UNIVERSITÀ
DEGLI STUDI
DI BRESCIA

DIPARTIMENTO DI INGEGNERIA
DELL'INFORMAZIONE

Corso di Laurea Magistrale

in Ingegneria Informatica

A CRITICAL ANALYSIS OF THE IDS/IPS LITERATURE RELATING TO OT / IOT NETWORKS

Professore:

Chiar.mo Prof. Cerutti Federico

Studente:
Andreoli Gregorio

Matricola n. 719045

Anno Accademico 2021/2022

TABLE OF CONTENTS

List of symbols and abbreviations

Abbreviation	Explanation
IDS	Intrusion detection system
IPS	Intrusion prevention system
IoT	Internet of things
OCT	Operational Technology
SIDS	Signature-based intrusion detection systems
AIDS	Anomaly-based Intrusion Detection System
NIDS	Network-based Intrusion detection system

Chapter 1: introduction

1.1 Context

One of the most critical parts of a manufacturing company's infrastructure from a cybersecurity point of view is the OT and IoT networks, mainly because the Internet of Things is notoriously insecure, but not because the technology to build or protect it is immature but because the security in industrial IoT suffers from a lack of willpower. Neither the buyers nor the technology providers have yet invested the same efforts in other areas of the tech world to create and take measures that make everyone safer. Many technical issues contribute to the fragmented and uncoordinated nature of security in the IIoT market, but the focus and security habits of the end-users in charge of projects emerge in almost all discussions about it.

Traditional IT staff deal with end users, tackle malware and Infosec issues and go back to dealing with multiple users, who are a constant source of access for attackers.

Operational Technology personnel come from a very different environment, one that has been automated and digitally controlled as part of an industrial control system whose Internet connections would have been carefully filtered or non-existent. To reduce the risk in these types of environments, various solutions can be evaluated, some of which provide mechanisms for protection, prevention and monitoring. The current market offers solutions more focused on IT network protection and monitoring than the OT / IoT network.

1.2 Purposes

This paper presents a critical review of the literature for IDS / IPS with a focus on IoT that can be relied upon to develop an intrusion detection system to monitor OT / IoT networks.

The content of this paper is divided into three sections: the first is indicated in chapter 2 of the report and includes a complete view of the literature on IDS in the IoT field, and the second section includes an analysis of the open-source software, which is reported on an excel sheet, and the conclusions are reported in Chapter 3 of the word document.

Chapter 2: literature review

Figure 1 Classification of IDSs for IoT shows the IDS techniques, deployment strategy and validation strategy covered by the research papers.

