TIRESIAS: Predicting Security Events Through Deep Learning

Group Name: Aw, Snap!

Group Members: Weijia Sun, Xinyu Lyu, Mengmei Ye

contents

1. **Problem**

2. **Motivation**

3. **Approach**

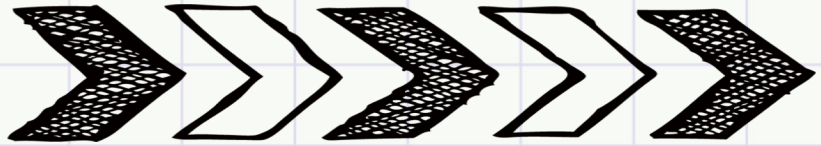
4. **Experiments results**

5. **Related Work**

6. **Future Work**

Part 1

Problems

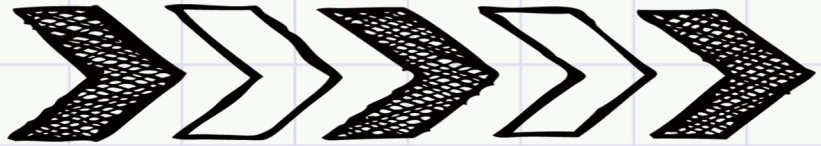


Problems

- Attack techniques-> unprecedented sophistication.
- Detecting malicious activity when happens is not enough.
- Need to predict following specific steps taken by adversary.

Tiresias : A security system, which can predict the security event sequence based on the previous observations and Deep Learning Models.

Challenges & Approach



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 $s_i = \{e_1, e_2, \dots, e_j\}$ is a security event sequence observed in an endpoint ordered by observation.
- We define the target event towards prediction as e_{tgt} which 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}.

S_1 $e_{14}e_{15}$ … $e_{10}e_{20}e_{11}e_8e_{12}e_4e_5$ … $e_{12}e_{11}e_0e_3e_9e_{23}e_4e_9e_3e_4e_3e_9e_{23}e_3e_9e_{19}e_{24}e_{25}e_{26}e_{12}e_{13}$

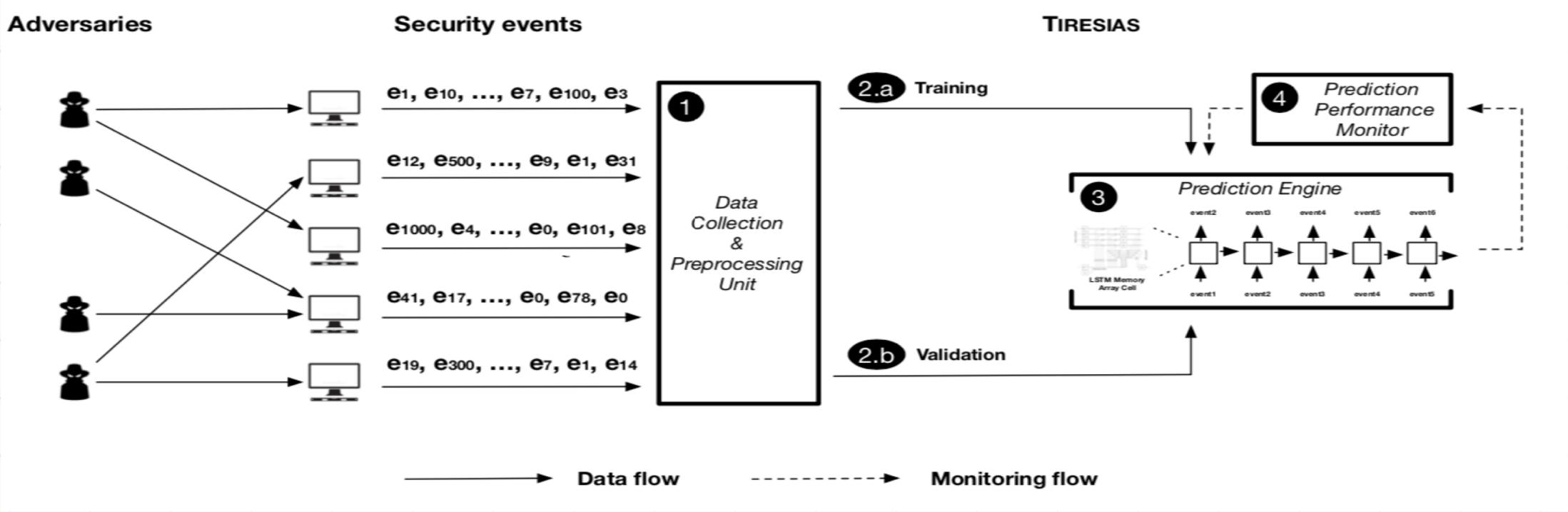
S_2 $e_4e_{27}e_{10}e_{11}e_{12}e_{28}e_5e_{21}e_7$ … $e_4e_{19}e_{30}e_{25}e_{24}e_{31}e_{12}$

