TIRESIAS: **Predicting Security Events Through Deep Learning**

Group Name: Aw, Snap!
Group Members: Weijia Sun, Xinyu Lyu, Mengmei Ye

Problem:

Nowadays, the attack techniques to computer systems have reached an unprecedented sophistication. Therefore, only detecting malicious activity when it happens is not enough for defenders. There is a need to predict the specific steps which will be taken by adversary when performing an attack. The authors think of the Tiresias, a security system, which can predict the security event sequence based on the previous observations.

Motivation:

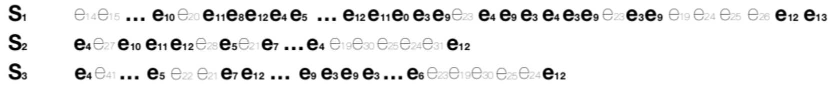
Firstly, previous research focused on predicting whether malicious events will happen or not, based on the binary outcome. For instance, they may predict whether a data breach would happen and whether hosts will get infected with the malware in the future. However, they cannot provide any insight information on the following techniques and modus operandi which will be made by attackers. Secondly, it is hard to collect enough labeled data to train the module. Thirdly, the attackers may change their modus operandi from time to time. Therefore, there is a need to retrain the models against the attackers.

Challenges:

Firstly, it is hard to find a visible pattern that a particular event will follow or precede another event among the events sequence. Therefore, it is difficult to predict each step of the multi-step attack. Secondly, the security events sequence will always get mixed with irrelated events (noisy events), which will prevent the system from making the correct forecast. Thirdly, there will be multi-attacks from different adversary groups happening at the same time. So, it is hard to recognize multi-attacks.

Problem formulation:

A security event $e_j \in E$ is a timestamped observation recorded at timestamp j, where E denotes the set of unique events. Each $s_i = \{e_1, e_2, \dots, e_l\}$ is a security event sequence observed in an endpoint ordered by observation. We define the target event towards prediction as e_{tgt}. Each target e_{tgt} is associated with a number of already observed security events. The problem is to learn a sequence prediction model which takes a variable-length security events sequence $\{e_1, e_2, \dots, e_l\}$ as input and predicts the e_{tgt}.


S1 e14 e15 ... e10 e20 e11 e8 e12 e4 e5 ... e12 e11 e0 e3 e9 e23 e4 e9 e3 e4 e3 e9 e23 e3 e9 e19 e24 e25 e26 e12 e13
S2 e4 e27 e10 e11 e12 e28 e5 e21 e7 ... e4 e19 e30 e25 e24 e31 e12
S3 e4 e41 ... e5 e22 e21 e7 e12 ... e9 e3 e9 e3 ... e6 e23 e19 e30 e25 e24 e12

Solution:

They design a system called Tiresias leveraging RNN to predict future events based on previous observations. Moreover, if there is a sharp drop on the prediction precision towards the new data, the module will get retrained automatically.

1. Module selection

In order to make the model more adaptable to new data and keep the training session more efficiently, they use Rocki-LSTM which behaves better on noisy sequential input data with a stochastic memory array.

The architecture follows a stochastic design which takes the initial output from output gate into softmax activation function. Then, among the probability score distribution of softmax, it finds the most likely memory cell and updates the hidden layer.

2. Data collection and preprocessing

TIRESIAS reconstruct the data collected from millions of machines from Symantec's intrusion prevention product into $\mathrm{D} = \{s_1, s_2, \dots, s_m\}$, where m denotes the number of machines. And each $s_i = \{e_1, e_2, \dots, e_l\}$ denotes a sequence of security events ordered by timestamps. Then, they split the data into D_T (training data) and D_V (validation data), where $D_T \cap D_V = \emptyset$. From each event, they extract anonymized machine ID, timestamp, security event ID, event description, system actions and other information as features.

3. Model training and validation

With the affine transformation to the hidden layer of LSTM, TIRESIAS specifies the probability distribution of e_{w+1} given historical observed events $\{e_1, \ldots, e_w\}$ with the softmax as the activation function:

$$\Pr(e_{w+1}|e_{1:w}) = \frac{e^{(h^w \cdot p^j + q^j)}}{\sum_{j' \in E} e^{(h^w \cdot p^{j'} + q^{j'})}}$$

Then, it makes the prediction based on the highest probability score among the probability score distribution of softmax:

$$\Pr[e_{i+1}|e_{0:i}] = \{e_1: p_1, e_1: p_1, \ldots, e_{|E|}: p_{|E|}\}$$

Moreover, they choose log-likelihood as the loss function of the training process:

$$\mathcal{L} = \sum_{t=1}^{|D_T|} \Pr\,(e_t|e_{1:t-1}: \theta)$$

Finally, they use the D_V to determine when the training process should end and select the best model.

4. Module evaluation

They made model performance evaluation based on some metrics such as Precision, Recall, F1 score. Moreover, if the precision drops quickly, they will retrain the module.

Experiments and analysis:

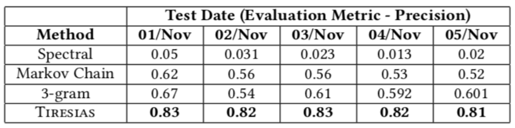
1. Experiments

D1 data contains 27 days of data composed of over 2.2 billion security events. Moreover, the first seven days of D1 from Nov 1st- Nov 7th is used to train the model. The rest ten days of D1 from Nov 8th- Nov 17th is used to evaluate the module performance. Finally, the first five days of D1 from Nov 1st- Nov 5th is used to choose the best model and compared to the baseline models.

