Machine learning for IDS log analysis

Joseph Azar (FEMTO-ST)
Malcolm Bourdon (EDF R&D, LAAS-CNRS)
Alexandre Debant (Univ Rennes, CNRS, IRISA)
Julien Girard-Satabin (CEA, INRIA)
Yuxiao Mao (LAAS-CNRS)

October 25, 2019
IEC 104 protocol

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>53</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>54</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>57</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>58</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>59</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>61</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>62</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>63</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>65</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>66</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>67</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>68</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>69</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
<td>184476</td>
<td></td>
</tr>
</tbody>
</table>

Log analysis

October 25, 2019,
IEC 104 protocol

Log analysis

October 25, 2019
IEC 104 protocol - attacks

Log analysis October 25, 2019,
Several classes of attacks

The attacker is able to:

- replay valid packet already sent
- forge and send an invalid packet
- sending arbitrary messages of the protocol
- sending many packets quickly
How can we detect malicious behaviours using machine learning techniques?
ML blitz
The big question

Why do we use machine learning today?
Drowning in a sea of information

about $10^6$ terabytes per day
Hard to specify sometimes

No specification of what is a pedestrian: learn from examples
What is machine learning?

$\tilde{f}$: ideal function
$
\tilde{X}$: ideal representation of data

**Goal**
learn $f$ approximating $\tilde{f}$, using an approximation of data $\tilde{X}$
Supervised learning

Dataset $\mathcal{X}$ is labelled

Approximated function: classifier between the different labels

Remark: labelling data is costly!
Supervised learning

Dataset $\mathcal{X}$ is labelled

Approximated function: classifier between the different labels

Remark: labelling data is costly!
## Some standard algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Explainability</th>
<th>Generalization</th>
<th>Learning cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Tree (DT)</td>
<td>very good</td>
<td>poor</td>
<td>cheap</td>
</tr>
<tr>
<td>Support Vector Machine (SVM)</td>
<td>poor</td>
<td>good</td>
<td>cheap</td>
</tr>
<tr>
<td>Neural Networks (NN)</td>
<td>poor</td>
<td>very good</td>
<td>expensive</td>
</tr>
</tbody>
</table>
Unsupervised learning

Dataset $\mathcal{X}$ is **not labelled**

Rely on the inherent structure of the data

Approximated function: a representation of the data
Some standard algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Generalization</th>
<th>Learning cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clustering (k-nn)</td>
<td>good</td>
<td>cheap</td>
</tr>
<tr>
<td>Dimensionality reduction (PCA, t-SNE)</td>
<td>poor</td>
<td>cheap</td>
</tr>
<tr>
<td>Neural networks (auto-encoders)</td>
<td>very good</td>
<td>expensive</td>
</tr>
</tbody>
</table>

Log analysis October 25, 2019,
Data analysis
## Data analysis: first analysis

- **Data size:** 863 rows × 27 columns, 147 kB
- **Attack / Non-attack:** 610 / 253
Data analysis: first analysis

- attack
- captured_length, dst_port, frame_length, ip_checksum, ip_checksum_status, ip_size, ip_dest, ip_src, src_port, tcp_size, timestamp
- addr, causetx, ioa, nega, numix, oa, proto_name, proto_size, sq, typeid, test, qoi, siq, sco, dco
Data analysis: first analysis

- Important fields:
  - **typeid** (type identification)
  - **causetx** (cause of transmission)
  - **ioa** (information object address)
  - **ip_checksum** (IPs, sequence of transmission)
Data analysis: preprocessing

- normalization
  - why? different range
  - what? numeric fields
  - how? mean = 0, standard deviation = [-1,1]

- one-hot encoding
  - why? strings
  - what? non-numeric fields (type, address...)
  - how? e.g., IP address

192.168.1.1
192.168.1.2
192.168.1.3
[0, 0, 1]
Data analysis: preprocessing

- normalization
  - why? different range
  - what? numeric fields
  - how? mean = 0, standard deviation = [-1,1]

- one-hot encoding
  - why? strings
  - what? non numeric fields (type, address...)
  - how? e.g., IP address
    - 192.168.1.1 [1, 0]
    - 192.168.1.2 [0, 1]
Data analysis: preprocessing

