Machine Learning Techniques to Detect Known and Novel Cyber-attacks

Mehrnoosh Monshizadeh
Machine Learning Techniques to Detect Known and Novel Cyber-attacks

Mehrnoosh Monshizadeh

A doctoral thesis completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Electrical Engineering, at a public examination held at Maarintie 8 the lecture hall R037/2005 TU2 of the school on 22 May 2023 at 12:00.

Aalto University
School of Electrical Engineering
Department of Communication and Networking
Network Security and Trust
Supervising professor
Professor Raimo Kantola, Aalto University, Finland

Thesis advisor
Professor Zheng Yan, Xi'dian University, China

Preliminary examiners
Professor Timo Hamalainen, University of Jyvaskyla, Finland
Professor Francesco Palmieri, University of Salerno, Italy

Opponent
Professor Jerry Chun-Wei Lin, Western Norway University of Applied Sciences, Bergen, Norway
**Author**
Mehrnosh Monshizadeh

**Name of the doctoral thesis**
Machine Learning Techniques to Detect Known and Novel Cyber-attacks

**Publisher**
School of Electrical Engineering

**Unit**
Department of Communication and Networking

**Series**
Aalto University publication series DOCTORAL THESES 31/2023

**Field of research**
Networking Technology

**Permission submitted**
27 September 2022

**Date of the defence**
22 May 2023

**Permission for public defence granted (date)**
19 December 2022

**Language**
English

**Abstract**
Intrusion detection systems are considered well-known tools for monitoring and detecting malicious traffic in communication networks. However, traditional intrusion detection systems rely on known signatures and lack the ability to detect novel attacks. Therefore, machine learning techniques are introduced to complement intrusion detection and to dynamically identify the relevant data of interest and intelligently find out the security threats. However, in order to train algorithms in machine learning based intrusion detection systems, obtaining reliable datasets with appropriate characteristics is a major challenge. Due to the lack of labelled datasets, machine learning based intrusion detection systems suffer from overfitting problem which makes them inefficient for real time intrusion detection. Furthermore, in real-life scenarios, considerable amount of incoming data does not belong to any known category; and for unknown traffic, dividing data into the classes without having information on the nature of the traffic is challenging. In addition, annotating a large dataset is very costly and hence in practice we can label only a few examples manually. On the other hand, the 5G+ and 6G networks are expected to deliver massive connectivity to numerous IoT/IIoE devices, where a huge amount of data needs to be analyzed by artificial intelligence enabled mechanisms. Consequently, a mature and scalable architecture must be considered as a mandatory objective in machine learning based intrusion detection systems. This thesis explores machine learning techniques to handle mentioned issues in the cyber-security domain. The thesis proposes an intelligent, modular, robust and scalable security solution to dynamically detect known and unknown cyber-attacks targeting mobile networks. This project takes the intrusion detection to the next level with a hybrid machine learning based mechanism namely Hybrid Anomaly Detection Model that employs a protocol analyzer and various supervised and unsupervised techniques to filter network traffic and identify malicious activities in high load communication networks. The protocol analyzer classifies and filters vulnerable protocols to avoid unnecessary computation load, the classifiers detect known cyber-attacks, while clustering algorithms use these attributes and features to detect novel attacks.

**Keywords**
Machine learning, intrusion detection, overfitting, cyber-security, cyber-attack

**ISBN (printed)**
978-952-64-1174-3

**ISBN (pdf)**
978-952-64-1175-0

**ISSN (printed)**
1799-4934

**ISSN (pdf)**
1799-4942

**Location of publisher**
Helsinki

**Location of printing**
Helsinki

**Year**
2023

**Pages**
181

**urn**
I was 6 years old; it was the first day of school, I haven't been previously in any kindergarten, it was my big day; my father was lovingly waiting for me wearing my new red shoes that war sirens echoed everywhere; it was the beginning of eight years of fear, flee from village to village and refuge in mountain as it has been said generation to generation, Kurds have no friend but the mountain. And yet every moment carrying the heavy pain in my little heart for father, my most precious, who was left behind with the other firefighters. Those days, looking at rockets and bombs falling around us, I learnt how to write to him, how to pray for him, and how to keep my hopes.

This thesis is dedicated to my parents: First to my father who from the very beginning taught me that being successful means being a good-hearted person, he also imparted to me that learning is a never-ending well. I know how proud he would have been to see me graduate. And I dedicate this work to my mother as well who originally encouraged me to pursue a doctoral degree, and with unconditional love supported me along this journey.

I would like to deeply thank my supervisor Professor Raimo Kantola who gave me guidance and freedom in my research direction, patiently providing me with numerous manuscripts and supporting my ideas. I also gladly thank my instructor Professor Zheng Yan for her continuous advice during my PhD from the day first. She valuably supported and challenged me during my study and taught me how to best write a scientific paper.

This work would not have been possible without support from a long-term colleague and a very dear friend Vikramajeet Khatri. I am thankful to him for many good moments and discussions. We have closely worked together on research approaches, implementations and writing scientific publications. Special thanks also to another very dear colleague, Serge Papillon who always welcomed me with unconditional support and valuable technical guidance.

I am grateful to Nokia that supported my PhD and provided me a friendly environment, a unique culture and the equipments to conduct my research. Furthermore, I would like to express sincere gratitude to my manager Samuel Dubus that constantly supported me and made sure I am on pathway for completing my PhD in different professional and personal situations.

Finally, I am thankful to all my colleagues at Nokia, friends, and my family for the support they have shown over the years.

Mehroosh Monshizadeh,
Paris, September 21, 2022
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Cloud computing network security</td>
<td>68</td>
</tr>
<tr>
<td>4.2</td>
<td>IoT security</td>
<td>70</td>
</tr>
<tr>
<td>5.</td>
<td>Conclusions and Future Study</td>
<td>74</td>
</tr>
<tr>
<td>5.1</td>
<td>Summary of contributions</td>
<td>76</td>
</tr>
<tr>
<td>5.2</td>
<td>Future study</td>
<td>76</td>
</tr>
<tr>
<td>References</td>
<td></td>
<td>78</td>
</tr>
<tr>
<td>Publications</td>
<td></td>
<td>87</td>
</tr>
</tbody>
</table>
Abbreviations

5G  5th Generation
6G  6th Generation
Adam Adaptive moment estimation
ADBC Associated Density Based Clustering
AI  Artificial Intelligence
ANN Artificial Neural Network
AUC Area Under the Curve
B  Botnet
BGP Border Gateway Protocol
CP  Control Plane
CVAE Conditional Variational AutoEncoder
CVAEwRF Conditional Variational AutoEncoder with Random Forest
DaaS Detection as a Service
daddr1 Destination IP address, 1st octet
daddr2 Destination IP address, 2nd octet
daddr3 Destination IP address, 3rd octet
daddr4 Destination IP address, 4th octet
DBSCAN Density Based Spatial Clustering of Applications with Noise
DDC1 Destination MAC address, 4th octet
DDC2 Destination MAC address, 5th octet
DDC3 Destination MAC address, 6th octet
DDoS Distributed Denial of Service
DM Data Mining
DMC1 Destination MAC address, 1st octet
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMC2</td>
<td>Destination MAC address, 2nd octet</td>
</tr>
<tr>
<td>DMC3</td>
<td>Destination MAC address, 3rd octet</td>
</tr>
<tr>
<td>DNN</td>
<td>Deep Neural Networks</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name System</td>
</tr>
<tr>
<td>DoS</td>
<td>Denial of Service</td>
</tr>
<tr>
<td>DSCP</td>
<td>Differentiated Service Code Point</td>
</tr>
<tr>
<td>DT</td>
<td>Decision Tree</td>
</tr>
<tr>
<td>ELM</td>
<td>Extreme Learning Machine</td>
</tr>
<tr>
<td>eNB</td>
<td>evolved Node B</td>
</tr>
<tr>
<td>FIN</td>
<td>Finish</td>
</tr>
<tr>
<td>FMI</td>
<td>Fowlkes Mallows Index</td>
</tr>
<tr>
<td>FN</td>
<td>False Negative</td>
</tr>
<tr>
<td>FNR</td>
<td>False Negative Ratio</td>
</tr>
<tr>
<td>FP</td>
<td>False Positive</td>
</tr>
<tr>
<td>FPR</td>
<td>False Positive Ratio</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>FW</td>
<td>Firewall</td>
</tr>
<tr>
<td>HADM</td>
<td>Hybrid Anomaly Detection Model</td>
</tr>
<tr>
<td>HIDS</td>
<td>Host-based Intrusion Detection System</td>
</tr>
<tr>
<td>HTTP</td>
<td>HyperText Transfer Protocol</td>
</tr>
<tr>
<td>HTTPS</td>
<td>HyperText Transfer Protocol Secure</td>
</tr>
<tr>
<td>IDS</td>
<td>Intrusion Detection System</td>
</tr>
<tr>
<td>IoD</td>
<td>Internet of Drones</td>
</tr>
<tr>
<td>IoE</td>
<td>Internet of Everything</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>IPS</td>
<td>Intrusion Prevention System</td>
</tr>
<tr>
<td>KM</td>
<td>K-means</td>
</tr>
<tr>
<td>k-NN</td>
<td>k-Nearest Neighbor</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>LI</td>
<td>Lawful Interception</td>
</tr>
<tr>
<td>LR</td>
<td>Logistic Regression</td>
</tr>
<tr>
<td>M</td>
<td>Malicious</td>
</tr>
<tr>
<td>MAC</td>
<td>Medium Access Control</td>
</tr>
<tr>
<td>ML</td>
<td>Machine Learning</td>
</tr>
<tr>
<td>MLP</td>
<td>Multi-Layer Perceptron</td>
</tr>
<tr>
<td>NFV</td>
<td>Network Function Virtualization</td>
</tr>
<tr>
<td>NIDS</td>
<td>Network-based Intrusion Detection System</td>
</tr>
<tr>
<td>NN</td>
<td>Neural Network</td>
</tr>
<tr>
<td>OPEX</td>
<td>Operation Expense</td>
</tr>
<tr>
<td>PoC</td>
<td>Proof of Concept</td>
</tr>
<tr>
<td>R2L</td>
<td>Remote to Local</td>
</tr>
<tr>
<td>RF</td>
<td>Random Forest</td>
</tr>
<tr>
<td>RFE</td>
<td>Recursive Feature Elimination</td>
</tr>
<tr>
<td>RMSProp</td>
<td>Root Mean Squared Propagation</td>
</tr>
<tr>
<td>ROC</td>
<td>Receiver Operator Characteristic</td>
</tr>
<tr>
<td>RTSP</td>
<td>Real Time Streaming Protocol</td>
</tr>
<tr>
<td>saddr1</td>
<td>Source IP address, 1st octet</td>
</tr>
<tr>
<td>saddr2</td>
<td>Source IP address, 2nd octet</td>
</tr>
<tr>
<td>saddr3</td>
<td>Source IP address, 3rd octet</td>
</tr>
<tr>
<td>saddr4</td>
<td>Source IP address, 4th octet</td>
</tr>
<tr>
<td>SDC1</td>
<td>Source MAC address, 4th octet</td>
</tr>
<tr>
<td>SDC2</td>
<td>Source MAC address, 5th octet</td>
</tr>
<tr>
<td>SDC3</td>
<td>Source MAC address, 6th octet</td>
</tr>
<tr>
<td>SDN</td>
<td>Software Defined Network</td>
</tr>
<tr>
<td>SGD</td>
<td>Stochastic Gradient Descent</td>
</tr>
<tr>
<td>SIM</td>
<td>Subscriber Identity Module</td>
</tr>
<tr>
<td>SMC1</td>
<td>Source MAC address, 1st octet</td>
</tr>
<tr>
<td>SMC2</td>
<td>Source MAC address, 2nd octet</td>
</tr>
<tr>
<td>Acronym</td>
<td>Definition</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
</tr>
<tr>
<td>SMC3</td>
<td>Source MAC address, 3rd octet</td>
</tr>
<tr>
<td>SSH</td>
<td>Secure shell</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Sockets Layer</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>SVMo</td>
<td>Support Vector Machine online</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>TLS</td>
<td>Transport Layer Security</td>
</tr>
<tr>
<td>TN</td>
<td>True Negative</td>
</tr>
<tr>
<td>TNR</td>
<td>True Negative Rate</td>
</tr>
<tr>
<td>TP</td>
<td>True Positive</td>
</tr>
<tr>
<td>TPR</td>
<td>True Positive Rate</td>
</tr>
<tr>
<td>U2R</td>
<td>User to Root</td>
</tr>
<tr>
<td>UAV</td>
<td>Unmanned Aerial Vehicles</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>UE</td>
<td>User Equipment</td>
</tr>
<tr>
<td>UP</td>
<td>User Plane</td>
</tr>
<tr>
<td>VAE</td>
<td>Variational AutoEncoder</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
</tbody>
</table>
Publications

This doctoral thesis consists of a summary and of the following publications that are referred to in the text by their numerals.


Author’s Contribution

Publication 1: An adaptive detection and prevention architecture for unsafe traffic in SDN enabled mobile networks.

The author designed the entire architecture as well as defining the implementation scenarios in this publication. The author also carried out the survey of related works and identified the research problems. The author led the writing (writing the entire paper and integrating contributions of other authors to the paper) of this publication.


The author designed the entire architecture in this publication. In order to identify research problems, the author carried out the survey of related works. Furthermore, the author defined various implementation scenarios. The author led the writing of this publication.


The author designed the entire architecture in this publication. The author also carried out the survey of related works to identify the research problems. The author analyzed various network traffic datasets to define their advantages and disadvantages and to select the best datasets among them. In addition, the author defined the execution procedures and implemented the architecture (data preprocessing and algorithms scripts). Defining the performance requirements and analyzing the results was also done by the author. The author led the writing of this publication.


The author designed the entire architecture in this publication. In order to define the research problems, the author carried out the survey of related works. The author also defined the execution scenarios, analyzed the datasets (prepossessing), reviewed the algorithms codes as well as identified the performance requirements. In addition, the author carried out the analysis of the experimental results in order to find the best algorithms. The author led the writing of this publication.
**Publication 5:** Improving Data Generalization with Variational Autoencoders for Network Traffic Anomaly Detection.

The author designed the entire architecture in this publication. The author carried out the survey of previous studies to define the research problems. The author implemented the algorithms initial scripts and contributed to dataset prepossessing. The author defined the execution scenarios and performance requirements as well as analyzed the results. The author led the writing of the publication.

**Publication 6:** A Deep Density Based and Self-determining Clustering Approach to Label Unknown Traffic.

The author designed the main architecture for the idea of employing DBSCAN. The author carried out the survey of the related works to define the research problems. Furthermore, the author defined the execution scenarios and contributed to the preparation of the datasets. The author also provided the initial codes and reviewed the scripts. In addition, the author defined the performance requirements and analyzed the results. The author also contributed to the design and analyzing of a demonstrator; the demonstrator dashboard main design, performance evaluation and architecture optimization were done by the author. The author led the writing of this publication.
1. Introduction

1.1 Background

As society becomes increasingly connected to networks, both for work and leisure, users grow more reliant than before on the availability, integrity, and confidentiality of networking services. Not only mobile devices such as smartphones are heavily dependent on secure mobile network services, but also a growing number of smart household appliances, smart utilities, smart cities, and e-health require secure connectivity. Household appliances along with home security systems, building management systems, vehicles control systems, and many electronic devices make up a growing market referred to as Internet of Things (IoT). IoT produces lots of data that needs to be collected using a network and processed. When a cellular network is used for IoT data collection, it adds the benefits of mobility of the sensors and secure connectivity. If the cellular network is based on 5G, it adds the benefit of edge computing allowing to run collected data processing close to the sensors. Thus, this leads to low delay in the processing and a significant reduction in the amount of data that needs to be carried to a remote data center. Besides, in order to upgrade the software for processing the data in an agile manner and with low cost, the industry has adopted the cloud computing paradigm. Just as it is taking an ever-wider role in the office and offering savings in operational expenditure (OPEX), it can be applied for IoT and offer the same benefits. The lower OPEX comes from better and wider scale automation than with the traditional approach, where complex software is placed in all kinds of devices and for all kinds of purposes. Although, the cloud computing presents many potential positive advancements for society, it also poses several security challenges for network service providers and operators. Cyber-attackers using malicious programs or malware can compromise an entire network, leading to service disruption, loss of data or identity theft. Accordingly, a need exists for improved systems security and methods of detecting and preventing cyber-attacks on communication networks [1].

1.2 Motivation

In order to protect networks against cyber-attacks, various defense mechanisms are typically designed to stop malicious traffic at the border of the network. While these mechanisms aim to ensure network security, there are differences
between their functionalities and the level of security they can provide. Consequently, the main motivations of this thesis are listed in the followings.