D2 data contains the 8th and 23rd day of each month between Nov 2017-Feb 2018 composed of 1.2 billion security events to evaluate if the module retains stability after a long time.

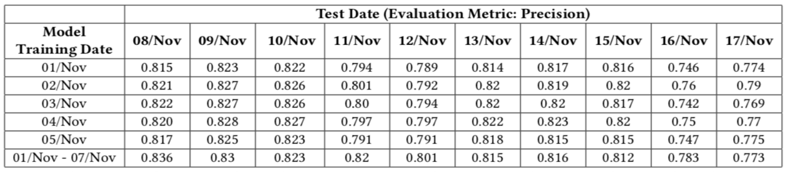
2. Analysis

Precision over 80% precision. Moreover, precision and recall are well balanced with a similar scale (from 0.87 to 0.795). Firstly, Tiresias can easily beat the performance of baselines such as Spectral, Markov Chain and 3-gram.



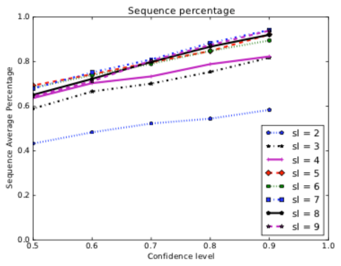
	Test Date (Evaluation Metric - Precision)				
Method	**01/Nov**	**02/Nov**	**03/Nov**	**04/Nov**	**05/Nov**
Spectral	0.05	0.031	0.023	0.013	0.02
Markov Chain	0.62	0.56	0.56	0.53	0.52
3-gram	0.67	0.54	0.61	0.592	0.601
TIRESIAS	**0.83**	**0.82**	**0.83**	**0.82**	**0.81**

Secondly, Tiresias is stable over time because the model quickly converges towards high accuracy with only one or a few days of training data and the model ages very well even months after it was built.



	Test Date (Evaluation Metric: Precision)									
Model Training Date	**08/Nov**	**09/Nov**	**10/Nov**	**11/Nov**	**12/Nov**	**13/Nov**	**14/Nov**	**15/Nov**	**16/Nov**	**17/Nov**
01/Nov	0.815	0.823	0.822	0.794	0.789	0.814	0.817	0.816	0.746	0.774
02/Nov	0.821	0.827	0.826	0.801	0.792	0.82	0.819	0.82	0.76	0.79
03/Nov	0.822	0.827	0.826	0.80	0.794	0.82	0.82	0.817	0.742	0.769
04/Nov	0.820	0.828	0.827	0.797	0.797	0.822	0.823	0.82	0.75	0.77
05/Nov	0.817	0.825	0.823	0.791	0.791	0.818	0.815	0.815	0.747	0.775
01/Nov - 07/Nov	0.836	0.83	0.823	0.82	0.801	0.815	0.816	0.812	0.783	0.773

Thirdly, long-term data training seems slightly more robust to anomalies on a specific day of data. 0.3% higher on precision.



Pros and Cons:

1. Pros

Firstly, the prediction precision is about 0.819 with one-week training data, which is 0.3% higher than one-day training. The long-term data training seems slightly more robust to anomalies on a specific day of data than the short-term data training. Moreover, the standard deviation for long-term training is about 0.02 over ten days of prediction. Secondly, TIRESIAS performs the efficiency and stability prediction of the security event over a long time. Thirdly, TIRESIAS behaves robustness towards noise while detecting multistage attacks. Fourthly, the system may need to retrain only if the data present radical changes, while its precision does not decrease.

2. Cons

The limitations of Tiresias: Firstly, the training data is unbalanced because there are some rare events data. Such unbalanced training data will make the training incomplete, and the model will be unable to forecast such events correctly. Secondly, the retrain of LSTM module takes a long time for about 10 hours, when new security events come out.

The limitations of Data: It can only predict the pre-labeled security event. Since a new security event will get signed after some time, the Tiresias cannot timely predict such latest events until they get labeled.

Think differently:

Firstly, we will use the attention mechanism to replace the LSTM, which is recently very popular for sequential prediction. As shown in results, the training model with one day or a week data behaves similarly in the prediction precision. That is because of the primary disadvantage of the LSTM that it makes prediction mainly based on the last previous input. In other words, it will forget the memory from farther cells due to the gradient disappear problem from "tanh" activation function. Therefore, it is better to use attention which adds global mapping features onto the temporal association among all the inputs and outputs. Attention can furtherly extract the essential association feature from subsequent security events. Finally, the model can make the prediction based on the enhanced feature from attention and LSTM.

Secondly, we think they should add original LSTM into baselines for the comparison of precision and training time. Since LSTM itself has a kind of ability to recognize and ignore the noise input, we should make the comparison between Rocki-LSTM and the original LSTM to prove that Rocki-LSTM behaves better on noise data. Also, it is meaningless to compare the training time between baselines with CPU and TIRESIAS with GPU. They should compare the training time between Rocki-LSTM and the original LSTM to prove the TIRESIAS is training efficiently.

Thirdly, for Long/short term data training, they only tell us that the long-term data training seems slightly more robust to anomalies with 0.3% higher on precision. However, they only show the static analysis to precision between Long/short term data. However, they do not show much static analysis on standard deviation between Long/short term data. Therefore, they had better draw a table of deviation contrast between Long/short term data training to convince us that long-term data training is more robust to anomalies.