Information & Security

Published since 1998

A. Kok, I. Ilic Mestric,
G. Valiyev & M. Street
vol. 47, no. 2 (2020): 203-220
https://doi.org/10.11610/isij.4714
ISSN 0861-5160 (print), ISSN 1314-2119 (online)

Research Article

Cyber Threat Prediction with Machine Learning

***Arvid Kok* (✉), *Ivana Ilic Mestric*,**
Giavid Valiyev*, *Michael Street

Service Strategy & Innovation NATO C & I Agency, The Hague, Netherlands,
https://www.ncia.nato.int/

ABSTRACT:

In this paper we address the approaches, techniques and results of applying machine learning techniques for cyber threat prediction. Timely discovery of advanced persistent threats is of utmost importance for the protection of NATO's and its allies' networks. Therefore, NATO and NATO Communication and Information Agency's Cyber Security service line is constantly looking for improvements. During Coalition Warrior Interoperability Exercise (CWIX) event data was captured on a Red-Blue Team Simulation. The data set was then used to apply a variety of Machine Learning techniques: deep-learning, auto-encoding and clustering with outliers.

ARTICLE INFO:

RECEIVED: 08 JULY 2020
REVISED: 14 SEP 2020
ONLINE: 22 SEP 2020

KEYWORDS:

cybersecurity, machine learning, deep learning, auto-encoding, DBSCAN, clustering with outliers, MITRE ATT@CK framework, KNIME Analytics Platform

Creative Commons BY-NC 4.0

Introduction

The NATO Communication and Information Agency (NCIA) Data Science Team supports projects related to data, (advanced) analytics, machine learning and visualization. This paper describes an increasingly popular area of Cyber Security and how Machine Learning techniques were used for Threat Detection – or perhaps more accurate: Threat Prediction.

Timely discovery of Advanced Persistent Threats (APT) is of utmost importance for the protection of NATO's and its allies' networks. Hackers apply

✉ E-mail: arvid.kok@ncia.nato.int

different techniques over longer periods when targeting a network. Cyber Security experts are tasked to detect the techniques, determine the tactics and recognize the APT.

The experiments executed and described in this paper address data preparation and machine learning for technique and tactic prediction; potentially preparing for APT discovery. Experiments for both known and unknown techniques are explored.

Cyber Security Simulation

NATO Allied Command Transformation (ACT), supported by NATO Communication and Information Agency (NCI Agency), performed a Cyber Security exercise, including a Red-Blue Team Simulation, during the 2019 Coalition Warrior Interoperability Exercise (CWIX) in Poland. During this exercise's simulation the red team was using known hacking techniques, where the blue team was trying to detect. This simulation took place on an isolated network with simulated user activity.

During the simulation NATO's Cyber Security experts compared implemented detection methods with MITRE and the MITRE ATT@CK™ framework enriched methods. MITRE ATT&CK™ is a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations. The ATT&CK knowledge base is used as a foundation for the development of specific threat models and methodologies in the private sector, in government, and in the cybersecurity product and service community.[5]

Resulting from the simulation exercise – among other results – extracts of all captured windows event logs and MITRE threat detection logs were made available to NCI Agency's Data Science team. In total 93 million windows event log entries and 75 thousand MITRE threat detection log entries were provided.

Windows events logs included:

- Application
- Microsoft-Windows-PowerShell/Operational
- Microsoft-Windows-Sysmon/Operational
- Microsoft-Windows-TerminalServices-LocalSessionManager/Operational
- Microsoft-Windows-Windows Firewall with Advanced Security/Firewall
- Security
- System.

MITRE threat detection logs indicated:

- ATT@CK Tactics
- ATT@CK Techniques
- References to windows event logs.

Information on the applied cyberattack tactics and techniques (red-team activity) was deliberately kept behind, so results could not be optimized to better reflect reality. Also, no information was shared about what attacks were detected by the blue team.

Platform for Experimentation and Implementation

To ingest, pre-process, experiment with various machine learning techniques and also reduce the implementation effort, the *KNIME Analytics Platform* – or simply *KNIME* [1] – was selected as the core platform.

KNIME *Analytics Platform* is a free, open source workflow editor following the familiar drag-'n'-drop paradigm with double-click accessible dialogue windows to edit component (node) properties. KNIME implements its own workflow enactment orchestrated by the Java virtual machine but also supporting component (node) enactment in Python and R, as well as parallel enactment through modern Hadoop cluster technologies (including Apache Spark) and Cloud-based services.

The platform provides a graphical composition framework for data preparation, model fitting, and result analysis. It relies on GUI-configurable nodes, symbolizing varied data processing steps which can be arranged arbitrarily to create complex workflows. Through its interface KNIME significantly reduces the need for low-level programming, making the data mining process accessible to a larger group of data analysts. For illustration, Figure 1 shows an example workflow for a basic machine learning approach.

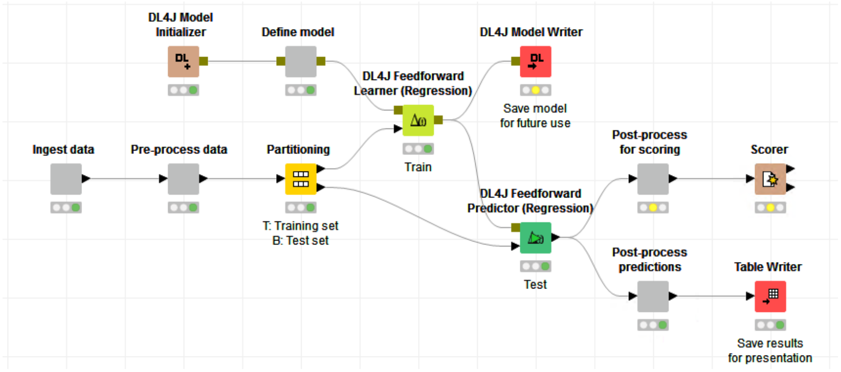


Figure 1: Illustration of a workflow in KNIME.

Data preprocessing

KNIME comes with a complete solution of nodes for handling small and large datasets and provides a possibility of transforming raw data in more normalized and structured format which cover a typical data preprocessing step in data analytics projects.