### 1.2.1 Practical demands

Virtualization of computing has been adopted in 5th Generation (5G) as basis for the networking software, where the network function virtualization (NFV) technologically adopts tools and methods from cloud computing. The term cloud computing in addition to technology also includes the business model of selling computing as a service while NFV leaves the business model open. It remains to be seen how deeply cloud computing business models will penetrate the edge computing platforms that are offered by mobile systems vendors and how 3rd party software will run on the same platforms that run the 5G network itself. In any case, NFV brings security challenges which must be addressed in a smart way. Furthermore, due to the programmability of Software Defined Network (SDN), if an attacker gains access to an SDN controller, he can exploit the whole network. The attackers may change forwarding paths and pass malicious traffic to infect the SDN enabled network [2-4]. Many studies have been conducted and several methods have been introduced to block such malicious traffic from passing through the network. Majority of these techniques detect known attacks based on the static signatures that they receive from internal processes or a third party. However, for a mobile operator it is necessary to best address the security threats and provide detection mechanism for known and unknown attacks in an intelligent way. In such case, the underlying network is based on Internet technology where the risks caused by the security treats in the domain are ever higher because of both the breadth of the impacts on society, people's lives as well the businesses and the depth of the impact [5].

With 6th Generation (6G) technology, massive number of IoT devices will be connected to the networks meaning that huge amounts of data collected from numerous devices must be analyzed in order to detect threats. On the other hand, due to extreme low latency character of next generation networks, new critical use cases will be arising for this technology. The IoT devices will be regularly deployed for critical applications such as emergency services, health, and self-driving cars, where an anomaly in the original function may mean the difference between life and death. The attacker may exploit vulnerabilities, inject malicious codes, or steal data that will result in full or partial loss of service [6]. Furthermore, Internet of Drones (IoD) is increasingly being used by both government entities as well as private enterprises for vast domain of applications such as military, smart factories and agriculture, transportation, health emergencies, etc. However, IoDs are considerably vulnerable to malicious attacks and could be employed by an attacker as targets or a vector of cyber-attack. As a result, recently a lot of attention is being paid to deploying robust, scalable, and intelligent security mechanisms to detect the anomalies in real time and before they manipulate network and interrupt services.

And finally, adversarial attacks on Artificial Intelligence (AI) systems are becoming more and more of a security concern in 6G networks. Moving toward an
intelligent network either by deploying intelligent radio, intelligent edge computing or by delivering Internet of Everything (IoE) applications, require utilizing AI as an essential component in the architecture, products, and services [7]. However, in such intelligent networks, AI is not only an enabler, but it can be employed by the attackers to launch intelligent attacks. AI-driven attacks operate at scale and become stealthier. These attacks blend into the background and imitate user behavior; due to the adaptable structure, they can switch between attack techniques and easily bypass many defense mechanisms. Hence, mitigating these attacks require more intelligent defense systems empowered by AI methods that in real time and with minimum human interaction detect the malicious input, learn the attacker behavior and weaken its process [8-11].

1.2.2 Related work

While most state-of-the-art studies have a limited focus on the scalability and robustness of the IDSs, this thesis offers a comprehensive review on the domain and proposes an optimized ML-based architecture to efficiently detect known and unknown attacks. Many studies in the field of attack detection have been focusing on the signature-based techniques, whereas some of them have been more intelligent and applied ML algorithms in their structure. These studies have employed a model of normal traffic behavior and compared the input data with the expected normal traffic; if the input was different, a signature would have been generated for the attack class to train their model with new information and for the next detection phase. However, this approach may have high false positive rate since it compares input traffic only with an expected model [12]. Other anomaly detection methods have combined their ML algorithms with honeypot to collect labelled malicious traffic. However, these techniques only rely on the received attacks by honeypot; in addition, there is a risk that honeypot node is identified to attacker [13].

Furthermore, most of the related studies have applied ML for applications other than cyber-security, whereas studies that have utilized ML for intrusion detection, have not usually evaluated the performance, scalability, and robustness [14-15]. Simultaneously, their presented solutions may have improved the detection rate in overall and only for a specific dataset while the performance has been varying considerably for another dataset and per each type of attack [16-25]. On the other hand, they have presented limited evaluation metrics rather than a detailed analysis [26-28]. Finally, rarely these studies have discussed a method to categorize unknown attacks and even if the issue has been addressed, a high detection rate has not been achieved [29].

1.3 Research problems

Intrusion Detection Systems (IDS) are considered well-known tools for monitoring and detecting malicious traffic in communication networks. However, traditional IDSs rely on known signature and lack the ability to detect novel attacks. This problem has motivated researchers to incorporate machine learning (ML) algorithms in the IDS architecture. However, in order to train algorithms
in ML-based IDSs, obtaining reliable datasets with appropriate characteristics is a major challenge. Due to the lack of labelled datasets, these methods suffer from overfitting problem which may considerably degrade their detection rate. In ML models, overfitting occurs when a model is aligned too closely to a set of data points. It means a model that is trained with a dataset may not have the same outcome with another dataset. In other words, with the same parameter setting, a model must be able to adapt to new and previously unseen data. This concept in ML domain is referred as model or data generalization. While there are manual techniques to solve to some extent the overfitting problem, yet these techniques will not be efficient for real time intrusion detection. Consequently, in the field of ML, unlabeled data analysis is one of the well-known challenges and many studies have been conducted on different techniques to solve this issue. In real-life scenarios, considerable amount of incoming data does not belong to any known category; and for unknown traffic, dividing data into the classes without having information on the nature of the traffic is challenging. Furthermore, annotating large datasets is very costly and hence we can label only few examples manually. Therefore, clustering methods are introduced to gain some insight into the structure of the data. However, clustering techniques also have some drawbacks as clusters can appear with different sizes, shapes, density, and overlapping degrees [30-31].

This thesis focuses on the problems of:
1) Can we improve the performance of IDS by employing ML algorithms?
2) Can we solve the overfitting problem raised in ML-based IDS?
3) Can we solve the problem of unlabeled data analysis and unknown attack detection surged in ML-based IDS?

To demonstrate the effectiveness of the methods for 1, 2, and 3 the thesis develops an architecture that will integrate the different mechanisms and algorithms into one system such that a robust detection of known and unknown cyber-attacks is provided.

1.4 Contributions

The goal of this thesis is to propose a robust intelligent security platform to detects known and unknown cyber-attacks targeting mobile networks. In this direction, the thesis applies a set of algorithms, integrates the applied methods into a prototype architecture and finally evaluates the usefulness and effectiveness of the employed methods. The thesis takes the intrusion detection to the next level with a hybrid ML-based mechanism namely Hybrid Anomaly Detection Model (HADM) that employs a protocol analyzer, various supervised/un-supervised techniques, and deep learning algorithms in order to satisfy the mentioned goal. The main contributions of this dissertation are given in the original publications 3-6 and summarized below.

In order to solve the computation load (with respect to detection performance) issue inherited from IDSs, current thesis introduces an ML-based IDS platform namely HADM consisting of two main parts, where each part independently increases the efficiency of attack detection. While Part 1 of the model utilizes the
protocol analyzer and ML algorithms for traffic filtering and reducing the processing time, Part 2 applies a dynamic feature extraction and combination of supervised and unsupervised methods to classify known and cluster unknown attacks. The use of protocol analyzer, layered architecture and ML algorithms allow the platform to achieve improved efficiency in terms of the detection rate and computation time. In addition, five publicly available datasets with different sizes and diverse attacks have been used to check the architecture robustness and scalability. Thus, the computation time of Part 1 is evaluated in contrast to the scenario where the protocol analyzer is not used. Furthermore, different feature selection methods are applied in conjunction with various classifiers to find the best combination to provide improved detection rate and processing time. This contribution is adopted from Publications 3 and 4 of the thesis.

Secondly, the thesis contributes to the model generalization in order to overcome well-known overfitting challenge. To solve this problem this thesis applies a variant of variational auto encoder (VAE) namely conditional VAE (CVAE). To evaluate the CVAE effectiveness and robustness, the random forest (RF) classifier is applied on the output of the CVAE. The presented proof of concept (PoC) proves that the classifier performs very well (for different datasets with various attack classes) when CVAE is applied on the feature space to solve the overfitting problem as well for dimensionality reduction. In addition, the proposed architecture classifies each type of known attack with high detection rates. Details of this contribution are presented in Publication 5.

Finally, the HADM introduces a combined and unsupervised architecture to analyze unknown data. The proposed architecture categorizes unknown data into different clusters with minimum overlapping. The model applies multiple unsupervised algorithms and a co-association matrix. This approach has the advantage of finding clusters of any shape if their elements are density-connected. This advantage is important when dealing with a clustering problem of unknown protocol messages where the shape of the clusters is uncertain. The designed architecture introduces a novel metric and method to change the feature space, hence helping to achieve very high silhouette score. The new similarity metric is different from Euclidean distance. Details of this contribution are presented in Publication 6.

It must be noted that this thesis extensively evaluates the proposed platform (HADM) robustness and scalability over several supervised and unsupervised methods, different datasets, and various attacks to deliver a proof of concept. The architecture performance is evaluated for network traffic, per packet and based on various supervised and unsupervised metrics.

To summarize, this thesis provides the followings contributions:

1) Proposing an ML-based architecture comprising of various supervised and unsupervised ML algorithms in conjunction with protocol analyzer to efficiently detect cyber-attacks. The architecture demonstrates a considerable reduction in computation time and detection ratios where the protocol analyzer is applied to split the input data.
2) Presenting a deep learning-based model to extract discriminative features from original feature space, hence solving the overfitting problem resulted from feature selection methods. The model is robust against various datasets in classifying known attacks.

3) Introducing a novel similarity metric and a robust clustering method to categorize unknown traffic is such way that clusters are distinct with minimum overlapping. Hence the security investigator can analyze few packets from each cluster and generalize the result of analysis to entire cluster.

1.5 Structure

Overall, the second chapter of this thesis reviews the relevant background about IDS and the reason to move toward data mining (DM) based security mechanisms. Furthermore, the chapter reviews the pipeline of DM in network security. The concept of ML and structure of the ML algorithms that are used in the thesis methodology are also described in this chapter. Finally, the evaluation metrics for mentioned algorithms are defined and explained.

The research methodology is proposed in Chapter 3, and it includes the relevant contribution to computation time reduction, model generalization and unknown traffic analysis. The chapter also summarizes the implementation and performance evaluation of the proposed architecture in order to achieve the research objectives. This chapter is based on the Publication 3, 4, 5 and 6.

The Chapter 4 of this thesis discusses mobile network security concerns for cloud computing networks and IoT technologies, also the chapter describes the corresponding use cases emphasizing on AI-driven and adversarial attacks. This chapter is written based on the Publication 1 and 2.

The last Chapter of this thesis concludes the study, its limitation and finally proposes research direction and future work.
2. Data Mining for Intrusion Detection

This chapter discusses the relevant background about IDS and the reasons to move toward DM-based security mechanisms. Furthermore, the chapter reviews the pipeline of DM in network security. The concept of ML and structure of the ML algorithms that are employed in the thesis methodology are presented in this chapter. Finally, the evaluation metrics for mentioned algorithms are defined and described.

2.1 Introduction

In order to protect networks against cyber-attacks, traditional defense mechanisms such as IDS and Intrusion Prevention Systems (IPS) are typically designed to stop malicious traffic at the border of the network. While these mechanisms aim to ensure network security, there are differences between their functionalities and the level of security they can provide [32]. The IDS monitors the network traffic, compares it with predefined patterns, identifies suspicious activities and informs the administrator about deviations. Upon the detecting malicious activities, the IPS applies some actions such as blocking the intruders and preventing cyber-attacks entering the network.

The IDSs are often deployed at the gateway in order to detect intrusion while the traffic still passes through the network. As it is shown in Figure 1 [33], there are various IDS mechanisms for monitoring and analyzing the events, as well to block intruder and prevent malicious activities passing through the network. However, in general two main categories are considered for IDS: The misuse detection and anomaly detection.

In misuse detection the attack signatures are predefined for the system; if the input traffic matches any of these signatures, it will be labelled as malicious. In anomaly detection, the normal behaviour (or pattern of the safe traffic) is predefined for the system and any variation of this definition will be considered as anomaly. These techniques either rely on static signature from known attacks or the fixed pattern of the safe traffic [12].

Some of the prominent IDS software are compared in the Table 1 [32]. Though some of these IDS techniques such as Suricata look promising with comprehensive features, still they lack intelligent and dynamic detection mechanism to detect unknown attacks.
Table 1. Comparison between different intrusion detection systems.

<table>
<thead>
<tr>
<th>Name</th>
<th>IDS Type</th>
<th>Technique</th>
<th>Software Type</th>
<th>Features</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>SolarWinds Security Event Manager</td>
<td>NIDS</td>
<td>Signature-based</td>
<td>Proprietary</td>
<td>Log messages generated by Linux Windows PCs and by Mac-OS, capacity of 700 rules,</td>
<td>Prone to false positive, intensive configuration, not user friendly, complicated installation, not for unknown attack detection</td>
</tr>
<tr>
<td>Bro (Zeek)</td>
<td>NIDS</td>
<td>Signature-based, Anomaly-based</td>
<td>Open source</td>
<td>Packet logging, compatible with Linux – FreeBSD - and MacOS, real-time and off-line analysis, IPV6 support</td>
<td>Complicated installation, no GUI, difficult to interpret for non-analyst, passive network traffic analyzer, not for unknown attack detection</td>
</tr>
<tr>
<td>Snort</td>
<td>NIDS</td>
<td>Signature-based</td>
<td>Open source</td>
<td>Multi-threading for packet processing, easy installation, can be installed in any environment, OS fingerprinting, packet logging</td>
<td>Application layer is not decoded, don’t scale well for new protocols, slow packet processing, not for unknown attack detection</td>
</tr>
<tr>
<td>Suricata</td>
<td>NIDS</td>
<td>Signature-based, Anomaly-based</td>
<td>Open source</td>
<td>Data collection at application layer, low level protocol monitoring, compatible with various databases and input, multi-threading</td>
<td>Complicated installation, prone to false positive, network resource intensive, not for unknown attack detection</td>
</tr>
<tr>
<td>Security Onion</td>
<td>HIDS, NIDS</td>
<td>Signature-based, Anomaly-based</td>
<td>Open source</td>
<td>Multi analysis tools, good visualization, log management</td>
<td>Compatibility issue, complicated installation, not automated, not for unknown attack detection</td>
</tr>
<tr>
<td>Open WIPS-NG</td>
<td>NIDS</td>
<td>Signature-based</td>
<td>Open source</td>
<td>Specified for wireless networks, logs, and alert,</td>
<td>Only works on Linux, not for unknown attack detection</td>
</tr>
<tr>
<td>Sagan</td>
<td>HIDS</td>
<td>Signature-based</td>
<td>Open source</td>
<td>Log analysis, compatible with data collected from other IDS tools, can be installed on different operating systems</td>
<td>Complicated installation, log analyzer rather than IDS, not for unknown attack detection</td>
</tr>
<tr>
<td>McAfee Network Security Platform</td>
<td>HIDS, NIDS</td>
<td>Signature-based</td>
<td>Proprietary</td>
<td>File blocking, access blocking, DDoS prevention, data encryption</td>
<td>Prone to false positive, high network resources intensive, not for unknown attack detection</td>
</tr>
<tr>
<td>Palo Alto Networks</td>
<td>NIDS</td>
<td>Signature-based</td>
<td>Proprietary</td>
<td>Payload analysis, active policies, SSL decryption, file blocking</td>
<td>Lack of customizability, no visibility into the signatures, not for unknown attack detection</td>
</tr>
</tbody>
</table>
2.2 Data mining for detecting cyber-attacks

DM has evolved as a well-known technology to dynamically figure out the security threats, and many studies have been conducted in this domain to detect security problems, holes, intrusions, malware, etc. DM is the process of collecting data, extracting valid and useful information from collected data, analyzing the extracted information, discovering useful patterns, dynamically learning from extracted patterns, and identifying abnormal behavior based on the learnt patterns [34].

As it is shown in Figure 2, the collected raw data must be normalized and transformed into the proper format to be usable by DM algorithms. Data cleaning for removing redundant and irrelevant information, converting the values to the right types (e.g., converting IP address to separate numeric attributes), handling missing values and normalization are some of the functions carried out in the preprocessing and transformation phases [35]. Another important task in this phase is constructing useful attributes. An example is network traffic dataset, whereas each packet carries a wide variety of irrelevant or redundant features that must be removed to improve the model performance. For this purpose, the feature selection or extraction methods are applied to find the best features from the datasets [36].

The processed data will be then forwarded to the DM algorithms for further analysis. Based on the application and objectives (e.g., labeling network traffic and identifying unusual packets), the suitable DM algorithm must be selected.