• normalization
  • why? different range
  • what? numeric fields
  • how? mean = 0, standard deviation = [-1,1]

• one-hot encoding
  • why? strings
  • what? non numeric fields (type, address...)
  • how? e.g., IP address
  
    192.168.1.1 [1, 0, 0]
    192.168.1.2 [0, 1, 0]
    192.168.1.3 [0, 0, 1]
Data analysis: Principal Component Analysis (PCA)

- dimensionality reduction
Data analysis: split technics

- sequential split (75% training)
  - training: 398 normals, 249 attacks
  - evaluation: 212 normals, 4 attacks
Data analysis: split techniques

- **Sequential split (75% training)**
  - training: 398 normals, 249 attacks
  - evaluation: 212 normals, 4 attacks

- **Random split (75% training)**
  - training: 448 normals, 199 attacks
  - evaluation: 162 normals, 54 attacks
Data analysis: limitations of dataset

- small dataset with only 863 IEC104 packets
- repetitive legitimate behaviours
- unbalanced attacks behaviours
  - many Denial-of-Service (DoS) attack packets
  - few occurrences of each attack
  - 2 fields to draw out 1/4 attacks
  - 1 field with sequence to draw out most of DoS attacks
ML without sequence
Problem statement and limitations

• inputs: one packet for one output

• limitation: no context (DoS attacks indistinguishable)
Dataset $\mathcal{X}$, classes $\mathcal{Y} = \{y_1, y_2\}$, $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \in \mathcal{X}$

$x_1 > a \rightarrow P(\mathbf{x} \in y_1)?$ \\
$x_1 \leq a \rightarrow P(\mathbf{x} \in y_2)?$
Dataset $\mathcal{X}$, classes $\mathcal{Y} = \{y_1, y_2\}$, $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \in \mathcal{X}$

$x_1 > a \rightarrow \mathcal{P}(\mathbf{x} \in y_1)$?
$x_1 \leq a \rightarrow \mathcal{P}(\mathbf{x} \in y_2)$?

Decision tree answer those questions
Decision Trees (DT)

Dataset $\mathcal{X}$, classes $\mathcal{Y} = \{y_1, y_2\}$, $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \in \mathcal{X}$

$x_1 > a \rightarrow \mathcal{P}(\mathbf{x} \in y_1)$?

$x_1 \leq a \rightarrow \mathcal{P}(\mathbf{x} \in y_2)$?

Decision tree answer those questions
Decision Trees (DT)

Dataset $\mathcal{X}$, classes $\mathcal{Y} = \{y_1, y_2\}$, $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \in \mathcal{X}$

$x_1 > a \rightarrow \mathcal{P}(\mathbf{x} \in y_1)$?
$x_1 \leq a \rightarrow \mathcal{P}(\mathbf{x} \in y_2)$?

Decision tree answer those questions
Decision Trees (DT)

- case split on feature using different criterion (Gini, entropy)
- no parameter tuning, easy to train
- sensitive to data variations, can overfit fast
• dataset is small $\Rightarrow$ sensitive to bad data balancing
• mitigation: train \textbf{multiple} models on \textbf{multiple} splits
Decision Trees: results

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} : \text{number of correct predictions}
\]

\[
\text{Recall} = \frac{TP}{TP + FN} : \text{number of detected anomalies}
\]

- training time is less than 2ms on a Intel I7-8850H
- sequential split: recall is 0%
- random split: recall is 94,3%, accuracy: 96,6%
Accuracy = \frac{TP + TN}{TP + TN + FP + FN}: number of correct predictions

Recall = \frac{TP}{TP + FN}: number of detected anomalies

- training time is less than 2ms on a Intel I7-8850H
- sequential split: recall is 0%
- random split: recall is 94.3%, accuracy: 96.6%

Works surprisingly well. Why?
Our decisions trees are overfitting
The two culprits features

![Graph showing features over samples]

Log analysis October 25, 2019,
Support Vector Machine (SVM)

image source: wikipedia
• multiple kernels used
• accuracy: 79.7%, recall: 26.3%
Dense NN