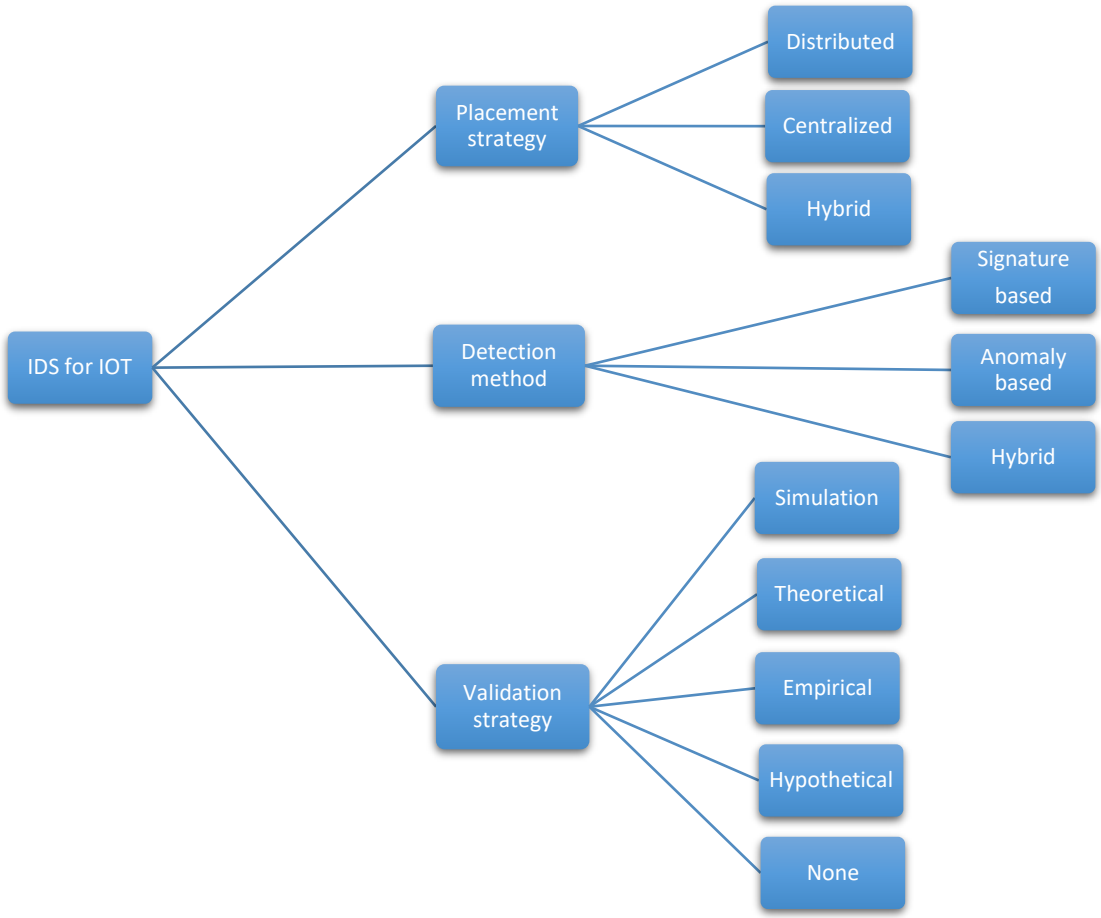

Figure 1 Classification of IDSs for IoT

2.1 IoT intrusion detection systems methods

Any unauthorized action or activity that damages the IoT ecosystem is known as an "IoT intrusion." In other words, any attack that compromises the availability, confidentiality, or integrity of information is regarded as an intrusion.

An IDS is a software or hardware device that monitors computer systems for malicious activity to keep the system secure. IDS' primary goal is to detect harmful network traffic and unauthorized computer use, which is impossible with a conventional firewall.

IDS system has two main sub-categories: Signature-based Intrusion Detection System (SIDS) and Anomaly-based Intrusion Detection System (AIDS).

2.1.1 Signature intrusion detection systems

Signature intrusion detection systems (SIDS) utilize pattern matching techniques to find a known attack; these are also known as Knowledge-based Detection or Misuse Detection. In SIDS, matching methods are used to find a previous intrusion. In other words, when an intrusion signature matches the signature of a previous intrusion that already exists in the signature database, an alarm signal is triggered. The fundamental concept is to create a database of intrusion signatures, compare the current set of actions to the signatures already in use, and raise the alarm in the event of a match. If (source IP address=destination IP address) then label as an attack, for instance, may result from a rule written in the format "if antecedent -then: consequent".

For previously known incursions, SIDS typically provides good detection accuracy. However, until the new attack's signature is extracted and saved, SIDS cannot identify zero-day attacks because there isn't a matching signature in the database. Numerous widely used tools, including Snort and NetSTAT, use SIDS.

2.1.2 Anomaly-based Intrusion Detection System

In an Anomaly-based intrusion detection system, a normal model of the behaviour of a computer system is created using machine learning, statistical-based or knowledge-based methods. Anomalies are considered to be intrusions when there is a sizable difference between the observed behaviour and the model.

This method takes advantage of the fact that malicious behaviour is different from the user's usual behaviour. The key benefit of AIDS is its capacity to detect zero-day attacks because it does not require a signature database to identify unusual user activity. When the observed behaviour departs from what is expected, AIDS sends out a warning signal.

However, AIDS can lead to a significant false-positive rate because anomalies may only be brand-new typical behaviours as opposed to actual intrusions.

The distinctions between anomaly-based and signature-based detection are presented in *Table 1*. The primary distinction between these two is that SIDS can only identify previously known attacks, whereas AIDS can identify zero-day attacks. However, due to the possibility that abnormalities could simply be new normal behaviours rather than actual invasions, AIDS can lead to a significant false-positive rate.

Table 1 Comparisons of intrusion detection methodologies (Khraisat and Alazab)

		Advantages	Disadvantages
Detection Methods	Signature-based IDS	•Very useful in identifying intrusions with minimum false alarms. •Promptly identifies the intrusions. •Superior for detecting known attacks. •Simple design	•It needs to be updated frequently with a new signature. •SIDS is designed to detect attacks for known signatures. When a previous intrusion has been altered slightly to a new variant, then the system would be unable to identify this new deviation of a similar attack. •Unable to detect the zero-day attack. •Not suitable for detecting multi-step attacks. •Little understanding of the insight of the attacks
	Anomaly-based IDS	•It could be used to detect new attacks. •Could be used to create intrusion signature	•AIDS cannot handle encrypted packets, so the attack can stay undetected and can present a threat. •High false positive alarms. •Hard to build a normal profile for a very dynamic computer system. •Unclassified alerts. •It needs initial training.