![Figure 2. Data Mining Pipeline](image)

The final step is to verify that the detected patterns and produced labels by the algorithms are valid. The evaluation process uses a test set of data on which the algorithms were not trained. The algorithm output for the test set will be compared with the expected output in order to measure the performance of the algorithm. For example, a DM algorithm trying to distinguish trojan infected packet from benign packet would be trained on a training set of network traffic including both trojan samples and safe traffic packets. Once trained, the learned patterns would be applied to the test set of the network traffic on which it had not been trained. The accuracy of the attack detection can then be measured from how many trojan infected packets the algorithm correctly detected. Based
on the application, various metrics may be used to evaluate the algorithm performance.

**Figure 3. Network traffic data collection**

Figure 3 shows network traffic data collection [37] and Figure 4 illustrates overall procedure of attack detection where DM techniques are applied. In the beginning of the process, the data will be normalized in a standard format. In the next phase, based on the attack type, important features are extracted from input data. Later, the reduced features will be used in DM analysis, to filter suspicious traffic and label data as safe traffic, new attacks or known attacks.

**Figure 4. Overall process of attack detection based on data mining.**
The network traffic consists of the header and the payload. The header is a structured data with distinct information (features) about packets. On the contrary, the payload is the actual intended data in unstructured format and with unspecific features. Consequently, in cyber-security domain, IDS can be applied on network traffic packet or flow. In packet level analysis each individual IP packet is analyzed. Whereas the flow represents aggregated information of packets that have similar header fields such as source and destination addresses, port number and protocol. In current thesis packet-based analysis is applied since this approach allows to have full information about network activities and makes real-time analysis possible [38-39]. All the analysis in current thesis is applied on the network traffic header.

Furthermore, in current thesis cyber-attack or intrusion stands for any type of action that threatens network security and stability. It can be unauthorized access to some information in network, occupying network resources, software corruption, and so on. Some mechanisms of the intrusion, that are frequently referred in this thesis are introduced in the following [40].

*Bot* is an attack tool that runs automated tasks with malicious intent such as manipulating or disrupting applications on end-users.

*Brute force* is an attack approach consisting of trying many passwords or encryption keys in order to eventually find the login credentials correctly. The credential can be further used for unauthorized access to a network.

*Denial of Service* is an attack type to congest networks leading to denial of service to the legitimate users and applications. Some examples are SYN attack where attacker overpowers the target with TCP synchronize message requests; and Ping Flood where attacker overpowers the target with Internet Control Message Protocol (ICMP) echo-requests (pings).

A *Distributed DoS (DDoS)* is type of DoS attack that uses multiple attack nodes to overwhelm the network resources. DDoS can take place either on network or packet transport layer or on the application layer. Example is HTTP attack that overpowers targeted server with HTTP requests. Other examples are FIN and RST attacks when numerous fake's terminate (or reset) messages are sent to kill a connection.

*Exploit* is an attack that takes advantage of application or system’s vulnerability to perform malicious actions. Usually, exploit is a script or part of a software.

*Fuzzer* is an automated process that injects random data in the software and programs in order to find their bugs and vulnerabilities.

*Generic* attacks run independently of the details of the targeted systems or programs implementation.

*Heartbleed* attack tricks vulnerable targets to leak their information.

*Malware* refers to all kinds of software codes that are programmed to perform malicious operations on a networked device. Backdoors, infiltration, worm, Sasser and shellcode are some of the Malware.
**Man-in-the middle** attack occurs when an attacker gains access to the communication channel established between two legitimate users. The attacker is capable of performing unauthorized activities such as intercepting and modifying communications. Examples include Browser attack where a web browser used by one of the trusted parties is eavesdropped.

**Remote to Local (R2L)** threat tries to access target machines without having an account and permissions on that machine. The access is made possible by exploiting a vulnerability and other related means. An example is File Transfer Protocol (FTP) write attack, which exploits a common anonymous FTP misconfiguration. Other examples include dictionary attacks, HyperText Transfer Protocol (HTTP) tunnel attacks and Xsnoop attacks.

**Scanning** attacks gather the targets’ information before launching the main attack. There are three types of scanning attacks: port scanning where the attacker detects open ports, network scanning where attacker discovers IP addresses and vulnerability scanning to obtain information about a target’s vulnerabilities (reconnaissance).

**User to Root (U2R)** is an attempt to get administrator or super privilege access while the user has only local access to a victim machine. A vulnerability in the victim machine is exploited in order to gain root access. An example is yaga attack, which adds the attacker to domain admins group by hacking the registry and can crash a service on the victim machine.

### 2.3 Machine learning algorithms

DM techniques apply various ML algorithms and improve their parameters according to the experience. They build a model based on the training data that will be updated during the learning process. The learning process could be supervised, semi-supervised or unsupervised [41-43].

**Supervised learning.** In this technique, labeled data is used to train the model; classification methods belong to this category. Classification is the process of dividing data into different classes. The classes are predetermined and supervised, which means the set of classes are known in advance and training data is labeled. Logistic Regression (LR), Decision tree (DT) and Random Forest (RF) are some of the supervised algorithms.

**Unsupervised learning.** Clustering algorithms belong to this category. In this technique, training data is not labeled; the characteristics of clusters are beforehand unknown, and they must be discovered in the clustering process. The clustering techniques distribute data into the groups, where the objects with similar properties go to the same group.

**Semi-supervised learning.** In this technique, unlabeled data will be labeled with the perception from a small number of labeled data points.

It must be noted that the techniques such as neural networks and deep learning algorithms can be applied for both classification and clustering operations.
Consequently, unsupervised techniques such as k-Nearest Neighbor (kNN), MultiLayer perceptron (MLP) and Extreme Learning Machine (ELM) can be applied as attack classifier while Variational AutoEncoders (VAE) can be employed with their unsupervised functionality to extract the best features from input data.

However, for real-life applications there is no single best method since different data have different characteristics and also diverse algorithms have various requirements. In spite of this fact, the supervised algorithms (such as RF) have better performance if they are used along with unsupervised algorithms (such as VAE). This implies that the best performance could be achieved when an unsupervised algorithm is used first to filter data and extract important attributes and then output of these algorithms are applied as input of supervised algorithms. Nevertheless, the complexity and scale of learning algorithms are still questionable and out of scope of this thesis [44].

2.3.1 Classification algorithms

In this section, the structure of some of the well-known classification algorithms that are applied in the thesis methodology are explained. For IDS application, these algorithms can classify various types of known cyber-attacks. These classifiers are first trained with labeled network traffic including benign and attack packets whereas they can be used later for testing and to identify malicious packets based on the trained model [45].

Decision Tree. DT is a tree-structured classifier, which is based on binary splitting features. As illustrated in Figure 5, DT comprises the internal nodes representing the features of a dataset, branches representing the decision rules and each leaf node as the model outcome [46-47].

![Figure 5. Simple Decision Tree with 5 outcomes.](image)

If the dataset consists of $n$ classes, then it will be complicated to decide in which node the classes must be located. In order to solve this problem various metrics such as entropy are introduced. The entropy measures the disorder and
evaluates the quality of the splits in decision trees. For a dataset with \( n \) classes, the entropy is a value between zero and one and calculated by Equation (2.1).

\[
E = - \sum_{i=1}^{n} p_i \log(p_i) \tag{2.1}
\]

Where, \( p_i \) is the probability of randomly selecting an example in class \( i \).

While decision trees are easily interpretable, require low computation resources and processing time, yet they are not quite stable since they are affected by small variations in the training samples [48].

**Random Forest.** RF comprises many decision trees. Each tree gives a classification, and the tree has a vote for that class. The forest chooses the classification having the most votes over all other trees. Compared to a decision tree, a random forest is considered more stable and robust against overfitting.

To classify a new sample, it is placed in each of the trees. The node probability can be calculated by the number of samples that reach the node, divided by the total number of samples. The higher the value the more important the feature. For each decision tree, the node importance is calculated with the same formula as Gini Importance (GINI). GINI determines how well the trees are split and therefore measures each feature importance [49-50].

\[
n_j^i = w_j C_j - w_{left(j)} C_{left(j)} - w_{right(j)} C_{right(j)} \tag{2.2a}
\]

where,

- \( n_j^i \) is the importance of node \( j \)
- \( w_j \) is the weighted number of samples located in node \( j \)
- \( C_j \) is the impurity value of node \( j \)
- \( left(j) \) is the child node from left split on node \( j \)
- \( right(j) \) is the child node from right split on node \( j \)

The importance for each feature in a decision tree is calculated as:

\[
f_i = \frac{\sum_{node \ splits \ feature \ i} n_j^i}{\sum_{k=1}^{all \ nodes} \frac{n_j^i}{n_k}} \tag{2.2b}
\]

where,

- \( f_i \) is the importance of feature \( i \).
- \( n_j^i \) is the importance of node \( j \).

The final feature-importance, at the random forest level is the average over all trees. The sum of importance of features on each tree is calculated and divided by total number of trees as seen in (2.2c):

\[
RF f_i = \frac{\sum_{i=1}^{all \ trees} \frac{norm \ f_{ij}}{T}}{T} \tag{2.2c}
\]

where,
- $RF_{fi}$ is the importance of feature $i$ calculated from all trees in the random forest model.
- $normf_{ij}$ is the normalized feature importance for $i$ in tree $j$.
- $T$ refers to the total number of trees.

Figure 6. Random Forest

Although random forests are more robust and to some extent reduce the overfitting problem in decision trees, yet they require more computational resources than decision tree during the training phase. Therefore, data filtration and dimensionality reduction must be applied on the input traffic in advance [30].

Multi-Layer Perceptron. An MLP is a deep, feed-forward, artificial neural network including more than one perceptron and different layers. As it is shown in the Figure 7, it includes an input layer to receive the signal (input data), an output layer to give a probability vector for predictions or only one prediction and a different number of hidden layers in order to represent the input vector in a more abstract form. A single perceptron in each layer calculates a weighted sum of the input and applies a nonlinear activation function to this weighted sum. The output of one perceptron is fed as an input to the perceptron of the next layer.

During training, MLP accepts the input $x$, forwards the information from layer to layer using its parameters $\theta$ (weights and biases) and produces an output $y'$ as well as a scalar cost $J(x; y; \theta)$ between the original class $y$ and the predicted class $y'$. With a back-propagation algorithm, it calculates the partial derivative of the cost function (gradient) with respect to its parameters. It updates weights
and biases using gradient values. The backpropagation is applied in each iteration (epoch) until the convergence of the parameters, or the convergence of test error is reached. In ML a model converges when additional iteration will not improve the model anymore. For parameter update, different gradient optimization techniques can be used such as stochastic gradient descent (SGD), momentum, root mean squared propagation (RMSProp) or adaptive moment estimation (Adam) [5, 36].

While neural networks perform well even with large number of input and even nonlinear data, yet they require quite high computation time in training phase. However, once they are trained, the testing would be fast. On the other hand, these algorithms are highly dependent on the training data which leads them to be prone to overfitting problem.

### 2.3.2 Clustering algorithms

The structure of some of the clustering algorithms is described in this section. In ML-based IDS, where the algorithms have no prior information about input traffic, clustering methods will be employed to categorize unknown traffic. These algorithms divide unknown traffic in distinct clusters in such a way that packets with similar properties go to the same group. These clusters will be further used by security investigator to detect malicious packets and generalize the investigation to the entire cluster [31].

**k means.** In this algorithm n data is divided into K clusters such that new data entry is assigned to the cluster with the highest similarity (mean); therefore, it has lower similarity to other clusters. The similarity is defined based on the distance function, which is a metric that gives the distance between each pair of points of a set. K-means algorithm has three steps: assigning the new data to the closest cluster, re-estimating the mean, and finally iteration or normalization of data [14, 44].

For observations \(x_1, x_2, \ldots, x_n\) where each observation is a d-dimensional real vector and for Euclidean distance function, k-means clustering aims to partition the n observations into k (\(\leq n\)) sets \(S = \{S_1, S_2, \ldots, S_k\}\) so as to minimize the within-cluster sum of squares.

\[
\arg \min \sum_{i=1}^{k} \sum_{x \in S_i} \|x - u_i\|^2 = \arg \min \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i
\]  

(2.3a)
Where,
\( u_i \) is the mean of points in \( S_i \). This is equivalent to minimizing the pairwise squared deviations of points in the same cluster.

\[
arg_{S} \min \frac{1}{2|S_i|} \sum_{x,y \in S_i} \|x - y\|^2
\]

This algorithm is easy to implement and understand. Also, it doesn’t require high computational resources. However, in k Means number of clusters must be defined in advance. Furthermore, changing the dataset in normalization will highly affect the results. On the other hand, the size of clusters is uniform regardless of dataset while for IDS application, the attack clusters may appear with different sizes, shapes, data sparseness, and overlapping degrees and therefore these methods are not able to identify all the cluster forms and structures encountered in real-life scenarios.

**Density-Based Spatial Clustering of Applications with Noise.** DBSCAN is a density-based clustering algorithm as its name suggests. For a clustering problem with noise, where clusters have arbitrary shapes, this algorithm groups connected regions of high density into the same cluster. The outcome of the clustering depends on two hyperparameters: the radius of the neighbourhood (the maximum distance between two samples for one to be considered in the neighbourhood of the other) further referred to as \( \epsilon \) and the minimal number of points MinPts for a region to be considered as dense. The value for epsilon is defined by calculating the distance to the nearest n points for each sample [52-53].

Let a region of the space with a radius of \( \epsilon \) centred at a point \( p \) of the dataset \( D \) be a dense region if it contains at least MinPts points. For each point \( p \), an \( \epsilon \) neighbourhood \( N_\epsilon \) can be defined as follows:

\[
N_\epsilon (p) = \{ q \in D | dist(p,q) \leq \epsilon \}
\]

where the \( \epsilon \) neighbourhood presents the set of points inside a hypersphere of radius \( \epsilon \) centred at \( p \).

A point is called a core-point if its \( \epsilon \) neighbourhood \( N_\epsilon \) contains at least MinPts points.

\[
|N_\epsilon (p)| \geq MinPts
\]

DBSCAN looks for clusters by checking the neighborhood of each object in the dataset. If the neighborhood of an object \( p \) contains more points than MinPts, a new cluster with \( p \) as the core-point is created. It then iteratively collects the objects directly density-reachable from these core points, which may involve merging multiple clusters. The process ends when no new objects can be added to any cluster.

Though DBSCAN is slower than k means, this algorithm can be very fast once it is properly implemented. The DBSCAN is a very powerful clustering algorithm and does not require to have the number of clusters predefined. DBSCAN has
the advantage of finding clusters of any shape, as long as their elements are density-connected. This advantage is important when dealing with a clustering problem of unknown protocol messages where the shape of the clusters is uncertain. However, the created clusters by DBSCAN (if applied alone) often are overlapping that makes it difficult for a security investigator to distinguish different attack categories.

**Co-Association Matrix.** This technique is like a voting mechanism that is used to combine the clustering results, leading to a new measure of similarity between samples/data points [14].

The idea behind the voting is that data points belonging to the same classes (having the same nature) are very likely to be assigned to the same cluster in different data partitions/data clustering. Taking the co-occurrences of pairs of samples/packets in the same cluster as votes for their association, the $N$ data partitions (clusters) of $n$ samples are mapped into a $n \times n$ co-association matrix:

$$C(i, j) = \frac{n_{ij}}{N} \quad (2.5)$$

where:
- $C$ is the co-association matrix.
- $N$ is the number of partitions created by the $N$ clusterings.
- $n_{ij}$ is the number of times that the pattern packet pairs $(i, j)$ is assigned to the same cluster among the $N$ partitions.

In order to create this co-association matrix, each clustering is represented by a $n \times n$ matrix ($n$ is the total number of samples) where the $(i, j)$ position is either 1 if observations $i$ and $j$ belong to the same cluster and 0 otherwise. The average of all these matrices constitutes the co-association matrix.

### 2.3.3 Feature selection and extraction algorithms

Feature selection and extraction techniques play an important role in ML-based IDS as they influence learning processes of ML algorithms. Each data set includes hundreds of features that may cause performance degradation in the detection process. To overcome this problem, feature selection or extraction methods are used to select a smaller number of features and reduce the dimension of the dataset. The selected or extracted features will be then provided to the classification and clustering algorithms in order to help them in classifying the known attacks and clustering unknown traffic.

The feature selection techniques have been widely used as an initial stage of ML-based intrusion detection techniques, yet due to the lack of labeled datasets, these methods are weak in data generalization which may considerably degrade the accuracy of the ML algorithms. While there are manual techniques such as cross-validation to solve the overfitting problem to some extent, yet they will not be efficient for real time intrusion detection. In ML models, overfitting occurs when a model is aligned too closely to a set of data points. It means a model
(e.g., feature selection method) that is trained with a dataset may not have the same outcome with another dataset [30].