S_3 e_4e_{41} … $e_5e_{22}e_{21}e_7e_{12}$ … $e_9e_3e_9e_3$ … $e_6e_{23}e_{19}e_{30}e_{25}e_{24}e_{12}$

Approach

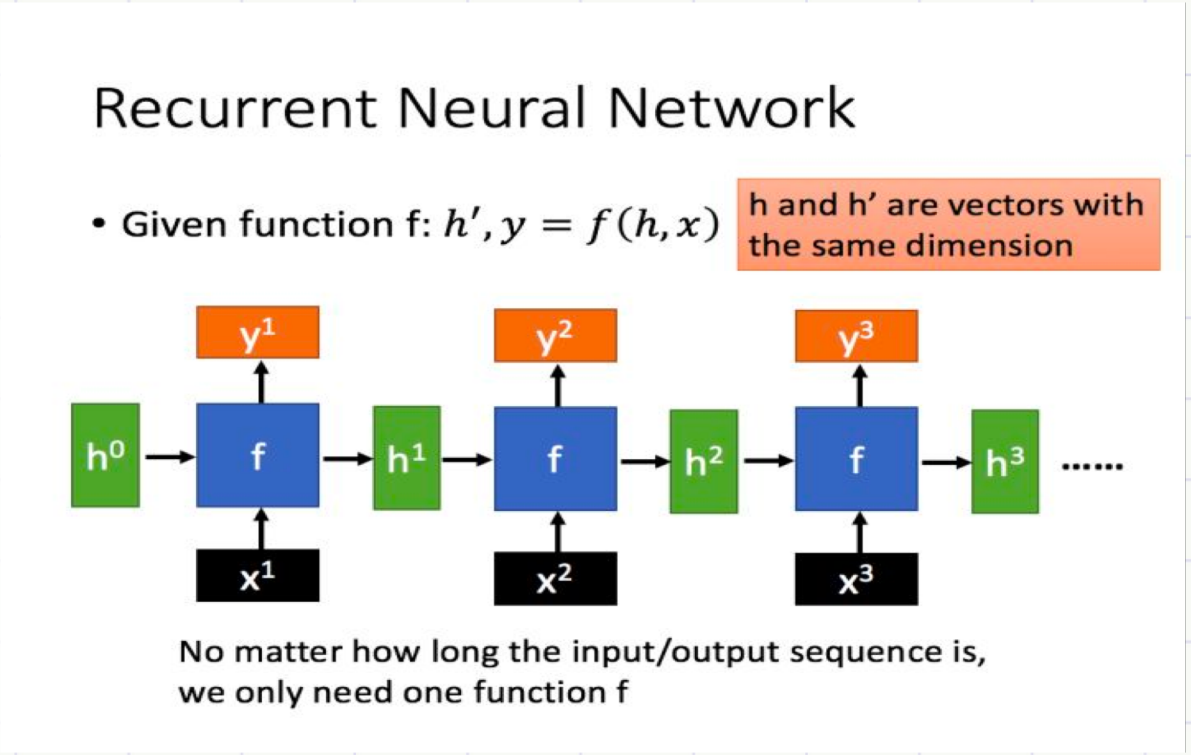
Tiresias

- Data collection: Collecting previous security events sequence.
- Training & Validation : Training Rocki-LSTM to predict future events based on previous observations. Validation to choose best model.
- Prediction Engine: LSTM predicts security event sequence based on previous observations.
- Performance Monitor: Monitor a sharp drop on precision, retrain the module will get automatically.



