REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

DESIGN AND INTEGRATION OF AN IPS STRUCTURE ON CENTRALIZED IDS-BASED NEXT GENERATION FIREWALLS

21011097 — İrem ÇELİK

SENIOR PROJECT

Advisor
Dr. Furkan ÇAKMAK

January, 2026

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Dr. Furkan Çakmak, who served as my advisor on this project, provided assistance at every stage, closely followed the project, and supported my work with his valuable contributions.

I would also like to thank Siemens for granting permission and providing unwavering support throughout the realization of this project. The working environment and technical guidance provided within Siemens greatly contributed to the development of this project.

İrem ÇELİK

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Network
DoS	Denial-of-Service
FOA	Falcon Optimization Algorithm
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
LSTM	Long Short-Term Memory
ML	Machine Learning
SIEM	Security Information and Event Management
SSM	SINEC Security Monitor
TC	Traffic Capturing
TM	Traffic Monitoring
XAI	Explainable Artificial Intelligence

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Design and Integration of an IPS Structure on Centralized IDS-Based Next Generation Firewalls

İrem ÇELİK

Department of Computer Engineering
Senior Project

Advisor: Dr. Furkan ÇAKMAK

New types of attacks are being developed. Network security is becoming increasingly important. In the industrial sector, network security has become a very risky issue. Therefore, there is a growing need for systems that can detect attacks early and take preventative measures. This project aims to detect attacks in traffic.

The architecture developed in this project is designed for use in Siemens security devices. The topology at Siemens was simulated using virtual machines.

The UNSW-NB15 dataset was used in this project. Various preprocessing operations were performed on the features in the dataset. Then, an Autoencoder model was developed using normally labeled data. This model was tested with various parameters to improve it. The results of the model were evaluated using precision, recall, accuracy, and F1-score metrics.

An offline Proof of Concept study was conducted on a virtual system to verify the model's performance. Different traffic scenarios were created. The model was applied to these traffic scenarios. The model's decisions were examined.

Model was tested with flowing traffic to determine its suitability for real-world situations. Detections were made using the model applied to the flowing traffic. Traffic identified as attack or suspicious was routed to the SIEM. This allowed for continuous monitoring of the traffic, enabling the detection of attacks and the investigation of suspicious traffic.

Keywords: IDS, IPS, Network, Security, Anomaly, Autoencoder, SIEM, UNSW-NB15.

ÖZET

Merkezi IDS Tabanlı Next Generation Firewall'lar Üzerine IPS Yapısı Geliştirilmesi ve Entegre Edilmesi

İrem ÇELİK

Bilgisayar Mühendisliği Bölümü
Bitirme Projesi

Danışman: Dr. Furkan ÇAKMAK

Yeni saldırı çeşitleri ortaya çıkmaktadır. Network güvenliğinin önemi her geçen gün artmıştır. Endüstriyel alanda network güvenliği çok riskli bir konu haline gelmiştir. Saldırıları erken tespit edebilen ve önlem alınabilen sistemlere ihtiyaç da artmıştır. Bu projede trafikteki saldırıların tespitinin yapılması amaçlanmıştır.

Projede geliştirilen yapı Siemens'e ait olan güvenlik cihazlarında kullanmak amacıyla tasarlanmıştır. Sanal makineler yardımıyla Siemens'teki topoloji simüle edilmiştir.

Projede UNSW-NB15 veriseti kullanılmıştır. Verisetine yer alan featurelar üzerinde çeşitli ön işlemler yapılmıştır. Daha sonrasında normal olarak etiketlenen veriler ile Autoencoder modeli eğitildi. Bu modeli iyileştirmek adına çeşitli parametrelerle model test edilmiştir. Gerçekleştirilen model sonuçları analiz edildi. Precision, recall, accuracy ve F1-score metrikleri kullanılarak değerlendirildi.

Model performansını doğrulamak için sanal ortamda oluşturulan sistemde offline bir Proof of Concept çalışması yapılmıştır. Farklı trafik senaryoları oluşturulmuştur. Bu trafiklere model uygunlanmıştır. Modelin verdiği kararlar incelenmiştir.

Modelin gerçek dünyaya uygunluğunu tespit edebilmek için sanal ortamda akan trafikle model test edilmiştir. Akan trafiğe uygulanan model ile flowlarda tespitler yapılmıştır. Atak ya da şüpheli olan trafikler SIEM'e yönlendirilmiştir. Trafik sürekli olarak izlenmiştir. Bu sayede atakların tespit edilebilmesi ve şüpheli trafiklerin incelenmesi sağlanmıştır.

Anahtar Kelimeler: IDS, IPS, Network, Güvenlik, Anomali, Autoencoder, SIEM, UNSW-NB15.

1
INTRODUCTION

Network security is important today. It is of great importance in both corporate and industrial infrastructures. Devices used in industrial areas affect the continuity. They ensure safety of production processes. So protecting these systems against attacks is important.

Different threat detection systems are used to achieve network security. Threat detection methods protect the system against various types of attacks. However, it is not possible to provide protection against zero-day attacks with existing systems.

IDS (Intrusion Detection System) and IPS (Intrusion Prevention System) technologies play an important role at network security. IDS detects potential threats in the system against known vulnerabilities. IPS responds to threats. It is used to prevent possible damage. The goal of the system to be developed is to use these two technologies together. This will increase network security.

1.1 Objective

The main purpose of this project is to develop an IPS. Then integrate into a centralized IDS based system. It aims to increase system security and provide a system better prepared for attacks.

The IDS monitors all traffic and detects anomalies. The designed IPS will use machine learning algorithms to provide protection against attacks. The system is intended for use in the real world where traffic flow is high. So it will be simulated using virtual machines. At the end the performance metrics of the system will be examined.

1.2 Preliminary Review

1.2.1 Rationale for the Project

Network infrastructure is evolving every day. New threats are emerging. With this evolution traditional firewalls and detection systems are no longer able to provide adequate protection. Industrial network are at high risk due to the continuous flow of data and real time operations.