From the literature, it is possible to classify Anomaly-based Intrusion Detection System methods in the categories illustrated in *Figure 2*

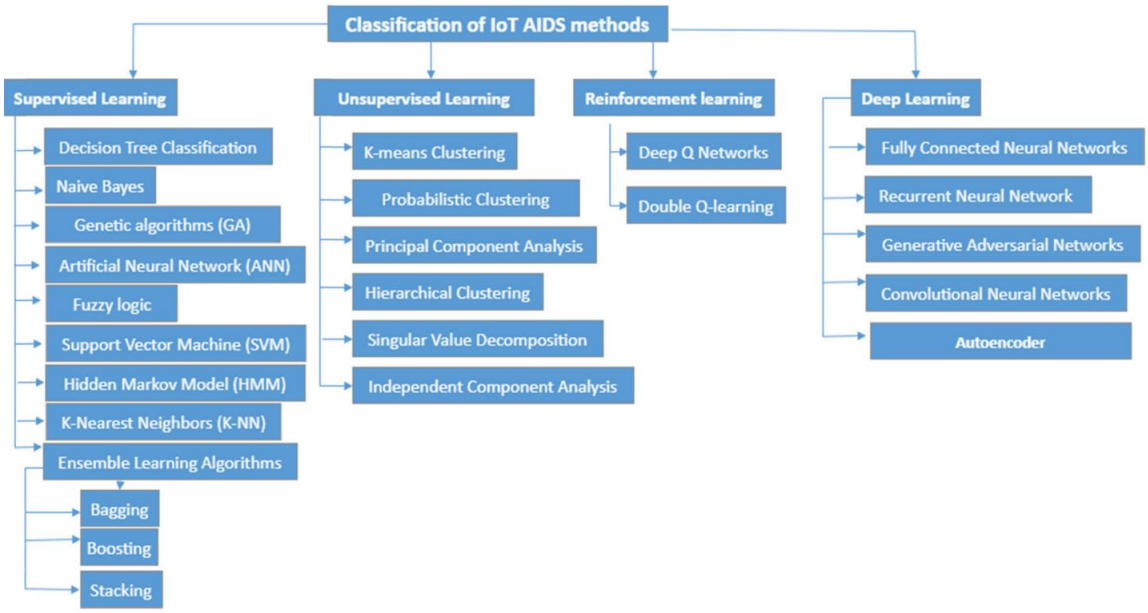

Figure 2 Classification of AIDS methods (Khraisat and Alazab)

The field of AIDS has seen substantial use of machine learning techniques. Different algorithms and approaches, including clustering, neural networks, association rules, decision trees, genetic algorithms, and closest neighbour methods, are used to extract knowledge from intrusion datasets. The major goal of applying machine learning techniques is to develop IDS that are more accurate and requires human knowledge. The primary goal of machine learning-based intrusion detection system research is to identify patterns and construct an intrusion detection system based on the dataset. Unsupervised and supervised machine learning techniques are the two main groups.

2.1.3 Supervised learning in the intrusion detection system

IDS approaches based on supervised learning find intrusions by using labelled training data. Training and testing are the two phases of a supervised learning approach. Relevant features and classes are found during the training step, and the algorithm subsequently learns from these data samples. Each record in supervised learning IDS is a pair that includes a network or host data source and an output value (also known as a label), such as minimal intrusion or normal. Next, obtrusive features can be removed via feature selection. A classifier is then trained using supervised learning to discover the underlying relationship between the input data and the labelled output value using the training data for certain features.

There are numerous classification techniques, including a k-nearest neighbour, decision trees, rule-based systems, neural networks, and support vector machines. To create a categorization model, each strategy applies a learning technique. However, a successful classification method must be able to reliably identify the class of records it has never seen before in addition to handling the training data.

The learning algorithm's main objective is to produce classification models with trustworthy generalizability. According to research, combining various classifier algorithms is an efficient way to address the drawbacks that traditional IDSs have when used for IoT.

2.1.4 Unsupervised learning in the intrusion detection system

Machine learning techniques are known as "unsupervised learning" use input datasets without class labels to retrieve useful data. Typically, the input data points are viewed as a collection of random variables. The data collection is subsequently turned into a joint density model. In supervised learning, the machine is trained to provide the desired outputs for a hidden data point using the output labels that have been provided. In contrast, no labels are provided in unsupervised learning; instead, the learning process automatically divides the data into several classes. Unsupervised learning in the context of creating an IDS refers to the employment of a process to detect intrusions utilising unlabelled data to train the model. Without the requirement to establish these groups beforehand, IoT network traffic is clustered into groups based on how similar the traffic is.

Several unsupervised learning problems and methods:

- Clustering: the objective is to create clusters of similar instances. This is a great tool for data analysis, customer segmentation, recommender systems, search engines, image segmentation, semi-supervised learning, dimensionality reduction, and more.

  One of the most popular methods for clustering data is the K-means technique, which divides a set of n data points into k clusters, with each point being chosen for the cluster with the closest mean. K denotes the use of an iterative clustering technique to help find the greatest value throughout each iteration. It is a distance-based clustering algorithm, therefore there is no need to calculate the distances between every possible set of record pairing.

