A Deep Learning Approach for Intrusion Detection System in Industry Network

Ahmad HIJAZI
Univ.Grenoble Alpes, G-SCOP,
F-38000 Grenoble, France
ahd.hjz@gmail.com

EL Abed EL SAFADI
Univ.Grenoble Alpes, G-SCOP,
F-38000 Grenoble, France
Abed.safadi@grenoble-inp.fr

Jean-Marie FLAUS
Univ.Grenoble Alpes, G-SCOP,
F-38000 Grenoble, France
Jean-marie.Flaus@grenoble-inp.fr

***Abstract*— Network has brought convenience to the world by allowing flexible transformation of data, but it also exposes a high number of vulnerabilities. A Network Intrusion Detection System (NIDS) helps system and network administrators to detect network security breaches in their organizations. Identifying anonymous and new attacks is one of the main challenges in IDSs researches. Deep learning (2010's), which is a subfield of machine learning (1980's), is concerned with algorithms that are based on the structure and function of brain called artificial neural networks. The progression on such learning algorithms may improve the functionality of IDS especially in Industrial Control Systems to increase its detection rate on unknown attacks. In this work, we propose a deep learning approach to implement an effective and enhanced IDS for securing industrial network.**

Keywords—Intrusion Detection System, Deep Learning, SCADA, Modbus, Industrial Control Systems, Artificial Neural Networks.

I. INTRODUCTION

Targeted attacks on industrial control systems are the biggest threat to critical national infrastructure, says Kaspersky Lab. Today's industrial control systems (ICS) face an array of digital threats. Two in particular stand out. On the one hand, digital attackers are increasingly targeting and succeeding in gaining unauthorized access to industrial organizations. Some actors use malware, while others resort to spear-phishing (or whaling) and other social engineering techniques [1]. The main challenge is linked to the fact these systems typically control physical processes that relate to power, transport, water, gas and other critical infrastructure. Because the output of ICS relates to physical processes, the effects of any downtime – such as a power outage – can affect millions of people [2].

Signature-based and anomaly-based Intrusion Detection System is one aspect of an effective network security monitoring strategy. Very few asset owners have IDS/IPS deployed and configured appropriately at the boundary between the Enterprise IT and ICS networks.
However, network intrusion detection has been criticized for its propensity to generate a perceived large amount of false positives and false negatives. Signature-based IDS lacks the capability of detecting new forms of attacks that it had not seen before, and anomaly based produces high amount of false positive rates, in addition to that it is difficult to select normal behavior of traffic dataset in the network.

Various machine learning techniques have been used to develop NIDSs, such as Articial Neural Networks (ANN), Support VectorMachines (SVM), Naive-Bayesian (NB), Random Forests (RF), Self-Organized Maps (SOM), etc. The NIDSs are developed as classifiers to differentiate the normal traffic from the anomalous traffic [3].

In this paper, an intrusion detection system using the deep learning is proposed to secure the ICS network. The proposed technique uses multi-layer perceptron with binary classification and trains high-dimensional Modbus packet data after a network simulation and label the data with normal and malicious in order to the neural network to understand the underlining structure of the normal and anomalous behavior of the network.

II.ICS AND IDS

A. ICS overview

Industrial control system (ICS) is a general term that encompasses several types of control systems, including supervisory control and data acquisition (SCADA) systems, distributed control systems (DCS), and other control system configurations such as Programmable Logic Controllers (PLC) often found in the industrial sectors and critical infrastructures.

ICS have different performance and reliability requirements, and also use operating systems and applications that may be considered unconventional in a typical IT network environment. Security protections must be implemented in a way that maintains system integrity during normal operations as well as during times of cyber-attack.

A typical ICS contains numerous control loops, human interfaces, and remote diagnostics and maintenance tools built using an array of network protocols on layered network architectures. A control loop utilizes sensors, actuators, and controllers (e.g., PLCs) to manipulate some controlled process. A sensor is a device that produces a measurement of some physical property and then sends this information as controlled variables to the controller. The controller interprets

the signals and generates corresponding manipulated variables, based on a control algorithm and target set points, which it transmits to the actuators.

Industrial control systems underpin the critical national infrastructure and are essential for the success of industries such as:

- Electricity production and distribution
- Water supply and treatment
- Food production
- Oil and gas production and supply
- Chemical and pharmaceutical production
- Telecommunications
- Manufacturing of components and finished products
- Paper and pulp production [5].

