University of Amsterdam

Research Project 1

Network Anomaly Detection in Modbus TCP Industrial Control Systems

February 9, 2020

Students:
Philipp Mieden
philipp.mieden@os3.nl

Rutger Beltman
rutger.beltman@os3.nl

Contents

Abstract

This research project evaluates an approach to network intrusion detection inside of a modern water treatment facility, using machine learning to model normal traffic behaviour and test the classification on 36 different recorded attacks. A deep neural network with sequential dense layers is compared to a configuration with Long Short Term Memory (LSTM) layers, performing classification on the packet-based network traces and alerting if an attack could be identified. We demonstrate that this approach can be used to identify malicious behaviour within industrial networks with a high success rate, indicate performance implications and discuss applicability and challenges in deployment.

keywords: industrial control systems, network security, machine learning

1 Introduction

Industrial Control Systems (ICS) are interconnected devices responsible for the monitoring, control and automation of industrial physical processes. As some of these systems are used for providing vital resources such as water and electricity, they are considered part of the national critical infrastructure. While it was common practice to deploy ICS on networks that were completely isolated from the internet, nowadays more and more ICS networks are interconnected, with or without knowledge of their operators, and can be found online with search engines such as SHODAN [10]. This trend of deploying interconnected IoT devices to automate the logistics and supply chain management is generally referred to as *Industry4.0* [14]. However, even systems that are correctly air-gapped can be attacked, as attacks such as STUXNET and the attack on the Ukrainian power grid have shown in the past [1] [3] [2], especially with regard to insider attacks. In these systems availability and safety is of utmost importance since failures can have a huge economic impact or endanger human lives. Computer security however, was not considered early in the development process of industrial devices, and due to a combination of complications when applying patches and the long lifetime of this equipment, many production systems are vulnerable to attacks. To protect such an infrastructure from cyber threats network segmentation and monitoring have to be correctly applied, in order to successfully isolate vulnerable devices.

1.1 Problem Description

Security research is usually not permitted on live systems due to the consequences it could have on its operational availability, even short downtimes are often impossible. A majority of technologies and protocols are proprietary and need to be reverse engineered for security research, due to the lack of source code or documentation. Common communication protocols in industrial networks are lacking authentication and encryption, which makes them an easy target for malicious activities. Some PLCs even crash when being pinged, which leads to passive monitoring being dominant in this area. [26] Blocking malicious behaviour in an automated fashion is often not considered, in order to not interrupt the manufacturing process, as the wrong decision of an intrusion prevention system could have fatal consequences. Rule based intrusion detection systems like SNORT are retrofitted to be usable in an ICS context. This creates drawbacks, either through loss of visibility due to limited parsing support for domain specific protocols, or being unable to describe complex chains of legitimate commands that can be used to manipulate the physical process, as outlined in [1]. While the approach of passive monitoring was the most adopted solution for the past years, the industry is now moving to a hybrid approach where data about process variables is actively queried from the involved Programmable Logic Controllers (PLCs). Some attacks however, cannot be detected only by passive network analysis, as a chain of valid commands can sometimes be used to create a dangerous state in production equipment [1]. Network anomaly detection has the potential to identify such changes in behavioural patterns, and has shown great success in related research [18].

1.2 Modbus TCP

The Modbus protocol is an industry standard that was developed by Modicon (now Schneider Electric) in 1979, and due to its free licensing terms is now used all over the globe in industrial facilities. Using Modbus, data from various sensors can be reported by the PLC to a supervisory control and data acquisition (SCADA) system, in order to be monitored by process engineers. To use this serial protocol for communications over TCP/IP networks, the Modbus TCP variant is used, which encapsulates the Modbus payloads in TCP packets. The protocol does not specify any form of authentication or encryption, and as a consequence it it vulnerable to man-in-the-middle attacks. Figure 1 displays the Modbus frame format when sent over TCP:

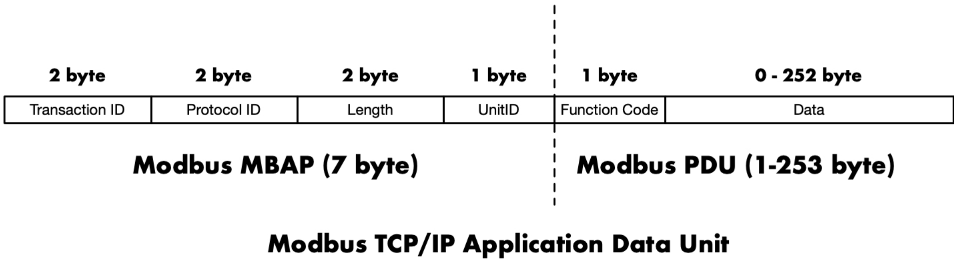


Figure 1: Modbus TCP frame format

1.3 Research Questions

We evaluated different anomaly detection algorithms to classify malicious behaviour in the context of an ICS network, in order to determine which one provides the best overall detection performance. Malicious behaviour is a broader term than just malware, as it also includes insider threats and other forms of process disruption, such as before mentioned combination of legitimate actions with malicious intent. The following research questions shall be answered during our research:

- How does malicious behaviour look like on an ICS network?
- How do the observed attributes differ from regular IT systems?
- Can machine learning based solutions help to identify such malicious behaviour based on multiple attack categories?

2 Related Work

Gonzalez et al. 2008 [11] developed an intrusion detection system (IDS) for passively monitoring network traffic of the Modbus protocol. The system works by inspecting the packet headers and payloads, and maintaining a table of states for the monitored PLCs and other devices. Their approach is based on the often static and predictable patterns in network traffic of industrial control systems and inspired us to investigate available behavioral approaches to intrusion detection in the context of industrial control systems.

Goh et al. 2016 [17] use the Secure Water Treatment testbed (SWaT) to generate a data set to support research that in done the in area of industrial control systems. The main objective was to produce a dataset, comprising network traffic recordings and device state information, obtained directly from a real-life industrial facility. The authors make an in-depth description of how the testbed is designed and what kind of data is collected. A total of 36 different attacks have been used with a thorough notation of where and when the attack has taken place. The in-depth documentation and extensive amount of provided traces, lead to the decision of evaluating this data set in our research project.