IDS systems can detect known threats. But they are insufficient for detecting previously unseen attacks. So, it's not enough to just detect. A security architecture capable of responding in real time is needed. This project will be developed on a centralized IDS based infrastructure. A machine learning based IPS mechanism will be integrated into centralized infrastructure. The goal is to improve attack detection and prevention.

1.2.2 Scope of the Project

The main features of the system to be developed in the project are listed below:

1. This project aims to design a security system that uses IDS and IPS together.
2. Real traffic data will be used in the project. IPS algorithms will be trained and tested with this data.
3. The system to be designed will examine and implement the detection methods used in IPS. These methods will also include machine learning based approaches.
4. Performance and accuracy analyses will be conducted. These analyses will be compared to create a more reliable system. The most suitable method will be selected.
5. Attack packets containing vulnerabilities or suspicious packets will be identified. A warning will be sent to the user to provide protection against potential threats.
6. The created system is intended to be used in industrial environments.

2
LITERATURE REVIEW

Digitalization is on the rise today. Billions of devices are connected to each other via the internet. This create a wider area for cyberattacks. And significantly increases the importance of security. Modern cyberattacks are no longer limited to data theft. They lead to economic losses and production disruptions. So, network security no longer simply means preventing threats. It also includes continuous monitoring, analysis, and anomaly detection. In this context, IDS IPS systems based on artificial intelligence and machine learning are necessary. Systems must adapt to a dynamic threat environment. Providing real time attack detection and explainable decision making mechanisms are important.

2.1 Similar Studies

2.1.1 Intrumer

It is an IDS/IPS model developed to ensure network security in cloud environments. This system goal to detect cyberattacks based on machine learning (ML) and explainable artificial intelligence (XAI).

There are three main modules. First real time network traffic is collected. Then the data is transmitted to a module called TC/TM. In this module, the Falcon Optimization Algorithm (FOA) selects the most important features. The Naïve Bayes algorithm classifies the traffic as normal or malicious. The selected features are then passed to the Heterogeneous Attention Transformer module. It analyzes the contextual relationships between the features. In the third module, the Explainable Prevention Module used traffic is classified as malicious. The Shapley Additive Explanations (SHAP) method is used to explain why it was classified as malicious. The alert is sent by the system to other modules and servers. This model has gained an important place in the literature due to its multi module and explainable nature [1].

2.1.2 FlowTransformer

This study purpose to analyze network traffic flow using transformer based models. It is different from traditional machine learning based IDS. This system analyzes flow sequences over time. It enables the detection of more complex attack patterns. Various transformer models, including BERT, GPT2, and shallow encoder decoder variants were compared. The study revealed that deep language learning models are too complex for network traffic analysis. In conclusion, Flow Transformer provides a research framework for transformer based models. It fills a significant gap in the literature [2].

2.1.3 CNN Based Network Intrusion Detection against DoS Attacks

This study goals to implement a system against Denial of service (DoS) attacks. Network traffic data was converted into images. This system was built using a Convolutional Neural Network (CNN). CNN has been shown to provide higher accuracy than Recurrent Neural Network (RNN) and classical ML methods. This study has attracted attention in the literature. Because its focus on low level discrimination of DoS types [3].

2.1.4 An Optimized LSTM Based Deep Learning Model for Anomaly Network Intrusion Detection

Previously developed IDS have a high false alarm rate. The goal of this research is to develop an optimized LSTM based IDS for detecting network anomalies. The goal is to reduce the false alarm rate. The hyperparameters of the LSTM model were optimized. It uses three different metaheuristic algorithms. The optimized models were compared and the highest performance was identified. This study shows the importance of hyperparameter optimization in deep learning based IDS research [4].

2.1.5 Anomaly Detection based on Isolation Forest and Local Outlier Factor

This study investigated the Isolation Forest algorithm for detecting anomalies. It was hypothesized that this unsupervised algorithm could be faster and more effective than traditional algorithms. It was compared to the commonly used, density based Local Outlier Factor algorithm. The results showed that the Isolation Forest algorithm achieved more successful results in normality detection. Isolation was better in terms of both speed and false positive rate. [5].

2.1.6 AI Based Techniques for Zero Day Attacks Detection

There are many AI based studies in the literature that use methods against attacks. The goal of this study is to compare and analyze the methods in the literature. Many AI based models (Autoencoder, LSTM, CNN etc.) are examined in this study. Various datasets have also been analyzed. The study shows the inadequacy of signature based methods against zero day attacks. It shows the importance of AI based models [6].

2.1.7 Kitsune

This study will work on network devices such as routers. It proposes an ANN based network intrusion detection system. The goal is to create a system that combines unsupervised learning, online processing, and low complexity. A suitable model has been implemented. Using an autoencoder based structure the model continuously monitors network traffic. It performs packet based feature extraction. It improves itself as new traffic arrives. This system uses numerous small models instead of a single large model. Among the limitations of this model is the possibility of performance degradation in very dense networks. The online learning model can become overly sensitive to noise. It may lead to an increase in false positive rates [7].

2.2 Conclusion

Preliminary research has revealed various solutions for improving network security. These suggestions include models developed. They are using machine learning and deep learning algorithms. These have been presented as effective solutions. The accuracy and performance of the models used have been improved. It leading to the development of more secure systems.

The literature shows that studies frequently uses multiple datasets. This approves the generalizability of the developed system. Its ability to demonstrate similar success in different network environments.

Studies applying transformer-based models to network traffic data were reviewed. It was found that these models generate high computational costs and complexity. So these models were not preferred in this study.

The study "Utilizing IDS and IPS to Improve Cybersecurity Monitoring Process" appears in the literature emphasizes that the combined use of IDS and IPS systems strengthens network security. It is effective in reducing false alarm rates [8].

Similarly this project presents an IDS IPS framework that analyzes network traffic

in real time. It takes preventive action against detected threats. Therefore, the developed system purposes to be an application that translates existing approaches in the literature into practice.

3
SYSTEM ANALYSIS AND FEASIBILITY

This section will detail the methods and techniques to be used to successfully complete the system. First the software and hardware requirements will be described in Technical Feasibility section. Then followed by the Legal, Economic, and Time Feasibility evaluations. It has determined the most suitable method to meet the system requirements.