Machine Learning

KNIME is offering a flexibility of applying latest Machine Learning and Deep Learning frameworks such as PyTorch, TensorFlow, Keras and Deeplearning4J. These deep learning extensions allow user to read, create, train and execute machine learning and deep neural networks within KNIME. Another important feature provided by KNIME is the possibility to use GPU (NVIDIA CUDA) acceleration which facilitates the possibility of applying machine learning models to large datasets as they require intense computational power and possibly being processed in parallel. For these reasons, when it comes to large volume datasets, a GPU can help to speed up the process.

Visualization

For limiting the effort and convenience reasons, the results of the experimentation were brought to a Microsoft Power BI dashboard. Power BI allows for interactive exploration of results. This tool in particular was selected before for its completeness of visual solutions (ships with over 20 chart types and over 200 custom chart types available from store) and in general for its ease of use.

Experiments and Findings for Cyber Threat Prediction

The performed experiments using the provided CWIX Cyber Simulation log data were aiming to find and test an approach for two situations:

1. known tactics and techniques that are applied in a cyber-attack
2. unknown (new) tactics and/or techniques that are applied in a cyberattack.

The machine learning techniques available are growing in number and time restrictions guided the team to search the realm of supervised deep-learning for the first category (known tactics and techniques). The section on "Supervised Machine Learning for training an ATT@CK framework Threat Prediction Model" provides more details. For the second category (unknown tactics and techniques) clustering techniques were considered, next to unsupervised learning. More details on these approaches in paragraph "Unsupervised Machine Learning for Anomaly Detection." These techniques and their target problem are shown in Fig. 2.

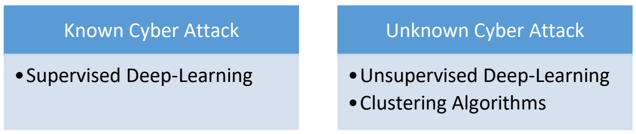


Figure 2: Machine Learning techniques explored.

All approaches to be executed required the provided data sets to be preprocessed. Which turned out to be a small challenge on its own given the volume and variety of the contents. Measures were taken. In order to reduce the required processing duration, the team used distributed computing capabilities on more than one occasion. On top fail-save measures were taken, saving intermediate results were possible.

Preprocessing

Starting with the windows event logs (source for features) and MITRE detection logs (source for labels), the aim for preprocessing was to prepare the appropriate training and test sets for all machine learning techniques to be applied. Following the common (sub) steps in data science:

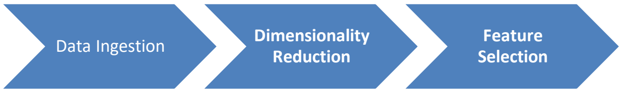


Figure 3: Common steps to prepare for Machine Learning.

The resulting features were aggregated into event time windows to get to a meaningful starting point for machine learning. The preprocessing steps are described in more detail in paragraph "Preprocessing."

As reference for later results, the preprocessed MITRE threat detection logs were visualized to allow overlays and comparisons with machine model results.

Since the actual red-team activity was not shared and Cyber Security (deliberately) did not share the planned and executed attacks, there was no model-trained prediction directly based on that. This more accurately reflects the typical scenario for cyber incidents.

Data Ingestion

The team used KNIME to ingest the provided windows event logs and MITRE detection logs. Both data sets were shared with the team as comma-separated-value (CSV) files, extracted from a Splunk set-up that was collecting during the CWIX Cyber Simulation.

Splunk (the product) captures, indexes, and correlates real-time data in a searchable repository from which it can generate graphs, reports, alerts, dashboards, and visualizations.[6]

93 million windows event log entries and 75 thousand MITRE threat detection log entries were in data sets and processed to a native KNIME table format. The full data sets were filtered, including only the 5-day period of the exercise. "Noise" from the setting up of the simulation system at the exercise was filtered, resulting in almost 49 million windows event log entries to prepare for machine learning (reduction of 47 %).

At this point all windows event log attributes for all entries were made available for processing steps to follow. From the MITRE threat detection log only the timestamp, ATT@CK tactic and ATT@CK technique were kept.

Dimensionality Reduction

Real-world data, such as event log collections, have a high dimensionality (attributes). In order to handle such real-world data adequately, its dimensionality needs to be reduced. Dimensionality reduction is the transformation of high-dimensional data into a meaningful representation of reduced dimensionality.

$$intrinsic\ dimensionality\ d \ll dimensionality\ D$$

Ideally, the reduced representation should have a dimensionality that corresponds to the intrinsic dimensionality of the data. The intrinsic dimensionality of data is the minimum number of parameters needed to account for the observed properties of the data.[7, 8] Figure 4 shows more common techniques for dimensionality reduction.

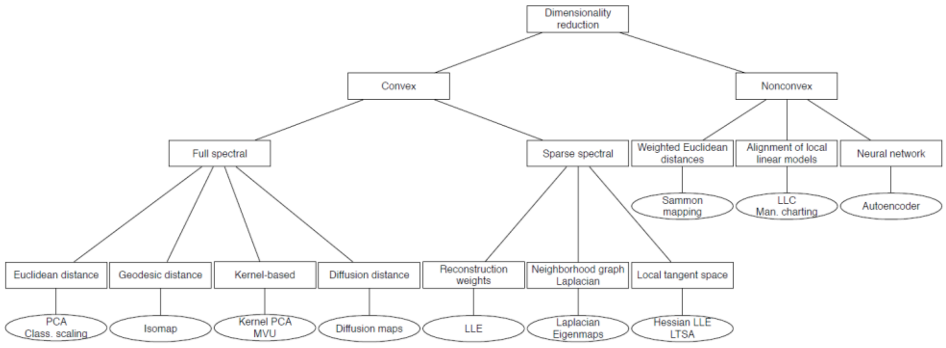


Figure 4: Taxonomy of dimensionality reduction techniques.[8]

No indications or SME knowledge was provided on the dimensions with valuable indicators for threats, therefor they had to be discovered in a way not depending on Cyber Security expertise and still in a controlled manner.