Kravchick et al. 2018 [18] propose an unsupervised deep learning approach to detect cyber attack in industrial control systems. This is done using convolutional neural networks. To analyse the performance of the algorithm the SWaT [17] data set was used. They found that a convolutional network that works over time data was able to successfully identify the majority of the 36 attacks in the SWaT data set. With the most effective model an F1 score of *0.775* was achieved, while performing analysis over all of the process stages. Classification involved only the physical part of the data set, therefore we decided to pick the network portion for our experiments.

Hijazi et al. 2019 [15] used a deep machine learning approach purely focused on Modbus/TCP traffic. The feature set includes features from the IP header, TCP header, and Modbus protocol. In their machine learning approach, they use multi-layer perceptrons together with binary classification. The authors chose to generate the data by creating a synthetic test on which attacks are performed. As a realistic test bed is important for drawing conclusions about the applicability of an approach, we chose the SWaT data set over creating a synthetic test bed ourselves.

3 Dataset

For our experiments, we chose the SWaT from the Singapore University of Technology and Design, iTrust institute [17]. It ensembles a modern water treatment facility with network segmentation and process monitoring. Raw water is converted to drinkable water in a 6 stage process, that involves mechanical filtering and chemical cleaning. The testbed and conducted attacks are extensively documented, and recordings for network and physical data are provided to researchers. Besides that, unmodified network captures in PCAP format are provided for parts of the data set.

3.1 Testbed

The test bed contains six different processing stages, through which the water will have to pass, until it is suitable again for human consumption. The authors of the data set defined four different attack categories, which we will use during our experiments. Stage 1 contains the raw water tank and supplies it to the system, stage 2 is responsible for chemical dosage, stage 3 performs ultra filtration, stage 4 is used for dechlorination, stage 5 performs reverse osmosis and stage 6 is taking care of RO permeate transfer and ultra filtration backwash [27]. Throughout the different stages, flow indicators and water level indicators provide a view on the current system state. Figure 2 shows a schematic of the test bed and the different stages:

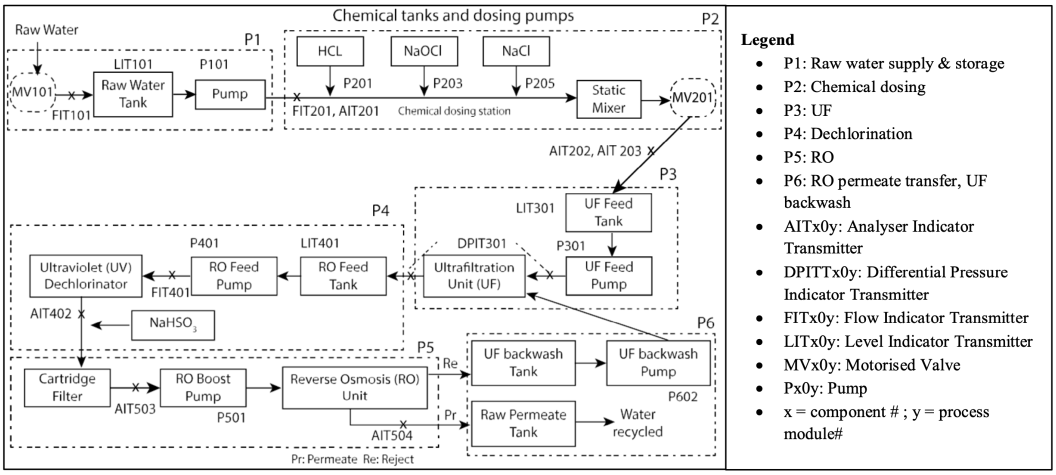


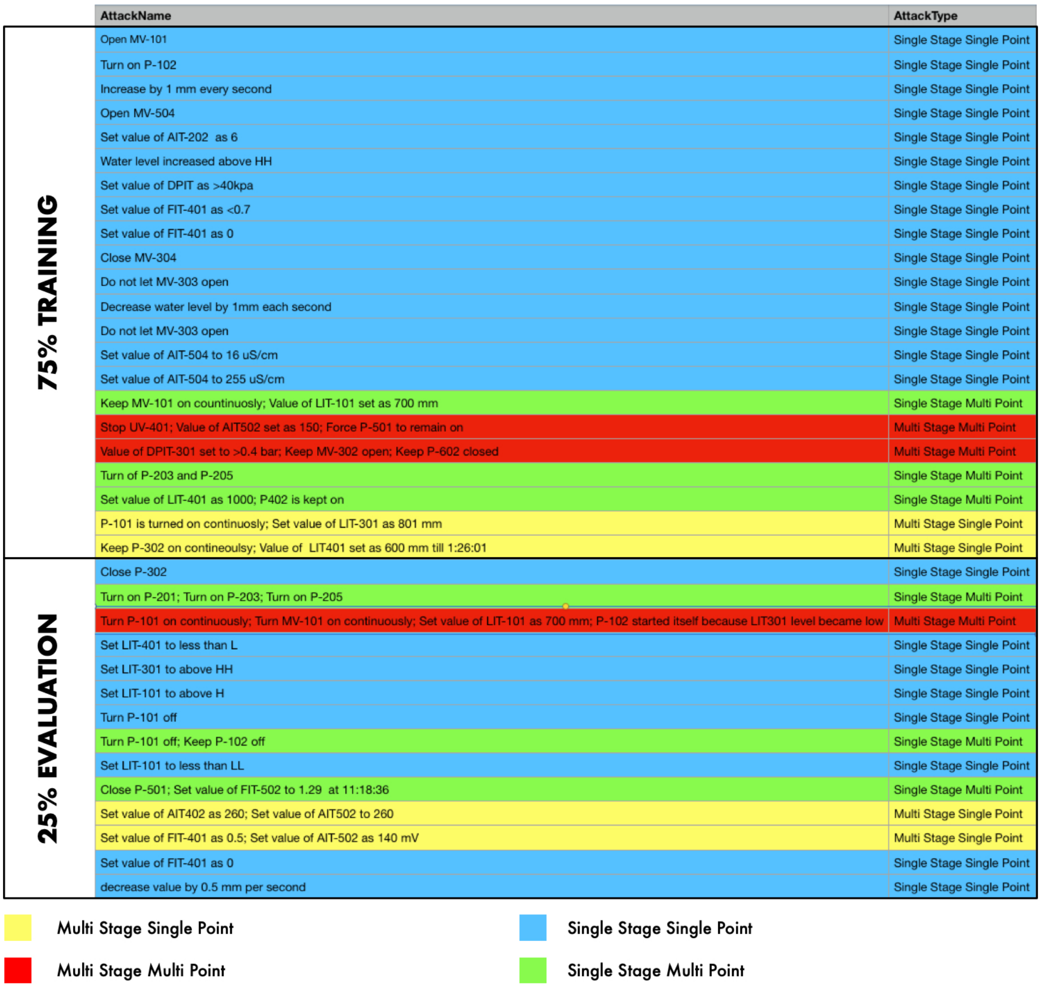
Figure 2: SWaT testbed schematic

3.2 Devices

- PLC: Programmable Logic Controller(s): for controlling valves and pumps, manufactured by Allen Bradley
- HMI: Human Management Interface(s): for displaying sensor values, such as water level or flow throughput indication
- Engineer Workstation: for configuring PLCs, running the Windows operating system (exact version unknown)
- Historian Server: for process monitoring of the physical sensor readings

3.3 Attack Distribution

In 2015 a collection of 36 different attacks have been recorded and data from the physical state and network traffic have been released to researchers to do experimentation for intrusion detection [17]. Documentation has been released for the 2015 attacks with all of the timings and locations of the attacks. The physical and network data from the 2015 attacks are provided in the CSV format. Figure 3 shows the attack distribution. The evaluation part of the data sets contains attacks that have not been seen during the training phase by the neural network. With this split we intend to test the ability of the neural network to learn the pattern of an attack class and whether it succeeds detecting a variant of the attack in another part of the system.



	AttackName	AttackType
75% TRAINING	Open MV-101	Single Stage Single Point
	Turn on P-102	Single Stage Single Point
	Increase by 1 mm every second	Single Stage Single Point
	Open MV-504	Single Stage Single Point
	Set value of AIT-202 as 6	Single Stage Single Point
	Water level increased above HH	Single Stage Single Point
	Set value of DPIT as >40kpa	Single Stage Single Point
	Set value of FIT-401 as <0.7	Single Stage Single Point
	Set value of FIT-401 as 0	Single Stage Single Point
	Close MV-304	Single Stage Single Point
	Do not let MV-303 open	Single Stage Single Point
	Decrease water level by 1mm each second	Single Stage Single Point
	Do not let MV-303 open	Single Stage Single Point
	Set value of AIT-504 to 16 uS/cm	Single Stage Single Point
	Set value of AIT-504 to 255 uS/cm	Single Stage Single Point
	Keep MV-101 on countinuosly; Value of LIT-101 set as 700 mm	Single Stage Multi Point
	Stop UV-401; Value of AIT502 set as 150; Force P-501 to remain on	Multi Stage Multi Point
	Value of DPIT-301 set to >0.4 bar; Keep MV-302 open; Keep P-602 closed	Multi Stage Multi Point
	Turn of P-203 and P-205	Single Stage Multi Point
	Set value of LIT-401 as 1000; P402 is kept on	Single Stage Multi Point
	P-101 is turned on continuosly; Set value of LIT-301 as 801 mm	Multi Stage Single Point
	Keep P-302 on contineoulsy; Value of LIT401 set as 600 mm till 1:26:01	Multi Stage Single Point
25% EVALUATION	Close P-302	Single Stage Single Point
	Turn on P-201; Turn on P-203; Turn on P-205	Single Stage Multi Point
	Turn P-101 on continuously; Turn MV-101 on continuously; Set value of LIT-101 as 700 mm; P-102 started itself because LIT301 level became low	Multi Stage Multi Point
	Set LIT-401 to less than L	Single Stage Single Point
	Set LIT-301 to above HH	Single Stage Single Point
	Set LIT-101 to above H	Single Stage Single Point
	Turn P-101 off	Single Stage Single Point
	Turn P-101 off; Keep P-102 off	Single Stage Multi Point
	Set LIT-101 to less than LL	Single Stage Single Point
	Close P-501; Set value of FIT-502 to 1.29 at 11:18:36	Single Stage Multi Point
	Set value of AIT402 as 260; Set value of AIT502 to 260	Multi Stage Single Point
	Set value of FIT-401 as 0.5; Set value of AIT-502 as 140 mV	Multi Stage Single Point
	Set value of FIT-401 as 0	Single Stage Single Point
	decrease value by 0.5 mm per second	Single Stage Single Point

🟨 Multi Stage Single Point　🟦 Single Stage Single Point

🟥 Multi Stage Multi Point　🟩 Single Stage Multi Point

Figure 3: Attack Distribution

3.4 Attack Scenarios

Figure 4 displays the different attack categories defined by the authors of the data set.

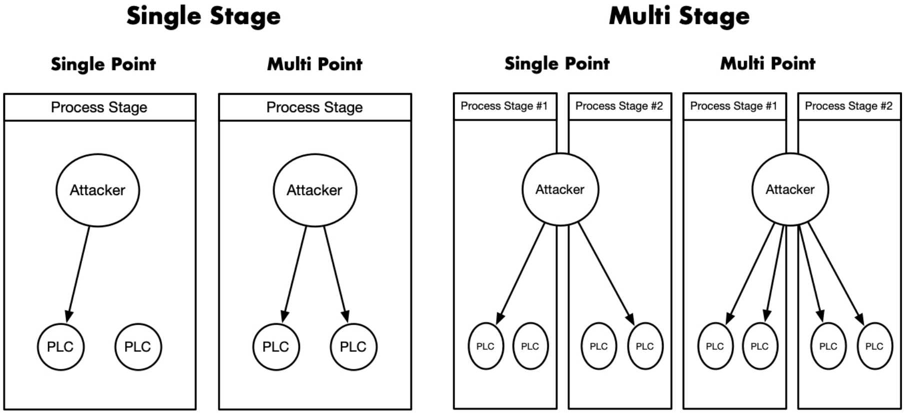


Figure 4: Attack Types

The following attack categories have been defined:

- Single Stage Single Point: a single process stage is being attacked, at a single point. For example a motorized valve is opened for too long, in order to cause a tank overflow
- Single Stage Multi Point: a single process stage is being attacked, at multiple points. For example a motorized valve is opened for too long, and the corresponding values displayed on the HMI are being manipulated to obscure the attack
- Multi Stage Single Point: multiple process stages are being attacked, each at a single point.
- Multi Stage Multi Point: multiple process stages are being attacked, each at multiple points

Attacked devices include the raw water inlet valve (MV-101), raw water level meter (LIT-101), raw water pumps (P-101, P-102), ultra filtration feed level meter (LIT-201), reverse osmosis feed level meter (LIT-401), reverse osmosis feed pumps (P-402), reverse osmosis pump (P-501) and more. Some devices named in the attack descriptions could not be located in the diagrams of the testbed technical architecture document.

The data set has an unbalanced distribution of normal and attack labels, as displayed in table 1. We will address this by choosing an appropriate metric for the evaluation.

Dataset	normal	SSSP	SSMP	MSSP	MSMP	rows	files
train	75.94%	20.68%	0.95%	1.90%	0.51%	24819975	50
eval	87.56%	7.92%	2.55%	0.57%	2.55%	7939824	16

Table 1: Dataset imbalance

3.5 Features

The provided data contains 19 features in total, of which 16 features have been used during our experiments. Features include IP address information, network interface names and flow direction (ingress, egress), protocol names, SCADA device tag, service name and port, Modbus function code and transaction ID. We did not use the provided hex encoded binary payload of the Modbus protocol for our experiments, as we intended to focus on behavioural aspects and beyond deep packet inspection. Table 2 lists all features used in our research and their data types.

Feature #	Feature Name	Type	Description
1	unixtime	numeric	UNIX timestamp
2	orig	categorical	origin IP address
3	type	categorical	record type
4	i/f_name	categorical	interface name
5	i/f_dir	categorical	flow direction
6	src	categorical	source ip address
7	dst	categorical	destination ip address
8	proto	categorical	protocol name
9	appi_name	categorical	application layer info
10	proxy_src_ip	categorical	proxy source ip
11	modbus_function_code	numeric	modbus protocol function code
12	modbus_function_description	categorical	description for modbus code
13	modbus_transaction_id	numeric	modbus transaction identifier
14	scada_tag	categorical	SCADA device name
15	service	numeric	service port
16	s_port	numeric	client port

Table 2: Features used in the experiments

4 Methodology

The following section outlines our methodology during the research. In order to identify attacks within an industrial system, we will evaluate a deep neural network with Long Short Term Memory (LSTM) layers, with network data recorded in the SWaT testbed.
We will measure the *performance* of the neural networks, by comparing the number of true positive alerts, with the number of false positives. Complementary, we will use the F1 Metric to score how effective the neural network is at identifying attacks against the physical infrastructure.

4.1 Attack Aggregation

In order to apply the attack scenario labels to the network CSV data, additional information about the attacked endpoints was required. We collected the network addresses of the devices involved in an attack from the technical architecture document [27], and created logic to label the records inside the provided CSV data if they are part of an attack. The devices of interest during an attack are the primary PLC and the primary Remote I/O (RIO) PLC, both communicate over the Ethernet/IP protocol.

4.2 Data Preprocessing

Figure 5 displays the processing pipeline for the provided network data. First, various typos had to be fixed across the entire data set as well as a column with missing data was found

(Referrer_self_uid). Since this column was not present on a majority of the data we decided to remove it as a feature. After cleaning, the data set was analyzed again to determine the distribution of values and calculate mean and standard derivation for all numeric columns. For columns with categorical data, all unique values have been collected and indexed, to allow experiments with different encoding and normalization strategies. In the labeling phase, the previously aggregated attack information CSV is loaded and attacks are mapped to the CSV records. During this phase, the data points are encoded if necessary and normalized.

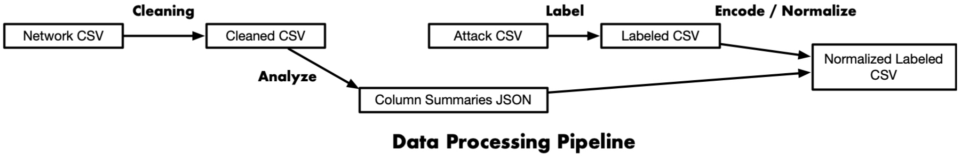


Figure 5: Processing Pipeline

4.3 Labeling

Labeling is done by using the aggregated attack information, that contains start and end time of an attack, and the affected devices IP addresses. Bidirectional communication between an IP address of an affected device during the timeframe of an attack is considered an instance of it. After the first match for an attack the logic stops, that means no collection of labels is taking place.

4.4 Feature encoding

Deep Neural Networks can only operate with numerical values, therefore categorical data needs to be transformed prior to the experiments. Additionally, the numeric data needs to be normalized, to prevent a bias on features than have a larger range of values [22].

4.4.1 Categorical Values

For categorical values the one-hot encoding strategy is used, which translates each unique categorical into a separate column that is set to one for each record that is part of the category. This increases the input dimension of the DNN, which can lead to problems further down in the analytical process [28] [29].