3.1 System Analysis

System analysis is an important process. It involves identifying the projects main requirements, evaluating available resources, and determining the most suitable solution for designing the targeted system. At this stage, technical requirements were assessed to clarify the project scope. Performance criteria were defined.

The goal of this project is to design an IPS architecture. Then integrate this system into a centralized IDS system. The designed IPS will enhance system security. It develops methods for responding to attacks.

3.1.1 Main Objectives of the Project

1. Improve network security by detecting attacks on the system with IDS IPS.
2. Use a large traffic data to training and testing model.
3. Find the best performing model by analyzing the performance and accuracy of different IPS detection methods.
4. Analyze the behavior of the developed IDS IPS system in different environments.
5. Create a system more protected against attacks with the resulting system.

3.2 Feasibility

The purpose of feasibility study to evaluate the feasibility of project in terms of Technical, Legal, Economic and Time.

3.2.1 Technical Feasibility

3.2.1.1 Hardware Feasibility

The Siemens devices intended for use in this project were not used. The system created with these devices was simulated in a virtual environment. The project was developed in this virtual environment. The hardware specifications of the computer used in the model development phases and in simulating the environment are listed below.

Table 3.1 Hardware Specifications

Specifications	**Hardware Requirement**
RAM	16 GB
Processor	8 A modern 8-core processor
Graphics	AMD Radeon™ RX 5500M
Storage	At least 512 GB SSD

3.2.1.2 Software Feasibility

In this project, the IDS IPS model was developed using the Python programming language within the Google Colab environment. After training and testing the model in the Colab environment, it was saved for use in a virtual environment. The scripts necessary to use the model in virtual environment were also written in Python.

3.2.2 Legal Feasibility

The system used in this project was created by simulating a system at Siemens. The RX1400 series network security device and SSM software, both Siemens products. They are proprietary Siemens products. The devices were used under the hardware and software license terms provided by Siemens. There is no commercial activity or product development purpose. In this context, the project was conducted in accordance with intellectual property rights, license agreements, ethical research principles. Necessary permissions were obtained.

Also the data used in the project does not contain any actual user or personal information. So the project complies with the provisions of the Personal Data

Protection Law (KVKK).

3.2.3 Economic Feasibility

The RX1400 series network security device and SSM software used in this project were previously supplied by Siemens. These are systems already available in the test environment. So no new devices or licenses were purchased during the project. Development was carried out by simulating the existing hardware and software infrastructure. Since open source software was used, no license fees were charged.

The high performance GPU required for model training and testing was provided using the Google Colab Pro environment. This system cost 165 TL.

3.2.4 Labor and Time Feasibility

The project is planned to be completed within 15 weeks. The Gantt chart in Figure 3.1 include the stages of the process. It shows how the time allocated to these stages is managed.

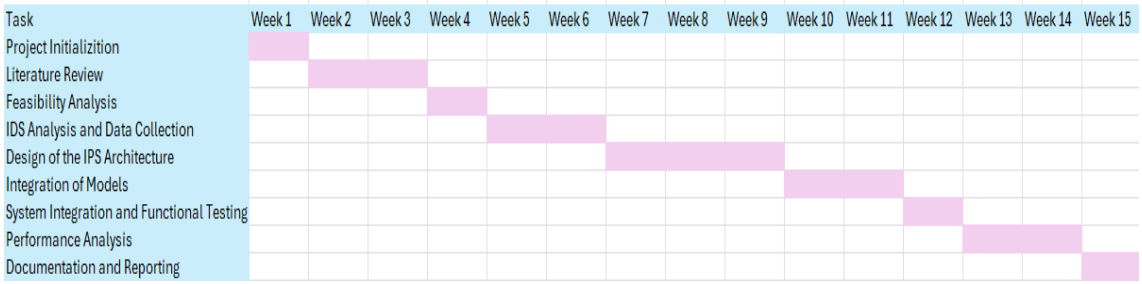

Figure 3.1 Gantt Chart

4
System Design

This section will describe the features of the dataset to be used in the project. Then the methods to be applied will be explained in detail.

4.1 Materials and Dataset

This section will give information about the selected dataset and discuss the attributes chosen to suit the model to be used.

4.1.1 Dataset Description

The **UNSW-NB15** dataset, frequently used in the literature. It was selected as the dataset in the project. Raw network packets of the UNSW-NB15 dataset were generated at the UNSW Canberra Cyber Range Laboratory using the IXIA PerfectStorm tool. In this process, realistic and current normal network activities were combined with modern attack behaviors. Tcpdump was used to capture network traffic. Traffic features were extracted from the collected data. It uses to Argus and Bro IDS (Zeek) tools. Twelve different algorithms were developed in this process. A total of 49 features were generated [9]. It includes both normal traffic and different attack categories. Similar to real IDS systems each log is represented on network flow. It includes various protocols and statistical features.

Dataset Split: The dataset contains 93,000 normal traffic records. Of these, 65,100 were used to train the model. A total of 5,580 normal traffic data were reserved for threshold selection. 22,320 normal traffic records and 82,337 attack records were used for testing.

This dataset contains nine different attack types. This are Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. The distribution of these attacks in the dataset is shown in the Table 4.1.

Table 4.1 Attack Distribution in Test Dataset

Attack	Count
Analysis	1338
Backdoor	1164
DoS	8177
Exploits	22263
Fuzzers	12123
Generic	29436
Reconnaissance	6994
Shellcode	755
Worms	87
Total	82337

Attribute The dataset contains 49 flow based attributes extracted using the Argus and Bro IDS (Zeek) tools. These features are categorized as follows:

Protocol and Connection Attributes: These attributes define the structural characteristics of the network connection. This indicates the protocol used for communication, the service used, and the status of the connection (protocol, service, status).

Statistical Traffic Attributes: These attributes include volumetric values. For example the number of packets and bytes sent and received. It allow for the numerical assessment of traffic density, magnitude, and potential anomalies (spkts, dpkts, sbytes, dbytes, rate, sload, dload).