Approach-Module Selection(RNN)

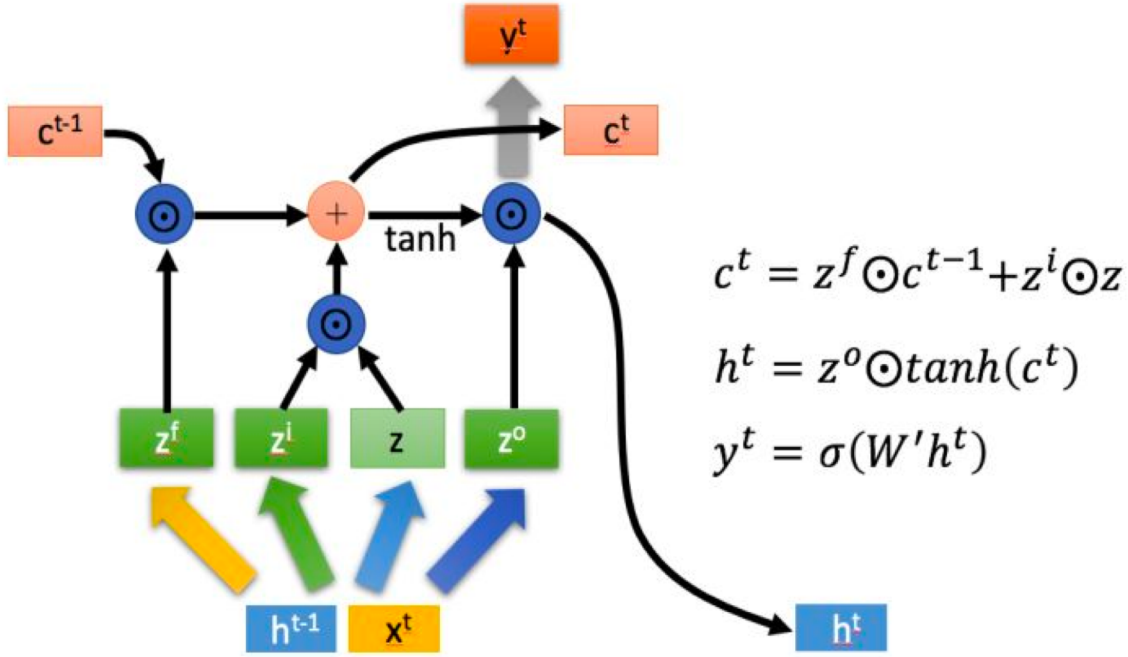
- Input: Take s $= \{x_1, x_2, \ldots, x_l\}$ as sequential input.
- Train: Recurrently update the hidden layer h_t based on each current input x_t and on the previous hidden layer h_{t-1}.
- Prediction: Based on the previous hidden layer h_{t-1} and current input x_t to predict current output y_t.
- Disadvantage: Shot-term memory reliable.



Approach-Module Selection(LSTM)

- Forget : Use forget gate $Z^f(0,1)$ to selectively forget the unimportant previous memory c^{t-1}, from previous cell state.
- Remember: Use input gate $Z^i(0,1)$ to selectively remember the important input Z.
- Output: Use output gate $Z^o(0,1)$ to determine which outputs will be treated as the current states output.
- Transfer: Transfer c^t and h^t to next state.
- Advantage: Process a longer input sequence than RNN without short-term memory reliable problem.