4.4.2 Numerical Values

For numeric values, the *z-score* function will be applied to the data set. This function first subtracts the mean from all of the variables and then divides it by the standard deviation from the mean.

$$f(X) = \frac{X - \mu}{\sigma}$$

Alternatively the *minmax* approach is used to normalize the data between o and 1. This makes the data usable with the *relu* activation function, as it treats all negative values as zero, and therefore ignores many values when using *z-score*. To allow working with *z-score* encoded values, the *LeakyReLU* activation must be used. The following displays the formula for *minmax* encoding:

$$f(X) = \frac{x - min}{max - min}$$

4.5 Deep Neural Network

Initially, the extracted features will be fed to two different Deep Neural Network (DNN) types, utilizing a network with sequential layers. As this type of supervised classification has proven to be effective in similar research [15] [16], we aim to establish a baseline with our results from this experiment, in order to compare it with a Neural Network layer type known as Long Short Term Memory (LSTM). LSTMs have been developed to work with time correlated data streams, since normal DNN layer types can not represent this kind of relation. To test the effectiveness of this system design, experiments with both neural network types will be compared.

4.5.1 Sequential Dense Layers

Neural networks with sequential dense layers are known to not be able to work well with time series data [6]. This is due to the fact that they have no memory to be able to correlate sequential events.

The core component of a neural network are the neuron itselves, also referred to as nodes of the network. The first part of a neuron are the inputs. All inputs are multiplied with a weight that is unique. The outcomes of all of the multiplications are then summed up together and provided as an input for a so called "activation function". The final output of the neuron can be provided as input for the next layer.

A deep neural network is a neural network where there are multiple hidden layers connected to each other. The first layer is the input layer, it's main responsibility is to provide the input data to the subsequent hidden layers. The hidden layers are filled with multiple rows of neurons. These hidden layers make predictions of what type of input is given. Finally the hidden neurons forward their output to the output layer. In this layer a prediction is made for the type of information from the input. To correct the weights within the neurons back propagation is used. This is a phase where the weights within the models are adjusted to better learn how to predict data [22].

4.5.2 Long Short Term Memory Layers

To understand LSTM architectures we have to look at recurrent neural networks first. With Recurrent neural networks output from previous step is given as an input for the next step. This allows the model to make predictions on what information it expects next [8]. A LSTM works with memory cells instead of neurons as seen in regular deep neural networks. Each of these memory cells has a lane that allows information to be preserved across many memory cells. Those mnemonic capabilities are the reason for the long term memory of this layer type.

4.5.3 Classification

Binary classification is a scheme where objects are classified between two different classes. With such a classification type traffic would be distinguished to be either normal or abnormal. With multi class classification more advanced classification becomes possible. Rather than only classifying between normal and abnormal it becomes possible to classify the type of abnormality as discussed in figure 4. The advantage of this classification type is that an alert generated contains more information on want kind of abnormality or attack is happening, potentially providing further information to the human analyst and reducing incident response time. With binary classification the only information about the attack is the input vector.

4.5.4 Experiment Configuration

For the configuration of the DNN we followed the best practices for an imbalanced data set with lots of categorical data, as discussed by Bhattacharyya et al. in [22] and Hossain in [24]. We used the *categorical_crossentropy* loss function for mutli-class classification, in combination with the *softmax* activation function on the final layer. For binary classification, the *binary_crossentropy* function was used, together with the *sigmoid* output layer activation. The *leakyrelu* activation was configured with an alpha of *0.3*. When using dropout layers, one dropout layer is inserted after each sequential dense layer, with a rate of *0.5*, except for the first dropout layer which has a rate of *0.8*. Table 3 displays the parameters for different experiments. Each DNN has a dense input layer, followed by the first wrapping layer, the number of core layers configured, the final wrap layer and the output layer. For the number of neuron per layer as well as the input vector size for the neural network, we chose values from the geometric progression of 2.

#	Wrap Neurons	Core Neurons	Core Layers	Activation	Optimizer	Dropout
0	2	4	1	relu	adam	false
1	2	4	1	relu	sgd	false
2	2	4	1	relu	adam	true
3	8	32	1	relu	sgd	true
4	8	32	1	leakyrelu	adam	true
5	16	64	1	relu	sgd	true
6	16	64	1	relu	adam	false
7	16	64	3	relu	sgd	false
8	16	64	3	relu	adam	true
9	8	32	3	relu	sgd	true
10	8	32	3	leakyrelu	adam	false

Table 3: Experiment DNN configuration

Each experiment configuration was tested using binary classification and multi-class classification an. Training was either performed on single file with 500000 records and evaluation on a different file with the same number of records, or training was run on 50 files and 16 different files have been used for evaluation. The experiments have been tested during development with a total number of epochs ranging from *3 - 50*. Due to time constraints the experiments in the evaluation used epochs ranging from *3 - 10*.

5 Evaluation

With the size of the training set exceeding the memory capacity of our server we will need to batch the files up as shown in figure 6. For every epoch we load a set of files at a time into the model. When the model is done training on the file batch we load in the next set of files and train on those. This process is repeated until all data from all files in the training set has passed through the DNN. We use Tensorflow version 2.1.0 with Keras to create the deep neural networks, our servers for executing the experiments are a Dell PowerEdge R240 with 8GiB DIMM DDR4 2666 MHz, Intel(R) Xeon(R) E-2124 CPU @ 3.30GHz and a Dell PowerEdge R230 with 8GiB DIMM DDR4 2133 MHz and Intel(R) Xeon(R) CPU E3-1240L v5 @ 2.10GHz. Both servers are running Linux x86_64 ubuntu xenial with kernel version 4.15.0-74-generic.

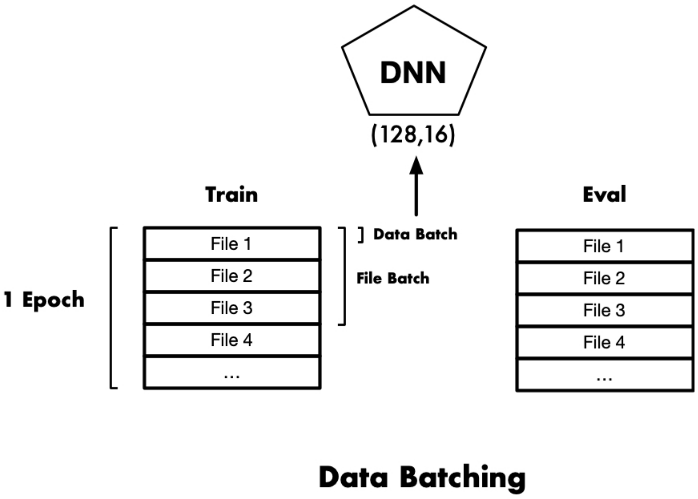


Figure 6: Data Batching

The evaluation of the models prediction performance on the evaluation set is done similarly to the procedure of feeding records into the training phase. First a set of attacks that have never been seen by the neural network are loaded and split in batches. Metrics are collected in multiple confusion matrices. After the whole evaluation set has been processed, the confusion matrices are aggregated and used to calculate precision, recall and the F1 score.