Unfortunately, time permitted only for using statistical reduction methods already available from KNIME, being: Low Variance, High Correlation and (High Variance) Masking.

First step applied was discarding dimensions with low variation. In practice, zero variation tolerance was used on a bootstrap sample of 6 million (12 %) log events evenly picked from the full data set.

Next correlating dimensions were discarded, leaving a single dimension per set of correlating dimensions. Within the 6 million log events sample all records must correlate to be reduced.

As a last step dimensions with high variation (>100k unique values in 6 million log events sample) were masked. The masking process takes all values of a single dimension and partitions them. For each sequential partition the absolute number variances is calculated. If the variances count above the applied threshold, the section value is replaced with a joker-character (e.g. an asterisk '*'). This results in a highly reduced number of unique values and generalizes the data values.

Dimensionality Reduction brought the dimensions of the data set down from 226 to 147 (reduction of 35%), with high confidence that no valuable indicators for Cyber threats were lost.

Feature Extraction and Selection

Having reduced the dimensionality as much as feasible, feature extraction delivered a set of derived values: features. At this point 30.069 features. Feature extraction addresses one attribute at a time and creates a feature column per unique value. The feature column representing the attribute value is set to 1.0 (selecting the value); all other feature columns are left 0.0 (not selected). At this point the earlier masking of attributes limited the resulting features severely.

Despite the effort of dimensionality reduction, over 30 thousand features is still far more than desirable. Therefor the team applied a subset selection method for feature selection.

Subset selection evaluates a subset of features as a group for suitability. Subset selection algorithms can be broken up into wrappers, filters, and embedded methods:[9]

- *Wrappers* utilize the learning machine of interest as a black box to score subsets of variable according to their predictive power.
- *Filters* select subsets of variables as a pre-processing step, independently of the chosen predictor.
- *Embedded* methods perform variable selection in the process of training and are usually specific to given learning machines.

A for our purpose an optimized version of a wrapper was used. The wrapper used a small deep neural network (DNN) consisting of two hidden layers (resp. 100 and 50 fully connected units with ReLu activation), trained using all 30.069 features during 100 epochs to predict the 32 MITRE ATT@CK Techniques. Even though the predictive skills of the model were not optimal, it was expected to highlight the most relevant features. In a follow-up step all feature-groups (features related to a single dimension) were tested. By running the test set through the model ones with the feature-group set to 0.0 and ones set to 1.0, the distance in the model results and therefor the influence of the features was measured. This led to the top 10 shown in Figure 5.

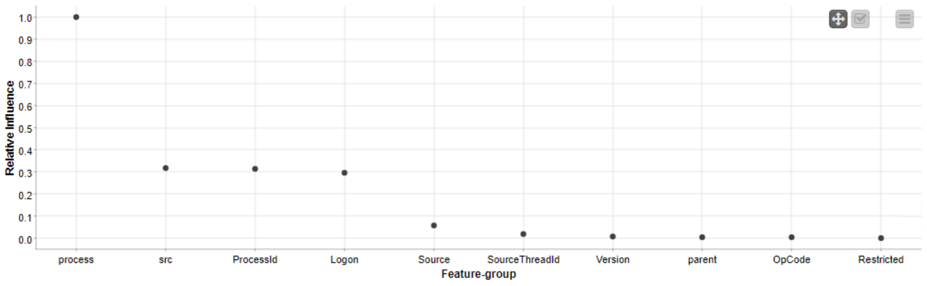


Figure 5: Feature-group influence on DNN prediction error.

For machine learning down-stream the features of the top 5 dimensions were selected, coming to a total of 8042 features (reduction of 73 %, coming from over 30 thousand features). This allowed for smaller models and faster models.

Event Time Window Generation

With the assumption that Cyber threats are recognized in combinations of actions – and therefor log multiple event log entries on these actions – a set of time windows were created from the training and test data set. Each window aggregating the features and labels of a 15-minute time window, shifting the window in 2-minute steps (giving 13 minutes overlap from one window to the next).

Size of the time windows and step size from one window to the next are estimated to be optimal for the scenario based on Cyber SME input. Other window sizes and step sizes are not tested to confirm these assumptions due to time limitations.

The generation coming from the 49 million windows event log entries resulted in 3234 time windows, each composed of 8042 features (related to logged events), plus the label (related to MITRE ATT@CK framework).

As a final step the features were normalized, getting them ready for machine learning. Here the normalizer was set to normalize between -0.07 and 1.02 ("over-normalizing" if you like), which eases the training of the deep neural network (DNN) by taking out the 0.0 "multipliers" out of the equation improving the gradients.

Visualization of MITRE Threat Detection

As a reference for the Machine Learning to be conducted and measuring the results, the MITRE threat detection logs were processed and visualized. The visualizations—and the prepared numbers they are based on—show the number of ATT@CK framework techniques and tactics detected on a timeline.

The visualization allows users to check the results coming from machine models visually. This was used to have an easy indicator to check results and share progress. The numbers behind the visualizations allow for statistically scoring the machine model outcomes.

Supervised Machine Learning for training an ATT@CK framework Threat Prediction Model

In order to predict known tactics and techniques applied in a cyber-attack, supervised Machine Learning (ML) was conducted. The aim was to see if ML is able to predict (or reproduce) the threats detected by MITRE per time window (of 15 minutes), *based on Windows Event Logs.*

In general Machine Learning approaches address problems as classification problem or regression problem.[10]

- Classifications problems are trying to assign class labels to the records of a provided data set. A classification problem can be binary (assign class A or B), multi-class (assign one of many possible classes), or multi-label (assign multiple classes per record).