LSTM(Long short-term memory)

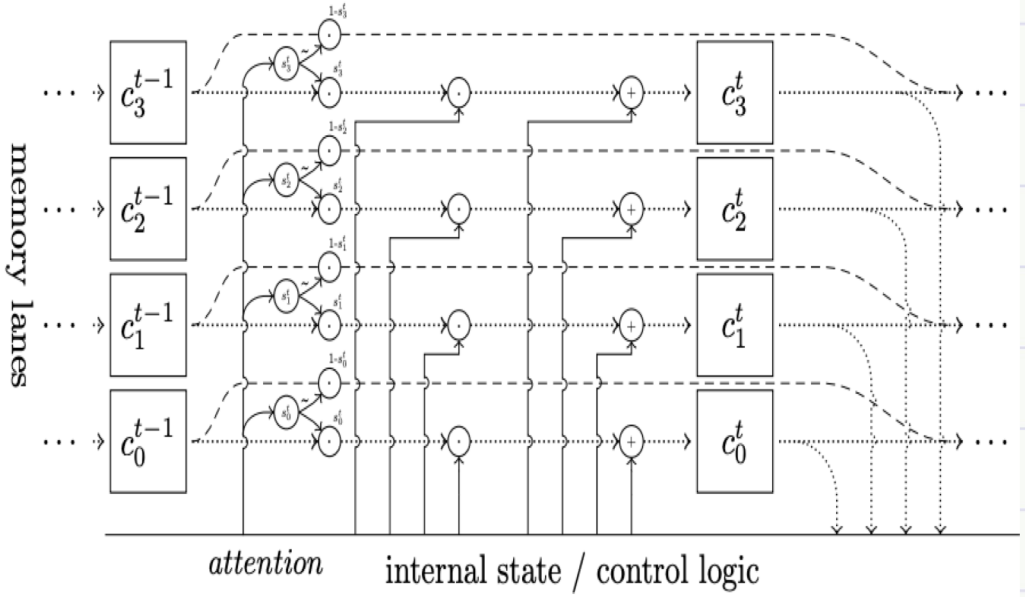


$$c^t = z^f \odot c^{t-1} + z^i \odot z$$

$$h^t = z^o \odot tanh(c^t)$$

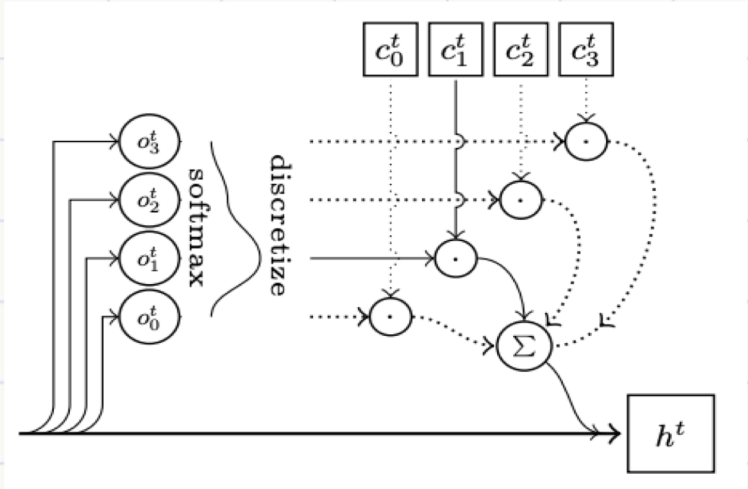
$$y^t = \sigma(W' h^t)$$

Approach-Module Selection(Rocki-LSTM)



$$\tilde{c}_k^t = tanh(W_{ck}x^t + U_{ck}h^{t-1} + b_{ck})$$
$$c_k^t = f_k^t \odot c_k^{t-1} + i_k^t \odot \tilde{c}_k^t$$
$$h^t = \sum_k o_k^t \odot tanh(c_k^t)$$

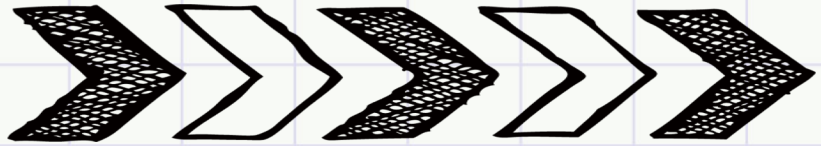
$$p(i=k) = \frac{e^{o_k^t}}{\sum_k e^{o_k^t}} \qquad h^t = o_i^t \odot tanh(c_i^t)$$



Rocki-LSTM

- A single layer RNN:
  Training session computational efficiently.
- More complex memory structures (array-like structure):
  k controls the number of cell memory vectors.
- Stochastic design:
  More resilient on noisy sequential input data.