5.1 Metrics

In order to understand the metrics it is neccessary to understand the four different classification types that could occur in the evaluation as show in table 4. A true positive is a classification that has been correctly labeled as an attack or abnormality, while a true negative is normal traffic that has been correctly classified as normal. A false positive happens when normal traffic would get incorrectly classified as an attack. Finally, a false negative is an attack that should have been classified as such, but the prediction model classified it as normal.

		actual value	
		attack	normal
predicted value	attack	true positive	false positive
	normal	false negative	true negative

Table 4: Explanation of all possible classes.

Due to the imbalanced nature of the data set, as shown in table 1. accuracy is not a suitable metric for evaluating the performance of the DNN. Instead, recall and precision can be used to describe the the performance of the resulting prediction model more precisely.
For recall we look at the balance between true positives and false negatives using the following formula:

$$recall = \frac{true\,positives}{true\,positives + false\,negatives}$$

Recall results in the percentage of predictions that have been correctly classified by the neural network.
Precision is calculated by taking the amount of true positives and comparing it to the amount of false positives:

$$precision = \frac{true\,positives}{true\,positives + false\,positives}$$

For the evaluation of our experiments, false positives, true positives and false negatives will be taken into account to calculate the precision and recall metrics. The true negatives are not important for abnormality prediction models as this is normal traffic that has been correctly classified as normal. The F1 score is a metric that combines takes the harmonic mean between the precision and recall. The harmonic mean is a suitable metric for unbalanced data sets, because it penalizes the model when either the recall or precision is low [7].

$$F1\,score = 2 * \frac{recall * precision}{recall + precision}$$

The loss function in neural networks is used to evaluate the current performance of the model. The loss function takes multiple aspects of the model and create a number that represents how well the model is performing. A result close to zero indicates that the model is performing better [9]. During early stopping the loss value is checked after each full epoch, and the training is stopped if the loss becomes smaller than *0.001*, in order to avoid overfitting the model.

5.2 Results

Several model configurations with smaller network sizes did not succeed in identifying anomalies in the evaluation data, those models predicted exclusively a single class: normal. These models did not show any decrease in the loss over several epochs, and had a loss between *0.65 - 0.71.* Successful training phases have shown a decreasing loss over time, an example of this for single file training runs of the LSTM layer variant are displayed in figure 7. LSTM v6 has shown a very low loss value, and therefore was early stopped after the third epoch. LSTM v7 was only trained for 3 epochs, before being used for validation. We observed that for an input vector length of less than 128 records, the neural network would never produce a model that could identify the attack class and assume this is related to the imbalanced nature of the data set. The loss development shows that the model is capable of learning from the training data, most likely the varying prediction results are related to the relatively low amount of epochs.

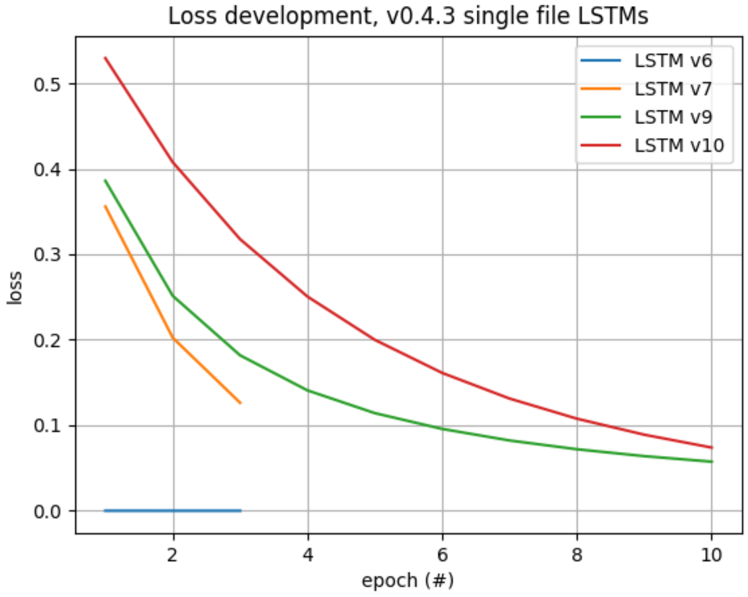


Figure 7: Loss development over multiple epochs, LSTMs

In our experiments we discovered that it took on average longer to the LSTM on the full data sets than the DNN. Figure 8 shows this observations. What should be noted is that the average is created as an average over multiple different models, with different network complexities and configurations. Therefore these numbers should be treated with a grain of salt.

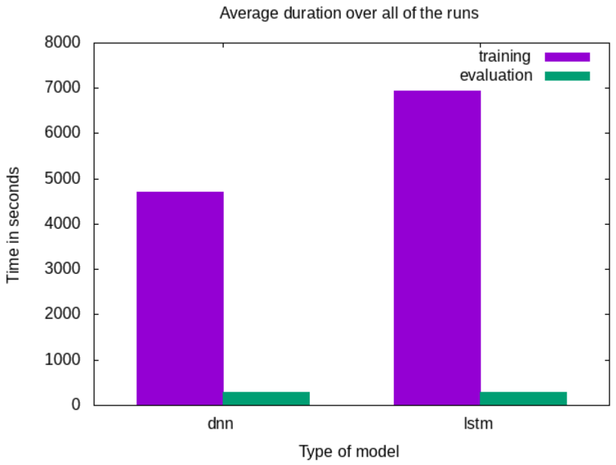


Figure 8: Average duration over all experiment runs

5.2.1 Sequential Dense Layers

The Deep Neural Network with sequential dense layers performed overall faster and consumed slightly less memory, most likely due to the fact that the timestamp feature column was not used and the model did not memorize previous information. Configurations that gave predictions other than exclusively zero are listed in table 5.