Inputs

\[ b_{k-1}^{\ell-1} \]

\[ b_k^{\ell} \]

\[ b_{j-1}^{\ell} \]

\[ b_j^{\ell} \]

\[ b_{j+1}^{\ell} \]

\[ b_{k-1}^{\ell+1} \]

\[ b_k^{\ell+1} \]

Outputs

\[ b_j^{\ell} = \sum_k^K w_{kj}^{\ell} a_k^{\ell-1} + b_j^{\ell} \]  
(output before activation)

\[ a_j^{\ell} = \sigma(b_j^{\ell}) \]  
(output after activation)

\[ C \]  
(Cost function)

gradient of \( C(\text{Outputs}) \) with respect to weights

updating weights \( w^{\ell} \)
Dense NN: parameters and results

- fully connected network
- 4 layers and $10^6$ neurons
- recall : 26.3%, accuracy : 90.9%
Supervised learning works because of over-fitting
K-means and PCA
Variational Auto Encoder

Goal: learn the probability distribution of the input

Training objective: input $x$, learn a code $s$, an encoder $Q$ and a decoder $P$ such that $\hat{x}$ is a good reconstruction
(Bad) results for VAE
1. Strong similarity between legitimate and attack packets
2. Unsupervised learning cannot separate efficiently
## Summary of results

<table>
<thead>
<tr>
<th></th>
<th>Supervised</th>
<th>Unsupervised</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Name</td>
<td>Acc.</td>
</tr>
<tr>
<td>No-seq.</td>
<td>SVM</td>
<td>80%</td>
</tr>
<tr>
<td></td>
<td>DT</td>
<td><strong>96%</strong></td>
</tr>
<tr>
<td></td>
<td>DNN</td>
<td><strong>91%</strong></td>
</tr>
</tbody>
</table>
ML with sequences
Sequence classification

- order is important and must be respected
- predicting a class label for a given input sequence
- limitation of classical ML and MLP: Unaware of temporal structure
Supervised sequence classification: LSTM

- recurrent connections
- avoid the problems that prevent the training and scaling of other RNN
- memory cells contain weights and gates
IDS using Bidirectional LSTM

- **loss**: binary cross-entropy, **optimizer**: Adam
- **epoch**: 500, **batch**: 20
- **train**: 595 ($\approx$ 155 anomalies), **test**: 256 ($\approx$ 65 anomalies)
- **training time**: 5min (no GPU)
Evaluation: the beginner’s mistake

- fit to training; evaluate on test; report skill: Wrong!
- deep learning models are stochastic
- LSTM’s use randomness while being fit on a dataset
- same model may give different predictions

```python
scores=list()
for i in repeats:
    train, test = random_split(data)
    model.fit(train.X, train.y)
    predictions=model.predict(test.X)
    skill=compare(test.y, predictions)
    scores.append(skill)
final_skill=mean(scores)
```
IDS using Bidirectional LSTM: results

![Graph showing accuracy over epochs for training and validation datasets.](image)

![Graph showing sigmoid output for normal and attack labels.](image)
accuracy = \frac{TP + TN}{TP + TN + FP + FN}

precision = \frac{TP}{TP + FP}

recall = \frac{TP}{TP + FN}

F1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}

Confusion matrix = \begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix}

Confusion matrix = \begin{bmatrix} 183 & 4 \\ 3 & 66 \end{bmatrix}
IDS using unsupervised learning

- what if you have no labelled data at all?
- binary analysis requires hours of fingerprinting and study per sample
- incident investigation requires huge resources and bureaucratic layers to triage
- infers hidden latent structure from unlabelled training data
- objective: learn from unlabelled data while respecting the temporal order
IDS using unsupervised learning: strategy

- data preparation
- build an auto-encoder on the normal (negatively labelled) data
- use it to reconstruct a new sample
- if the reconstruction error is high, we label it as an attack
IDS using unsupervised learning: data preparation