Temporal attributes: These time based attributes include parameters such as duration of the flow, time differences between packets, and latency measurements (dur, sinpkt, dinpkt, tcprtt, synack, ackdat).

Behavioral/connection counters: These features identify how many times a specific source or destination communicates over time or detect recurring patterns (`ct_srv_src`, `ct_state_ttl`, `ct_dst_ltm`, `ct_src_ltm`, `ct_dst_src_ltm`).

In this study, the main goal is for the model to be able to detect anomalous behavior in traffic. In the model, the attacks appear as behavior that deviates from normal network behavior. For this purpose the identifier (id) and the label related fields were excluded. Other attributes provided by the UNSW-NB15 dataset were included in input. The use of these features allows the model to learn a comprehensive representation of network behavior. This is important for detecting attacks.

4.1.2 Data Preprocessing

In this project, a detailed preprocessing step was applied to the dataset. This is to ensure that the machine learning model learns network traffic behavior well.

Filtering: The model was trained only normal traffic. The inputs to be given to the model were selected. Unnecessary columns were removed.

Separation of Numerical and Categorical Features: The features in the dataset were divided into two groups: numerical and categorical.

Because categorical features cannot be directly processed by most machine learning algorithms. They must be converted into a numerical representation. So the **One-Hot Encoding** method was used to categorical features. In the One-Hot Encoding approach each categorical value is represented by its own binary vector. This prevents the introduction of an artificial order or magnitude relationship between categories. It helps the model avoid misinterpreting these features [10].

The numerical features in the dataset have different scales and units. These features can take values across a wide range of magnitude. This can lead to features with high values becoming disproportionately dominant in the model. To eliminate this problem, **StandardScaler** was applied to all numerical features. This method allows all numerical features to be rescaled. This ensures that features with different units and scales have a balanced impact on the model.

4.2 Methodology

The goal of this study is to develop an IDS IPS system capable of detecting attacks especially zero day attacks. Zero day attacks are previously unseen. They dont have known signatures. It is not possible to detect them using traditional signature based methods. For this reason the anomaly based **Autoencoder** was chosen for this study. To analyze model performance, the detection rates of different attack types were examined. Model evaluation was performed by looking at the obtained accuracy, recall, and F1-score metrics.

4.2.1 Autoencoder

Autoencoder is a deep learning structure. It learns the normal behavior of data without using labels (unsupervised). In this study, the Autoencoder model was trained using only normal traffic and tested in different attack scenarios.

Autoencoder models this behavior using the compress reconstruct principle. The goal here is not to classify the data. The goal is understand its underlying structure. It learns the characteristics of network traffic.

Autoencoder is trained with normal traffic so all attacks are considered zero day attacks by this system. This is because autoencoder does not learn attack behavior. An autoencoder cannot reproduce behavior it does not recognize. It allows to detect behaviors it has never seen before [11]. It only looks at the difference between incoming traffic and normal traffic. The model makes inferences based on the error rate obtained in abnormal behavior. It detects traffic that deviates from normal behavior through high reconstruction error. It labels such records as attack.

There are three main components in an Autoencoder:

Encoder: It preserves the most distinctive information from high-dimensional input. It is used to compress it into a lower-dimensional representation.

Latent Space: When trying to learn normal behavior, it focuses on the most important and generalizable features. Normal traffic has a basic pattern and relationships between features. The latent space forces the model to learn this pattern. It prevents the reconstruction of abnormal behavior. The correct latent size is one of the most important aspects of the model.

Decoder: It uses data compressed by the encoder. It is used to produce an output that is as similar as possible to the original.

$$\text{Reconstruction Error} = \text{distance}(x, \hat{x})$$

Model success depends on the reconstruction error between the result produced by the decoder ($\hat{x}$) and the input data (x). The difference is measured by the reconstruction error. The choice of different distance metrics affects the performance of the model.

$$(label) = \begin{cases} 0, & err < t_1 \quad \text{(Normal)} \\ 1, & err \geq t_1 \quad \text{(Attack)} \end{cases}$$

The generated reconstruction errors are classified according to the threshold value selected in the model. In general, reconstruction errors related to attacks are high. And reconstruction errors related to normal traffic are low. Reconstruction errors actually show how closely the traffic resembles the training data. So, choosing a

threshold based on the characteristics of the data used to train the model is very effective in model performance. Below is an example of how the generated errors are classified according to the thresholds.

4.3 Virtual System

This section will discuss the system in which the autoencoder is implemented. The developed model is intended for use with Siemens Layer 3 router devices. Specifically the RX1400 and the SRM devices currently under development. To this end, systems using these devices have been minimized. It simulated in a virtual environment. Figure 4.1 below shows the simulated topology intended for real world use.

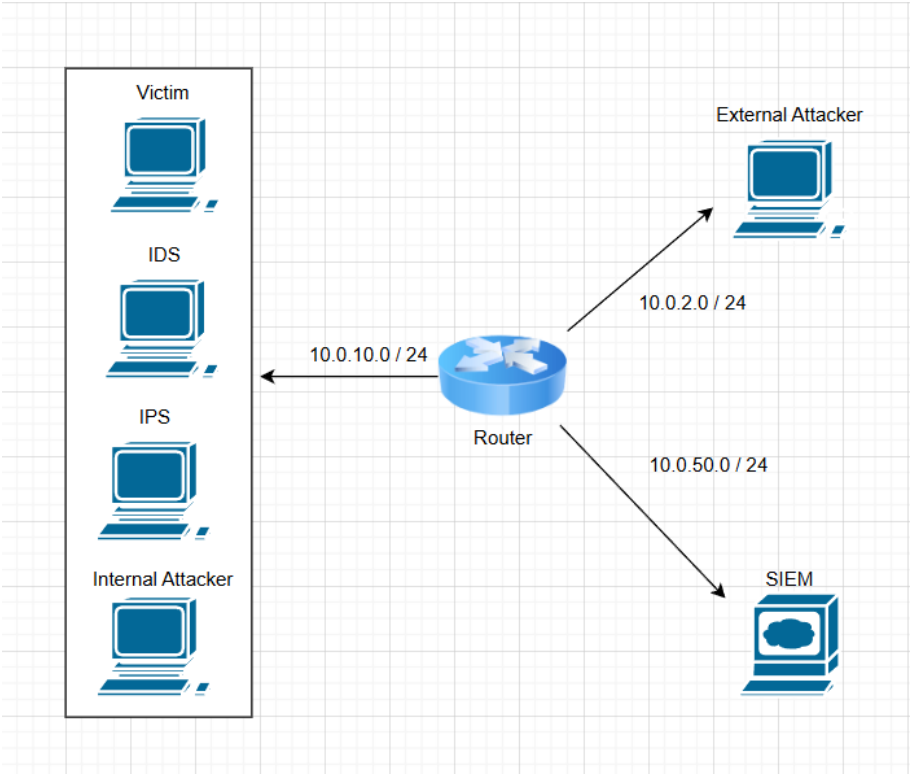

Figure 4.1 Virtual Enviroment

The topology contains 7 virtual machines. Each machine has different functions.