- Detecting anomalies: the goal is to become familiar with the appearance of "regular" data so that you can use it to spot anomalous occurrences like faulty products on a production line or a new trend in a time series.

- Estimating the probability density function (PDF) of the random process that produced the dataset is known as density estimation. Instances found in areas with extremely low densities are frequently utilised to identify anomalies. For data analysis and visualisation, it is also helpful.

2.1.5 Comparison of K-Means and Gaussian Mixture Model

K-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centres or cluster centroid), serving as a prototype of the cluster.

When the blobs have very different diameters, the K-Means algorithm does not perform well because all it cares about when assigning an instance to a cluster is the distance to the centroid.

Instead of assigning each instance to a single cluster, which is known as hard clustering, it can be more effective to simply assign each instance a score per cluster, which is known as soft clustering.

The score could, for example, be the distance between the instance and the centroid, or it could be a similarity score (or affinity) such as the Gaussian Radial Basis Function.

The computational complexity of the algorithm is generally linear in terms of the number of instances m, clusters k, and dimensions n. This is true, however, only when the data has a clustering structure. If it does not, the complexity can grow exponentially with the number of instances in the worst-case scenario. In practice, however, this is rare, and K-Means is one of the fastest clustering algorithms.

Inertia measures how well a dataset was clustered by K-Means. It is calculated by measuring the distance between each data point and its centroid, squaring this distance, and summing these squares across one cluster. When trying to choose the number of clusters k, inertia is not a good performance metric because it decreases as k increases. Indeed, the greater the number of clusters, the closer each instance is to its nearest centroid, and thus the lower the inertia. Selecting the number of clusters k using the "elbow rule". In cluster analysis, the elbow method is a heuristic used in determining the number of clusters in a data set. The method consists of plotting the explained variation as a function of the number of clusters and picking the elbow of the curve as the number of clusters to use.

This technique for choosing the best value for the number of clusters is rather coarse.

The silhouette score, which is the mean silhouette coefficient over all instances, is a more precise but also computationally expensive approach. The silhouette coefficient of an instance is equal to (b - a) / max(a, b), where an is the mean distance to the other instances in the same cluster (the mean intra-cluster distance) and b is the mean distance to the instances in the next closest cluster (defined as the one that minimises b, excluding the instance cluster).

Despite its many advantages, most notably its speed and scalability, K-Means is not without flaws.

As we saw, running the algorithm several times is required to avoid suboptimal solutions, and you must also specify the number of clusters, which can be time-consuming.

Furthermore, K-Means do not perform well when the clusters have varying sizes, densities, or non-spherical shapes.

A Gaussian mixture model (GMM) is a probabilistic model that assumes the instances were generated by a mixture of several Gaussian distributions with unknown parameters. All the instances generated by a single Gaussian distribution form a cluster that resembles an ellipsoid. Each cluster can be different in terms of ellipsoidal shape, size, density, and orientation.

It is quite simple to detect anomalies using a Gaussian mixture model: any instance located in a low-density region can be considered an anomaly. You must specify the

density threshold to be used. For example, in a manufacturing company attempting to detect defective products, the defective product ratio is usually well known. If it is equal to 4%, you can set the density threshold to be the value that results in 4% of the instances being in areas below that density. You can lower the threshold if you notice that you are getting too many false positives (perfectly good products that are flagged as defective). If you have a high number of false negatives (defective products that the system does not flag as defective), you can raise the threshold. This is the usual precision/recall tradeoff.

With K-Means, you could use the inertia or the silhouette score to choose the appropriate number of clusters, but with Gaussian mixtures, these metrics are unreliable because the clusters are not spherical or have different sizes. Instead, seek the model that minimises a theoretical information criterion such as the Bayesian information criterion (BIC) or the Akaike information criterion (AIC).

K-Means is a simple and fast clustering method, but it may not truly capture the heterogeneity inherent in Cloud workloads. Gaussian Mixture Models can discover complex patterns and group them into cohesive, homogeneous components that are close representatives of real patterns within the data set. Experiments with Google cluster trace and Bitbrains' business-critical workloads show that clusters obtained using K-Means provide very abstract information. The Gaussian Mixture Model improves clustering by defining usage boundaries. Although the Gaussian Mixture Model takes longer to compute than K-Means, it can be used for finer-grained workload characterization and analysis.

2.2 IoT IDS deployment strategies

According to the deployment utilised to find IoT attacks, IDS may also be categorised into three categories: distributed, centralised, and hybrid.

2.2.1 Distributed IDS

Advanced intrusion detection systems, packet analysis, and incident response are all supported by a central server, or several IDS spread across a large IoT ecosystem that is collectively known as distributed IDS.