Experiments & results

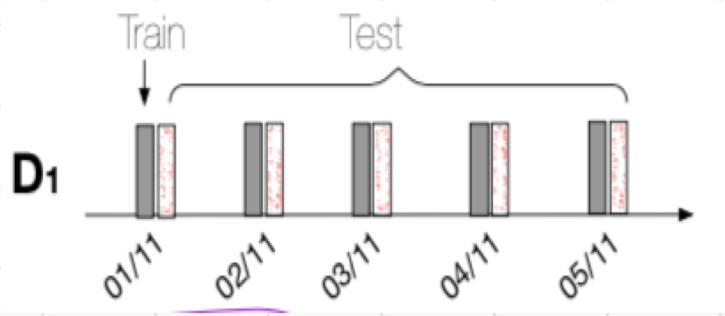


Experiments & results

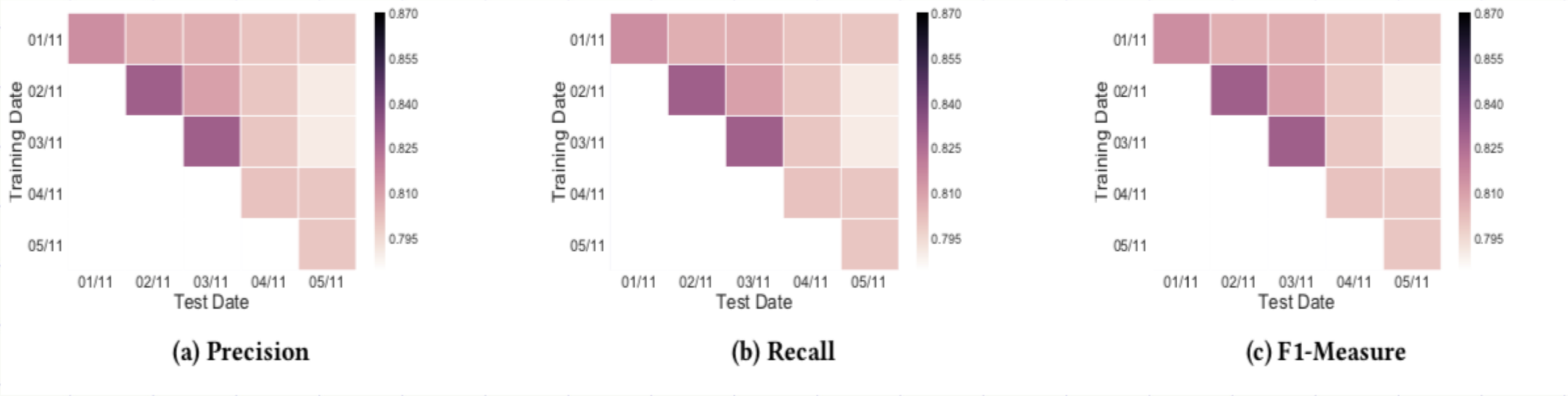
Overall Experiments and results:

- Experiment:

  From D1, they train model using on day of data and evaluate the model using the same day and the followings days until 5 Nov 2017.



- Results:
  - Precision over 80% precision.
  - Precision and Recall are well balanced with a similar scale (from 0.87 to 0.795).
  - Sudden decrease 6.9% retrained.



(a) Precision (b) Recall (c) F1-Measure

Experiments & results

Comparison study:

- Experiment:

  The comparison study uses daily data(1 Nov – 5 Nov) from D1.

  Training, validation and test data are all from the same day of different machines.

- Results:

  Tiresias can easily beat the performance of baselines such as Spectral, Markov Chain and 3-gram(has values higher than 0.8).