Router: The intended Layer 3 Router (RX1400) is represented by this virtual machine. It connects the three networks in the topology and represents the traffic flow point.

Attacker: As seen in the topology, there are two attackers in the system. One is the external attacker, representing an attacker located on the outside network. The other represents an attacker that may be located on the inside network. Various attack scenarios will be simulated on these devices. These machines were built with Kali

Linux. Because it enables these attack simulations.

SIEM (Security Information and Event Management): The virtual machine created to represent Siemens' SINEC Security Monitor (SSM). It collects logs from traffic. It provides a management system for managing this information. To ensure the security of these logs, the SIEM is stored on a separate LAN. Wazuh is a open source SIEM. It is used to represent the SSM. This system allows traffic to be classified not only as alert or normal, but also as suspicious. Alerts and warnings detected by the model will be directed here. This machine allows the user to manage the system. It also ensures that normal traffic is not interrupted due to false positive results.

Victim: It represents the target machine. Attacks are carried out on this machine. It hosts various services, such as web services. This allows it to function as a real target within the system.

IDS: The virtual machine added to create the IDS based system. It uses Suricata. Traffic detected by this machine is routed to the SIEM.

IPS: This is the virtual machine on which the model will run. Zeek is installed on this machine. It continuously monitors traffic. It collects traffic and extracts features from the traffic data. They are used in the developed model. It continuously evaluates the traffic flow using the model. Depending on the model's decision, it reports detected traffic to the SIEM as alerts and warnings.

The virtual system described above was used to test the model developed in the Colab environment. And used to examine its real world application. Simulations were performed in this virtual environment. In these simulations, two types of traffic flows were run on the system. These traffic flows were monitored and evaluated by the IPS. One of these flows was a PCAP based offline analysis. The other was created to represent the flowing traffic.

The Zeek tool was used to record this traffic. Traffic log files were generated using Zeek. Thanks to Zeek, features revealing traffic behavior were created. These features were analyzed in the model.

Offline Pipeline: This scenario was created to test model performance. The process from collecting PCAP files to obtaining results is as follows:

1. The generated traffic was listened to with Tcpdump. Each was recorded in a separate PCAP file.
2. PCAP files were converted into log files using Zeek.

3. conn.log, http.log, and dns.log files were used from the generated log files.
4. Feature extraction was performed on the log files using Zeek. CSV files containing 35 features were obtained.
5. Reconstruction errors were generated using the model based on the data obtained from these traffic events.
6. The threshold values created in the model were used. Decisions were made regarding the resulting reconstruction values.

Online Pipeline: This pipeline was created to evaluate model performance in flowing traffic. Several methods were employed to realize the real time system.

1. In this virtual system, Zeek continuously monitors traffic. Log files are collected in the background.
2. The script that was created then performs feature extraction on the data.
3. A reconstruction error is generated for the flows.
4. Decisions are determined for each flow based on the defined thresholds.
5. Flows classified as warnings and alerts are routed to the SIEM.

5
Experimental Results

This section will provide information about the results obtained from the improvements and enhancements made to the chosen model.

5.1 Performance Analysis

One of the most significant factors affecting model performance is parameters selection. The impact of these parameters on the model was observed while project. Various parameters were tested. The goal was to select the parameters that would yield the best results for the model.

5.1.1 Threshold Selection

The autoencoder works by generating a reconstruction error for each piece of data. How these errors are classified depends on the threshold. Threshold is the decision making mechanism of the system. If the threshold is too low, every piece of traffic is mistaken for an attack. And if it's too high, attack traffic data is perceived as normal and overlooked. So the choice of threshold affects the accuracy of the model's detections.

Trying different thresholds is important to finding the right thresholds. There are many methods for deciding on thresholds. The most common are selecting based on a normal validation distribution or target oriented selection. To make this selection the dataset is divided into three parts: train, val, and test. Reconstruction errors are calculated for the data in the val dataset. Then, thresholds are determined according to the chosen method. These two methods are compare and select the best threshold.

Normal Validation Distribution: The error distribution of normal traffic based on quantile values is used to generate thresholds.

Target Oriented: Targeted performance criteria are defined for IPS and SIEM. The

resulting error evaluations are examined. Among the threshold values that meet these targets, the values that yield the best results are selected.

In the normal validation distribution, reconstruction error values are sorted from smallest to largest. The thresholds were chosen such that %85 of these errors would be below the first threshold. And %90 would be below the second threshold. In target oriented, the goal was to identify %90 of attacks as attacks and %97 of all attacks as attacks or suspicious. Table 5.1 below shows the recall values obtained for the normal and attack labeled classes as a result of the threshold selection method.

Table 5.1 Threshold Selection

Recall	**Validation**	**Goal**
Normal	0.84	0.49
Attack	0.81	0.89

Table 5.1 shows that target based selection is actually better at attack recall. The goal was to increase the attack detection rate. However, more than half of the normal traffic was misinterpreted as an attack. A trade off situation exists here. Attack recall is lower in the validation option. But thresholds in the model were chosen based on the distribution. Because normal and attack recalls are more balanced in this option.