The distributed IDS system provides the analyst with a faster, simpler, and more effective way to recognise coordinated attacks across several network segments and track the attackers' movements. This could enable the early discovery of a well-planned and coordinated attack against the business in question, allowing the security personnel to verify that targeted systems are safeguarded, and offending IPs are denied access. Having the ability to detect Internet worms as they enter a corporate network is another benefit that has been demonstrated. With the help of this information, infected systems could be located and cleaned, and the worm could be stopped from spreading further throughout the network, minimising any resulting financial losses.

The second key benefit is the reduction in the number of incident analysis teams necessary to complete tasks that previously needed many teams owing to physical distance. As a result, the organization's offices no longer need to hire separate incident analysis teams for each different geographic location.

2.2.2 Centralized IDS

Centralized IDS are made up of several monitors that keep an eye on network traffic and host activity. These monitors exchange their data with a central analysis unit; the data can either be derived from the local network traffic or alarms as a consequence of local detection. Therefore, the analysis unit is either performing standard detection algorithms on top of incoming network traffic data or alert correlation algorithms on top of received alerts. Centralized IDS do not scale with the expansion of the system that needs to be secured. Additionally, the central analytic unit is a single point of failure and a performance bottleneck.

2.2.3 Hierarchical IDS

In hierarchical IDS the network is grouped into clusters. The sensor nodes that are next to one another are classified as part of the same cluster. For each cluster a leader is designated, the so-called cluster leader, who oversees the analysis at the network level and filters the member nodes.

Figure 3 represents an overview of centralized, hierarchical, and distributed IDS architectures that consist of monitors (M) and analysis units (A).

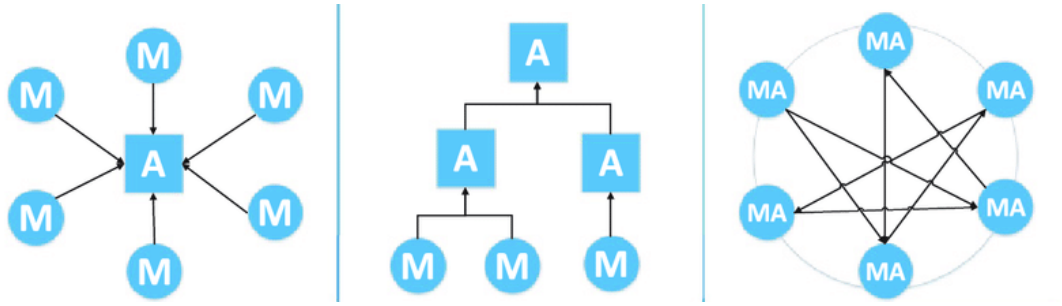

Figure 3 Overview of centralized, hierarchical, and distributed IDS

2.3 IoT System and Security Issues

Numerous use cases have been developed for the IoT system and every IoT system application has its own unique negative security risks. The IoT system and the associated security issues are covered in more detail in this section.

IoT system is composed of networked mechanical, electronic, or other objects that may transport data over a network without the need for human-to-human or human-to-computer interaction. IoT systems connect to diverse devices or things using unique network address schemes. Due to resource limitations, the majority of Internet-connected IoT devices are vulnerable to cyberattacks since their security measures are inadequate. However, the majority of IoT systems run independently over the Internet and have unreliable network connections, which exposes the network to cyberattacks. When compared to the bright future of IoT devices and cyber threats and network attacks, security issues need to be resolved right away.

IoT applications and technologies are predicted to advance more than anyone could have imagined. However, IoT technology development is still in its early stages and has not reached its full security protection maturity. IoT systems face several security hurdles, including inconsistent manufacturing standards and issues with update management by IoT software developers. Critical challenges also include the physical management of security issues and users' ignorance of security risks related to IoT systems due to their lack of awareness.

There is also no agreed standard defined security architecture in the IoT ecosystem. Various security designs are used depending on the use case, system requirements, available technology, and IoT network size. Is compelling that customisation is necessary for NIDS created to target the diverse architecture and use cases in IoT. In addition to using encryption techniques to secure data transmission the network and surroundings of IoT systems must also be protected,

Unfortunately, because of the nature of the resource limitations, typical network security techniques are not appropriate for IoT systems. In addition, a variety of network attacks have also emerged as a result of the IoT system applications' quick development and widespread use. As IoT use cases grow, the number of attacks will also increase. The likelihood of a network security and data breach is greatly reduced by being able to notice and understand the dramatic increase in cyber threats in the IoT system.

