EP2420/EP272V Project 2
IT Intrusion Detection Using Testbed Measurements

Rolf Stadler, Xiaoxuan Wang

November 22, 2023

Project Objective

We consider an intrusion detection use case which involves the IT infrastructure of an organization (see Figure 1). The infrastructure includes a set of servers that run client applications and an Intrusion Detection System (IDS), which logs events in real-time. Clients access the applications through a public gateway, which also is open to an attacker. The attacker intrudes on the infrastructure and compromises a set of its servers. The defender continuously monitors the infrastructure through accessing IDS and other statistics. The defender does not directly observe the attacker but only indirectly learns about an attack through the monitoring data. Using this data, the defender has the following objectives:

1. Predict whether there is an ongoing attack, and if true, predict the start time of the attack.
2. If there is an ongoing attack, predict the sequence of attack actions.

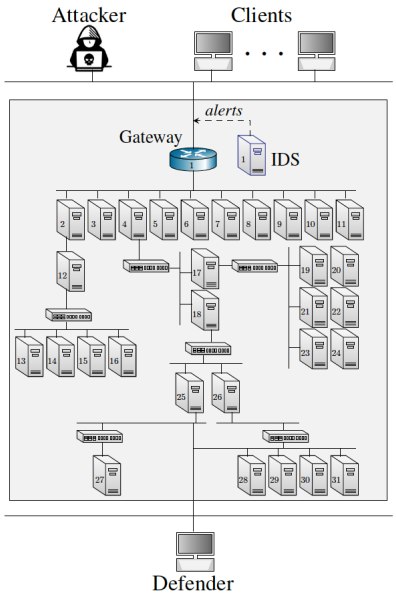

Figure 1: IT infrastructure

In this project, you study and evaluate intrusion detection methods. The methods are based on Random Forest Classifier (RFC) [1], Hidden Markov Models (HMM) [2], and Long Short-Term Memory (LSTM) [3].

They allow a defender to train models based on observations offline and to predict a sequence of attack stages online. The predictions allow a defender to estimate whether an attack is occurring and in which attack stage it is.

For the project, you will use the Anaconda environment (anaconda.com), including Jupyter notebook, and the scikit-learn machine learning packages (scikit-learn.org).

Project tasks

Our group has developed an emulation system where key functional components of the target infrastructure in Figure 1 are replicated. This system can be used to run attack scenarios and produce system measurements and logs. The attacker and the defender observe the infrastructure continuously and take actions at discrete time-steps ($t = 1, 2, ..., T$) of length 30s. You will receive data sets with traces collected from the emulation system. More information about the emulation platform is available at [4].

The project includes five tasks. Task 1 to Task 4 are mandatory. At each deadline shown below, you submit two files, a report in pdf and a Jupyter notebook file with the code. Name your files Your_name.Task(s).pdf or Your_name.Task(s).ipynb, respectively. For all numerical results in your reports, give three significant digits, for instance, 52.3 or 5.23e+01.

1. Week 1: Task 1.
2. Week 2: Task 2.
3. Week 3: Task 3.
4. Week 4: Task 4 and final report.
5. The optional task can be submitted any week.

Task 1 - Literature study and data set preparation

1. The objective is to get familiar with HMM and prepare the data set.
2. Read the document about HMM (available in 'Other resources' on Canvas) so that you can explain the forward-backward algorithm, the Viterbi algorithm, and the supervised training process.
3. The trace contains 2000 sample pairs. One such pair includes one attacker action sequence and the corresponding defender observation sequence. An element of an attacker sequence can either be 'Continue', which means that an attack has not yet occurred, or it can be an attack action, which means that an intrusion is ongoing. In the defender observation sequence, there is an alert value at each time step. This value is based on the number of IDS alerts that occurred during the time-step weighted with priority.
4. Plot the distribution of alert values for different attack actions as a histogram and as a density plot. Choose one sample pair and produce the time-series plot of the alert value. Describe your observations of the plot.
5. Model the system states and the corresponding observations with a HMM model. In the attacker action sequence, you map the actions into different hidden states in the HMM model. For the defender observation sequence, find a mapping from the alert values to corresponding observation symbols (see the HMM document mentioned above). Describe the mapping.
6. Transform the traces you have received into sequences of hidden states and corresponding observation symbols using the above mapping.