5.1.2 Latent Dimension

As explained in the previous section, the Autoencoder compresses the data it receives as input. Latent dimension is the size of the intermediate vector representing the input during the compression phase. Choosing the correct latent dim affects the accurate representation of the data. So model performance was analyzed using different latent dims.

Table 5.2 below shows the results obtained according to the latent dim selection. Based on these results dim 16 was chosen as latent dim.

Table 5.2 Latent Dimension Selection

	16	**8**	**4**
Normal	0.84	0.84	0.83
Attack	0.81	0.76	0.70

5.1.3 Loss Function

The Loss function determines how the reconstruction error of the data is evaluated. Each Loss function has its own specific approach.

Mse: It severely punishes major errors.

Mae: It punishes errors linearly.

Huber: It behaves like Mse for small errors and like Mae for major errors.

The loss functions described in Table 5.3 were evaluated. Huber function was selected based on the results obtained.

Table 5.3 Loss Function Selection

	Huber	**Mse**	**Mae**
Normal	0.84	0.80	0.84
Attack	0.81	0.78	0.68

5.1.4 Analysis

Based on the comparisons and evaluations given above, the parameters to be used in the model were determined. In the 2 threshold model, Q-85 was chosen as the 1st threshold value and Q-90 as the 2nd threshold value. In the model 16 was selected as the latent dim. Huber was used as the loss function. Then the model was applied to the selected 2 threshold dataset.

$$(label) = \begin{cases} 0, & err < t_1 \quad \text{(Normal)} \\ 1, & t_1 \leq err < t_2 \quad \text{(Warning)} \\ 2, & err \geq t_2 \quad \text{(Attack)} \end{cases}$$

The decision making mechanism described above was used. Classifications were made on the test dataset. Tables 5.4 and 5.5 below show the evaluation of the model according to the selected performance criteria.

In Table 5.4, only data labeled with label 2 for the attack class were considered found. Based on this evaluation, the model's recall values were examined. Model can capture %89.5 of normal traffic and %79.45 of attack traffic. When we examine the precision values, %54 of the captured normal traffic is correct. And %96.5 of the captured attacks are correctly detected.

Table 5.4 Performance of the Alert Decision

Class	Precision	Recall	F1-score	Support
Normal (0)	0.5414	0.8950	0.6746	22320
Attack (1)	0.9654	0.7945	0.8716	82337
Accuracy			0.8159	104657
Macro avg	0.7534	0.8447	0.7731	104657
Weighted avg	0.8750	0.8159	0.8296	104657

Looking at Figure 5.1 below, we observe that out of 22,320 examples belonging to the "normal" class, 19,962 were correctly identified.And out of 82,337 attack data points, 64,321 were correctly classified. 18,016 attack data points were missed. They labeled as normal.

Since a total of 37,978 samples were labeled as normal, the precision value for the normal class was lower than expected. This reveals the dominance of imbalance between classes on the precision.

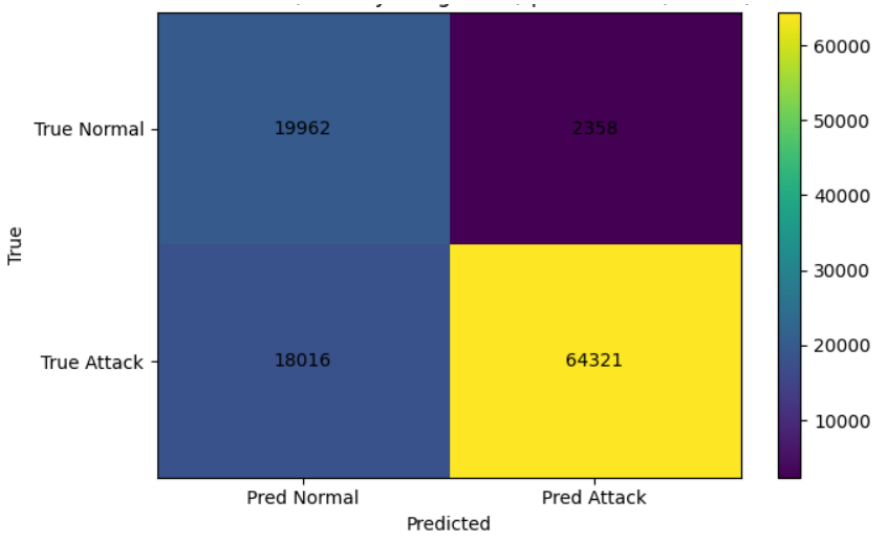

Figure 5.1 Confusion Matrix (0/2)

As seen in the confusion in Figure 5.1, class imbalance reduces the precision of the normal class. To observe how the imbalance affects the results, the test data in the dataset was balanced. Table 5.5 shows the results when the imbalance between classes is corrected. As can be seen, the normal precision value has increased to %79.7.

Table 5.5 Balanced Data Classification Report

Class	Precision	Recall	F1-score	Support
Normal (0)	0.7972	0.8475	0.8216	22320
Attack (1)	0.8539	0.8052	0.8288	24701
Accuracy			0.8253	47021
Macro avg	0.8255	0.8263	0.8252	47021
Weighted avg	0.8270	0.8253	0.8254	47021

Although the results have improved, the goal of this project is not to increase the numerical accuracy of the model. The goal is to ensure the model accurately classifies as much traffic as possible. So the data belonging to the attack class in the dataset has not been reduced. Model performance has been evaluated using various metrics, taking into account the imbalance between the classes.

In Table 5.5, all data labeled with a 1 or 2 tag are counted as detected. This is because data with a 1 tag is actually considered suspicious so not normally labeled. When evaluated in this way, the attack detection rate has increased to %81. Also %95 of the detected attacks were correctly classified. This model shows a low false positive rate for attacks. Since some normal data was classified as a warning, the recall value for the normal category in this table has decreased to %84.75.

Table 5.6 Performance of Detection (Warning+Alert) Decision

Sınıf	Precision	Recall	F1-score	Support
Normal (0)	0.5480	0.8475	0.6656	22320
Attack (1)	0.9515	0.8105	0.8754	82337
Accuracy			0.8184	104657
Macro avg	0.7497	0.8290	0.7705	104657
Weighted avg	0.8654	0.8184	0.8306	104657

5.2 Comparative Analysis

5.2.1 Type of Attack

The dataset used contains 9 different types of attacks. To observe the model's performance according to attack type, the distribution of data according to attack types was examined.