Contrary to the feature selection methods that select a subset of features and eliminate the others, feature extraction techniques create new features by combining the existing features. The new reduced features preserve most of the information contained in the original features. Some examples are deep generative models and autoencoders that apply nonlinear transformation to provide a latent space of lower dimension than the input space [54].

In the following, some of the feature selection and extraction methods are described.

**Support Vector Machine online (SVMo).** Incremental SVM calculates the loss and re-trains linear SVM in every batch using stochastic gradient descent. It assigns SVM weights to each feature and selects those with highest absolute value as best discriminative features. Although SVMonline relies on linear dependency of features and labels, still it is more robust than F-Score, since it splits the dataset into small batches and calculates the average of model coefficients further increasing the robustness [30, 55-56].

**Variational Auto Encoder.** VAE is an unsupervised latent-variable-based deep generative model. VAE comprises two neural networks: an encoder network and a decoder network.

Encoder is a neural network that inputs a data point $x$ and outputs a latent representation $z$. This latent variable $z$ belongs to a latent space of lower dimension than the input space. The encoder has weights and biases $\phi$. We denote the encoder as $q(z|x; \phi)$, the distribution of the latent variable $z$.

A Decoder is a neural network that receives the latent variable $z$ as input and reconstructs $x$ from the probability distribution $p(x|z; \theta)$. The decoder has weights and biases $\theta$.

Loss function is a negative loglikelihood with a regularizer:

$$D_{KL}(q(z|x; \phi) || p(z; \theta)) - \mathbb{E}_{z \sim q}\left[\log p(x|z; \theta)\right] \quad (2.6a)$$

where,

$p(z)$ is the expected distribution (the prior) of $z$ which is specified as a standard normal distribution with mean at zero and variance at one.

An observation $x$ is assumed to be distributed according to $p(x|z; \theta \ast)$, where the decoder takes as input $z$ and outputs $p(x|z; \theta)$. The choice of this distribution depends on the type of data. Herein, we applied a multivariate gaussian distribution as it is usually used when the input data is continuous. In order to estimate $\theta$, to get the closest possible $p(x|\theta)$ to the true data distribution, the decoder can be fit by maximizing the marginal likelihood as seen in (11):

$$p(x; \theta) = \int p(x|z; \theta)p(z)dz \quad (2.6b)$$

Unfortunately, this likelihood cannot be evaluated or approximated as it is intractable. Even trying to use $p(z|x; \theta)$ will not solve this problem because $p(z|x; \theta)$ is intractable too.
Variational autoencoder model solves this problem by using variational inference which uses majorization-minimization principles to solve this optimization problem. The approach is to approximate \( p(z|x; \theta) \) using an encoder network and to use this approximation to estimate a lower bound on the marginal log-likelihood. As a result, the model will learn its parameters by maximizing this lower bound (the Evidence Lower Bound).

We consider \( q(z|x; \varphi) \) as the approximating distribution of \( p(z|x; \theta) \) where \( q(z|x) \) is a multivariate gaussian distribution. It is parametrized with the encoder that takes as input \( x \) and outputs \( q(z|x; \varphi) \).

The marginal log-likelihood of an observation \( x \) and for any variational distribution \( q(z|x; \varphi) \) over the latent variables \( z \) can be expressed as follows:

\[
\log p(x; \theta) = L(x; \varphi, \theta) + D_{KL}(q(z|x; \varphi)\|p(z|x; \theta))
\]

where,

\( L(x; \varphi, \theta) \) represents the Evidence Lower Bound (ELBO) as seen in (13):

\[
L(x; \varphi, \theta) = E_{z \sim q\theta}[\log p(x, z; \theta) - \log q(z|x; \varphi)]
\]  

(2.6d)

As it is shown in the (2.6e), the Kullback-Leibler divergence is non-negative:

\[
\log p(x; \theta) \geq L(x; \varphi, \theta) \text{ with equality only when } q(z|x; \varphi) = p(z|x; \theta) \tag{2.6e}
\]

Therefore, the objective function maximized in variational inference is:

\[
L(x; \varphi, \theta) = E_{z \sim q\theta}[\log p(x, z; \theta) - \log q(z|x; \varphi)]
\]

\[
= -D_{KL}(q(z|x; \varphi)\|p(z; \theta)) + E_{z \sim q\theta}[\log p(x|z; \theta)]
\]

(2.6f)

As it is shown in Figure 8, the ELBO has two terms. The first is the KL divergence which is a regularization term. It ensures that the encoder stays close to a standard normal distribution. The second is the reconstruction term. Even if we don’t always have an analytical expression of the ELBO, we can have an approximation of it using Monte Carlo estimate [30, 57].

Figure 8. Variational Autoencoder architecture
**Conditional Variational Auto Encoder.** CAVE is a conditional version of Variational Auto Encoder (VAE) where the decoder network takes label $y$ as an additional input in order to generate a sample that belongs to the class indicated by the label, i.e., label $y$ is concatenated with latent vector $z$. Therefore, instead of having $p(x|z; \theta)$ as the likelihood that is parametrized by the decoder, we will have $p(x|z; \theta, y)$ which is a conditional probability that depends on input label $y$.

This CVAE helps to make classes of input data more distinguishable as it forces the VAE to take class labels into account in latent space. CVAE can be seen in Figure 9 [30].

**Lambda VAE.** The architecture of the lambda VAE is a variation of VAE. It modifies the $KL$ divergence of the original VAE which is responsible for encouraging all latent embeddings to cluster around the origin. In fact, this behaviour is not desirable in our application because it can increase class overlapping in the latent space.

The idea behind lambda VAE is to change the prior $p(z)$ used in the $KL$ divergence $KL(q(\theta(z|x_i))||p(z))$ depending on the label so that each class is placed on a separate dimension in the latent space.

For example, if the input sample has a label of 2 and the dimension of the latent vector $z$ (as defined in the VAE) is 5, we will enforce the mean of the encoded latent vector of this sample to be $[0, 0, \lambda, 0, 0]$. In this way, all samples with a label of 2 will be clustered around the direction $[0, 0, 1, 0, 0]$ of the latent space. Note that the vector $[0, 0, \lambda, 0, 0]$ is an example of a lambda-hot encoding vector.

This behaviour is enforced by replacing the prior $p(z)$ in the $KL$ divergence with the lambda-hot encoding vector. So, instead of including $KL(q(\theta(z|x_i))||p(z))$ in the loss of the VAE, we replace it with $KL(q(\theta(z|x_i))||v(x))$ with $v(x)$ the lambda-hot encoding vector of the input sample $x$.

Notice that $\lambda$ is a hyperparameter that needs to be set in a way that keeps a balance between pushing each class far from the others in the latent space to avoid overlapping and pushing classes so far that the similarity between them is lost. For our experiments, we set lambda $= 5$ after testing other values [31, 58].
2.3.4 Evaluation metrics

Evaluating effectiveness and detection performance of applied algorithms requires selecting proper evaluation metrics. The definitions and evaluation metrics that are widely used in IDS and in this thesis, are described in the following [30–31, 36].

True Positive. For binary classification, $TP$ represents the malicious traffic that is correctly identified as an attack.

True Negative. For binary classification, $TN$ represents the benign traffic that is correctly identified as not malicious.

False Positive. For binary classification, $FP$ represents the safe traffic that is incorrectly identified as an attack.

False Negative. For binary classification, $FN$ represents the malicious traffic that is incorrectly identified as safe traffic.

Computation time. In real-time IDS, the CPU consumption and memory usage must be considered as an essential evaluation metric since a lower processing time than the rate of the arriving traffic will cause packet drop and performance degradation [48].

Accuracy score. This metric indicates the count of correctly predicted samples out of entire datapoints and is calculated by the following equation.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.7}$$

Precision. It determines the ability of a classifier not to wrongly label a negative sample as positive. In other words, how many of the selected objects were correct. Precision is calculated with:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.8}$$

Recall. It refers to the ability of a classifier to find all positive samples. In other words, how many of the objects that should have been selected were actually selected. Recall is calculated with:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.9}$$

$F1$ score. It is the weighted average of the precision and recall and is calculated with:

$$F1 \text{ score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.10}$$

Confusion matrix. It summarizes the performance of a classification algorithm and is represented in table format. The diagonal elements represent the number of points for which the predicted label and true label are equal, whereas non-
diagonal elements represent mislabeled prediction by classifier [59-60]. An example can be seen in Figure 10, where the x axis represents the predicted values and y axis illustrates the actual values. This figure helps in determining TP, FP, TN, and FN as well.

Figure 10. Confusion Matrix

Receiver Operating Characteristic. ROC curve shows how the number of correctly classified positive samples varies with the number of incorrectly classified negative samples and present a view of the discriminant ability of a classification model. The goal of a classifier is to be in the upper-left-hand corner in ROC space [61-62].

Figure 11. ROC curve

In the ROC graph illustrated in Figure 11, the x-axis represents the False Positive Rate (FPR), and the y-axis represents the True Positive Rate (TPR) where:
True Positive Rate = \frac{TP}{TP+FN} \quad (2.11a)

True Positive Rate is the fraction of positive examples that are correctly classified.

False Positive Rate = \frac{FP}{FP+TN} \quad (2.11b)

False Positive Rate is the fraction of negative samples that are misclassified as positive.

Area Under the ROC Curve. AUC is a measure between 0 and 1 that describes the discriminant ability of a classifier. It is the probability that a model ranks a randomly chosen positive sample higher than a randomly chosen negative one. The AUC value close to 1 means the detection results are credible [61-62].

In order to evaluate the quality of the classifier and effectiveness of correctly identifying the intrusion, below criterion is considered for AUC value:
- AUC = 1, accurate results
- AUC = [0.85, 0.95], good results
- AUC = [0.7, 0.85], general result
- AUC = [0.5, 0.7], less accurate results
- AUC = 0.5, random prediction
- AUC < 0.5, worse than random prediction

Log loss. It is also called binary cross-entropy loss or logistic loss (log loss) and is defined as the negative log-likelihood of a classification model [63-64].

Let’s consider a classification task with n classes. Suppose that \{(x_1, y_1), \ldots, (x_i, y_i), \ldots, (x_n, y_n)\} is a training dataset of n samples, where \(y_i\) is the label of the \(i^{th}\) sample \(x_i\). Suppose that \(p_i\) is a vector in which the \(j^{th}\) (element \(j \in \{1,2,\ldots,n\}\)) is the probability that sample \(x_i\) is assigned to the \(j^{th}\) class. \(T\) is ground-truth class or the class label for correct classification. Then, the log loss can be defined as the following:

\[ Log \, loss = -\frac{1}{n_{\text{samples}}} \sum_{i=1}^{n_{\text{samples}}} (y_i^T \log(p_i)) \quad (2.12) \]

Cross-entropy loss. Entry \(i\) and \(j\) in a confusion matrix are the number of observations in group \(i\), but predicted to be in group \(j\). The diagonal elements represent the number of points for which the predicted label is equal to the true label, while off-diagonal elements are those that are mislabeled by the classifier. The higher the diagonal values of the confusion matrix the better, indicating many correct predictions [36].

If the actual probability is \(p_i\) but the predicted probability is \(q_i\), each event will occur with the probability of \(p_i\) but a surprisal will be given by \(q_i\) in its formula. The weighted average surprisal, in this case, is cross-entropy loss \(c\) and it is calculated as:

\[ c = \sum_{i=0}^{n} p_i \log \left( \frac{1}{q_i} \right) \]  

(2.13a)
In the case of binary classification where we have only two classes, we name it as binary cross-entropy loss and the above formula becomes:

\[ c = n \sum_i p_i \log \left( \frac{1}{q_i} \right) = p_0 \log \left( \frac{1}{q_0} \right) + p_1 \log \left( \frac{1}{q_1} \right) \]

\[ = p_0 \log \left( \frac{1}{q_0} \right) + (1 - p_0) \left( \frac{1}{1 - q_0} \right) \]

(2.13b)

**Homogeneity.** A clustering result satisfies homogeneity if all of its clusters contain only data points which are members of a single class. This score checks clustering results, and a satisfactory homogeneity is achieved if a cluster contain only samples belonging to a single class. That is, the class distribution within each cluster should be skewed to a single class, that is, zero entropy. We determine how close a given clustering is to this ideal by examining the conditional entropy of the class distribution given the proposed clustering. Homogeneity is calculated using the following formulas [31].

\[
\begin{align*}
    h &= \begin{cases} 1 & \text{if } H(C | K) = 0 \\ 1 - \frac{H(C | K)}{H(C, K)} & \text{else} \end{cases} \quad (2.14a) \\
    H(C | K) &= -\sum_{k=1}^{[K]} \sum_{c=1}^{[C]} \frac{A_{ck}}{N} \log \frac{A_{ck}}{\sum_{c=1}^{[C]} A_{ck}} \quad (2.14b) \\
    H(C, K) &= -\sum_{k=1}^{[K]} \sum_{c=1}^{[C]} \frac{A_{ck}}{N} \log \frac{A_{ck}}{N} \quad (2.14c)
\end{align*}
\]

where,
- \( N \) is total number of data points
- \( C = \{c_i\} i = 1, \ldots, n \) is a set of labels
- \( K = \{k_i\} i = 1, \ldots, m \) is a set of clusters
- \( A \) represents the contingency table produced by the clustering algorithm, such that \( A = a_{ij} \) where that \( a_{ij} \) the number of data points that are members of label \( c_i \) and elements of cluster \( k_j \).
- \( H(C | K) \) the conditional entropy of the class distribution

Note that \( H(C | K) \) is maximal (and equals \( H(C) \)) when the clustering provides no new information — the class distribution within each cluster is equal to the overall class distribution. \( H(C | K) \) is zero when each cluster contains only members of a single class, a perfectly homogenous clustering. Therefore, when all samples in a cluster \( K \) have the same label \( C \), the homogeneity equals 1.

**Silhouette Score.** This metric shows how well clusters are apart from each other and clearly distinguished. It ranges between -1 to 1. Values near 0 indicate overlapping clusters. Negative values generally indicate that a sample has been assigned to the wrong cluster, as a different cluster is more similar [31, 65]. We define a silhouette score of a data point \( i \) as follows:
\[ \text{Silhouette score } (i) = \frac{b_i - a_i}{\max{(b_i, a_i)}} \] (2.15)

where,
- \( b_i \) represents the smallest mean distance of \( i \) to all points in any other cluster.
- \( a_i \) is the mean distance of a point to all data points from the same cluster.
- if \( b_i > a_i \), then a point is well separated from its neighboring cluster whereas it is closer to all points from the cluster it belongs to.

A silhouette score above than 0.5 is considered as reasealable, while a value below 0.2 indicates deficiency in clustering function [66].

2.4 Summary

In order to protect networks against cyber-attacks, IDSs are designed to detect malicious traffic. The IDS monitors the network traffic, compares it with predefined patterns, identifies suspicious activities and informs an administrator about deviations. Some of the IDS techniques such as misuse detection and anomaly detection were discussed in this chapter. Furthermore, the chapter compared some of the prominent IDS software. Though some of these techniques provide comprehensive functionalities, still they lack intelligent and dynamic mechanisms to detect unknown attacks. Therefore, this chapter further discussed DM technology as a complementary to the IDS to dynamically identify the relevant data of interest and intelligently figure out the security threats.

DM comprises various phases such as data collection, data preparation, data transformation, modelling and evaluation that were described thoroughly in the chapter. In the data preparation, transformation and modelling phases, DM uses ML algorithms that were later discussed in the chapter. Based on the learning process, ML algorithms are divided in different categories such as supervised and unsupervised techniques.

In supervised techniques, the labeled data is used to train the model; classification methods belong to this category. For IDS application, these algorithms can classify various types of known cyber-attacks. The classifiers are first trained with labeled network traffic including benign and attack packets whereas they can be used later for testing and to identify malicious packets based on the trained model. Hence, the chapter explained the structure of some of the well-known classification algorithms and briefly discussed their advantages and disadvantages. These algorithms will be later employed by the thesis methodology.

In unsupervised techniques, the training data is not labeled; the clustering algorithms belong to this category. In the ML-based IDS, where the algorithms have no prior information about input traffic the clustering methods will be employed to categorize unknown traffic. These algorithms divide unknown traffic in distinct clusters in such a way that packets with similar properties go to the same group. These clusters will be further used by a security investigator to detect malicious packet and generalize the investigation to the entire cluster. Consequently, the structure of some of the clustering algorithms was described in this chapter and their advantages and disadvantages were briefly discussed. These algorithms will be later employed by the thesis methodology.
A data set may include several features that cause performance degradation in detection process. To overcome this problem, feature selection or extraction methods are used in data preparation and data transformation phases of the DM pipeline to select a smaller number of features and reduce the dimension of the dataset. The selected or extracted features will be then provided to the classification and clustering algorithms in order to help them in classifying the known attacks and clustering unknown traffic. Hence the chapter described some of the feature selection and extraction methods that will be later employed by the thesis methodology.