SCADA and industrial protocols, such as Modbus/TCP, are critical for communications to most control devices. Unfortunately, many of these protocols were designed without security built in and do not typically require any authentication to remotely execute commands on a control device.

B. IDS for ICS

For a long time, ICS/SCADA was an area that relied on different embedded devices and clear-text communications such as Modbus/TCP, without taking into consideration the security approach which made it vulnerable to different types of attacks and it becomes a target of cyber threats. This resulted in a new focus on the security issues related to industrial control systems.

Intrusion Detection System are capable of providing visibility and detection of any breach on the network, IDS can alarm in response to network security or endpoint security events.
IDSs for ICT networks have become very popular; especially for identifying the signatures of many pieces of known malicious code (e.g. SNORT rules), other IDSs utilize model-base anomaly detectors. Modern ICS equipment does not normally fall in the same category as computer systems in modern-day ICT networks. ICS equipment is not typically designed with security logging and processing in mind. It does not usually run standard operating systems used in ICT desktops and servers. Network-based IDSs are a network device that collects network traffic directly from the network, often from a central point such as a router or switch. Data from multiple network sensors can be aggregated into a central processing engine, or processing may occur on the collection machine itself. The network traffic can also be analyzed for unsatisfactory traffic or behavior patterns; either patterns that are anomalous to a previously established traffic or behavior model, or specific traffic patterns that display non-conformity to standards, e.g. violations of specific communication protocols.

C. Deep learning and IDS

Signature based IDS is effective in the detection of known attacks and results in a high detection accuracy and less false-alarm rates. However, its performance suffers during detection of unknown or new attacks due to the limitation of rules that can be installed beforehand in an IDS. On the other hand, anomaly based IDS, is well-suited for the detection of unknown and new attacks. Although Anomaly Detection IDS produces high false-positive rates, its theoretical potential in the identification of new attacks has caused its wide acceptance among the research community. There are primarily two challenges that arise while developing an effective and flexible NIDS for the unknown future attacks. First, proper feature selections from the network traffic dataset for anomaly detection is difficult. As attack scenarios are continuously changing and evolving, the features selected for one class of attack may not work well for other classes of attacks. Second, unavailability of labeled traffic dataset from real networks for developing an NIDS.

Deep learning belongs to a class of machine learning methods, where employs consecutive layers of information-processing stages in hierarchical manners for pattern classification and feature or representation learning. Usually deep learning plays the important role in image classification results. In addition, deep learning is also commonly used for language, graphical modeling, pattern recognition, speech, audio, image, video, natural language and signal processing. There are many deep learning methods such as Deep Belief Network (DBN), Restricted Boltzman Machine (RBM), Deep Boltzman Machine (DBM), Deep Neural Network (DNN), Auto Encoder, Deep / stacked Auto Encoder, etc… [6].

The advancements on learning algorithms might improve IDS ability to reach higher detection rate and lower false alarm rate. It is envisioned that the deep learning based approaches can help to overcome the challenges of developing an effective NIDS.

In this work, we will use Multi-layer Perceptrons with binary classification which we found the most useful type of neural network where the only two output classes will be normal and malicious ones. A Perceptron is a single neuron model that was a precursor to larger neural networks.

The power of neural networks come from their ability to learn the representation in your training data and how to best relate it to the output variable that you want to predict. In this sense neural networks learn a mapping. Mathematically, they are capable of learning any mapping function and have been proven to be a universal approximation algorithm. The data structure can pick out (learn to represent) features at different scales or resolutions and combine them into higher-order features. For example from lines, to collections of lines to shapes.

III. APPLICATION OF DEEP LEARNING ALGORTHM TO NETWORK TRAFFIC

The steps for building a good deep learning approach consists of preparing the data, defining and compiling the model, fitting the model, and evaluation (prediction) the model. We

will start with a brief overview concerning the deep learning structure.

A. Overview of deep neural networks

1) Neurons

The building block for neural networks are artificial neurons. These are simple computational units that have weighted input signals and produce an output signal using an activation function.

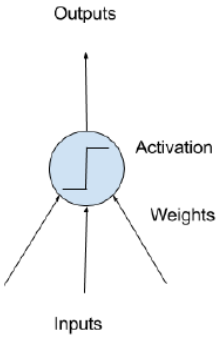


Fig. 1. Model of a Simple Neuron

2) Neuron Weights

Each neuron has a bias which can be thought of as an input that always has the value 1.0 and it too must be weighted. For example, a neuron may have two inputs in which case it requires three weights. One for each input and one for the bias. Weights are often initialized to small random values, such as values in the range 0 to 0.3, although more complex initialization schemes can be used. Like linear regression, larger weights indicate increased complexity and fragility of the model. It is desirable to keep weights in the network small and regularization techniques can be used.