Recent examples of the most prevalent attacks launched against IoT systems include:

- *Spoofing*: Attackers pose as a legitimate Internet of Things system in a network to take over or obtain unauthorised access to the network. After gaining access, the attacker launches DoS and Man-in-the-Middle attacks on the targeted devices.
- *Denial of Service (DoS):* IoT systems or network resources become unavailable to the intended, authorised users due to a cyberattack. The purpose of these assaults is to interfere with an IoT system’s services temporarily or permanently.
- *A distributed denial-of-service (DDoS):* In a distributed denial-of-service attack the incoming traffic flooding the victim originates from many different sources. More sophisticated strategies are required to mitigate against this type of attack, as simply attempting to block a single source is insufficient.
- *Jamming*: Several IoT devices communicate with other devices through wireless networks. Attackers breach the targeted IoT system and broadcast a false signal to sabotage radio communication, exhausting the system's bandwidth, processing power, and memory.
- *Man-in-the-Middle:* A man-in-the-middle attack is a cyberattack where the attacker secretly relays and possibly alters the communications between two parties who believe that they are directly communicating with each other, as the attacker has inserted themselves between the two parties.
- *Privacy leakage*: In the communication of IoT systems, data security and privacy should come first. Manufacturers and consumers of IoT systems typically give little thought to the information that is saved on their devices, how that information is analysed, and how it is used by third parties. Wearable technology and other Internet of Things technologies that gather user data on their

whereabouts, circumstances, and health have significantly increased the potential for personal privacy leaks.

- *Mirai Botnet Attack*: Mirai is malware that turns networked devices into remotely controlled bots that cyber-attackers use as part of a botnet in a large-scale network. It primarily targets online consumer electronics, including routers and IP cameras. Attacks like DoS/DDoS also frequently used Mirai as an initiator.
- *Sybil Attack*: Sybil attacks are common in peer-to-peer networks. A Sybil attack subverts the identity of an IoT device to create many anonymous identities and consumes disproportionate power. It is named after Sybil, the author of the book Sybil, which is a case study of a woman dealing with a dissociative identity disorder. An IoT device in a network that operates multiple identities frequently undermines authorised network access in reputation systems. Sybil attacks exploit this vulnerability in the IoT system network to launch initial attacks.
- *AI-Based Attacks*: Hackers are now developing AI-powered tools that are faster, easier to scale, and more efficient than traditional human-centric attacks. This poses a serious threat to the IoT ecosystem. While the techniques and characteristics of traditional IoT risks may appear to be the same, the scale, automation, and customisation of AI-powered attacks will make them increasingly difficult to eradicate.

  Table 2 Common Attacks in IoTsummarises some common IoT system attacks and their modes of initiation.

Table 2 Common Attacks in IoT

ATTACKS	*MODE OF ATTACK INITIATION*
Spoofing	Impersonation
Denial of service	Network Flooding
Distributed denial-of-service	Network Flooding
Jamming	Fake Signaling
Man-in-the-middle	Eavesdropping Packets
Privacy leakage	Attack Authentication Storage
Marai botnet attack	Malware Implant on Devices
Sybil attack	Creates Anonymous Identities
Ai-based attacks	Creates AI-powered Tools

2.4 NIDS Design Strategies for IoT Systems

NIDS design methodologies can be divided into three groups: encrypted traffic analysis, payload analysis, and packet parsing-based. NIDS examines the payload information of transmitted packets in the IoT network. The payload data is inaccessible during encrypted

transmission; hence this technique occasionally fails. Researchers use encryption traffic analysis to overcome the constraints. The majority of NIDS created to safeguard IoT systems employ lightweight design concepts due to the issues with resource constraints in IoT.

This section explores the various design approaches and the method for lightening the NIDS to meet the system's resources.

2.4.1 Packet Parsing-Based NIDS Design for IoT

Network packets are the data that are used for NIDS and form the fundamental structure of network communication. They contain various types of binary data that must first be parsed. The header and application data are both present in network data. The header contains IP addresses, ports, and other protocol-specific information. The following are the primary benefits of using packet parsing to design NIDS data sources:

- Packets contain communication content and can be used to detect User to Root Attacks (U2R) and Remote to local (R2L) attacks. Remote to Local (R2L) is one type of computer network attack, in which an intruder sends a set of packets to another computer or server over a network that he does not have permission to access as a local user. User to Root Attacks (U2R) begins by gaining access to a normal user while sniffing around for passwords to gain access as a root user to a computer resource.
- TCP/IP data content and timestamps in packets provide NIDS with precise information about attack sources.

Meanwhile, each packet does not represent the entire communication state or the contextual information of each packet. As a result, some attacks, such as DDoS, are difficult to detect. Packet analysis detection methods, which primarily include packet parsing and payload analysis methods, support packets.

In network communications, packet parsing-based detection makes use of various protocols such as HTTP and DNS. These protocols take various forms; the packet parsing-based detection methods focus primarily on the protocol header section.

The standard procedure is to use parsing tools to extract the header fields and convert the most important fields' values into feature vectors. The header section also uses classification algorithms to extract necessary features from packets to design attack detection systems.

Because the packet headers do not need to be analysed, payload analysis-based methods work well in a multi-protocol network environment. Payloads, as a type of unstructured data, can be processed directly by deep learning models.

This method, however, does not support encrypted payloads.