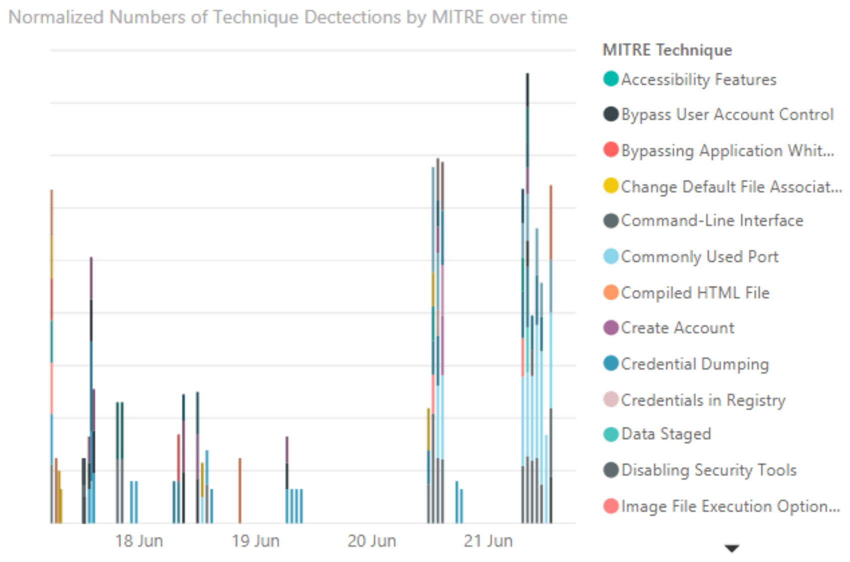


Figure 6: Visualization of MITRE Threat Detection (normalized) for Cyber red-blue team simulation.

- Regression problems require the prediction of quantities. A regression can have real value or discrete input variables, but expect real value outputs.

The main difference between regression and classification is that the output variable in regression is numerical (or continuous) while that for classification is categorical (or discrete).[12]

Problem was recognized as a regression problem, where there are multiple threats happening within the same time window and to a certain extend. Common neural network architectures for regression problems are Convolutional Neural Network (CNN), Multi-Layer Perceptron Neural Network (MLP) and Recurrent Neural Network (RNN). Given the data set the team deemed a MLP – the basis form of a neural network and commonly referred to as Deep Neural Network (DNN) – the best suitable architecture.

- CNNs work well with data that has a spatial relationship (e.g. images).
- RNNs work well when predicting sequences (not the case in the prediction scenario at hand, but would be the case when moving up to APT-prediction).

A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers.[18, 19] The DNN finds the correct mathematical manipulation to turn the input into the output, whether it be a linear relationship or a non-linear relationship. The network moves through the layers calculating the probability of each output. Complex DNN have many layers, hence the name "deep" networks.

Attaching the regression problem, the team first trained a small DNN (consisting of two hidden layers [100 units, 50 units]) during 100 epochs. The fairly quickly trained model allowed for getting a feeling of the problem and if the selected features and chosen architecture were fit for purpose. Results looked very promising to continue down this road.

Assumption at this point was that many combinations of features can indicate a threat, rather than a single feature. To allow the network to better coop with this the first hidden layer was extended to 1000 units (10 times earlier size). Given only five dimensions were included, adding an extra hidden layer was not thought to be beneficial to the accuracy. No extra layers were added; this would have allowed for more complex combinations of indicators to be recognized. The extended model was trained for 1000 epochs, which resulted in an accuracy of 99.7% (on the training set; see Figure 9 for full scoring, showing the Euclidian distance scoring matrix per threat technique) for predicting the different threats present in per time window. Figure 7 shows a comparison of the model's results with the MITRE logging.

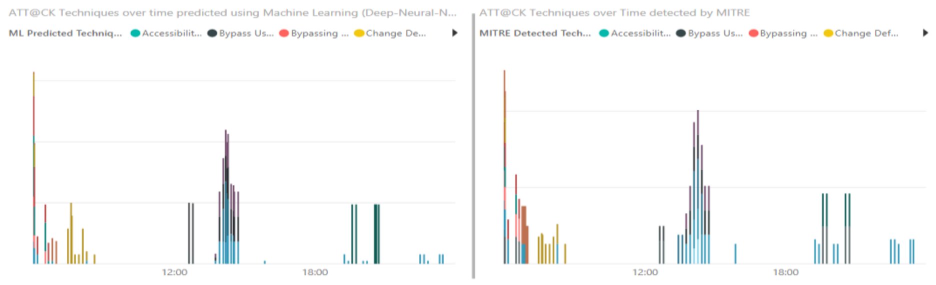


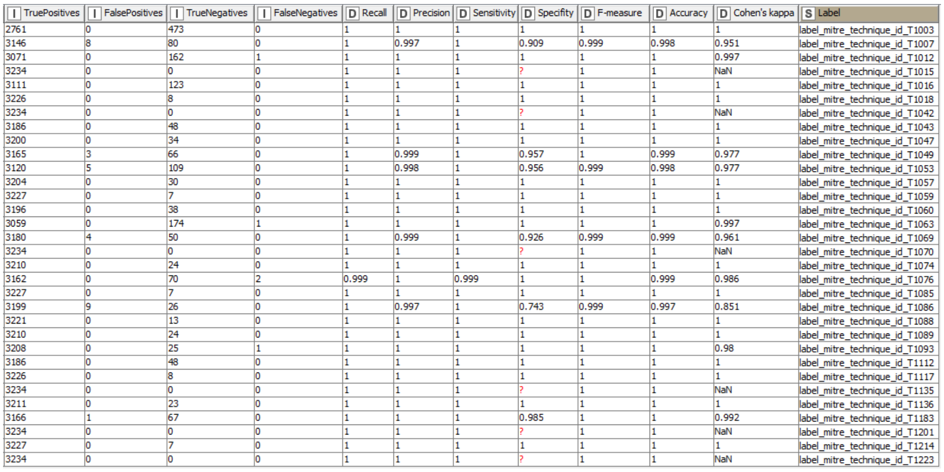
Figure 7: Detail comparison Machine Learning threat prediction vs MITRE threat detection (showing June 17 only; left Machine Learning model, right MITRE threat detection visualization).

The visualizations already show high resemblance, and scoring the model confirmed this. Using Euclidian distance per threat technique, a median distance of 0.017 (75% percentile: 0.021) was measured (0.0 distance being fully aligned; 1.0 opposite results).