3) Activation

The weighted inputs are summed and passed through an activation function, sometimes called a transfer function. An activation function is a simple mapping of summed weighted input to the output of the neuron. It is called an activation function because it governs the threshold at which the neuron is activated and the strength of the output signal. Historically simple step activation functions were used where if the summed input was above a threshold, for example 0.5, then the neuron would output a value of 1.0, otherwise it would output a 0.0.

4) Network of Neurons

DL involves making very large and deep (i.e. many layers of neurons) neural networks to solve specific problems, as shown in Fig.3. Thus, similar to how neurons are organized in layers in the human brain cells, neurons in neural networks are often organized in layers as well. So, an algorithm is deep if the input is passed through several non-linearities before being output.

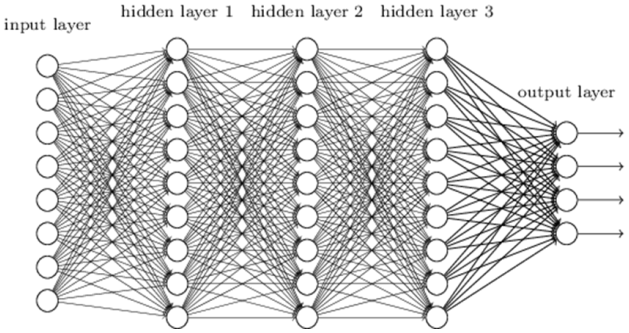


Fig. 2. An example of deep neural network with five layers

a) Input Layer

The first layer that takes input from some dataset is called the input or visible layer, because it is the exposed part of the neural network. Often a neural network is characterized with an input layer with one neuron per each input value in the dataset.

b) Hidden Layer

After the input layer, we have the hidden layers, they are called hidden because they are not directly exposed to the input. The simplest example of a neural network is to have a single neuron in the hidden layer that directly outputs a value. With the increase in computing power and very efficient libraries, very deep neural networks can be built. Neural network can have many hidden layers in it.

c) Output Layer

The last layer is called the output layer and it is responsible for exporting the value or vector of values that correspond to the format required for the problem.

B. Training The Network

a) Data Classification

In order to use binary classification, we should capture two types of data, in our case it will be normal and malicious packets to train the neural network on. As neural networks can only work with numerical data, we have to label the network packets with 0 or 1 for normal and malicious packets. We captured a big dataset that is composed of normal network traffic, i.e. a normal behavior of the ICS devices. In order to get the malicious packets, we prepared a table consisting of the opposite functions and values of the normal ones, that is different IP sources, IP destinations, port numbers, protocol numbers, Modbus (functions, values, registers, coils) etc… And then we captured almost the same number of packets. After that, we combined the normal and malicious packets into one dataset and added a column labeling the packets 0 for normal and 1 for malicious one.

b) Data Values

Data must be numerical, for example real values. If we have categorical data, such as a sex attribute with the values male and female, we can convert it to a real-valued representation

called a one hot encoding. This is where one new column is added for each class value (two columns in the case of sex of male and female) and a 0 or 1 is added for each row depending on the class value for that row.
Neural networks require the input to be scaled in a consistent way. We can rescale it to the range between 0 and 1 called normalization. Another popular technique is to standardize it so that the distribution of each column has the mean of zero and the standard deviation of 1. Scaling also applies to image pixel data. In our case, the data will be a captured PCAP file where the fields consists of IP addresses, port numbers, hexadecimal Modbus values as shown in Fig. 4.