Experiment #	Attack Type	Precision	Recall	F1 Score
4	SSSP	0.053	0.415	0.094
6	SSSP	0.579	1.000	0.733

Table 5: Average training and evaluation times over all different runs

5.2.2 LSTM Layers

The LSTM layers have shown a more varying classification performance, compared to the sequential dense layers. This comes at a price of increased time for training and evaluation. The best observed model is comparable to the sequential dense layers variant. Configurations that gave predictions other than exclusively zero are listed in table 6.

Experiment #	Attack Type	Precision	Recall	F1 Score
0	SSSP	0.578	0.995	0.731
3	SSSP	0.036	0.267	0.063
6	SSSP	0.111	0.009	0.016
9	MSMP	0.060	0.583	0.108
9	MSSP	0.013	0.441	0.025
9	SSSP	0.080	1.000	0.148

Table 6: Results LSTM multi-class

6 Conclusion

We conclude that Deep Neural Networks are applicable to network of intrusion detection in industrial control systems, but require careful configuration and adaption to the operational environment. In the modern industrial world, we have different priorities, but share similar technologies. Based on our analysis of the 2019 network traces, we conclude that the anatomy of an intrusion is identical to the corporate world, in terms of workstation infection, lateral movement and reconnaissance behaviour. Network intrusion detection provides data for incident and forensic analysis, and anomaly detection has made its way into the industry [25] [19]. Common Network Intrusion Detection Systems can be deployed, such as Snort, Suricata and Bro / Zeek [23], but need to extend parsing support for ICS protocols, such as Modbus, Ethernet/IP and the Common Industrial Protocol (CIP). For rule-based intrusion detection systems several industrial rule sets exist [20] [21]. The LSTM layer type seems to be applicable based on the results from the data set, although we observed an increased learning and evaluation time of 148% on average in our experiments. Multi-class classification for attack types is difficult and might confuse the DNN if not sufficient patterns can be found in the provided data. The data set should therefore contain sufficient amount of well suited training data. Detecting an intruder in his early stages of lateral movement and reconnaissance can prevent further damage to industrial systems. As these systems are highly complex and in-depth knowledge about them is required to cause lasting damage, this phase is often longer than in the corporate world. Although changes can also be detected based on measurements from the physical sensor data in the historian, we argue that if changes occur on this level it is too late, because damage was already done to the physical process. A solid network monitoring approach is therefore key to discover anomalies, that can unveil the behaviour of an attacker in his early stages.

7 Discussion

A challenge in using DNNs in the context of network intrusion detection is that their internals are a black box for an analyst, and it is therefore not clear how a decision came to place. Ensemble learning methods can provide increased decision transparency, due to their expert voting based model. Furthermore, having the DNN unlearn something requires fine grained check pointing during the training phase, and precise knowledge of the training data and structure. We expect the configuration of the neural network to be highly specific to the target environment, and that it possibly requires updates on major traffic pattern changes or after installation of new equipment. The hyper parameter configuration was based on best practices, but should be subject to further research. Due to time constraints we could only choose small DNN sizes, with less than 100 neurons per layer. It would be interesting to see if a larger and more complex network performs better, and if it is worth the increased processing time. Other optimizers besides *adam* and *sgd* should be considered as well. The *ReLU* and *LeakyReLU* activation functions did seem to perform well for our use-case, but also here different alternatives should be evaluated. It important to remember, that not every anomaly is an attack, and that attacks may affect normal system behaviour, which will lead to more anomalies during consecutive operation. Although no perfect F1 score could be achieved, we argue that the presented anomaly detection mechanism can provide value for the protection of such a facility. This is because at the network level, an attack and caused communication can range from several to multiple thousand packets, and even when detecting only parts of a malicious stream as anomalous, it can reveal the presence of an intruder to the Security Operation Center. Further work needs to done however to determine whether the high data volume from packet-based records is suitable for a DNN, or if the use of summary structures, such as flows or specific events delivers more accurate results and produces less noise. Machine learning is not a silver bullet and will always require human supervision to judge over generated alerts and take appropriate action.

8 Future Work

As Deep Packet Inspection (DPI) can reveal further patterns in the input data, future work will include the use of the Modbus payload data for feature engineering techniques, such as Principal Component Analysis. The effectiveness of the proposed solution will be evaluated also on the remaining parts of the data set. A comparison to unsupervised methods in terms of training duration and prediction performance should also be considered. Each experiment should be executed multiple times to establish a baseline performance and variance of the model stability. The experiments should be repeated with a higher number of epochs, to allow more time for the deep neural network to recognize patterns in the data.

Glossary

Deep Neural Network A set of algorithms designed to recognize patterns, inspired by the human brain. A deep neural network is an artificial neural network (ANN) with multiple layers between the input and output layers.

Demilitarized Zone In computer security, a DMZ or demilitarized zone (sometimes referred to as a perimeter network) is a physical or logical subnetwork that contains and exposes an organization's external-facing services to an untrusted network, usually a larger network such as the internet.

Machine Learning The science of training computers to learn and behave like humans. By supplying data and information in the form of observations, the goal is to improve the learning process over time in autonomous fashion.