The model was trained and test on the same data set, because there is no/too little redundancy in the data set. As a result, the model is not validated, with a risk of having over-fitted the model. The model does show the windows event logs captured, and the selected features, allow for precise prediction of the techniques and tactic as available from the MITRE ATT@CK framework.

Unsupervised Machine Learning for Anomaly Detection

In order to predict if unknown (new) tactics and/or techniques are applied in a cyber-attack, unsupervised Machine Learning (ML) was conducted. Aim was to see if ML is able to *predict* (or reproduce) the *threats* detected by MITRE per time window (of 15 minutes), *based on Windows Event Logs*, with no pre-know-



I TruePositives	I FalsePositives	I TrueNegatives	I FalseNegatives	D Recall	D Precision	D Sensitivity	D Specifity	D F-measure	D Accuracy	D Cohen's kappa	S Label
2761	0	473	0	1	1	1	1	1	1	1	label_mitre_technique_id_T1003
3146	8	80	0	1	0.997	1	0.909	0.999	0.998	0.951	label_mitre_technique_id_T1007
3071	0	162	1	1	1	1	1	1	1	0.997	label_mitre_technique_id_T1012
3234	0	0	0	1	1	1	?	1	1	NaN	label_mitre_technique_id_T1015
3111	0	123	0	1	1	1	1	1	1	1	label_mitre_technique_id_T1016
3226	0	8	0	1	1	1	1	1	1	1	label_mitre_technique_id_T1018
3234	0	0	0	1	1	1	?	1	1	NaN	label_mitre_technique_id_T1042
3186	0	48	0	1	1	1	1	1	1	1	label_mitre_technique_id_T1043
3200	0	34	0	1	1	1	1	1	1	1	label_mitre_technique_id_T1047
3165	3	66	0	1	0.999	1	0.957	1	0.999	0.977	label_mitre_technique_id_T1049
3120	5	109	0	1	0.998	1	0.956	0.999	0.998	0.977	label_mitre_technique_id_T1053
3204	0	30	0	1	1	1	1	1	1	1	label_mitre_technique_id_T1057
3227	0	7	0	1	1	1	1	1	1	1	label_mitre_technique_id_T1059
3196	0	38	0	1	1	1	1	1	1	1	label_mitre_technique_id_T1060
3059	0	174	1	1	1	1	1	1	1	0.997	label_mitre_technique_id_T1063
3180	4	50	0	1	0.999	1	0.926	0.999	0.999	0.961	label_mitre_technique_id_T1069
3234	0	0	0	1	1	1	?	1	1	NaN	label_mitre_technique_id_T1070
3210	0	24	0	1	1	1	1	1	1	1	label_mitre_technique_id_T1074
3162	0	70	2	0.999	1	0.999	1	1	0.999	0.986	label_mitre_technique_id_T1076
3227	0	7	0	1	1	1	1	1	1	1	label_mitre_technique_id_T1085
3199	9	26	0	1	0.997	1	0.743	0.999	0.997	0.851	label_mitre_technique_id_T1086
3221	0	13	0	1	1	1	1	1	1	1	label_mitre_technique_id_T1088
3210	0	24	0	1	1	1	1	1	1	1	label_mitre_technique_id_T1089
3208	0	25	1	1	1	1	1	1	1	0.98	label_mitre_technique_id_T1093
3186	0	48	0	1	1	1	1	1	1	1	label_mitre_technique_id_T1112
3226	0	8	0	1	1	1	1	1	1	1	label_mitre_technique_id_T1117
3234	0	0	0	1	1	1	?	1	1	NaN	label_mitre_technique_id_T1135
3211	0	23	0	1	1	1	1	1	1	1	label_mitre_technique_id_T1136
3166	1	67	0	1	1	1	0.985	1	1	0.992	label_mitre_technique_id_T1183
3234	0	0	0	1	1	1	?	1	1	NaN	label_mitre_technique_id_T1201
3227	0	7	0	1	1	1	1	1	1	1	label_mitre_technique_id_T1214
3234	0	0	0	1	1	1	?	1	1	NaN	label_mitre_technique_id_T1223

Figure 8: Scoring of Machine Learning model for threat technique prediction.

ledge at training time (MITRE detected threat logs not to be used as labels while training the machine models). The assumption is that threats will be seen as anomalies.

Two approaches for detecting anomalies were addressed and explained in following paragraphs:

1. Auto-encoding
2. Clustering with outliers

Auto-encoding

An auto-encoder is a type of artificial neural network used to learn efficient data encodings in an unsupervised manner.[5, 14] The aim of an auto-encoder is to learn a representation (encoding) for a set of data, typically for dimensionality reduction, by training the network to ignore signal "noise". Along with the reduction side, a reconstructing side is learnt, where the auto-encoder tries to generate from the reduced encoding a representation as close as possible to its original input, hence its name. Examples are the regularized auto-encoders (Sparse, De noising and Contractive auto-encoders), proven effective in learning representations for subsequent classification tasks.[16]

The Auto-encoder approach for detecting anomalies applied the following steps:

1. Select the time windows that are to be considered "normal" behaviour.
2. Train the auto-encoder model on the "normal" behaviour time windows to learn how to reproduce the input.
3. Test the auto-encoder model on all time windows, including time windows with potential cyber threats, and collect the output.

Measure the distance between input and output (high distances indicate anomalies)

The Auto-encoders—a specific form of a deep neural network—were composed of three hidden layers. First layer for encoding, second layer a bottleneck and third layer for decoding. During the experiment three auto-encoder models were tested:

I. A model with 100-10-100 units in the hidden layers which was trained for 200 epochs
II. A model with 100-50-100 units in the hidden layers which was trained for 200 epochs
III. A model with 1000-10-1000 units in the hidden layers which was trained for 1000 epochs.

For step 1, "Normal" behaviour was taken from the time periods in between red team activity (read from MITRE Threat Detection visualization). Simplified to 4:00pm to 6:00am, excluding an end-of-day reset of the system. User behaviour was automatically simulated and cycled and therefor expected to be constant, also during the night.