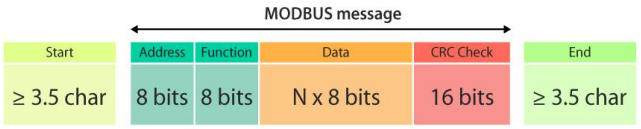


Fig. 3. Modbus Frame

Thus, data must be well-prepared before training the neural network on, we should convert the IP addresses, hexadecimal values, and all other non-decimal attributes into decimal ones, preferred between 0 and 1.

c) *Stochastic Gradient Descent*

The classical and still preferred training algorithm for neural networks is called stochastic gradient descent. This is where one row of data is exposed to the network at a time as input.
The network processes the input upward activating neurons as it goes to finally produce an output value. This is called a forward pass on the network. It is the type of pass that is also used after the network is trained in order to make predictions on new data.
The output of the network is compared to the expected output and an error is calculated. This error is then propagated back through the network, one layer at a time, and the weights are updated according to the amount that they contributed to the error. This clever bit of math is called the Back Propagation algorithm. The process is repeated for all of the examples in your training data. One round of updating the network for the entire training dataset is called an epoch. A network may be trained for tens, hundreds or thousands of epochs, an example of epoch round is shown in Fig. 5.

```
Epoch 1/100
   5/5499 [..............................] - ETA: 17s - loss: 0.6931 - acc: 0.6000
 290/5499 [>.............................] - ETA: 1s - loss: 0.6922 - acc: 0.5310
 625/5499 [==>...........................] - ETA: 1s - loss: 0.6823 - acc: 0.5888
 965/5499 [====>.........................] - ETA: 0s - loss: 0.6378 - acc: 0.6922
1280/5499 [=====>........................] - ETA: 0s - loss: 0.5812 - acc: 0.7438
1605/5499 [=======>......................] - ETA: 0s - loss: 0.5250 - acc: 0.7788
1930/5499 [=========>....................] - ETA: 0s - loss: 0.4792 - acc: 0.8026
2250/5499 [===========>..................] - ETA: 0s - loss: 0.4375 - acc: 0.8240
2560/5499 [===========>..................] - ETA: 0s - loss: 0.4041 - acc: 0.8398
2885/5499 [==============>...............] - ETA: 0s - loss: 0.3786 - acc: 0.8516
3205/5499 [================>.............] - ETA: 0s - loss: 0.3540 - acc: 0.8630
3525/5499 [==================>...........] - ETA: 0s - loss: 0.3362 - acc: 0.8709
3845/5499 [====================>.........] - ETA: 0s - loss: 0.3207 - acc: 0.8780
4180/5499 [=====================>........] - ETA: 0s - loss: 0.3092 - acc: 0.8835
4520/5499 [======================>.......] - ETA: 0s - loss: 0.2957 - acc: 0.8894
4860/5499 [========================>.....] - ETA: 0s - loss: 0.2895 - acc: 0.8922
5170/5499 [=========================>....] - ETA: 0s - loss: 0.2805 - acc: 0.8967
5495/5499 [==========================>...] - ETA: 0s - loss: 0.2705 - acc: 0.9008
5499/5499 [==============================] - 1s - loss: 0.2704 - acc: 0.9009
```

Fig. 4. Epoch example during network training

d) *Prediction*

Once a neural network has been trained it can be used to make predictions. You can make predictions on test or validation data in order to estimate the skill of the model on unseen data.
You can also deploy it operationally and use it to make predictions continuously. The network topology and the final set of weights is all that you need to save from the model. Predictions are made by providing the input to the network and performing a forward-pass allowing it to generate an output that you can use as a prediction [7].

C. *Model Approach*

a) *Preparing the Neural Network*

As deep learning structure is defined as a sequence of layers, we will create a sequential model and add layers one at a time until we are satisfied with our network topology. The first thing to get right is to ensure the input layer has the right number of inputs. In our case, the number of inputs will be the number of fields extracted from the network packets as shown in Fig.6, in addition to the last field which indicates if the packet is normal or malicious.

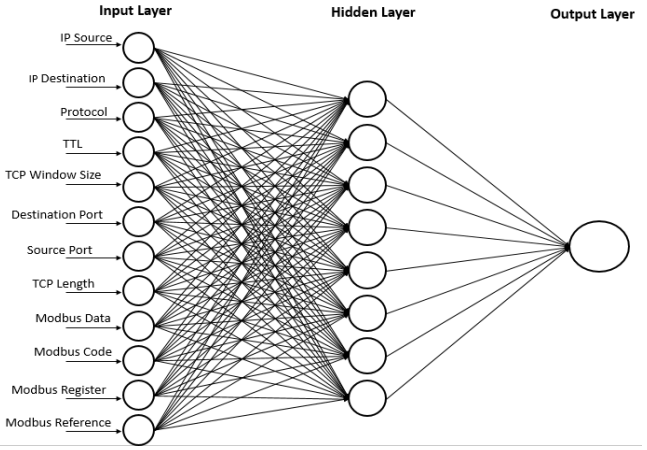


Fig. 5. Input parameters of the neural network

As shown in the above figure, we have 12 inputs including different types of fields (IP, TCP, and MODBUS). The neural network will try to train and learn using those attributes.