2.4.2 Encrypted Traffic Analysis-Based NIDS Design for IoT

There is a focus on attacks that are visible due to the increase in traffic. Scanning, brute force, and DoS/DDoS attacks are all examples of this type of attack. Many research studies have shown that such attacks can be detected in network traffic. To be visible

without decryption, the attack must exhibit a few characteristics. To begin, only traffic passing through the NIDS can detect attacks. Some network attacks can be carried out locally or via a different path to the target. In some cases, NIDS may be unable to detect the direct attack but may detect the indirect attack. Second, the attack must cause some changes in network flow. Zero-day and targeted cyber-attacks can be detected without relying on existing attack signatures by focusing on network traffic flow and modelling different detectable attack aspects

2.4.3 Lightweight NIDS Design in IoT Systems

Due to resource constraints in IoT systems, the concept of lightweight NIDS is gaining traction in the research domain to conform to the IoT device architecture. Lightweight NIDS does not imply simplicity, but they should be able to perform their mandated security duties without jeopardising resource constraints.

Using feature selection and reduction models is one method of achieving lightweight NIDS. Feature reduction techniques such as PCA, auto-encoder, Pearson correlation, and others are used to reduce dataset dimensionality, which affects the model size, computation, storage capacity, and complexity of the NIDS model on the IoT device.

Researchers use one or a combination of the following principles to create lightweight NIDS for IoT systems:

- Using new protocols to reduce the number of computational operations required by a traditional NIDS;
- Designing a security system using optimization techniques to reduce scalar multiplication, addition, and doubling.
- Offloading all or parts of computationally intensive algorithms to a proximal device in the same network with more resources than the IoT device;
- Using modern algorithms that require fewer computational resources rather than traditional, alternative methods;
- Using data feature dimension reduction. Researchers use various algorithms such as sparse auto-encoders, the Ranker method, principal component analysis (PCA), and others to reduce the dimension of the dataset to lightweight ML-based NIDS for IoT devices.

2.5 NIDS Design Datasets for IoT Systems

The dataset is a critical component of an ML-based NIDS design. It is made up of features observed during normal and abnormal operations of the targeted systems. Innovative methods and detecting algorithms for IoT networks necessitated a well-designed dataset.

Network packet extraction flows, system logs and sessions are the most common data generation sources for NIDS for IoT. Creating a dataset tailored to IoT NIDS can be difficult and time-consuming. *Table 3* provides an overview of several publicly available datasets that are widely used by the research community for NIDS design.

Table 3 Popular Public Datasets for NIDS for IoT.

Dataset Name	*Coverage*	*Attack Types*	*Year Created*
NSL-KDD	General Purpose		2009
UNB-ISCX 2012	General Purpose	DoS, DDoS, Brute Force, XSS, SQL Injection, Infiltration, Portscan, and Botnet	2012
UNSW − NB15	General Purpose	Analysis, Backdoor, DoS, Exploit, Fizzers, Generic, Reconnaissance, Shellcode, and Worm	2015
CICIDS2017	General Purpose	DoS, DDoS, Brute Force, XSS, SQL Injection, Infiltration, Portscan, and Botnet	2017
AWS(CSE-CIC-IDS2018)	General Purpose	DoS, DDoS, Brute Force, XSS, SQL Injection, Infiltration, Portscan, and Botnet	2018
IoT Network Intrusion Dataset	IoT	Benign, MITM-arp spoofing, DoS-syn flooding, scan-host port, scan-ports, Mirai-UDP flooding, Mirai-back flooding, Mirai-HTTP flooding,Mirai-hostbruteforce	2019
IoTID20 Dataset	IoT	DoS/DDoS attacks	2020
TON_IoT Dataset	IoT, IIoT	DoS, DDoS and ransomware	2019
OPCUA dataset	IoT	DoS, Eavesdropping, Man-in-the-middle, Impersonation, Spoofing attacks	2020
IOT DOS AND DDOS ATTACK DATASET	IoT	DoS and DDoS attacks	2021
IOT HEALTHCARE SECURITY DATASET	IoT, SDN, IIoT	Normal and IoT attack traffic	2021
Mqtt-iot-ids2020 DATASET	IoT, IIoT	Normal operation, aggressive scan, UDP scan, Sparta SSH brute-force, and MQTT brute-force attack.	2020

Edge-IIoTset DATASET	IoT, IIoT	DoS/DDoS attacks, Information gathering, Man in middle attacks, Injection attacks, and Malware attacks	2022
NSS MIRAI DATASET	IoT	Mirai-type botnet attack	2020
WUSTL-IIOT-2021	IoT	Command Injection, DoS, Reconnaissance, Backdoor	2021
IOT-BDA BOTNET ANALYSIS DATASET [	IoT	Port scanning, exploitation, C2 communications and DDoS	2021
THE BOT-IOT DATASET	IoT	DDoS, DoS, OS and Service Scan, Keylogging and Data exfiltration attacks	2019
X-IIoTID dataset	IoT, IIoT	Brute force attack, dictionary attack, and the malicious insider reverse shell and Man-in-the-Middle	2021

2.5.1 Dataset requirements

A dataset should possess certain qualities to be deemed fit to be used in the field of network intrusion detection and these have been identified and verified by researchers. This section highlights several key requirements and their importance, along with details on how the requirements have been addressed.