Step 2 (training) and 3 (testing) were likely to take more time to train. In order to select the appropriate auto-encoder configuration, first models I (tight bottleneck) and II (slight bottleneck) were trained. A tight bottleneck will generalize more than a slight bottleneck. Comparing the results from models I and II, more generalization was deemed better. The last model (III) was trained with a tight bottleneck for a longer time (1000 epochs).

Step 4 (measure), was performed by applying Euclidian distance. For each time window the input (features) and output (predictions) were treated as a dimensional vector and the distance was calculated. The error on "normal" event time windows was used to correct the measures. The results of the final model are visualized in Figure 9 (yellow: model output; blue: values corrected for error).

Comparing the pikes – indicating anomalies – with the MITRE reference visualizations (shown here as grey bars pointing down), most threats detected by MITRE are also recognizable from the auto-encoder results. Note that the grey bars do not indicate quantity of detections.

Clustering with outliers

Clustering is a Machine Learning technique that involves the grouping of data points. Given a set of data points, we can use a clustering algorithm to classify each data point into a specific group. In theory, data points that are in the same group should have similar properties and/or features, while data points in different groups should have highly dissimilar properties and/or features. Clustering is a method of unsupervised learning and is a common technique for statistical data analysis used in many fields.[4]

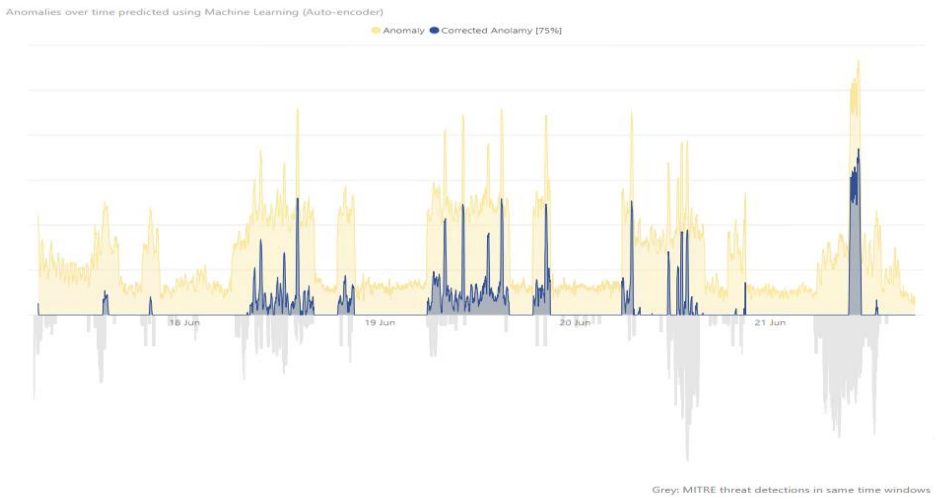


Figure 9: Anomalies over time predicted using Machine Learning (Auto-encoder).

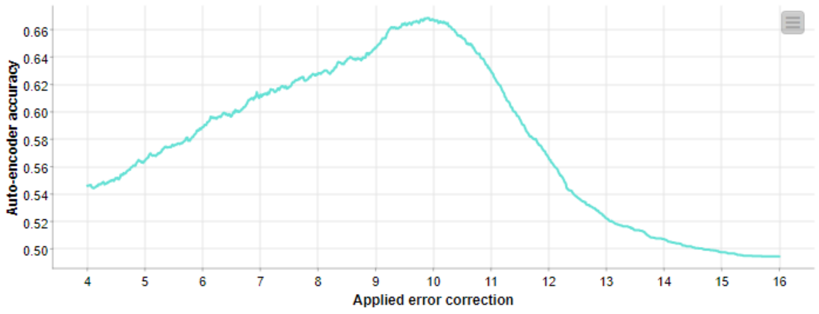
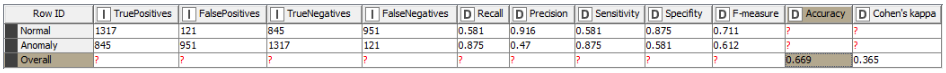


Figure 10: Auto-encoder accuracy by applied error correction (optimum: 66.9% at 9.9 error correction).



Row ID	I TruePositives	I FalsePositives	I TrueNegatives	I FalseNegatives	D Recall	D Precision	D Sensitivity	D Specifity	D F-measure	D Accuracy	D Cohen's kappa
Normal	1317	121	845	951	0.581	0.916	0.581	0.875	0.711	?	?
Anomaly	845	951	1317	121	0.875	0.47	0.875	0.581	0.612	?	?
Overall	?	?	?	?	?	?	?	?	?	0.669	0.365

Figure 11: Auto-encoder score (for 9.9 error correction).

For anomaly detection the focus shifts to the data point that cannot be assigned to the main clusters. The main clusters represent "normal" behavior, where smaller clusters and not-assigned data points represent potential anomalies. The clustering algorithm most likely to find anomalies is *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN).

DBSCAN clustering

DBSCAN is a data clustering algorithm proposed by Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu in 1996.[20] It is a density-based clustering non-parametric algorithm: given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away).

The clustering algorithm allows to fine-tune with two setting in particular:

1. Epsilon (E-value): is the maximum allowed distance between two data point in order to belong to the same cluster. Experiment E-value range: 2.0 to 8.0 (incl.), step size: 0.1.
2. Minimum cluster size (M-value): is the minimum number of data points that can form a cluster. Single data point and smaller clusters will be treated as being "noise." Experiment M-value range: 2 to 6 (incl.), step size: 1.

DBSCAN anomaly detection

For the experiment with DBSCAN clustering the team compared the results for the mentioned ranges combined, clearly showing that models settings have to be tuned for purpose. In order to visualize the scores were matched with the MITRE threat detections (labels), using an additional parameter (by the team named): L-value. The L-value is used to indicate the maximum number of MITRE threat detections in a time window that should trigger an anomaly to be predicted. For L-value the experiment used the range from 1 to 20 (incl.), step size of 1.