Finally, the evaluation metrics that are widely used in IDS and in this thesis, were described in this chapter.
3. Machine Learning Based Intrusion Detection

The current chapter proposes the thesis methodology, and includes the relevant contribution to computation time reduction, model generalization and unknown traffic analysis. The chapter summarizes the implementation and performance evaluation of the proposed architecture in order to achieve the research objectives.

3.1 Introduction

As it was discussed in the previous chapter, IDSs are considered well-known tools for monitoring and detecting malicious traffic in communication networks. However, traditional IDSs rely on known signatures and lack the ability to detect novel attacks. This problem has motivated researchers to incorporate DM technology and ML algorithms in the IDS architecture to dynamically identify the relevant data of interest and intelligently detect the cyber-attacks.

However, there is no single best algorithm to be considered for detection process, since different attacks have different characteristics and diverse algorithms have various requirements. This suggests a hypothesis that the best detection performance could be achieved when a combination of algorithms is used to provide advanced performance for cyber-attack detection [44].

Furthermore, in order to train algorithms in ML-based IDSs, obtaining reliable datasets with appropriate characteristics is a major challenge. Due to the lack of labeled datasets, these methods suffer from overfitting problem which may considerably degrade their detection rate. While there are manual techniques to solve to some extent the overfitting problem, yet these techniques will not be efficient for real time intrusion detection [30].

Consequently, in the field of ML, unlabeled data analysis is one of the well-known challenges and many studies have been conducted on different techniques to solve this issue. In real-life scenarios, considerable amount of incoming data does not belong to any known category; and for unknown traffic, dividing data into the classes without having information on the nature of the traffic is challenging. On the other hand, annotating large datasets is very costly and hence we can label only few examples manually. Hence, clustering methods are introduced to gain some insight into the structure of the data. However, clustering techniques also have some drawbacks as clusters can appear with different sizes, shapes, data sparseness, and overlapping degrees [31].
Therefore, this chapter introduces an architecture that comprises network traffic flow control and combination of different algorithms to provide a scalable and robust model for detecting cyber-attacks. While the proposed architecture addresses the mentioned challenges, yet the model complexity evaluation is out of scope of this thesis.

### 3.2 Hybrid anomaly detection model

We will present HADM that is an intelligent platform that filters network traffic and identifies malicious activities in high load communication networks.

**Figure 12. Hybrid Anomaly Detection Model.**

As shown in Figure 12, the platform compromises two main parts where each one independently increases the efficiency of attack detection based on the metrics such as computation time, precision, recall and so on. Overall, the proposed model utilizes the protocol analyzer and a combination of classification and clustering algorithms with supervised and unsupervised learning processes for network traffic filtering. The protocol analyzer classifies and filters vulnerable protocols to avoid unnecessary computation load. The classifiers detect known cyber-attacks, while clustering algorithms use these attributes and features to cluster unknown traffic.

In the followings, two parts of the HADM architecture are explained. Part 1 is written based on Publication 3 and 4, while Part 2 is based on Publication 5 and 6.

**Part 1 of HADM.** As it is illustrated in Figure 13 the Part 1 of the platform utilizes protocol analyzer to filter vulnerable protocols, divides the traffic based on the carried protocol and applies ML algorithms to detect attacks at a high level but not per attack type [1, 36, 67].
Figure 13. Part 1 of Hybrid Anomaly Detection Model.

The protocol analyzer mainly consists of various modules such as decision making, counter and prioritization and Algorithm 3. A feature selection method is also used with Algorithm 3 to reduce the dimensionality of data.

Decision-making module contains a list of vulnerable protocols which are pre-defined and dynamically updated based on the feedback that is collected from log file. Some protocols such as HyperText Transfer Protocol (HTTP) are well-known vulnerable protocols while others like Real Time Streaming Protocol (RTSP) could be considered as safe protocols. Algorithm D in this module checks, whether the traffic is carried on a vulnerable protocol. Depending on the decision-making module’s outcome, the traffic will be either forwarded to counter and prioritization or the Algorithm 3.

```plaintext
for each packet do:
    if packet is encapsulated then
decapsulate packet
else
    if packet contains vulnerable protocol then
        if packet is carried over UDP then
            forward packet to Algorithm 1
        elseif packet is carried over TCP then
            forward packet to Algorithm 2
        end if
    elseif packet does not contain vulnerable protocol then
        forward packet to Algorithm 3
    end if
end if
end for
```

Figure 14. Pseudocode for algorithm D
If input traffic is carried on a protocol that is not listed as vulnerable, traffic will be sent to Algorithm 3 for analysis and reconfirmation. Every time the Algorithm 3 in the protocol analyzer detects a new vulnerable protocol, it is recorded into the log file and feedback will be sent to decision module.

The counter and prioritization module checks how frequently a protocol carries malicious packet. If the protocol carries suspicious traffic for more frequently than a threshold value, then this module considers the protocol as vulnerable and forwards the suspicious traffic to the next layer for detection and labelling.

Later, the traffic carried on the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) will be separately analyzed by Algorithm 1 and Algorithm 2. This structure not only helps to reduce the load of the model, but also helps the analysis process with having specific algorithms according to the nature of the protocols and traffic carried over it. These algorithms detect attacks on a high level but not per attack class. For dimensionality reduction and to achieve the highest efficiency in attack detection, various feature selection methods are applied with ML algorithms in this phase. Yet, due to the lack of labeled datasets, the feature selection methods suffer from data generalization which may considerably degrade the accuracy. To solve this problem in this phase, a manual cross-validation is applied to solve (to some extent) the overfitting problem. However, manual techniques will not be efficient for real time intrusion detection.

**Part 2 of HADM.** As it was described, the Part 1 of the platform detects attacks on a high level, and for determining the type of attacks, traffic will be forwarded to the Part 2 that consists of classification and clustering algorithms. As it is shown in Figure 15, the first algorithm classifies traffic into attack classes and unknown traffic. Afterward, the second algorithm creates clusters to categorize unknown traffic thus security investigator would be able to analyze the cluster and define the attacks based on the cluster composition [30].

![Figure 15. Part 2 of Hybrid Anomaly Detection Model.](image)

For dimensionality reduction and to extract discriminative features, a CVAE is employed with a RF classifier. This combination automatically learns similar-
ity among input features, provides data distribution, and finally classifies various types of attacks. CVAE introduces the labels of traffic packets into a latent space in order to better learn the changes of input samples and distinguish the data characteristics of each class. This technique avoids the confusion between classes while learning the whole data distribution. The architecture of Conditional Variational AutoEncoder with Random Forest (CVAEwRF) is illustrated in Figure 16.

![Figure 16. Conditional Variational AutoEncoder with Random Forest architecture.](image)

Furthermore, in order to efficiently detect unknown attacks, a density-based method, namely Associated Density Based Clustering (ADBC) is proposed in the Part 2 of the platform [31].

![Figure 17. Associated Density Based Clustering Architecture.](image)

As it is shown in Figure 17, ADBC utilizes several unsupervised algorithms in conjunction with a Density Based Spatial Clustering of Applications with Noise (DBSCAN) in order to categorize unknown traffic in various clusters. As hypothesis, if two packets belong to the same attack category, it is more likely that they fall into the same cluster when we apply any clustering algorithm with any parameters. And when we apply multiple clustering methods, the more they fall into the same cluster, the more likely they belong to the same attack category. Specifically, given a set of packets, a new distance measure between data points is calculated based on the mentioned assumption and is represented in a co-association matrix, which is later used for DBSCAN clustering. DBSCAN has an advantage in finding clusters of any shape, as long as the elements are density connected. This advantage is important when we deal with a clustering problem.
of unknown protocol messages because the shape of clusters is uncertain. The created clusters will be generalized to the whole dataset and analyzed by a security investigator (human or an algorithm) to identify malicious clusters.

In order to evaluate the HADM performance, in the next sections, the implementation settings and experimental results are presented.

### 3.3 Experimental results

Overall, HADM introduces three enhancements to ML-based IDSs:

- **a.** Applying protocol analyzer to reduce the load and therefore computation time with respect to architecture detection rate. This content is extracted from Publication 3 and 4.

- **b.** Applying deep learning feature extraction methods for data generalization. This content is obtained from Publication 5.

- **c.** Applying unsupervised techniques for unknown traffic analysis. This content is written based on the Publication 6.

For the experiments and to measure the model efficiency, robustness and scalability over the previous studies, this thesis compares 16 datasets that have been publicly available since 1998. Majority of mentioned datasets are small; they do not have attack or traffic diversity and are usually anonymized. Following a survey presented in Publication 4, five datasets (ISCX-2012, UNSW-15 Jan, UNSW-15 Feb, ISCX-2017 and MAWILab-2018) that have less of the mentioned limitations and meet the real traffic criteria were selected to evaluate the platform performance and to provide a proof of concept (PoC) [38, 68-70].

While the network traffic payload may have different characteristics for every dataset, this thesis only analyzes the packet headers of the network traffic datasets that consist of similar attributes and protocols. Therefore, mixing datasets (where a customized dataset was needed) has not been an issue as the data points were also close in the feature space during the experiments. Furthermore, the experiments are done on packet-level datasets that allow to have full information about network activities, making real-time analysis possible and leading to realistic evaluation of performance. Table 2 presents the employed datasets characteristics [71].

In addition to the attack list presented in Table 2, for the experiments, the attacks are grouped into three categories: Local to Remote (L2R), Remote to Local (R2L), and Local to Local (L2L). Furthermore, in mentioned datasets, some packets are considered unknown as they cannot be correlated to any of the labels provided by the owners of the datasets. The packets are considered in normal (-2), unknown (0) and attacks (1,...) classes. In the field of IDS, unknown traffic refers to any packet that is not labeled and not classified to any predefined category. The packets in the subset of datasets are randomly selected but the proportion of each class is preserved as in original dataset.

For all the experiments, 2/3 of dataset is used for training and 1/3 is used for testing the algorithms.Datasets are classified into normal, unknown and attack
(n classes of attacks). Packets that do not have any label in the dataset are presented in an unknown class for further investigation by a clustering algorithm.

**Table 2.** Comparison between different publicly available datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Year</th>
<th>Labeled</th>
<th>Protocol</th>
<th>Attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAWILab</td>
<td>2018</td>
<td>Yes</td>
<td>HTTP, HTTPS, SSH, FTP, DNS, email, BGP</td>
<td>DoS, Scan, Worm, Browser, Brute force, RST, Sasser, SYN, FIN, Ping flood and other attacks</td>
</tr>
<tr>
<td>ISCX-2012</td>
<td>2012</td>
<td>Yes</td>
<td>HTTP, SSH, FTP, email</td>
<td>DoS, Scan, Backdoor, Brute force, Browser, and other attacks</td>
</tr>
<tr>
<td>ISCX-2017</td>
<td>2017</td>
<td>Yes</td>
<td>HTTP, HTTPS, FTP, SSH, email</td>
<td>Brute force, DoS, DDoS, Infiltration, Heart-bleed, Bot, Scan</td>
</tr>
<tr>
<td>UNSW-15 Jan</td>
<td>2015</td>
<td>Yes</td>
<td>TCP, UDP, ICMP</td>
<td>Fuzzer, Backdoor, DoS, Exploit, Generic, Reconnaissance, Shellcode, Worm</td>
</tr>
<tr>
<td>UNSW-15 Feb</td>
<td>2015</td>
<td>Yes</td>
<td>TCP, UDP, ICMP</td>
<td>Fuzzer, analysis, Backdoor, DoS, Exploit, Generic, Reconnaissance, Shellcode, Worm</td>
</tr>
</tbody>
</table>

Furthermore, data cleaning, converting the columns to the right types, handling missing values, splitting IP addresses into four fields, vectorizing categorical variables, normalizing the dataset, changing the labels of attack categories in order to differentiate different attack categories are carried out in the dataset pre-processing phase. For the normalization, statistical and scaling normalization are used [35]. In order to improve the performance of the algorithms, nominal attributes such as flag and protocol type (TCP, UDP and so on) are transformed into numeric attributes. Similarly, the IP addresses and hexadecimal Medium Access Control (MAC) addresses of the applied datasets are transformed into separate numeric attributes. These separate numeric attributes are named by the dataset owner [38] as the followings:

- **SMC1**: Source MAC address, 1st octet
- **SMC2**: Source MAC address, 2nd octet
- **SMC3**: Source MAC address, 3rd octet
- **SDC1**: Source MAC address, 4th octet
- **SDC2**: Source MAC address, 5th octet
- **SDC3**: Source MAC address, 6th octet
- **DMC1**: Destination MAC address, 1st octet
- **DMC2**: Destination MAC address, 2nd octet
- **DMC3**: Destination MAC address, 3rd octet
- **DDC1**: Destination MAC address, 4th octet
DDC2: Destination MAC address, 5th octet
DDC3: Destination MAC address, 6th octet
saddr1: Source IP address, 1st octet
saddr2: Source IP address, 2nd octet
saddr3: Source IP address, 3rd octet
saddr4: Source IP address, 4th octet
daddr1: Destination IP address, 1st octet
daddr2: Destination IP address, 2nd octet
daddr3: Destination IP address, 3rd octet
daddr4: Destination IP address, 4th octet

Considered features in applied datasets are shown in Table 3.

Table 3. Features in datasets.

<table>
<thead>
<tr>
<th>No.</th>
<th>Feature</th>
<th>No.</th>
<th>Feature</th>
<th>No.</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Frame length on the wire</td>
<td>15</td>
<td>UDP source port</td>
<td>29</td>
<td>DMC1</td>
</tr>
<tr>
<td>2</td>
<td>Frame epoch time</td>
<td>16</td>
<td>UDP destination port</td>
<td>30</td>
<td>DMC2</td>
</tr>
<tr>
<td>3</td>
<td>Time delta from previous captured frame</td>
<td>17</td>
<td>TCP source port</td>
<td>31</td>
<td>DMC3</td>
</tr>
<tr>
<td>4</td>
<td>IP Differentiated Services Codepoint</td>
<td>18</td>
<td>TCP destination port</td>
<td>32</td>
<td>DDC1</td>
</tr>
<tr>
<td>5</td>
<td>IP length</td>
<td>19</td>
<td>TCP ACK flag</td>
<td>33</td>
<td>DDC2</td>
</tr>
<tr>
<td>6</td>
<td>IP identification</td>
<td>20</td>
<td>TCP push flag</td>
<td>34</td>
<td>DDC3</td>
</tr>
<tr>
<td>7</td>
<td>IP Flags</td>
<td>21</td>
<td>TCP SYN flag</td>
<td>35</td>
<td>saddr4</td>
</tr>
<tr>
<td>8</td>
<td>IP time to live</td>
<td>22</td>
<td>TCP FIN flag</td>
<td>36</td>
<td>saddr3</td>
</tr>
<tr>
<td>9</td>
<td>IP protocol</td>
<td>23</td>
<td>SMC1</td>
<td>37</td>
<td>saddr2</td>
</tr>
<tr>
<td>10</td>
<td>IP Header Checksum</td>
<td>24</td>
<td>SMC2</td>
<td>38</td>
<td>saddr1</td>
</tr>
<tr>
<td>11</td>
<td>IP More fragments</td>
<td>25</td>
<td>SMC3</td>
<td>39</td>
<td>daddr4</td>
</tr>
<tr>
<td>12</td>
<td>IP Fragment Offset</td>
<td>26</td>
<td>SDC1</td>
<td>40</td>
<td>daddr3</td>
</tr>
<tr>
<td>13</td>
<td>ICMP type</td>
<td>27</td>
<td>SDC2</td>
<td>41</td>
<td>daddr2</td>
</tr>
<tr>
<td>14</td>
<td>ICMP code</td>
<td>28</td>
<td>SDC3</td>
<td>42</td>
<td>daddr1</td>
</tr>
</tbody>
</table>

It must be noted that network packets also carry a wide variety of irrelevant or redundant features that affect the efficiency and detection rate of our algorithms. For this purpose, the feature selection and extraction methods are applied to find the best features from the datasets.

All the experiments are carried out on a server with Intel® Xeon® 16 x E5-2623 CPU @3.0GHz (4 cores in each processor), 128 GB RAM and 1.6 TB HDD. The scripts were developed in Python in a Linux environment (Ubuntu) and utilized Scikit-learn library [72]; for deep learning algorithms, Tensorflow2 and TensorFlow-probability are used [73].