- the input to LSTMs are 3-dimensional arrays
- sliding window of size 6 and step = 1
IDS using unsupervised learning: LSTM auto-encoder
• trained on legitimate packets
• tested on legitimate and attack packets
• epoch: 3500, batch: 10
• training time: \( \approx 30 \) min (no GPU)
IDS using unsupervised learning: LSTM auto-encoder results
IDS using unsupervised learning: LSTM auto-encoder results

reconstruction error of normal packets

reconstruction error of attack packets
IDS using unsupervised learning: LSTM auto-encoder results

Reconstruction error for different classes

Data point index

Reconstruction error

- Benign
- Attacks
- Threshold
IDS using unsupervised learning: LSTM auto-encoder results

![Recall vs Precision](image1)

**auto-encoder**
(no sequence)

![Recall vs Precision](image2)

**LSTM auto-encoder**
(sequence)
IDS using unsupervised learning: LSTM auto-encoder results

- auto-encoder (no sequence)

- LSTM auto-encoder (sequence)
IDS using unsupervised learning: LSTM auto-encoder results

auto-encoder (no sequence)

LSTM auto-encoder (sequence)
IDS using unsupervised learning: LSTM auto-encoder results

auto-encoder
(no sequence)

LSTM auto-encoder
(sequence)
IDS using unsupervised learning: What can be done better on huge data?

- CNN LSTM Autoencoder
- LSTM Dropout (Dropout_U and Dropout_W)
- Gaussian-dropout layer
- SELU activation
- alpha-dropout with SELU activation
Conclusion
1. Preliminary work

- understand the protocol specification and the attacker model
- being able to identify (non-)legitimate packets
How did we tackle the problem using ML?

1. Preliminary work
   - understand the protocol specification and the attacker model
   - being able to identify (non-)legitimate packets

2. Data analysis
   - identify relevant fields (non-constant fields, principal component analysis...)
   - verify that legitimate/attack packets are balanced
1. Preliminary work
   - understand the protocol specification and the attacker model
   - being able to identify (non-)legitimate packets

2. Data analysis
   - identify relevant fields (non-constant fields, principal component analysis...)
   - verify that legitimate/attack packets are balanced

3. Apply ML techniques with single or sequence of packets
   - first, the simplest algorithms (SVM, decision trees, k-means)
   - then the more complex ones (DNN, LSTM, auto-encoders)
1. Preliminary work
   - understand the protocol specification and the attacker model
   - being able to identify (non-)legitimate packets

2. Data analysis
   - identify relevant fields (non-constant fields, principal component analysis...)
   - verify that legitimate/attack packets are balanced

3. Apply ML techniques with single or sequence of packets
   - first, the simplest algorithms (SVM, decision trees, k-means)
   - then the more complex ones (DNN, LSTM, auto-encoders)

4. Evaluation of the results
   - presentation of results
   - explanation of success/failures (e.g., identify over-fitting)
The different algorithm used

<table>
<thead>
<tr>
<th></th>
<th>Supervised</th>
<th></th>
<th></th>
<th>Unsupervised</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Name</td>
<td>Acc.</td>
<td>Rec.</td>
<td>Time</td>
<td>Name</td>
</tr>
<tr>
<td>No-seq.</td>
<td>SVM</td>
<td>80%</td>
<td>26%</td>
<td>&lt;1ms</td>
<td>k-means</td>
</tr>
<tr>
<td></td>
<td>DT</td>
<td>96%</td>
<td>97%</td>
<td>&lt;1ms</td>
<td>AE</td>
</tr>
<tr>
<td></td>
<td>DNN</td>
<td>91%</td>
<td>26%</td>
<td>2min</td>
<td>VAE</td>
</tr>
<tr>
<td>Seq.</td>
<td>LSTM</td>
<td>94%</td>
<td>89%</td>
<td>5min</td>
<td>LSTM AE</td>
</tr>
</tbody>
</table>
Results and advices

Results in a nutshell:

• considering sequences is mandatory
• similar results between unsupervised and supervised ML

Few advices for re-using our approach:

• generate an adapted dataset
• consider a more realistic network
• test the simplest algorithms first
Results and advices

Results in a nutshell:

• considering sequences is mandatory
• similar results between unsupervised and supervised ML

Few advices for re-using our approach:

• generate an adapted dataset
• consider a more realistic network
• test the simplest algorithms first

Thank you for your attention