How do we know the number of hidden layers to use and their types? This is a bit hard question. There are heuristics that we can use and often the best network structure is found through a process of trial and error experimentation. Generally, we need a network large enough to capture the structure of the problem if that helps at all. In our case we will use a fully-connected network structure with three layers as shown in Fig. 6.

Next, it's best to think about the structure of our layer, we have an input layer, some hidden layers and an output layer. As stated previously, a type of network that performs well on binary classification problem is a multi-layer perceptron. This

type of neural network is often fully connected. That means that we are looking to build a fairly simple stack of fully-connected layers to solve this problem. As for the activation function that you we will use, it's best to use one of the most common functions which is relu activation function [8].

The Rectified Linear Unit has become very popular in the last few years for logistic/continues output. It computes the function

$$f(x) = \max(0, x)$$

One way ReLUs improve neural networks is by speeding up training. The gradient computation is very simple (either 0 or 1 depending on the sign of x).

When we are building our model, it's therefore important to take into account that the first layer needs to make the input shape clear. The model needs to know what input shape to expect and that's why you'll always find the input shape, input dimension, input length arguments in the documentation of the layers and in practical examples of those layers Fig.7.

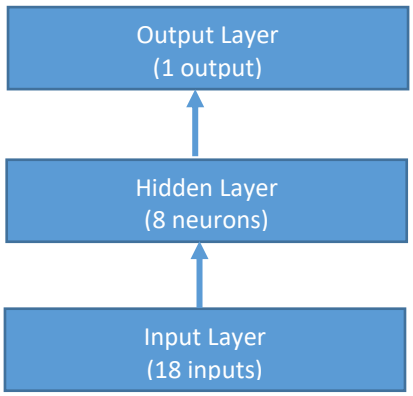


Fig. 6. Visualization of Neural Network Structure

b) Encoding

However, the training must be on numerical fields only, that is if we have an IP address which have the format xxx.xxx.xxx.xxx, the network wont understand it, same as if we have a hexadecimal Modbus data of FF00 for example.To Solve this problem, data must be converted into decimals, we used Excel plugins to convert IP addresses and hexadecimal values into numbers, so that all the fields became of decimal values.

As the scales of the different fields are wildly different, it may have a knock-on effect on network ability to learn. To overcome this, we used data standarization. Standardization is a scaling technique that assumes your data conforms to a normal distribution. If a given data attribute is normal or close to normal, this is probably the scaling method to use.

The result of standardization is that the features will be rescaled so that they'll have the properties of a standard normal distribution with a mean of =0 and a standard deviation of 1. This can be thought of as subtracting the mean value or centering the data. Standardization can be useful, and even required in some machine learning algorithms when the input data values are of different scales. Below is a table showing the network input conversion for a normal packet:

Table-1
Network packet different conversion stages

Attribute	Normal Value	Decemalized Value	Encoded Value
IP Source	192.168.1.5	3232235781	0.53640178
IP Destination	192.168.1.3	3232235779	0
Protocol	6	6	0
TTL	128	128	0.71646104
TCP Window Size	524288	524288	-1.06582338
Destination Port	56783	56783	1.0261182
Source Port	502	502	-1.01072698
TCP Length	0	0	-0.99563837
Modbus Data	FF:00	65280	-0.01348645
Modbus Code	5	5	-0.88003806
Modbus Register	0	0	-0.05902683
Modbus Reference	100	100	-0.13751838

c) Computation Time

The machine used to run the algorithm is a Intel® Core™ i7-3630QM @ 2.4GHz with 8GB installed memory (RAM) having x64-based processor with 4 cores and 8 Logical Processors. The total time for learning (Training + Testing) was 3228 seconds that is 54 minute (Fig.8).

```
196576/213083 [==========================>...] - ETA: 0s
199648/213083 [==========================>..] - ETA: 0s
202976/213083 [==========================>..] - ETA: 0s
206208/213083 [==========================>.] - ETA: 0s
209472/213083 [==========================>.] - ETA: 0s
212736/213083 [==========================>.] - ETA: 0s
acc: 99.98%
Saved model to disk

--- 3288.2358453273773 seconds ---
```

Fig. 7. Training computation time

IV.RESULTS AND DISCUSSION

A. *Description of the Network*

Our ICS network is composed of the SCADA, PLC, and a simulated heater process which triggers the network with a large amount of traffic for gathering and analyzing a real time data to be shown on the SCADA screen, the reactor diagram is shown in Fig.9.