Table 5.6 shows that some attacks are detected very well. And others are detected very rarely. This is because the reconstruction error produced by each type of attack

is unique to that attack. The errors produced by the undetectable attack types were observed to be close to those of normal traffic. Therefore the data for this type of attack remained below the thresholds and could not be detected.

Table 5.7 Attack Category Based Results

Attack	Total	Alert	%	Warning	%	Missed	%
Analysis	1338	1156	86.40	24	1.79	158	11.81
Backdoor	1164	1060	91.07	7	0.60	97	8.33
DoS	8177	7455	91.17	75	0.92	647	7.91
Exploits	22263	19533	87.74	301	1.35	2429	10.91
Fuzzers	12123	4440	36.62	694	5.72	6989	57.65
Generic	29436	29192	99.17	34	0.12	210	0.71
Reconnaissance	6994	2442	34.92	143	2.04	4409	63.04
Shellcode	755	65	8.61	41	5.43	649	85.96
Worms	87	71	81.61	3	3.45	13	14.94

5.2.2 1 Threshold vs. 2 Threshold

Table 5.7 below shows a 1 threshold structure implemented for the decision making mechanism. While the goal here is to increase attack recall by selecting a target based threshold, the threshold remains too high. It causes half of the normal traffic not being detected. Our goal is to detect attacks. However, marking half of the normal traffic as attacks significantly increases false positives. So this structure needs to be improved.

Table 5.8 1 Threshold

Class	Precision	Recall	F1-score	Support
Normal (0)	0.6239	0.5393	0.5785	22320
Attack (1)	0.8795	0.9119	0.8954	82337
Accuracy			0.8324	104657
Macro avg	0.7517	0.7256	0.7370	104657
Weighted avg	0.8250	0.8324	0.8278	104657

Due to the shortcomings of the existing 1 threshold structure, it was decided to include 2 thresholds in the newly developed structure. Thanks to the SIEM in the developed system, data can be classified as suspicious instead of just attack or normal. As seen in Table 5.5, this structure has ensured a balance in recall values between classes. It reduced false positives.

5.2.3 Online vs. Offline

The developed model was trained and tested with the UNSW-NB15 dataset. Table 5.5 shows the results obtained on the UNSW dataset.

To observe the model's performance in the real world and to increase its generalizability by testing it in other environments. The model was also tested in a virtual environment. Its performance was evaluated.

5.2.3.1 PCAP Based Offline Testing

The purpose of this test is to conduct a Proof of Concept study in a controlled environment. The goal is to determine the accuracy and suitability of the model in the real world by examining its performance under defined traffic scenarios.

The attacks were carried out by generating traffic in a virtual environment. Five different traffic scenarios were implemented. One of these scenarios is a normal traffic scenario. The other four are attack scenarios. The attack scenarios used include Fuzzers, DoS, Reconnaissance, and Analysis. All of them also present in the dataset. The process from collecting PCAP files of the traffic to obtaining the results is described in detail in the System Design section. Detailed information about the methods used to carry out the attacks is given below.

DoS: Sends a large number of HTTP requests simultaneously to the target server. Its goal is to exhaust the server's resources.

Fuzzers: Sends simultaneous requests to HTTP services on the target IP. It tries different URL paths with each request. Its goal is to find hidden directories or endpoints.

Reconnaissance: Attempts to connect to a large number of different ports from the same source IP to the destination. These connections are short lived and often incomplete. Its purpose is to identify open ports.

Analysis: It doesn't look at as many ports. It sends many packets to the ports. It tries to learn about the services or versions available on those ports.

When the results were examined, it was observed that the model classified all 5 traffic data points as attacks. It was noted that the reconstruction errors generated by these traffic data points were significantly higher than the thresholds generated by the model. This indicates that fixed thresholds are insufficient under real world traffic conditions. Therefore, threshold calibration is necessary. The threshold values need

to be readjusted according to the environment.

5.2.3.2 Real Time Evaluation on Live Traffic (Streaming Test)

The main objective of this study is to create a model that can be used in industry. A virtual system, which is a simulation of the system to be used in industry, has been designed. The created model was used to implement and test it in real-time traffic.

The Online Pipeline structure is explained in the System Design section. This section includes flow traffic scenarios. There are three states in flow traffic. The model monitors the movement in the flow traffic. It makes decisions about the traffic flow based with selected thresholds. Flows detected as alerts or warnings are directed to the SIEM (Wazuh). The model's performance in flow traffic will be evaluated.

In Scenario 1 normal traffic is run on the victim virtual machine. The reconstruction error of this traffic is calculated. Since it remains below the thresholds, it is marked as OK. The traffic is not blocked and is not sent to the SIEM.

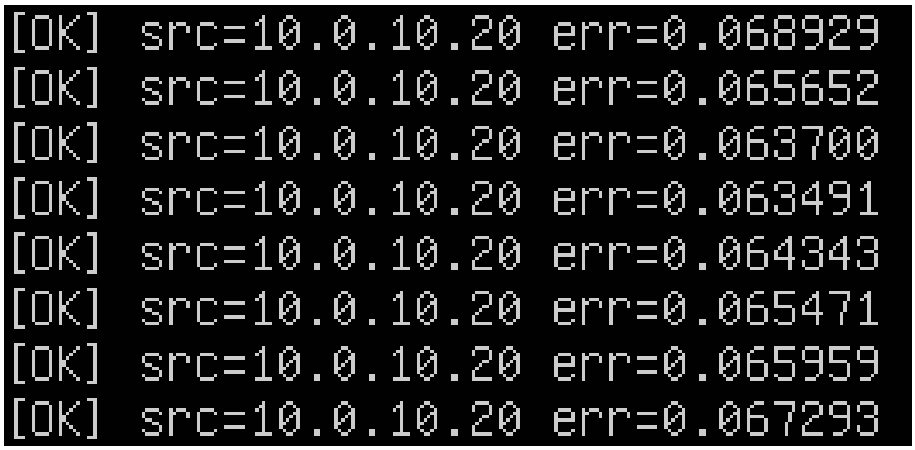

Figure 5.2 Normal Traffic

In scenario 2 attack traffic was routed from the attacker to the victim. Traffic flow suitable for both reconnaissance and analysis types was performed. This traffic remained within the selected thresholds. Therefore, it was classified as a warning. It was transmitted to the SIEM as a warning.