Acronyms

HTTP Hypertext Transfer Protocol

DNS Domain Name System

TCP Transmission Control Protocol

UDP User Datagram Protocol

IP Internet Protocol

TTL Time To Live

TLS Transport Layer Security

SSL Secure Socket Layer

NAT Network Address Translation

VPN Virtual Private Network

IMAP Internet Message Access Protocol

ICMP Internet Control Message Protocol

JSON Javascript Object Notation

SWaT Secure Water Treatment

GPU Graphics Processing Unit

ICS Industrial Control Systems

PLC Programmable Logic Controller

HMI Human Management Interface

CSV Comma-separated values

AV Anti Virus

HTTPS Hypertext Transfer Protocol Secure

DMZ Demilitarized Zone

PCAP Packet Capture

PCAPNG Packet Capture Next Generation

OS Operating System

DPI Deep Packet Inspection

IPS Intrusion Prevention System

IDS Intrusion Detection System

SOC Security Operation Centers

IoC Indicators Of Compromise

SIEM Security Information and Event Management

IPFIX Internet Protocol Flow Information Export

NSM Network Security Monitoring

BPF Berkeley Packet Filter

RFC Request For Comments

SVD Singular Value Decomposition

PCA Principal Component Analysis

DNN Deep Neural Network

ML Machine Learning

AI Artificial Intelligence

ANN Artificial Neural Network

HMM Hidden Markov Models

GA Genetic Algorithms

GP Genetic Programming

SVM Support Vector Machines

LSTM Long Short Term Memory

DoS Denial Of Service

DDoS Distributed Denial Of Service

MitM Man in the Middle

List of Figures

List of Tables

References

[1] W. Jardine, S. Frey, B. Green, and A. Rashid, "Senami: Selective non-invasive active monitoring for ics intrusion detection," in *Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy*, ser. CPS-SPC '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 23–34. [Online]. Available: https://doi.org/10.1145/2994487.2994496

[2] D. U. Case, "Analysis of the cyber attack on the ukrainian power grid," *Electricity Information Sharing and Analysis Center (E-ISAC)*, 2016.

[3] R. Langner. (2013) To kill a centrifuge. [Online]. Available: https://www.langner.com/wp-content/uploads/2017/03/to-kill-a-centrifuge.pdf

[4] W. S. (2002) ai-faq/neural-nets/part2. [Online]. Available: http://www.faqs.org/faqs/ai-faq/neural-nets/part2/

[5] P. Mieden. (2019) Netcap - a framework for secure and scalable network traffic analysis. [Online]. Available: https://github.com/dreadl0ck/netcap

[6] S. LABS. (2019) Understanding deep learning: Dnn, rnn, lstm, cnn and r-cnn. [Online]. Available: https://medium.com/@sprhlabs/understanding-deep-learning-dnn-rnn-lstm-cnn-and-r-cnn-6602ed94dbff

[7] S. Sateesh. (2018) Have you asked why f1-score is a harmonic mean(hm) of precision and recall. [Online]. Available: https://medium.com/@srinivas.sateesh/have-you-asked-why-f1-score-is-a-harmonic-mean-hm-of-precision-and-recall-febc233ce247

[8] N. Kumar. (2019) Recurrent neural networks (rnn) explained — the eli5 way. [Online]. Available: https://towardsdatascience.com/recurrent-neural-networks-rnn-explained-the-eli5-way-3956887e8b75

[9] J. Brownlee. (2019) Loss and loss functions for training deep learning neural networks. [Online]. Available: https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/

[10] "Shodan is the world's first search engine for internet-connected devices." https://www.shodan.io/, note = Accessed: 8-1-2020.

[11] J. Gonzalez and M. Papa, "Passive scanning in modbus networks," in *Critical Infrastructure Protection*, E. Goetz and S. Shenoi, Eds. Boston, MA: Springer US, 2008, pp. 175–187.

[12] M. Caselli, E. Zambon, and F. Kargl, "Sequence-aware intrusion detection in industrial control systems," in *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*, ser. CPSS '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 13–24. [Online]. Available: https://doi.org/10.1145/2732198.2732200

[13] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[14] A. Rojko, "Industry 4.0 concept: Background and overview," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 11, p. 77, 07 2017.

[15] A. Hijazi and J.-M. Flaus, "A deep learning approach for intrusion detection system in industry network," 02 2019.

[16] O. Linda, T. Vollmer, and M. Manic, "Neural network based intrusion detection system for critical infrastructures," 06 2009, pp. 1827–1834.

[17] J. Goh, S. Adepu, K. Junejo, and A. Mathur, "A dataset to support research in the design of secure water treatment systems," 10 2016.

[18] M. Kravchik and A. Shabtai, "Detecting cyber attacks in industrial control systems using convolutional neural networks," in *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*. ACM, 2018, pp. 72–83.

[19] M. James, P. Michael, S. Keith, T. CheeYee, Z. Timothy, B. William, O. Titilayo, W. Devin, and W. Johnathan. (2018) Securing manufacturing industrial control systems: Behavioral anomaly detection. [Online]. Available: https://www.nccoe.nist.gov/sites/default/files/library/mf-ics-nistir-8219.pdf

[20] (2020) Proofpoint emerging threats pro ruleset. [Online]. Available: https://www.proofpoint.com/us/threat-insight/et-pro-ruleset

[21] (2020) Quickdraw snort ruleset. [Online]. Available: https://github.com/digitalbond/Quickdraw-Snort

[22] D. K. Bhattacharyya and J. K. Kalita, *Network Anomaly Detection: A Machine Learning Perspective*, 1st ed. Boca Raton, Lodon, New York: CRC Press, 2014.

[23] R. Bejtlich, *The Practice Of Network Security Monitoring - Understanding Incident Detection and Response*, 6th ed. San Francisco: No Starch Press, 2013.

[24] M. Hossain, *Intrusion Detection with Artificial Neural Networks*, 1st ed. Saarbrücken: Lambert Academic Publishing, 2009.

[25] M. Collins, *Network Security Through Data Analysis - From Data to Action*, 2nd ed. Sebastopol: O'Reilly, 2017.

[26] K. Coffey, R. Smith, L. Maglaras, and H. Janicke, "Vulnerability analysis of network scanning on scada systems," *Security and Communication Networks*, vol. 2018, 02 2018.

[27] (2018) itrust swat technical details document. [Online]. Available: https://itrust.sutd.edu.sg/wp-content/uploads/sites/3/2018/10/SWaT_technical_details-051018-v4.2.pdf

[28] N. Venkat, "The curse of dimensionality: Inside out," 09 2018.

[29] M. Verleysen and D. François, "The curse of dimensionality in data mining and time series prediction," vol. 3512, 06 2005, pp. 758–770.