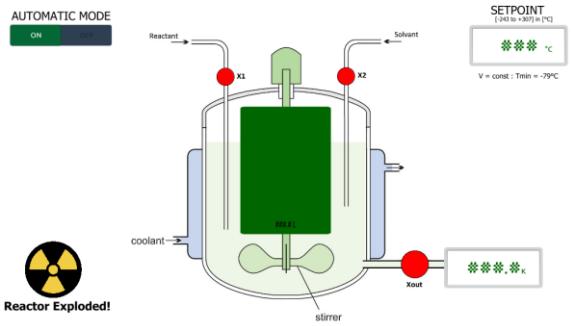


Fig. 8. Reactor diagram with inputs/outputs label

The following table summarizes the system inputs/outputs shown in the above figure.

Table-2
Reactor system inputs and outputs values

Variable	Value
X1	Opened/Closed
X2	Opened/Closed
Xout	Opened/Closed
Coolant Qc	[0; 500]
Liquid Height H	[0; 200]
Liquid Temperature T	[coolant temperature, undefined]
Reactant concentration	[0; undefined]
Explosion Notifier	True/False

Using existing approaches of a HIL system and a local network a hybrid approaches was designed respecting some constraints in order to simulate an industrial environment containing a PLC, a local network, a SCADA control and a virtual mockup built of electronic-designed parts, and a IHM for operator interaction. Fig.9 presents the generic schema of the system.

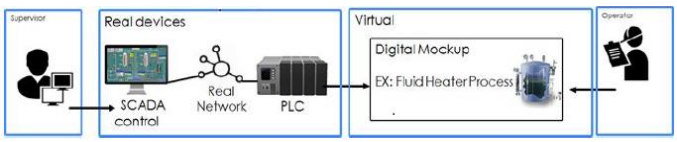


Fig.9. General ICS architecture

The PLC performs the control of the virtual mockup. It receives the data from the digital mockup as though it were a sensor capturing ongoing information of a physical process such as a fluid heater process. Then, it uses the received data to calculate a control signal that is sent to the mockup through an analog output.
The SCADA displays the system information for a supervisor that can access the major system information about the industrial process, the information comes from the PLC that gets information from the sensor and updates the system status. The supervisor uses a PC to control some functions of the systems such as the water temperature and the height. The real network is created by a Switch.

B. *Proposed Approach*

The proposed intrusion detection systems considers a general type of an attack scenario where malicious packets are injected into a SCADA network system composed of a heater and a PLC. The proposed intrusion detection monitors incoming packets and determines an attack.

In this work, we consider the most common industrial protocol, that is to say MODBUS protocol.

Our IDS design is composed of two main phases, the training phase and the detection phase. The training phase is performed offline as it is somehow time consuming. In the training phase, the Modbus packet is processed to extract a feature that represents the normal behavior of the network. Each trained Modbus packet has a label indicating either normal or malicious packet, that what we call the supervised learning. We adopt the Neural Network structure to train the features. The detection phase works almost the same, the same feature is extracted from an incoming packet and the Neural Network structure calculates with the trained parameters to predict the binary decision that is either normal or malicious.

In order to perform the training phase, we simulated a network traffic composed of real values to let the neural network train on.

a) *Preparing the simulation*

The simulation is composed of three virtual machines, the first one is the process that will be executed each 0.1s in order to generate high network traffic, the second one is the SCADA – HMI screen that will display the result and is capable of changing the temperature and finally the PLC controller who

is responsible for reading/writing from/to the registers and coils it is holding as shown in Fig.10, the PLC will control the cooling flow rate.