Figure 5.3 Warning in Wazuh

In scenario 3 attack traffic was routed from the attacker to the victim. There was traffic flow consistent with Fuzzers. This traffic exceeded thresholds. So it was sent to SIEM as an alert.

Figure 5.4 Alert in Wazuh

A two threshold structure has been implemented. According to this structure, traffic is constantly monitored. Suspicious and alert traffic is directed to the SIEM. This ensures that traffic flow is kept under control. Threshold calibration of the model is required. With threshold calibration, the model is expected to perform successfully in flowing traffic.

6
CONCLUSION

This section will evaluate the results obtained from this study and discuss planned future activities.

6.1 Evaluation

The autoencoder model was trained only with normal traffic. Normal validation was used for threshold selection. By comparing the results, potential enhancements for the system were identified. The model was further developed. Focusing on threshold selection various methods were used. This is the one of conclusions of the research. Various tools were employed. The trained model was tested on 3 systems. The ability to test the model in a simulated virtual system brought it closer to its goal of being used in industry.

One area that could be improved in this study is enhancing the traffic generated in the virtual environment. The generated traffic is short duration. Generating larger volumes and more diverse traffic will help us better understand the model's real world performance. Understanding the generalizability of the model by generating longer term and more comprehensive traffic data is area for improvement in the project.

6.2 Future Works

This section will discuss future improvements and goals related to this study.

6.2.1 Industrial Use

The primary goal of our study is to utilize the developed model in industry. To this end, the goal is to further develop the model and use it primarily in Siemens devices, specifically the RX1400 and subsequently in SRM devices.

6.2.2 Threshold Calibration

The threshold value was identified as the most important element in the model. The threshold value selection based on training data may not be compatible in other environments. In this study, it was observed to be unsuitable in a virtual environment due to field shift. Future studies goal to design a system that automatically performs threshold calibration at regular intervals.

6.2.3 2 Layer Model Structure

The model's 2 threshold structure can be improved by adding another model. The second model can only work on suspicious traffic classified as an alert. In this way, it is possible to make the data to be transmitted to the SIEM more controlled. It improves model performance by performing double verification.

6.2.4 Detection of Attack Type

It has been observed that the model exhibits limited performance against certain types of attacks. Therefore identifying the type of attack could be added. This would indicate that the model should behave differently against these attacks. Thus, by analyzing the behavior of attack types, their identification could be ensured.

6.2.5 Training with Attacks

The autoencoder model is only trained with normal traffic. There are an alternative approach. Training the model solely with attack traffic could be considered. This would allow the model to learn the behavioral basis of attacks. The model uses different types of attacks. The issue of the model failing to detect some attack types at all can be resolved.

References

[1] N. B. A and S. Sangeetha, "Intrumer: A multi module distributed explainable ids/ips for securing cloud environment," *Computers, Materials and Continua*, vol. 82, no. 1, pp. 579–607, 2025, ISSN: 1546-2218. DOI: https://doi.org/10.32604/cmc.2024.059805. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1546221825000347.

[2] L. D. Manocchio, S. Layeghy, W. W. Lo, G. K. Kulatilleke, M. Sarhan, and M. Portmann, "Flowtransformer: A transformer framework for flow-based network intrusion detection systems," *Expert Systems with Applications*, vol. 241, p. 122 564, 2024, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2023.122564. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741742303066X.

[3] J. Kim, J. Kim, H. Kim, M. Shim, and E. Choi, "Cnn-based network intrusion detection against denial-of-service attacks," *Electronics*, vol. 9, no. 6, p. 916, 2020.

[4] N. Dash, S. Chakravarty, A. Rath, N. Giri, K. Aboras, and G. N., "An optimized lstm-based deep learning model for anomaly network intrusion detection," *Scientific Reports*, vol. 15, Jan. 2025. DOI: 10.1038/s41598-025-85248-z.

[5] A. Fadul, "Anomaly detection based on isolation forest and local outlier factor," Ph.D. dissertation, May 2023. DOI: 10.13140/RG.2.2.17998.43843.

[6] S. Ali, S. U. Rehman, A. Imran, G. Adeem, Z. Iqbal, and K.-I. Kim, "Comparative evaluation of ai-based techniques for zero-day attacks detection," *Electronics*, vol. 11, no. 23, 2022, ISSN: 2079-9292. DOI: 10.3390/electronics11233934.

[7] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," *CoRR*, vol. abs/1802.09089, 2018. arXiv: 1802.09089.

[8] S. Ang, M. Ho, S. Huy, and M. Janarthanan, "Utilizing ids and ips to improve cybersecurity monitoring process," Jan. 2025. DOI: 10.63180/jcsra.thestap.2025.3.7.

[9] N. Moustafa and J. Slay, "Unsw-nb15: A comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6. DOI: 10.1109/MilCIS.2015.7348942.

[10] A. I. Bocknor Wisdom Samuels, *One-hot encoding and two-hot encoding: An introduction*, Jan. 2024. DOI: 10.13140/RG.2.2.21459.76327.

[11] Y. Song, S. Hyun, and Y.-G. Cheong, "Analysis of autoencoders for network intrusion detection," *Sensors*, vol. 21, p. 4294, Jun. 2021. DOI: 10.3390/s21134294.

Curriculum Vitae

FIRST MEMBER

Name-Surname: İrem ÇELİK
Birthdate and Place of Birth: 15.07.2003, Ankara
E-mail: irem.celik3@std.yildiz.edu.tr
Phone: 0553 677 9729
Practical Training: Siemens-Part Time Software Developer

Project System Informations

System and Software: Linux Operating System, Python, Google Colab Pro
Required RAM: 16GB
Required Disk: 16GB