Figure 13 shows an accuracy ranging from 19.3% up to 84.0% dependent on E, M and L-values applied. Resp. E=2.0, M=6, L=20 and E=8.0, M=2, L=19. The figure focusses on the influence of the epsilon size (E-value) setting. The larger the epsilon size, the more data point will belong to the same cluster (general clusters). Small values will cause more clusters to be formed (specialization clusters). For the simulation data set a larger epsilon size, building larger general clusters, clearly was beneficial to gain a higher accuracy. The accuracy is the percentage of discreet data points predicted "correct". The percentage includes – in case of the 84.0% score – 166 true positives of "correct" predicted anomalies. Still 319 threats labeled by MITRE were not predicted as anomalies (and therefor false negatives). The team considers the likelihood that the nature of the simulated data set plays a part here: an unrealistically high percentage (depended on L-value between 15% and 32%) of the time windows show abnormal activity. Supporting this theory is the missed large and wide MITRE threat peak showing for June 21st in the afternoon.

Figure 14 and Figure 15 show how minimum cluster size (M-value) and MITRE detections threshold (L-value) influence the accuracy, given the data set at hand.

The minimum cluster size proved to be less influential to the overall accuracy of the model. All values in the used range (2 to 6 incl.) allow to reach accuracies above 80.8%. Potentially a larger range would show its influence (not tested).

As already mentioned, the best accuracy seen in the experiment reaches up to 84.0%, predicting anomalies using the MITRE threat detection labels. Interesting to see is that increasing the L-value allows for both the best and worst accuracy. More importantly it shows that the influence of the L-value wears out, already allowing accuracy of 75.0% at L=5 and 80.0% at L=10.

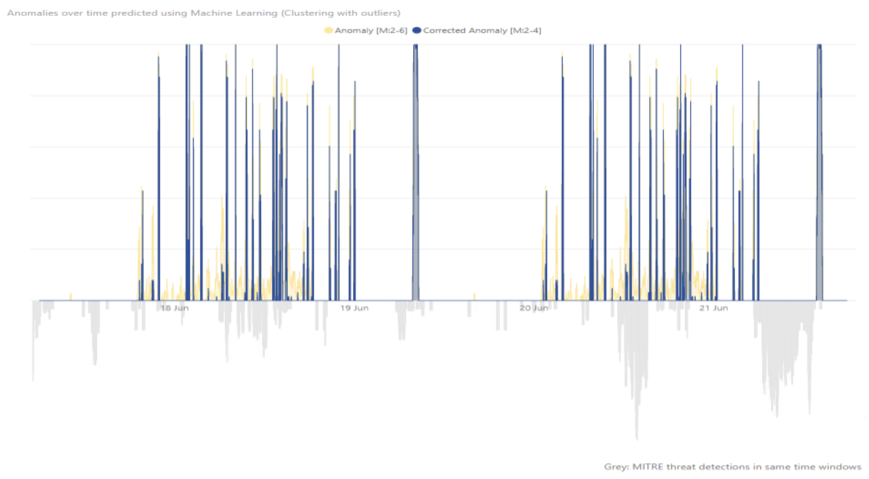


Figure 12: Figure 10: Auto-encoder accuracy by applied error correction (optimum: 66.9% at 9.9 error correction).

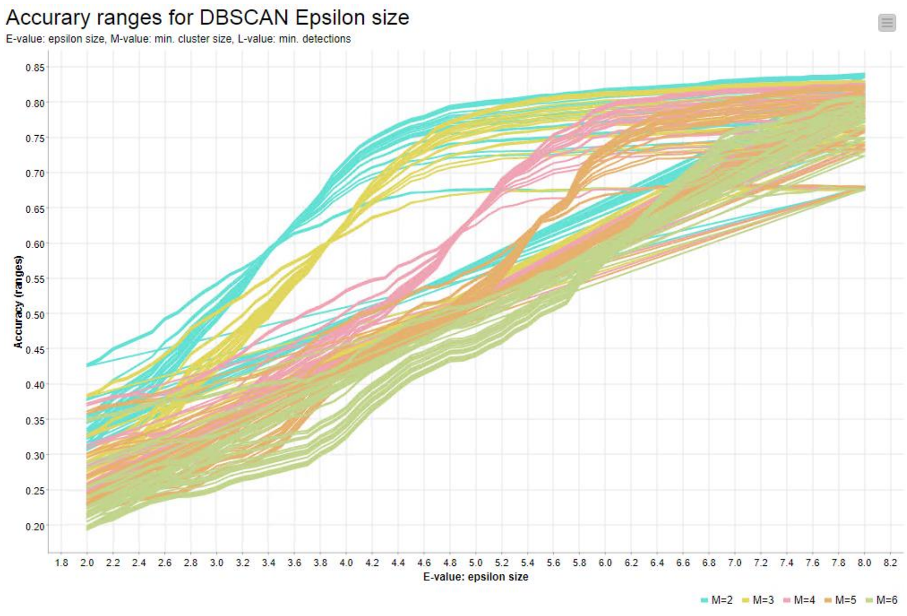


Figure 13: Accuracy ranges for DBSCAN epsilon size, by cluster size.

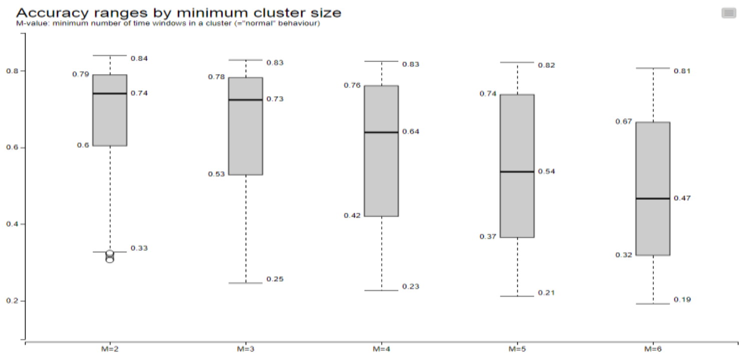


Figure 14: Accuracy spread by DBSCAN minimal cluster size.