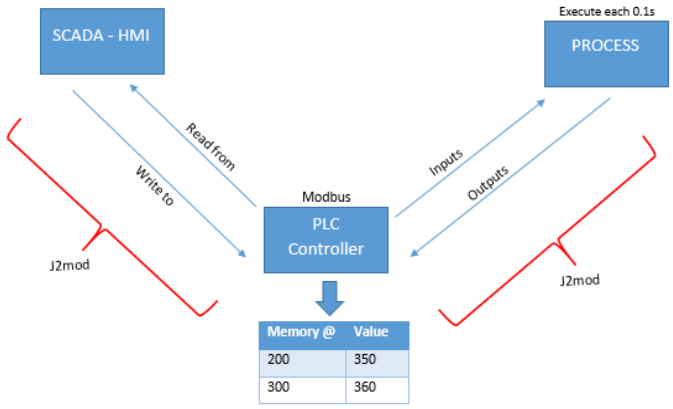
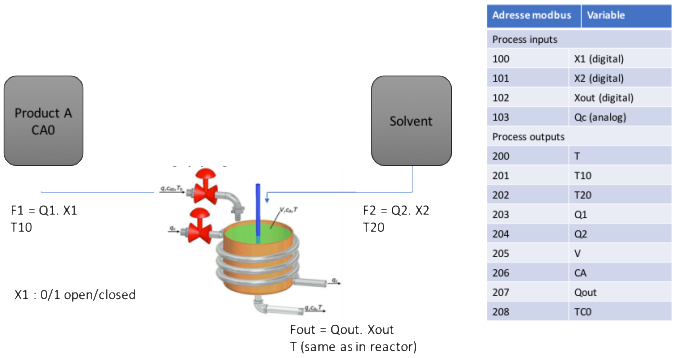


Fig. 10. Simulation of Modbus traffic using virtual machines

The process sends and receieves multiple input/output variables, these variables corresponds to modbus addresses in addition to the value sent for this variable, the addresses with their correspondant variables are shown in Fig.11.



Adresse modbus	Variable
Process inputs	
100	X1 (digital)
101	X2 (digital)
102	Xout (digital)
103	Qc (analog)
Process outputs	
200	T
201	T10
202	T20
203	Q1
204	Q2
205	V
206	CA
207	Qout
208	TC0

Fig. 11. Process input/output values

b) Capturing the traffic

Upon running the PLC, process and SCADA, a high volume of network packets can be captured using Wireshark, and then filtered in order to get the Modbus/TCP traffic only that is running between the machines. An example of those packets is shown in Fig.12.

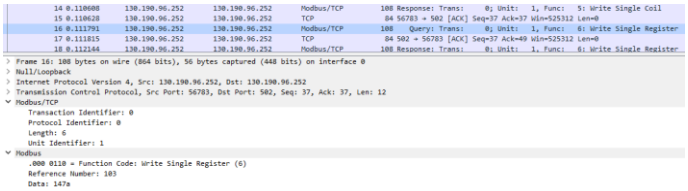


Fig. 12. Modbus/TCP packets capture using Wireshark

The captured PCAP file can be saved into CSV by using Tshark (A tool installed when installing Wireshark) where we can choose specific fields to be saved only (IP source, IP destination, Ports, Protocols, Modbus Data, etc…). Now after obtaining a good traffic and converting it into CSV file we can adjust and perform any operation on any field before training the neural network on.

C. Results

Upon training the neural network on the prepared dataset using Tensorflow and Keras, we can evaluate the performance of the network on the same dataset, this will give us the accuracy and the loss of the training after splitting the data into 70% for training and 30% for testing, these evaluations shows how well the network is doing on the data it is being trained, training accuracy usually keeps increasing throughout training. Using Tensorflow visualization on training and testing dataset, we can view the accuracy of our approach which is shown in Fig. 13.

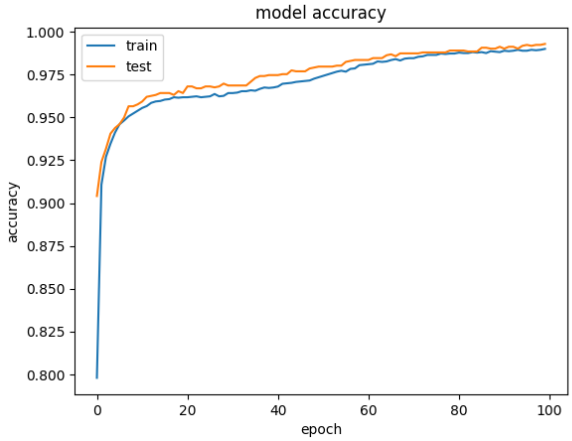


Fig. 13. Model accuracy during the training of the network

As we can see, the accuracy of the trained data is increasing as number of steps (epochs) is increasing, until it reaches approximately 99.89% of accuracy, which means that there is a change of 99.89% of detecting any malicious packet destined towards the network. It is good to note that the neural network performed very well while training, this can be noticed by viewing the speed by which the network learned to draw a pattern from the data given to him, so that between 0 and 40 epochs the accuracy reached approximately 100% of detecting. This is to ensure the importance of decimalizing and reshaping of the data before training the network on them.