In the following subsections, the experimental results for each enhancement are presented.
3.3.1 Data filtering with protocol analyzer

In the field of ML speed is a major challenge due to some heavy computation processes such as multi clustering or dimensionality reduction. The required time for running the algorithm grows higher as the number of clusters increases or for going through all iterations in order to obtain the best results [74]. While having many parameters makes an algorithm more flexible, yet the time required to process the data increases exponentially with these parameters (e.g., number of iterations, number of variants and so on). Similarly, having many features can slow down the algorithms. Furthermore, as it is shown in the Figure 18 the accuracy can only be improved by consuming more processing power and a compromise is needed to achieve the best results. Usually, linear algorithms (e.g., Logistic Regression) are simpler and faster, whereas the more complex algorithms with higher detection rate (e.g., Deep Learning) take much longer time for processing data [75].

![Figure 18. Comparison between computation time and detection rate in machine learning](image)

Consequently, Publications 3 and 4 aim to address the mentioned challenges with the following steps:

**Step 1.** Applying various algorithms to filter vulnerable protocols (e.g., UDP and TCP) from non-vulnerable protocols (e.g., streaming protocols) and processing the traffic based on the protocols with separate algorithms.

**Step 2.** Applying various feature selection techniques to select the most suitable features

**Step 3.** Applying manual cross validation technique for robustness purpose and to solve to some extent the overfitting issue.

**Step 1.** In this step, two scenarios are considered:

a. No protocol analyzer is applied, and all input traffic is forwarded to the ML algorithms. We have tested this scenario for both SVM and DT algorithms independently.

b. The protocol analyzer is applied, and the traffic carried on vulnerable protocols is split into UDP and TCP. The traffic carried over the UDP protocol
is forwarded to Support Vector Machine (SVM) algorithm and the traffic carried over TCP protocol is sent to DT algorithm for further analysis.

Figures 19 and 20 illustrate the comparison between these two scenarios with respect to computation time and detection rates. It is obvious that computation time decreased greatly in scenario two, while the accuracy, precision and recall factors are also improved. Hence, the architecture with the protocol analyzer considerably outperforms over the one without it.

These figures are the outcome of applying the architecture on a custom dataset (combining ISCX-2012 and UNSW-15), however, in Publication 3 and 4, the model performance is thoroughly evaluated for five various datasets; the complete results are presented in Tables 6 and 7.

**Figure 19.** Computation time comparison for two scenarios

**Figure 20.** Detection rate comparison for two scenarios

It must be noted that in the scenario two (protocol analyzer is applied), also Multi-Layer Perceptron (MLP) algorithm with 10 nodes is employed to analyze the traffic that is not carried over mentioned vulnerable protocols (UDP and TCP). As it is shown in Table 4, for non-vulnerable protocols different variants of MLP and Extreme Learning Machine (ELM) have been tested. Comparing the two algorithms with various metrics proved that MLP methods outperform the ELM approach. As a result, considering the HADM overall architecture with many preprocessing (protocol analyzer) and post-processing (deep learning algorithms) steps, the MLP with 10 hidden layer neurons is selected for this module to reduce the overall processing time.
**Table 4.** Protocol analyzer ML algorithm performance evaluation for custom dataset.

<table>
<thead>
<tr>
<th>Data</th>
<th>Accuracy score</th>
<th>Cross entropy error</th>
<th>False Negative score</th>
<th>Testing time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELM 10</td>
<td>0.80</td>
<td>0.84</td>
<td>2.35</td>
<td>0.72</td>
</tr>
<tr>
<td>ELM 50</td>
<td>0.58</td>
<td>3.01</td>
<td>0.95</td>
<td>1.06</td>
</tr>
<tr>
<td>MLP 10</td>
<td>0.87</td>
<td>0.66</td>
<td>0.11</td>
<td>0.09</td>
</tr>
<tr>
<td>MLP 50</td>
<td>0.88</td>
<td>0.57</td>
<td>0.12</td>
<td>0.34</td>
</tr>
</tbody>
</table>

*Step 2.* For dimensionality reduction and in order to select the best features, all of the applied algorithms are tested with four different feature selection methods. Thereby, one feature selection method is selected for each algorithm based on the achieved performance.

As it is shown in Figure 21, for MAWILab-2018 dataset SVMonline feature selection method has selected 10 features out of 33 features for DT algorithm that means 69.69% dimensionality reduction. This reduction will be beneficial in situations with a greater number of features such as extracting payload features.

![Selected features in MAWILab-2018 for DT with SVMonline.](image)

*Figure 21.* Selected features in MAWILab-2018 for DT with SVMonline.

The complete list of selected features applying various feature selection methods are shown in Table 5.

*Step 3.* In the previous step, the feature selection methods were introduced for dimensionality reduction. However, these methods usually lead to the overfitting problem.

As it is shown in Figure 22, applying same feature selection method on ISCX-2017 datasets has different outcome on selected features. This problem may also happen inside one dataset and between different classes of data.
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Algorithm</th>
<th>Feature selection</th>
<th>Selected features</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISCX-2012</td>
<td>DT / LR</td>
<td>Chi2</td>
<td>frame.len, ip.len, SMC3, tcp.srcport, saddr2, daddr2, SMC2, tcp.dstport, SDC1, daddr4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F-Score</td>
<td>SMC3, ip.len, frame.len, SMC1, daddr4, saddr4, tcp.srcport, daddr3, saddr2, ip.ttl</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SVMonline</td>
<td>daddr3, daddr4, SDC1, frame.time_epoch, tcp.dstport, SMC2, DDC1, DDC3, tcp.srcport, saddr2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RFE</td>
<td>saddr2, DDC3, SMC2, DDC1, DMC2, DMC3, tcp.flags.urg, ip.dsfield.dscp, daddr3, ip.ttl</td>
</tr>
<tr>
<td>UNSW-15</td>
<td>k-NN / SVM</td>
<td>Chi2</td>
<td>SDC2, ip.ttl, udp.dstport, saddr2, frame.time_epoch, saddr3, SMC1, daddr2, SMC3, udp.srcport</td>
</tr>
<tr>
<td>Jan</td>
<td></td>
<td>F-Score</td>
<td>saddr3, saddr2, daddr4, saddr2, daddr2, ip.ttl, SDC2, daddr1, frame.time_epoch, udp.dstport</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SVMonline</td>
<td>udp.dstport, frame.time_delta, saddr2, saddr3, frame.time_epoch, saddr4, ip.id, ip.ttl, daddr2, udp.srcport</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RFE</td>
<td>daddr2, saddr2, daddr4, udp.dstport, SDC2, frame.time_delta, ip.ttl, daddr4, ip.id, saddr4</td>
</tr>
<tr>
<td>UNSW-15</td>
<td>k-NN / SVM</td>
<td>Chi2</td>
<td>ip.flags, ip.ttl, SDC2, udp.dstport, saddr2, saddr3, SMC1, daddr2, SMC3, saddr4</td>
</tr>
<tr>
<td>Feb</td>
<td></td>
<td>F-Score</td>
<td>saddr3, ip.ttl, ip.flags, daddr4, saddr2, daddr2, saddr4, SDC2, daddr1, udp.dstport</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SVMonline</td>
<td>udp.dstport, ip.flags, frame.time_delta, saddr2, saddr3, ip.hdr_len, saddr4, daddr2, daddr3, frame.len</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RFE</td>
<td>saddr3, daddr2, udp.dstport, saddr2, ip.flags, daddr3, daddr4, saddr4, frame.time_epoch, SDC2</td>
</tr>
<tr>
<td>ISCX-2017</td>
<td>DT / LR</td>
<td>Chi2</td>
<td>DMC2, DDC1, tcp.dstport, tcp.flags.syn, DDC3, saddr3, tcp.srcport, SMC1, SDC1, DMC1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F-Score</td>
<td>saddr3, tcp.srcport, DDC3, tcp.dstport, DMC3, DMC2, DDC1, frame.len, ip.len, saddr4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SVMonline</td>
<td>tcp.dstport, daddr4, saddr4, ip.len, frame.len, ip.ttl, DDC1, frame.time_epoch, tcp.flags.push, SDC2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RFE</td>
<td>tcp.dstport, ip.len, daddr4, frame.time_delta, ip.dsfield.dscp, saddr3, saddr4, saddr1, ip.flags, DDC1</td>
</tr>
<tr>
<td>MAWILab-2018</td>
<td>DT / LR</td>
<td>Chi2</td>
<td>tcp.flags.syn, tcp.srcport, saddr1, SMC3, SDC1, SDC3, DMC2, DMC2, SMC1, saddr2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F-Score</td>
<td>saddr1, saddr2, ip.len, frame.len, tcp.srcport, tcp.dstport, saddr3, tcp.flags.syn, tcp.flags.ack, ip.flags</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SVMonline</td>
<td>ip.dsfield.dscp, ip.len, frame.len, daddr3, daddr4, saddr3, tcp.srcport, daddr2, saddr1, ip.flags</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RFE</td>
<td>ip.dsfield.dscp, ip.len, frame.len, frame.time_delta, saddr1, daddr4, daddr3, saddr2, SDC3, tcp.srcport</td>
</tr>
</tbody>
</table>
This problem is usually addressed with manual techniques such as cross validation. In this technique, ML model is trained several times by subsets of input data and as a result the best run is manually selected. This concept is illustrated in Figure 23, for Decision Tree (DT) and Logistic Regression (LR) algorithms. In this figure, each algorithm and each feature selection method have been applied on three different subsets of datasets. As a result, DT algorithm with SVMo feature selection method is selected as the best model since this combination performs the best on the third subset of the training data (cross-validation).
Table 6. SVM/DT performance evaluation for vulnerable protocol analysis (UDP/TCP)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Algorithm</th>
<th>FS method</th>
<th>FN score</th>
<th>Precision</th>
<th>Recall</th>
<th>F1 Score</th>
<th>Test time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNSW-15 Jan</td>
<td>SVM</td>
<td>Chi2</td>
<td>0.991967871</td>
<td>0.2000</td>
<td>0.00803</td>
<td>0.015444018</td>
<td>0.00061</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F-Score</td>
<td>0</td>
<td>0.5000</td>
<td>1</td>
<td>0.666666667</td>
<td>0.00056</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SVMonline</td>
<td>0.136546185</td>
<td>0.85317</td>
<td>0.86345</td>
<td>0.858283434</td>
<td>0.00056</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RFE</td>
<td>0.084337349</td>
<td>0.78621</td>
<td>0.91566</td>
<td>0.846011133</td>
<td>0.00056</td>
</tr>
<tr>
<td>UNSW-15 Feb</td>
<td>SVM</td>
<td>Chi2</td>
<td>0.039651838</td>
<td>0.60549</td>
<td>0.96035</td>
<td>0.74270755</td>
<td>0.00081</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F-Score</td>
<td>0.091876209</td>
<td>0.94183</td>
<td>0.90812</td>
<td>0.924667653</td>
<td>0.00074</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SVMonline</td>
<td>0.076402321</td>
<td>0.88100</td>
<td>0.92360</td>
<td>0.901794146</td>
<td>0.00071</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RFE</td>
<td>0.097678917</td>
<td>0.94721</td>
<td>0.90232</td>
<td>0.924219908</td>
<td>0.00090</td>
</tr>
<tr>
<td>ISCX-2012</td>
<td>DT</td>
<td>Chi2</td>
<td>0.032054299</td>
<td>0.97104</td>
<td>0.96795</td>
<td>0.969491423</td>
<td>0.01783</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F-Score</td>
<td>0.005405652</td>
<td>0.88277</td>
<td>0.99460</td>
<td>0.93535443</td>
<td>0.01167</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SVMonline</td>
<td>0.015229968</td>
<td>0.96026</td>
<td>0.98477</td>
<td>0.972360283</td>
<td>0.01282</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RFE</td>
<td>0.072885191</td>
<td>0.91735</td>
<td>0.92711</td>
<td>0.922206698</td>
<td>0.01417</td>
</tr>
<tr>
<td>ISCX-2017</td>
<td>DT</td>
<td>Chi2</td>
<td>0.031274898</td>
<td>0.99361</td>
<td>0.96873</td>
<td>0.981012113</td>
<td>0.02499</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F-Score</td>
<td>0.031274898</td>
<td>0.99361</td>
<td>0.96873</td>
<td>0.981012113</td>
<td>0.02470</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SVMonline</td>
<td>0.022266368</td>
<td>0.99964</td>
<td>0.97773</td>
<td>0.988566243</td>
<td>0.02485</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RFE</td>
<td>0.022361977</td>
<td>0.99978</td>
<td>0.97764</td>
<td>0.988586373</td>
<td>0.02921</td>
</tr>
</tbody>
</table>

Table 6 and Table 7 also show the performance evaluation for testing HADM model based on five metrics, FN score, precision, recall, F1 score and testing time. As it is shown in both tables, for UDP, though the detection rate in some results is a bit higher for k-NN still the SVM algorithm is selected considering lower computation time. And for TCP, the best performance is achieved with DT algorithm. It appears from the results that HADM did not have tremendous increase in computation time neither considerable decrease in detection factors while various datasets with different sizes and diverse attacks have been used. This means that the proposed model is scalable and robust.
Table 7. ELM/MLP performance evaluation for non-vulnerable protocol analysis.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Algorithm</th>
<th>Accuracy score</th>
<th>Cross-entropy loss</th>
<th>FN score</th>
<th>Testing time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISCX-2012</td>
<td>ELM10</td>
<td>0.53051</td>
<td>0.96743</td>
<td>0.51352</td>
<td>0.08776</td>
</tr>
<tr>
<td></td>
<td>ELM50</td>
<td>0.58761</td>
<td>0.90109</td>
<td>0.51352</td>
<td>0.14861</td>
</tr>
<tr>
<td></td>
<td>MLP10</td>
<td>0.86024</td>
<td>0.55731</td>
<td>0.14431</td>
<td>0.02163</td>
</tr>
<tr>
<td></td>
<td>MLP50</td>
<td>0.87229</td>
<td>0.58157</td>
<td>0.14590</td>
<td>0.06401</td>
</tr>
<tr>
<td>UNSW-15 Jan</td>
<td>ELM10</td>
<td>0.99452</td>
<td>0.55983</td>
<td>0</td>
<td>0.01031</td>
</tr>
<tr>
<td></td>
<td>ELM50</td>
<td>0.99452</td>
<td>0.56187</td>
<td>0</td>
<td>0.00842</td>
</tr>
<tr>
<td></td>
<td>MLP10</td>
<td>0.99452</td>
<td>0.03504</td>
<td>0</td>
<td>0.00099</td>
</tr>
<tr>
<td></td>
<td>MLP50</td>
<td>0.99452</td>
<td>0.03392</td>
<td>0</td>
<td>0.00302</td>
</tr>
<tr>
<td>UNSW-15 Feb</td>
<td>ELM10</td>
<td>0.95639</td>
<td>0.61755</td>
<td>0</td>
<td>0.03299</td>
</tr>
<tr>
<td></td>
<td>ELM50</td>
<td>0.95639</td>
<td>0.61819</td>
<td>0</td>
<td>0.04620</td>
</tr>
<tr>
<td></td>
<td>MLP10</td>
<td>0.95521</td>
<td>0.17596</td>
<td>0</td>
<td>0.00251</td>
</tr>
<tr>
<td></td>
<td>MLP50</td>
<td>0.95628</td>
<td>0.17583</td>
<td>0</td>
<td>0.00884</td>
</tr>
<tr>
<td>ISCX-2017</td>
<td>ELM10</td>
<td>0.70817</td>
<td>0.90051</td>
<td>0.10640</td>
<td>0.57638</td>
</tr>
<tr>
<td></td>
<td>ELM50</td>
<td>0.76518</td>
<td>0.80292</td>
<td>0.10640</td>
<td>1.02763</td>
</tr>
<tr>
<td></td>
<td>MLP10</td>
<td>0.78525</td>
<td>0.44047</td>
<td>0.10640</td>
<td>0.09491</td>
</tr>
<tr>
<td></td>
<td>MLP50</td>
<td>0.80387</td>
<td>0.41068</td>
<td>0.06230</td>
<td>0.12003</td>
</tr>
<tr>
<td>MAWiLab-2018</td>
<td>ELM10</td>
<td>0.66998</td>
<td>0.89468</td>
<td>0.32096</td>
<td>23.2952</td>
</tr>
<tr>
<td></td>
<td>ELM50</td>
<td>0.81291</td>
<td>0.78605</td>
<td>0.14894</td>
<td>43.9939</td>
</tr>
<tr>
<td></td>
<td>MLP10</td>
<td>0.88745</td>
<td>0.32922</td>
<td>0.10091</td>
<td>4.56287</td>
</tr>
<tr>
<td></td>
<td>MLP50</td>
<td>0.94309</td>
<td>0.29415</td>
<td>0.08888</td>
<td>15.9275</td>
</tr>
</tbody>
</table>

3.3.2 Data generalization

In the field of ML, feature selection is one of the well-known challenges. Many studies have been conducted with different techniques to solve the feature selection problem in overfitting contexts that have disastrous effects on anomaly detection performance. Figure 24 and Figure 25 show that overfitting happens both between different datasets as well in the same dataset and for different attack classes.