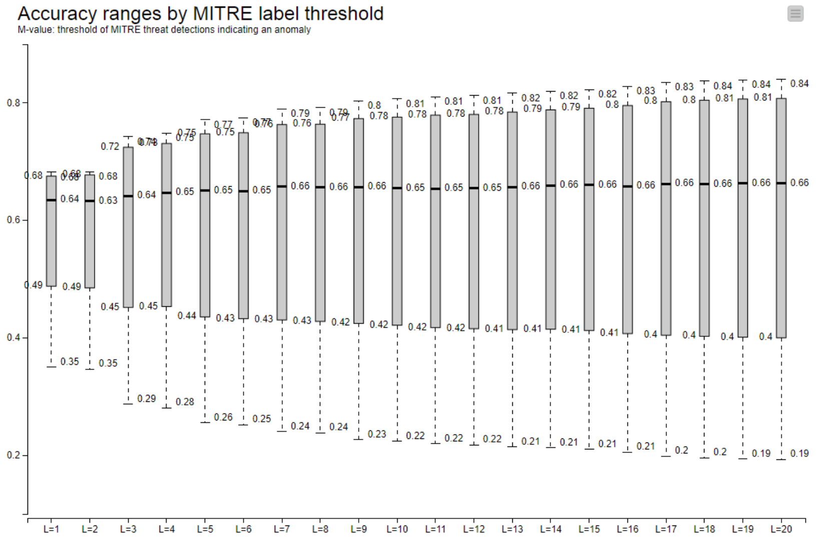


Figure 15: Accuracy spread by applied MITRE detections threshold.

Recommendations

The experiments executed clearly show that the event logs captured contain the information that allows for predicting of threat techniques with a high precision. They show promising results for the machine learning techniques.

The data sets were to a high extent artificial though, and therefor limited the possibility to assess how results relate to real-life situations. The results look promising and could even improve when applied to "real" situations (e.g. anomaly detection using clustering with outliers).

The work did not explore the possibilities of applying prediction techniques in operational systems, or linking results to operational challenges. Further work could bring valuable information and tools for the Cyber Security Centre and Cyber community in general and this will be explored in further work which is planned.

From this work the Data Science team recommends to:

- Follow up with developing a model for Advanced Persistent Threat prediction, on top of technique prediction (and their sequence)
- Analyse and determine the most influential logs and log entry types, allowing optimizations on the networks to best capture (potential) threats and attacks
- Repeat and improve models using genuine historic network data sets (also allowing to run MITRE detection) and records of detected attacks
- Research options for operationalizing prediction models as (near) real-time cyber threat alert support
- (from a scientific perspective) Look at alternative methods for dimensionality reduction and feature selection, improving the results potential.

Acknowledgment

The authors wish to thank Alberto Domingo and his colleagues at Allied Command Transformation for their support in applying Machine Learning techniques to the Cyber Exercise data set.

This work would not have been possible without the support of colleagues (past and present) across the NCI Agency who provided the environment and collected the data set used for post-exercise analysis, among many dedicated colleagues, Manisha Parmar deserves a special mention.

References

1 Michael R. Berthold, Nicolas Cebron, Fabian Dill, Thomas R. Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Christoph Sieb, Kilian Thiel, and Bernd Wiswedel, "KNIME: The Konstanz Information Miner," in *Studies in Classification, Data Analysis, and Knowledge Organization* (Springer, 2007), 319-326.

2 "Take on Data Science and Machine Learning Tools," *Gartner,* 2018, https://www.gartner.com/reviews/market/data-science-machine-learning-platforms.

3 "From Words to Wisdom," *KNIME Press*, 2018.

4 George Seif, "The 5 Clustering Algorithms Data Scientists Need to Know," 2018.

5 The MITRE Corporation, https://attack.mitre.org.

6 "Splunk," *Wikipedia*, 2019, https://en.wikipedia.org/wiki/Splunk.

7 Keinosuke Fukunaga, *Introduction to Statistical Pattern Recognition* (San Diego, CA: Academic Press Professional, 1990).

8 Laurens van der Maaten, Eric Postma, and Jaap van den Herik, "Dimensionality Reduction: A Comparative Review," Tilburg University, TiCC TR 2009–005, 2009.

9 Isabelle Guyon and André Elisseeff, "An Introduction to Variable and Feature Selection," *Journal of Machine Learning Research* 3 (2003): 1157-1182.

10 Jason Brownlee, "Gentle Introduction to Models for Sequence Prediction with RNNs," *Machine Learning Mastery*, July, 17, 2017, https://machinelearningmastery.com/models-sequence-prediction-recurrent-neural-networks/.

11 Fjodor van Veen, "Neural Networks," 2016, https://asimovintitute.org.

12 Michael J. Garbade, "Regression Versus Classification Machine Learning: What's the Difference?" 2018.

13 Cheng-Yuan Liou, Jau-Chi Huang, and Wen-Chie Yang, "Modeling Word Perception Using the Elman Network," *Neurocomputing* 71, no. 16-18 (2008): 3150-3157, https://doi.org/10.1016/j.neucom.2008.04.030.

14 Cheng-Yuan Liou, Wei-Chen Cheng, Jiun-Wei Liou, and Daw-Ran Liou, "Autoencoder for Words," *Neurocomputing* 139, no. 2 (September 2014): 84-96, https://doi.org/10.1016/j.neucom.2013.09.055.

15 Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning* (MIT Press, 2016).

16 Pascal Vincent and Hugom Larochelle, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion," *Journal of Machine Learning Research* 11 (2010): 3371–3408.

17 Diederik P. Kingma and Max Welling, "An Introduction to Variational Autoencoders," *arXiv:1906.02691* (2019).

18 Yoshua Bengio, "Learning Deep Architectures for AI," *Foundations and Trends® in Machine Learning* 2, no. 1 (2009): 1-127, https://doi.org/10.1561/2200000006.

19 Juergen Schmidhuber, "Deep Learning in Neural Networks: An Overview," *Neural Networks* 61 (2015): 85-117, https://doi.org/10.1016/j.neunet.2014.09.003.

20 Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, Evangelos Simoudis, Jiawei Han, and Usama Fayyad (eds.), "A density-based algorithm for discovering clusters in large spatial databases with noise," *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)* (AAAI Press, 1996).