Moreover, after each epoch, the model is tested against a validation set, Keras can separate a portion of the training data into a validation dataset and evaluate the performance of the model on that validation dataset after each epoch. The lower the loss, the better the model. Loss is not in percentage as opposed to accuracy and it is a summation of the errors made for each example in training or validation sets. Fig. 14 shows the loss upon training the network.

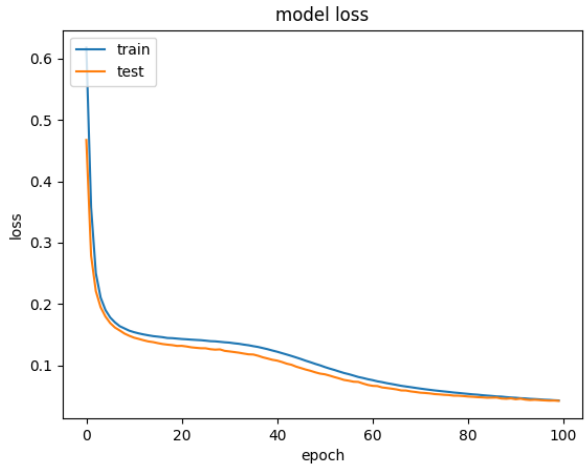


Fig. 14. Model loss during the training of the network

Similar to accuracy, loss will decrease as number of epochs increase till it reaches a value of 0.005% which is almost a negligible loss at the end of the training.

To test the neural network on malicious packets, we prepared a lot of anomalous packets with different IP addresses, ports, functions, and values combinations and injected the IDS with them, the IDS detects all the packets with a high accuracy of 99.9%, an example of the result Keras shows when injecting it with a normal packet is 0.99987454, which when rounded becomes 1 that is a normal one.

This result when compared to self-taught learning (STL) and soft-max regression (SMR) [9] shows a higher performance rate, where when using SMR the accuracy reached 97% and STR reached 98.4%, whereas our discussed approach reached 99.9% of accuracy.

V.CONCLUSION AND FUTURE WORK

We proposed a deep learning based approach to build an effective and flexible IDS. A multi-layer perceptron and binary based IDS was implemented. We used a network dataset that we simulated to evaluate anomaly detection accuracy. We observed that the IDS anomaly detection accuracy showed a very high percentage of detecting. The performance can further be enhanced by adding the ability to detect Denial of Service attacks and adding time stamps to the fields in order to learn the interval of times packets usually arrive by.

VI.REFERENCES

[1] David Bisson. (2016, Nov 13) *How to Approach Cyber Security for Industrial Control Systems.* [Online]. Available: https://www.tripwire.com/state-of-security/ics-security/approach-cyber-security-industrial-control-systems/

[2] Warwick Ashford. (2014, Oct 15) *Industrial control systems: What are the security challenges?* [Online]. Available: http://www.computerweekly.com/news/2240232680/Industrial-control-systems-What-are-the-security-challenges

[3] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin,*" Intrusion Detection by Machine Learning: A Review," Expert Systems with Applications*, vol. 36, no. 10, pp. 11994 - 12000, 2009.

[4] Keith Stouffer, Victoria Pillitteri, Suzanne Lightman, Marshall Abrams, Adam Hahn, *"Guide to Industrial Control Systems (ICS) Security"*, rev 2, NIST National Institute of Standards and Technology, U.S Department of Commere, May. 2015.

[5] *Characteristics of Industrial Control Systems*. [Online]. Available: https://www.citicus.com/Characteristics-of-Industrial-Control-Systems

[6] Muhamad Erza Aminantoa, Kwangjo Kimb, *"Deep Learning in Intrusion Detection System: An Overview"*, School of Computing, KAIST, Korea.

[7] Jason Brownlee, *"Deep Learning With Python: Develop Deep Learning Models on Theano and TensorFlow Using Keras"*, v1.7.

[8] Karlijn Willems (2017, May 2) *"Keras Tutorial: Deep Learning in Python".* [Online]. Available: https://www.datacamp.com/community/tutorials/deep-learning-python

[9] Quamar Niyaz, Weiqing Sun, Ahmad Y Javaid, and Mansoor Alam, *"A Deep Learning Approach for Network Intrusion Detection System"*, College of Engineering, The University of Toledo, USA.