In recent years, particularly for imaging applications, the dual structure of Variational Autoencoders (VAE) shows promising results on data compression or reconstruction. Furthermore, efficiency of VAE techniques, can be improved by data labelling adaptation, in their conditional version. These techniques mitigate overfitting and have nice potential for data generalization. However, they
are essentially used in a black-box way and their dimensioning is not well understood. On the other hand, they are not widely used outside of the imaging domain and hardly appear in network cyber-security applications. Therefore, generalizing and assessing autoencoders’ properties in a statistical framework would potentially be a breakthrough in cyber-security applications, where the false alarm rates, detection probabilities, and classification error guarantees are still missing when classically using ML or deep learning tools [30].

Consequently, Publication 5 addresses the following issues.

a. Due to the lack of labeled datasets, feature selection methods suffer from data generalization which may considerably degrade the accuracy.

b. The manual techniques such as cross-validation that solve to some extent the overfitting problem, will not be efficient for real-time intrusion detection.

c. The deep generative models can provide a feature representation by estimating of latent space of data and are becoming popular in the different domains such as image processing, though, they hardly appear in the cyber-security area.

![Figure 24. Overfitting problem between different datasets](image-url)
To meet the mentioned challenges, a combined architecture comprising a CVAE and a RF classifier is proposed in the Figure 26. The CVAE automatically learns similarity among input features, provides data distribution to extract discriminative features from original features, and finally RF efficiently classifies various types of attacks.

The CVAE introduces the labels of traffic packets into a latent space in order to better learn the changes of input samples and distinguish the data characteristics of each class. It avoids the confusion between classes while learning the whole data distribution.
In the following experiment a Random Forest classifier, CVAE feature extraction, and MAWILab-2018 dataset is used. In this experiment, Variational Auto-Encoder reduces dimensionality in preparation for the classification algorithm. The CVAE’s encoder is used after the training and the decoder will be used only during the training. For training CVAE and to handle sparse gradients on noisy samples, a Nadam optimizer (Nesterov-accelerated Adaptive Moment Estimation) that combines Adam and Nesterov’s Accelerated Gradient (NAG) is used [76-77]. The model applied partial fit which is not implemented with Random Forest in sklearn. Furthermore, warm start that takes the first model as initialization and retrains is applied [78-79].

Figure 27. Confusion Matrix comparison for SVMo and CVAE with MAWILab-2018 dataset
Figure 27 represents normalized confusion matrices that have the recall of each class on its diagonal. By comparing these confusion matrices, it is apparent the CVAE helps the RF to distinguish all the attack classes from normal traffic and correctly classify input samples.

A more complete characterization of the combined SVMo and RF performance is the receiver operator characteristic (ROC) curves depicted in Figure 28, along with area under ROC curve (AUC) scores for all classes (unknown, normal, and attack categories). The ROC curve of a random classifier (the worst scenario) is represented (in red) in this figure as a reference. The goal of the classifier is to be in the upper-left-hand corner in ROC space for each class. In this experiment, the classifier doesn’t have a very good discrimination ability for most of the classes as shown in this figure (a). The ROC curve of the attack class 5 is close to the
curve of a random classifier. This means that the model has no discriminative capacity to distinguish class 5 from other classes.

For CVAE and RF performance the ROC curve of a random classifier (the worst scenario) is represented (in red) in this figure as a reference. It is obvious that all these ROC curves dominate the ROC curves of the SVMo and RF classifier. This can be also checked by comparing AUC scores that are better for CVAEwRF model. As a result, the problem of class 5 is totally solved, and the RF is no longer classifying the samples of the latter class randomly.

Figure 29 compares the accuracy, precision, recall, and F1 score metrics which are obtained for RF with SVMo and for CVAEwRF when they are applied to MAWILab-2018. These metrics are represented for every class (normal, unknown, and attack) and for the overall classifier (through macro-averaging). All these figures show that the class performance and the overall performance of CVAEwRF is better than the performance of SVMo with RF.

In order to evaluate the architecture robustness, in the Publication 5, the experiments are also examined for two other publicly available dataset (ISCX-2017 and ISCX-2012) that have different size and diverse attacks, and similarly the experimental results proved that the CVAEwRF provides better performance than the RF with SVMo.

![Figure 29. Detection rate comparison for SVMo and CVAE with MAWILab-2018 dataset](image-url)
### 3.3.3 Unknown traffic analysis

In the field of intrusion detection, unknown data analysis is one of the well-known challenges and many studies have been conducted on different techniques to solve this issue. In real-life scenarios as shown in Figure 30 for a publicly available dataset (MAWILab-2018), a considerable amount of incoming data does not belong to any known category and subsequently, leads to high false positive and negative ratios. Furthermore, annotating large datasets is very costly and hence we can label only a few examples manually. In addition, for unknown traffic, dividing data into the classes without having information on the nature of the traffic is challenging. Therefore, clustering methods are introduced to gain some insight into the structure of the data. However, clustering techniques also have some drawbacks. In intrusion detection, clusters can appear with different sizes, shapes, density, and overlapping degrees. Even if there are many clustering algorithms, none of them is able to identify all the cluster forms and structures encountered in real-life scenarios [14, 31].

![Figure 30. MAWILab-2018 Data distribution](image)

Consequently, Publication 4 addresses the following issues.

a. Current IDSs lack a feature to analyze unknown attack.

b. Models which employ clustering algorithms, face challenges such as overlapping since clusters can appear with different shapes, sizes and density. Furthermore, it is difficult to generalize an algorithm for different datasets with the same hyperparameters tuning.

c. Available solutions lack a sophisticated unsupervised approach to automatically define the number of clusters when analyzing unlabeled data.

To solve the mentioned challenges, a combined unsupervised architecture is proposed as shown in Figure 31. The architecture utilizes several clustering algorithms in conjunction with co-association matrix and a DBSCAN in order to categorize unknown traffic into various clusters.
This method aims to change the data space by creating a more robust distance metric which replaces the Euclidean distance. Therefore, the distance between the data points in the new space will be computed according to clustering-based distance instead of the Euclidean distance.

In order to create this distance, the first step is to apply different clustering algorithms to the input data. For instance, this thesis applies the same clustering algorithm k-means, with different numbers of clusters. In other words, it applies k-means algorithm with several values for hyperparameters to create different clusterings. In the second step, the previous clusterings are used in order to create the Co-Association Matrix. The Co-Association Matrix allows to directly deduce the distance matrix which will be used as an input for DBSCAN.

The following explains the steps of ADBC architecture [14, 80].

Applying N different clustering methods. In order to produce N clustering sets, different clustering will be applied to a subset of the input data. This step avoids the drawbacks of a single clustering technique applied to a dataset that have classes with very different densities. In fact, the application of a single clustering algorithm is never sufficient nor stable because the result varies a lot with minor changes in the hyperparameters or the input data. Therefore, in this step, different clustering algorithms or even the same algorithm with different hyperparameters (or initializations) will be applied.

Creating a co-association matrix. In this step a matrix of obtained N clustering sets will be produced. Thus, a distance matrix will be derived from this co-association matrix. The distance represented in this matrix is different from the Euclidean distance that rarely reflects the similarity between packets in real scenarios. In fact, this distance is based on the idea that the more often two packets fall into the same cluster, when several clustering algorithms are applied, the more similar they are and the more likely they belong to the same cluster. The combination process that is used to obtain the co-association matrix is based on a voting mechanism.
Providing the co-association matrix as input to DBSCAN. As the resulting distance matrix represents the input data points in a new feature space where the distance between the points is a new measure of similarity, DBSCAN will be applied directly to this distance matrix in order to detect the clusters.

Analyzing created clusters. The created clusters will be generalized to the whole dataset and analyzed by a security investigator (human or an algorithm) to identify malicious points based on clusters composition. In other words, instead of analyzing all of the packets, it will be sufficient to analyze few packets belonging to each cluster and generalize the result to the entire set of packets belonging to same cluster.

Figure 32 shows the distance matrix, which is produced from the co-association matrix for a publicly available dataset (MAWILab-2018). Both axes represent Packet ID. This distance matrix represents created clusters. The more often two data points appear in the same cluster, the smaller is the distance between them. In fact, created clusters can be detected with white areas where the distance between data points is low. This figure reveals that the proposed architecture can detect the known attack classes without using the labels. In this figure, data points are grouped into clusters (attack classes).

![Distance matrix of MAWILab-2018](image)

**Figure 32.** Distance matrix of MAWILab-2018

The Figure 33 shows the performance of the introduced architecture where DBSCAN is applied on a distance matrix deduced from a co-association matrix. These figures depict the performance of the obtained clustering for MAWILab-
2018 dataset for various metrics such as Homogeneity score, Completeness score, Fowlkes mallows score, and Silhouette score.

If the homogeneity score is high, each cluster will be mainly composed of the same attack class. However, the overall number of clusters will be high which is, in general, a drawback for the security investigator as this means that more investigation time will be needed. If the homogeneity score is less, there will be fewer clusters to investigate but there is no guarantee that they are composed mainly of the same attack. Therefore, in order to select proper evaluation metrics, finding a good compromise between the homogeneity score and the silhouette score is important.

This thesis combines both supervised and unsupervised scores to have a more complete idea about the performance of the clustering and the coherence of the insights that will be given based on the known traffic. Consequently, this thesis focuses on the silhouette score because intuitively, a high silhouette score indicates that the clusters are compact (each point is similar to the other points in its cluster) and well separated (each point is unlike the points in other clusters) which gives an idea about the clusters for both unknown and known traffic. Moreover, this thesis considers the homogeneity score which ensures that the clusters contain mainly the same data points at each time (this score doesn’t include the unknown traffic as it is a supervised score).

From Figure 3, it is obvious that the architecture provides very high silhouette score (almost 0.99) with tuning hyperparameters correctly (0.05 epsilon for different sample points).

Similarly, analysis has been done for other publicly available datasets with different size and diverse attacks in the Publication 4 in order to verify architecture robustness and scalability. The architecture still provided very high silhouette score for all applied datasets and with the same hyperparameter setting. In addition, the highest homogeneity score is achieved for all datasets and with the same hyperparameter settings.

Finally, it can be claimed that one of the main achievements of this architecture is the robustness against different datasets where with the same hyperparameter epsilon (eps=0.05), the architecture provides the highest silhouette score. These results conclude that the architecture does not need to be tuned often.

In Figure 34 the first plot shows the classes in the input space where there are no natural clusters. Therefore, a clustering algorithm like DBSCAN cannot be applied directly to this input space. Therefore, the DBSCAN is applied on different feature spaces as described earlier. In the second and third experiments, DBSCAN was applied on the latent space of a VAE with two variations, vanilla VAE (basic VAE) and lambda VAE where the groups of packets look more like clusters.

However, these clusters have very distinct densities and are overlapping. Therefore, DBSCAN is applied on a co-association matrix in the last experiment in order to solve all the mentioned problems. As it is shown in this figure, this method helps to resolve the problem of varying densities of clusters and the overlapping problem.
In Figure 34 the first plot shows the classes in the input space where there are no natural clusters. Therefore, a clustering algorithm like DBSCAN cannot be applied directly to this input space. Therefore, the DBSCAN is applied on different feature spaces as described earlier. In the second and third experiments, DBSCAN was applied on the latent space of a VAE with two variations, vanilla VAE (basic VAE) and lambda VAE where the groups of packets look more like clusters.

However, these clusters have very distinct densities and are overlapping. Therefore, DBSCAN is applied on a co-association matrix in the last experiment in order to solve all the mentioned problems. As it is shown in this figure, this method helps to resolve the problem of varying densities of clusters and the overlapping problem.

**Figure 33.** Performance of ABDC for MAWiLab-2018
The unknown traffic contained in the created clusters will be analyzed by the security investigator to identify attack behaviours. Therefore, instead of analyzing thousands of packets, only few samples will be investigated, and result would be generalized to the entire cluster. This approach helps the security investigator with reducing the time and computation resources.

In order to determine the type of activity that each cluster corresponds, each cluster centroid or mean must be checked and inspected. The provided analysis will direct security investigator’s attention to new attacks and facilitate the analysis of known attacks. Therefore, it is important to provide the information on the composition of the clusters that can accelerate investigation. The malicious clusters and consequently, the attack packets will be identified based on some characteristics (e.g., the time stamp in relation to location of generated packets and so on) by the security investigator [81]. The detected attack packets will be fed into the architecture for training purposes. Hence, the architecture will be trained periodically with new packets after a time threshold (e.g., monthly).

### 3.4 Summary

ML-based IDS techniques have been widely used to detect cyber-attacks in a more intelligent way. However, there is no single best algorithm for detection process, since different attacks have different characteristics and diverse algorithms have various requirements. This suggests that the best detection performance could be achieved when a combination of algorithms is used to provide advanced performance for cyber-attack detection. Therefore, this chapter proposed a scalable and robust architecture comprised of various supervised and unsupervised ML algorithms in conjunction with protocol analyzer to detect cyber-attacks. The achieved results in the first set of experiments demonstrated a considerable reduction in computation time where the protocol analyzer was applied to split the input data. In order to verify the scalability and robustness, the architecture detection rate was thoroughly evaluated based on various metrics applying five publicly available datasets. It appears from the results that HADM did not have tremendous increase in computation time neither considerable decrease in detection factors while various datasets with different size...
and diverse attacks have been used. This means that the proposed model is scalable and robust.

Secondly, obtaining reliable datasets to train ML algorithms is a major challenge and these algorithms suffer from overfitting problem which may considerably degrade their detection rate. In ML models, overfitting occurs when a model is aligned too closely to a set of data points. It means a model that is trained with a dataset may not have the same outcome with another dataset. Overfitting happens both between different datasets as well in the same dataset and for different attack classes. To solve the mentioned challenge, a combined architecture comprising a CVAE and a RF classifier is proposed in this chapter. The CVAE automatically learns similarity among input features, provides data distribution to extract discriminative features from original features, and finally RF efficiently classifies various types of attacks. From confusion matrices illustrated in this chapter (second set of experiments), it is apparent that the CVAE helps the RF to distinguish all the attack classes and correctly classifies input samples. A more complete characterization of discrimination ability is also obvious in provided ROC and AUC curves where all attack classes are detected with high true positive rates. Furthermore, demonstrated graphs for various metrics (such as precision, recall and so on) show very high detection rates for every class. In order to evaluate the architecture robustness, the experiments are examined for three publicly available datasets and one custom dataset. These datasets have diverse attacks. The experimental results proved that the CVAEwRF overcome overfitting problem inherited from manual feature selection techniques such as SVMo and provides high detection rates regardless of selected datasets.

Finally, in the field of ML, unlabeled data analysis is one of the well-known challenges. In real-life scenarios, considerable amount of incoming data does not belong to any known category; and for unknown traffic, dividing data into the classes without having information on the nature of the traffic is challenging. Hence, clustering methods are introduced to gain some insight into the structure of the data. However, clustering techniques also have some drawbacks such as overlapping. On the other hand, as clusters can appear with different shapes, sizes and density, it is difficult to generalize an algorithm for different dataset with the same hyperparameters tuning. Therefore, to solve the mentioned challenges, a novel and combined unsupervised approach is proposed in this thesis [31]. The architecture utilizes several clustering algorithms in conjunction with co-association matrix and a DBSCAN in order to categorize unknown traffic into various clusters in real-time. The illustrated results in this chapter (Figure 33) prove that architecture provides very high silhouette score (almost 0.99) that means clusters (attack classes) are distinct and with minimum overlapping. Therefore, this method helped to resolve the problem of varying densities of clusters and the overlapping problem. Similar analysis (applying supervised and unsupervised metrics) has been done for other datasets with different size and diverse attacks in order to verify architecture robustness and scalability. The architecture still provided very high silhouette score for all applied datasets.
Therefore, it can be claimed that one of the main achievements of this architecture is the robustness against different datasets where with the same hyperparameter (epsilon), the architecture provides the highest silhouette score. Furthermore, a cluster composition analysis is presented based on the number of packets in each cluster and according to various attack categories. This analysis is intended to give insights to the cyber-security investigator. Accordingly, it will be sufficient for security investigator to analyze a few packets belonging to each cluster and generalize the result to all packets belonging to the cluster rather than analyzing every packet. Nevertheless, the complexity and scale of learning algorithms are still questionable and out of scope of this thesis.
4. Use-cases of HADM

In the current thesis the HADM was implemented and tested both standalone and for network traffic intrusion detection. Hence, this section introduces other use cases where HADM can be utilized for intrusion detection. In the following use-cases, HADM is applied in conjunction with IoT and SDN enabled networks in order to detect malicious activities targeting these networks. The author has initial work presented in Publication 1 and Publication 2.

