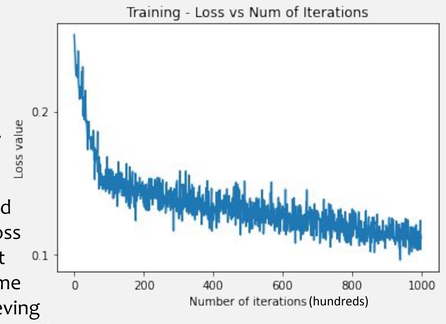
↓

Step 1: The predicted output is then compared to the actual target packet and the loss between both values is calculated. The model then determines if the sequence of packets is an anomaly or not based on a specified threshold value for the loss.

↓

Step 2: The total number of anomalies detected is then outputted to the screen, along with the overall RMSE score for the normal and attack sequences.

Results & Conclusions



The figure on the right shows the results of training the model on 100,000 random samples from the training dataset. It is a plot of the training loss vs the number of iterations in hundreds, and shows that the average loss starts relatively high but it decreases sharply over time as the model learns, achieving a final value of 0.1114. Copies of its optimized parameters were saved at the milestones of 40%, 80%, and 100% training progress so that its performance could be evaluated at different stages of training. Since only ~80,000 sequences were present in our training set, the model that had been trained for 100,000 iterations suffered from overfitting, meaning that it was fitted too closely to the training data and became less generalizable for other data. Out of the other two parameters, better performance was observed using the 80% trained model. A simple screenshot of output from a test using this model on our testing dataset (~60,000 sequences including attacks) is shown below. The model was able to successfully flag 2,080 sequences as potentially anomalous using an error threshold of 0.5 to determine normal vs anomalous sequences. Normal sequences were predicted with an average Root Mean Squared Error (RMSE) of ~0.33, while potential "attacks" were predicted with an RMSE of ~0.53, and the overall average RMSE of all predictions was ~0.34.

```
Potentially anomalous sequences:  2080
Overall RMSE score:  0.34039334843147934
Normal RMSE score:  0.33410887812161344
Attack RMSE score:  0.5301994586892161
---> Finished Testing Function <---
```

References

- https://ir.lib.uwo.ca/cgi/viewcontent.cgi?article=7635&context=etd
- https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html
- https://github.com/bentrevett/pytorch-seq2seq
- https://github.com/jeffheaton/t81_558_deep_learning/blob/master/t81_558_class_14_03_anomaly.ipynb

Acknowledgements

Academic Advisor: Vahid Behzadan, Ph.D
Faculty Advisor: Liberty Page
Sponsor: Secured and Assured Intelligent Learning (SAIL) Lab