	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**

Experiments & results

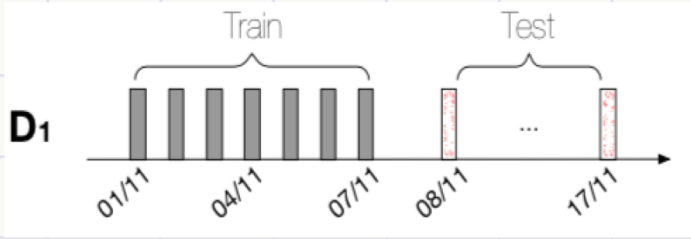
Influence of training period length:

- Experiment:

  Training:

  1. From D1, train five models using the one-day data from 01 Nov to 05 Nov.
  2. From D1, train one single model with the one-week data from 01 Nov to 07.
     (In contrast to prove a longer training period can offer better performance)

  Evaluation: Evaluate on the test data from 8 Nov to 17 Nov.



- Results:

  1. Precision:

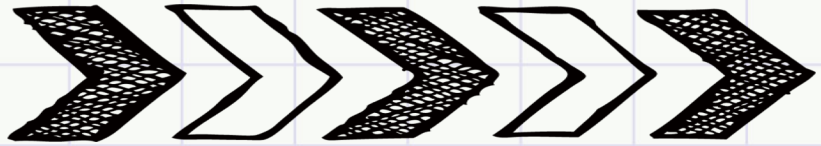
     One week data training (0.82), One day data training (0.79).

  2. Efficient : The same.
  3. Deviating :

     Week long train more stable compared with one day.

	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

Related Work



Related Work

1. System-level security event forecast:

- Soska trained the ensemble C4.5 classifiers to predict if a given website will become malicious in the future, based on the features from website.
- Bilge design the RiskTeller leverages both supervised and semi-supervised methods to predict if the machine is at risk, based on the binary log information of machines.

2. Organization-level security event forecast:

- Liu trained a Random Forest classifier to forecast security incidents, based on features including 258 features covering two main categories : mismanagement symptoms and malicious activities.
- Liu furtherly prove the malicious activities are stable indicators to build the predictable model. He train a SVM model based on intensity, duration and frequency.

3. Cyber-level security event forecast:

- Sabottke build a Twitter-based exploit detector to predict the vulnerabilities disclosed in Microsoft products.

4. Binary Analysis:

- Shin build RNN models to identify if the 256-vector one-hot encoding functions will happen or not, based on a binary classification output.

4. Anomaly Detection:

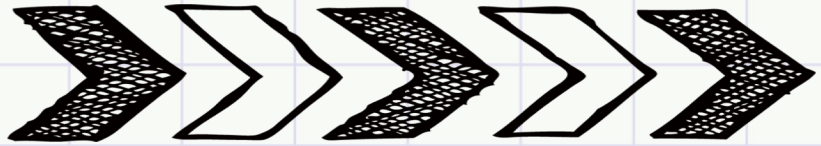
- Du proposed the DeepLog using LSTM to learn the system’s log patterns. And help identify abnormal log entries.

5. Malware Classification:

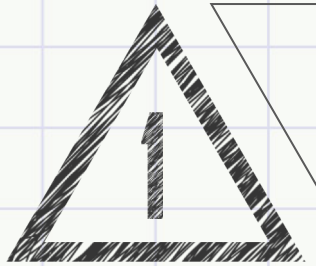
- Pascanu model malware API calls as a sequence and use RNN to predict the next API calls.

Part 6

Future Work

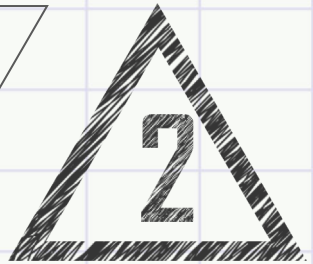


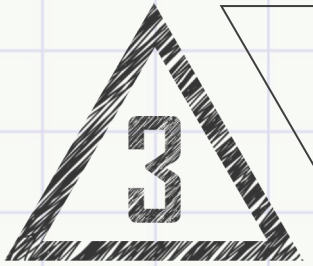
Future Work



Attention based LSTM to replace Rocki-LSTM.

Make a comparison between ml/dl methods and Rocki-LSTM about training time efficiency, model performance ...





Show more statistical analysis on standard deviation between Long/short term data to prove that long-term data training is more robust to anomalies.

Group Name: Aw, Snap!

Group Members: Weijia Sun, Xinyu Lyu, Mengmei Ye