4.1 Cloud computing network security

According to National Institute of Standard and Technology (NIST) [82], cloud computing is a process to enable on-demand access to a shared pool of configurable resources such as storage, applications, and services, which can be rapidly provisioned and released with minimum provider interaction [4, 83-84]. The cloud computing has three layers:

*Infrastructure as a Service (IaaS)* provides virtualized infrastructures. Mobile operators can rent out their network elements, storage resources, computing system and licenses to other operators.

*Platform as a Service (PaaS)* is an interface between applications in SaaS and IaaS. This virtual platform is provided to developers for programming and web management. This programming could be related to network optimization, adding new features and so on. The main added value for PaaS comes from providing easy to use mechanism to deploy customers’ software applications to the cloud service and providing scaling for server capacity.

*Software as a Service (SaaS)* is an application layer that provides different kinds of application and software services to mobile operators when they are relying on cloud base services. The applications can be used for bandwidth control, Quality of Service (QoS) management, network configuration, system backup and so on.

In cloudified networks, SDN is an approach to separate User Plane (UP) and Control Plane (CP) in mobile networks. Based on the functionality, SDN can be placed in infrastructure layer as SDN switch or in the platform layer as SDN controller. The SDN controller can be seen as an interface offered by the mobile platform layer to the network applications. SDN switching functionality belongs to the User Plane in 3GPP 5G architecture. In cloud computing terms it belongs to the infrastructure layer [5, 85].
However, security is a key issue of cloud services provided by mobile operators. Due to cloud characteristics such as virtualization and multi-tenancy, application sharing and open-source software, the associated security threats such as authentication, information leakage and data corruption are also growing in the cloud environment. In a cloud network, SDN controller cannot mitigate malicious traffic and even if it would try, the SDN management system is missing the detection module that could identify malicious flows [6].

For cloud computing network and specifically SDN security, often IDSs are deployed at the gateway to detect intrusions but still the traffic passes through the network if it comes from a subscriber and is destined towards the internet. In contrast, in Publication 1 of this thesis, an intelligent IDS is proposed to stop the malicious traffic from entering further and polluting the network elements. This architecture combines IDS functionality with programmability feature of SDN in order to create a redundant, reliable and scalable anomaly detection system for cloud network operators. As shown in the Figure 35, the proposed architecture mainly comprises a clustering algorithm (for traffic distribution and load balancing) and several ML-based IDS nodes namely Detection as a Service (DaaS). Each DaaS node represents the main security platform of this thesis (HADM). The DaaS nodes analyze the network traffic to detect the anomalies. Upon detecting malicious packets, SDN application and consequently, the SDN controller perform required actions (e.g., Flow removal), and thus such anomaly is dropped. The proposed architecture protects SDN from being overloaded and from resource abuse attacks. In addition, applied load balancing mechanism together with clustering on the sampled traffic reduce SDN controller load, computing resource use, hence lowering the computation cost [5, 85-86].

SDN architecture has a fundamental weakness where a single CPU core controller is able to set up probably 200 – 600 flows per second. If a malicious party is able trigger the establishment of more flows/s, the controller will choke. For this reason, SDN network must be tuned and designed to be as proactive as possible: the above weakness is present in the reactive mode. Even at 100Mbps and 1000 octet messages, if each message creates reactively a new flow, the controller will have to set up more than 12 000 flows/s.

In the architecture presented in the Figure 35 upon the taken approach, every packet or traffic sample is sent to clustering node. Based on the predefined criterion such as IP address, packet length and so on, the clustering node decides to which DaaS node the packet must be forwarded for further analysis. Anomaly detection is applied in DaaS nodes and results are sent to SDN Application. SDN Application communicates with the controller to decide whether the flows will be removed (malicious) or forwarded (normal). For an intelligent and real-time anomaly detection process, DaaS nodes represent the main security platform (HADM) of this thesis utilizing DM algorithms described in the previous section of this chapter.

Cloud network providers can utilize HADM for Detection as a Service (DaaS) in conjunction with software defined networking (SDN) controller in an SDN enabled network and mitigate malicious traffic using flow control techniques. The proposed architecture will protect SDN from being overloaded and from
resource abuse attacks. In addition, applying HADM in several instances (distributed) for load balancing and as network function (NF) would reduce SDN controller load and computing resources and therefore computation cost and latency.

Figure 35. SDN Security

Furthermore, the platform aims to detect in real-time known and unknown cyber-attacks that are targeting mobile networks. The platform can be deployed in 5G core network on N6 interface (similar to SGi in LTE network) in order to detect attacks on user plane traffic. For this application, traffic mirroring will be used to analyze user plane network traffic. In order to analyze control plane and signaling messages (e.g., DoS attack by creating fake bearer channel request) the platform must be placed on e.g., N2 interface.

The N6 interface provides connectivity between the User Plane Function (UPF) and any other networks or service platforms such as the Internet. Whereas the N2 interface connects the gNodeBs to the Access and Mobility Management Function (AMF).

4.2 IoT security

With an increased data rate, lower latency and high reliability, 5th Generation (5G) and 6G networks are becoming the key technology serving Internet of Things (IoT). A particular type of IoT is Internet of Drones (IoD) connectivity with extremely fast communication requirements. Connected Unmanned Aerial Vehicles (UAVs) or drones are more and more deployed by government entities as well as private enterprises for various applications namely smart city, transportation, industry and digital health. Some of these applications carry critical missions such as military, public safety, mobility, energy and emergency services, where an anomaly in the original function may mean the difference between life and death. In these networks, delay in setting up control plane security will considerably affect the delay in sensitive applications; and therefore, it
is important how and which security mechanisms are chosen since even the lowest latency in attack detection and prevention can have considerable effect on the communication [6, 87-88].

Hence, Publication 2 of this thesis introduces an intelligent and orchestrated security platform that collects data from IoT devices, performs data analysis, detects malicious devices and finally isolates them. Performing DM on data collected from IoT devices helps to detect the anomalies before they can cause harm. This architecture is applicable to entities that use mobile networks. The proposed IoT anomaly detection module in this publication represents the main security platform of this thesis (HADM) to label new attacks and detect known ones more efficiently than existing solutions. The analysed data would be used for intrusion detection in the same or different mobile networks.

While a drone or robot has been identified as malicious in a network (belonging to a service provider), a notification will be sent to all networks (belonging to other service providers) by an IoT orchestrator to block the malicious device and avoid the need for reanalysis of malicious robot in future communication. Therefore, the distributed security platform reduces the computation processing for a device that has been already identified as malicious in other networks.

![Diagram of IoT security platform](image)

**Figure 36. Internet of Drone-Robot Security**

As it is shown in Figure 36, IoT gateway collects data from connected devices, performs the first level data monitoring and finally transfers data to the cloud (mobile network in 5G) for further analysis. If the sensors include things like video cameras, they produce lots of data, where edge cloud (e.g., on master base station) can be used to extract the important data out of the video signals and send in the hierarchy just a small fraction of data, saving lots of network capacity and cost. In each site, the IoT service provider carries out advanced analysis of collected data via IoT anomaly detection module. IoT anomaly detection module
that represents the main security platform of this thesis (HADM) utilizes DM algorithms to label the data and detect malicious devices. As malicious traffic is detected, the mitigation mechanism would be applied to block the malicious device and inform the rest of the network. The analyzed data would be used for intrusion detection in the same or different sites. While a device has been identified as malicious in a site, a notification will be sent to all sites by an IoT orchestrator to block the malicious device and in case of the drone moving to other sites, avoid the reanalysis of malicious drone in future. Though, sending to other networks does not look very important at the first site, in 5G there may be many private networks with quite limited coverage. In this case the physical movement to the area of another network is a realistic scenario. Furthermore, this capability enhances the scalability of the proposed platform and makes it an efficient distributed security framework. Therefore, the proposed platform also offers a distributed mitigation strategy for malicious drones to prevent threat on other sites. IoT application nodes provide interfaces for various domains and hold information about devices’ missions in order to prioritize data analysis.

The nature of the cyber-attacks targeting IoT network is similar to the cyber-attacks that were described in the Chapter 2 of this thesis; however, they are employed by attackers with different strategy and impact. Table 1 shows the various attack scenarios targeting IoT networks [89-90].

**Table 8. Attacks scenarios in IoT network**

<table>
<thead>
<tr>
<th>Description</th>
<th>Impact</th>
</tr>
</thead>
</table>
| Attacker taps into connection between device and mobile network | • TLS misconfigured or broken  
• Access to private & sensitive data  
• Access and modify data (sensor data, configuration data)  
• Eavesdrop on the data  
• Tamper data (e.g., remove severe alarms) |
| Attacker access to device | • Access to private & sensitive data  
• Access and modify data (sensor data, configuration data)  
• Reconfigure device without admin rights, e.g., upload data to malicious server  
• Forwarding wrong or fake data  
• Tampered software  
• Denial of service attacks on devices and unsecure servers |
| Attack on mobile network | • Fake credentials  
• Modify system configuration  
• Stealing network information  
• Malware attack on system or admin devices  
• Denial of service |

Furthermore, as it is shown in Figure 37, for IoD networks, drones can be either employed by attacker as a weapon to attack other parties (other drones, cellular network and so on) or they can be a direct target of an attacker.
Figure 37. Drone Attack scenarios

The attacker directly targets a drone in order to:

a. Spoof the drone to prob or corrupt its data.
b. Take down, lockout or take over the drone.
c. Sabotage or destruct a drone (simcard or device theft).
d. steal the drone identity.
e. Manipulate the network by sending wrong data or the location

The attacker uses a drone as a weapon in order to:

a. Steal information of the other drones or poorly protected cellular (or Wi-Fi) network.
b. Probe or corrupt other drones’ collected data.
c. Corrupt smart IoT devices e.g., home appliances, light bulbs, car charging, sensors and so on.
d. Track other drones’ location.
5. Conclusions and Future Study

Though, the IDSs are considered well-known tools for monitoring and detecting malicious traffic in communication networks, yet they rely on known signature and lack the ability to detect novel attacks. This problem has motivated researchers to incorporate ML algorithms in the IDS architecture to dynamically detect the cyber-attacks. However, there is no single best algorithm to be considered for detection process, since different attacks have different characteristics and diverse algorithms have various requirements. This suggests that the best detection performance could be achieved when a combination of algorithms is used to provide advanced performance for cyber-attack detection.

Furthermore, in order to train algorithms in ML-based IDSs, obtaining reliable datasets with appropriate characteristics is a major challenge. Due to the lack of labeled datasets, these methods suffer from the overfitting problem which may considerably degrade their detection rate. While there are manual techniques to solve to some extent the overfitting problem, yet these techniques will not be efficient for real time intrusion detection.

And lastly, in the field of ML, unlabeled data analysis is one of the well-known challenges and many studies have been conducted on different techniques to solve this issue. In real-life scenarios, considerable amount of incoming data does not belong to any known category; and for unknown traffic, dividing data into classes without having information on the nature of the traffic is challenging. On the other hand, annotating large datasets is very costly and hence we can label only few examples manually. Hence, clustering methods are introduced to gain some insight into the structure of the data. However, clustering techniques also have some drawbacks as clusters can appear with different sizes, shapes, data sparseness, and overlapping degrees.

This thesis proposes HADM that uses a combination of ML algorithms along with a protocol analyzer to achieve improved efficiency in terms of the detection rate and computation time over existing solutions. The proposed model compromises two main parts, where each part independently increases the efficiency of attack detection based on various metrics. While Part 1 of the model utilizes the protocol analyzer and ML algorithms for traffic filtration, reducing the processing time, the Part 2 applies a dynamic feature extraction and combination of supervised and unsupervised methods to classify known and cluster unknown attacks.

In order to evaluate the performance of the Part 1, five publicly available datasets with different sizes and diverse attacks have been used. The performance of the Part 1 is compared with two scenarios where UDP is separated from TCP
protocol and where there is no protocol analyzer. Furthermore, for each algorithm, four different feature selection methods such as FScore, Chi2 and SVMonline and RFE are applied to find the best features. The best algorithms were selected based on the achieved detection rate and processing time.

Nevertheless, due to the lack of labeled datasets, feature selection methods suffer from overfitting and lack of data generalization which may considerably degrade the detection rate. Overfitting happens when an algorithm’s functionality corresponds too closely to a particular dataset (which algorithm is trained with) and irrelevant to any other dataset. While there are manual techniques such as cross-validation to solve the overfitting problem to some extent, yet they will not be efficient for real-time intrusion detection. Therefore, in the Part 2 of HADM, this thesis proposes an effective deep learning method, namely CVAEwRF. In this architecture, CVAE automatically learns the similarity among input features, provides data distribution in order to extract discriminative features from original features and finally RF efficiently classifies various types of attacks. The efficiency of the proposed model is evaluated against the well-known feature selection method, SVMonline. Applying various evaluation metrics, CVAEwRF demonstrates considerable improvement in the precision (mostly above 99%), regardless of the pattern of the applied dataset. These results show that the performance of intrusion detection is highly dependent on feature representation techniques.

Moreover, not only obtaining labeled and realistic dataset to train an algorithm is a major concern in the field of ML, real-time data analysis (or unlabeled data analysis) is also another well-known challenge. To solve this problem, unsupervised techniques are introduced by many related studies. However, these techniques cannot create clusters with different sizes, shapes, data sparseness, and without overlapping degrees and therefore these state-of-the-art proposals are not able to identify all the cluster forms and structures encountered in real-life scenarios. Therefore, in the Part 2 of HADM, this thesis proposes an effective density-based method, namely ADBC to categorize unknown traffic into various clusters and provide insights based on these clusters to a cyber-security investigator and to help in detecting unknown attacks. The model applies multiple unsupervised algorithms and a co-association matrix. This approach is based on the idea that if two packets belong to the same attack category, it is more likely that they fall into the same cluster when we apply any clustering algorithm with any parameters. Hence, multiple clusterings is applied in this architecture to check whether more often a datapoint falls into the same cluster by using different clusterings; this means more likely they belong to the same attack category. Specifically, given a set of packets, a new distance measure between data points is calculated based on the mentioned assumption and is represented in a co-association matrix, which is later used for DBSCAN clustering. DBSCAN has the advantage of finding clusters of any shape, as long as their elements are density-connected. This advantage is important when dealing with a clustering problem of unknown protocol messages where the shape of the clusters is uncertain. The architecture performance is evaluated based on the combination of supervised and unsupervised metrics and resulted in a high silhouette score regardless of
applied dataset. Furthermore, a cluster composition analysis is presented based on the number of packets in each cluster and according to various attack categories. This analysis is intended to give insights to the cyber-security investigator.

5.1 Summary of contributions

The main outcomes of this thesis are summarized in the followings.

We designed a robust AI-based IDS model comprising of protocol analyzer and various supervised and unsupervised algorithms to efficiently detect cyber-attacks in real time. Applying protocol analyzer considerably decreased the computation time.

We designed a CVAE algorithm for data generalization and in order to solve overfitting problem. The CVAE is a conditional VAE which means its hyperparameters can be designed based on the application; in this thesis the CVAE is optimized for data generalization. Rather than the prior art that solely focuses on the classification, we have applied RF classifier on the output of our CVAE to evaluate its effectiveness and robustness. Our PoC proved that the classifier performs very well for different datasets with various attack classes.

We designed a combined clustering method to analyze unknown data and detect new attacks. The designed architecture introduces a novel method (metric) to change the feature space, hence helping to achieve very high silhouette score. The new similarity metric is different from Euclidean distance. The robustness of the architecture is evaluated based on various supervised and unsupervised metrics and against different datasets with diverse attacks.

5.2 Future study

While we argue that this thesis presents an important enhancement to the present-time IDSs, it is also necessary to consider the future challenges and directions concerning the thesis topics.

Each user can make up their own dataset useful for their applications, yet generalizing the results however requires publicly available datasets. For example, applications such as IoT drones and robots, eHealth and lawful interception require different types of data than mobile network traffic and usually due to high level of privacy can be obtained only from publicly available resources that are rare in the domain. Therefore, advanced anonymization methods must be designed and applied on the data collected from these devices while conserving the most meaningful features. Otherwise, unconventional data generators must be designed to generate datasets that resemble real-life data.

The study shows for classes that have very few samples, the class imbalance stays a critical challenge. As it became evident for a class that does not have enough samples the classifier is not capable of learning the pattern of class correctly and classifies samples of the skewed class randomly. The mentioned class imbalance issue and a method to overcome the challenge will be addressed in our future work.
In addition, the complexity of the HADM will be evaluated in our future work as well the model performance will be assessed in detecting adversarial attacks.
References