Task 2 - Use HMM for intrusion detection

1. The objective is to use HMM to solve the problem of attack action prediction, which means decoding an observation sequence into a hidden state sequence.

2. Randomly select 70% of samples as the training set and the remaining 30% as the test set.

3. Train a HMM model using the supervised training method. List the learned parameters of the HMM model. Apply the Viterbi algorithm to predict the hidden state sequence for an observation sequence from the test set. Evaluate the learned HMM model regarding the prediction accuracy of the intrusion start time and attack action.
   We define:

   - Acc_{start}: the accuracy of correctly predicting the intrusion start time. t^i_{start} is the real intrusion start time step for the i-th sequence and $\widehat{t}^i_{start}$ is the predicted one. l stands for the number of sample sequences for testing.

$$Acc_{start} = \frac{1}{l}\sum_{i=1}^{l} \mathbb{1}\left\{t^i_{start} == \widehat{t}^i_{start}\right\} \tag{1}$$

   - Acc_{action}: the fraction of correctly predicted single actions. s^i_t is the real attack action at time step t of the i-th sequence and $\widehat{s}^i_t$ is the predicted one. T represents the sequence length.

$$Acc_{action} = \frac{1}{lT}\sum_{i=1}^{l}\sum_{t=1}^{T} \mathbb{1}\left\{s^i_t == \widehat{s}^i_t\right\} \tag{2}$$

Task 3 - Use LSTM for intrusion detection

1. The objective is to use LSTM to solve the problem of attack action prediction.

2. Use original alert values instead of the mapped symbols in observation sequences. Randomly select 70% of samples as the training set and the remaining 30% as the test set.

3. Train a LSTM model using the training set. List the architecture and hyper-parameters of the LSTM model. Use the trained neural network to predict the hidden state sequence for an observation sequence from the test set. Evaluate the LSTM model regarding the prediction accuracy of the intrusion start time and attack action. (Compute Acc_{start} and Acc_{action}.)

4. Compare the results from HMM and LSTM, and discuss your observations.

Task 4 - Use RFC for intrusion detection

1. The objective is to use RFC to solve the problem of attack action prediction.

2. Use original alert values instead of the mapped symbols in observation sequences. Randomly select 70% of samples as the training set and the remaining 30% as the test set. For the training set, take each alert value and corresponding attack action as a sample pair.

3. Train a RFC model using the training set where different attack actions are interpreted as different class labels. Use the trained model to predict the attack action for each given observation from the test set and form the predicted sequence. Evaluate the RFC regarding the prediction accuracy of the intrusion start time and attack action. (Compute Acc_{start} and Acc_{action}.)

4. Compare the results from all three methods and discuss your observations.

Optional Task - Use multiple HMMs for classifying the attack types

1. The objective is to use multiple HMMs to solve the problem of attack-type classification. The new trace contains 2000 sample pairs, 1000 for each attack type. We build one HMM model for each type of attack.

2. For the dataset of each attack type, randomly select 70% of samples as the training set and the remaining 30% as the test set. Train the two HMM models (HMM1 and HMM2) for the two attack types respectively. List the learned parameters of the HMM models learned through supervised training.

3. With the learned HMM models, apply the forward algorithm to predict the probability of occurrence for an observation sequence from the test set. If the probability value from HMM1 is higher than that from HMM2, the observation sequence is assumed to be generated by attack 1. Otherwise, it is assumed to be generated by attack 2. Evaluate the learned HMMs regarding the classification accuracy of the attack type.
   We define:

   - Acc_{type}: the fraction of correctly classified observation sequences. c represents the attack type.

$$Acc_{type} = \frac{1}{l} \sum_{i=1}^{l} \mathbb{1} \left\{ c^i == \widehat{c}^i \right\} \tag{3}$$

4. Choose one or more baselines for classifying the attack type. For example, random forest classifier. Compare the results from all methods and discuss your observations.

Resources

- **HMM**: `https://hmmlearn.readthedocs.io/en/latest/index.html`
- **LSTM**: `https://www.tensorflow.org/guide/keras/working_with_rnns`
- **RFC**: `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`

References

[1] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[2] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4. Springer, 2006.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] K. Hammar and R. Stadler, "Learning near-optimal intrusion responses against dynamic attackers," *arXiv preprint arXiv:2301.06085*, 2023.