- Realistic nature of Data

  Simulated attacks, generated using intrusion tools, are used to create intrusion traffic in most standard datasets. The exception is datasets created using honeypots where the traffic you get at the honeypot is always intrusion traffic. Both approaches produce realistic intrusion data and thus users are allowed to pick either option. This is achieved by operating the software package at any suitable network node and a honeypot, respectively.

- Proper labelling of the dataset

  Labelling an intrusion dataset properly is critical regardless of whether it is being used to train an IDS or used for research. Insufficient or inaccurate labelling requires users to utilize unsupervised learning methods to determine any attack classes or manually label the dataset based on anomaly identification which takes a considerable amount of time and effort. Our software does automatic labelling of records provided that the attack IPs are known.

- Accuracy of normal traffic

  When collecting data to be included in the normal traffic portion of the dataset, there is a possibility that intrusion traffic may get included in it, unknown to the dataset creators. This is due to undetected attacks present in traffic that are

assumed to be normal. This is avoided by generating normal traffic through simulation or getting a trustworthy group of volunteers to generate the normal traffic by interacting with the network habitually.

2.6 Strategies for IDS validation

IDS validation is the procedure used to check whether the IoT IDS model represents the system well enough to detect IoT attacks. Researchers have employed a variety of methodologies, including theoretical, empirical, and hypothetical strategies for validating their techniques.

The confusion matrix for a two-class classifier is shown in *Table 5*, and it can be used to assess how well an IDS performs. In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class.

Table 4 Terminology and derivations from a confusion matrix

Condition positive (P)	The number of real positive cases in the data
Condition negative (N)	The number of real negative cases in the data
True positive (TP)	A test result that correctly indicates the presence of a condition or characteristic
True negative (TN)	A test result that correctly indicates the absence of a condition or characteristic
False-positive (FP)	A test result which wrongly indicates that a particular condition or attribute is present
False-negative (FN)	A test result which wrongly indicates that a particular condition or attribute is absent

Table 5 Confusion matrix

		Prediction Condition	
	Total Population = P + N	Positive (PP)	Negative (PN)
Actual Condition	Positive (P)	True positive (TP)	False-negative (FN)
	Negative (N)	False-positive (FP)	True negative (TN)

The following common performance indicators are used to evaluate IDS:

- ***Precision***: It is used in information retrieval, and pattern recognition. Precision is all the points that are declared to be positive but what percentage of them are actually positive.

$$Precision = \frac{True\ positive}{Predicted\ Positive} = \frac{TP}{TP + FP}$$

- ***Recall***: It is all the points that are actually positive but what percentage declared positive.

$$Recall = \frac{True\ positive}{Actual\ Positive} = \frac{TP}{TP + FN}$$

- ***F1-Score***: It is used to measure test accuracy. It is a weighted average of precision and recall. When the F1 score is 1 it's best and on 0 it's worst.

$$\mathrm{F1} = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{TP}{TP + FN}$$

- ***Accuracy***: The accuracy measures how accurate the IDS is in detecting normal or anomalous traffic behaviour. It is described as the percentage of all those correctly predicted instances to all instances:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{\text{Total number of predictions made}} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

If we assess those errors caused by FPs are more undesirable, then we will select a model based on high precision. If we assess those errors caused by FN are more undesirable, then we will select a model based on high recall. However, if we assess that both types of errors are undesirable, then we will select a model based on F1-score.,

Accuracy is used when the True Positives and True negatives are more important while the F1-score is used when the False Negatives and False Positives are crucial.

Accuracy can be used when the class distribution is similar while the F1 score is a better metric when there are imbalanced classes. In most real-life classification problems, imbalanced class distribution exists and thus F1 score is a better metric to evaluate our model.

Network traffic is a collection of packets sent across a network. One method for detecting collective anomalies in network traffic is to divide the packets into subsections. Collective information about packets within the section can be gathered and used as input features in an anomaly detection algorithm by dividing network traffic into sets. The newly section network traffic also makes anomaly detection easier. In a nutshell, this transforms collective anomalies into point anomalies. As a result, a point anomaly detection algorithm can be used, allowing for a simpler algorithm that requires less computational power and memory.

One of the major issues when analysing complex data is the large number of features involved, to reduce the number of features required a dimensionality reduction technique:

- Independent component analysis
- Isomap
- Kernel PCA
- Latent semantic analysis
- Partial least squares
- Principal component analysis
- Multifactor dimensionality reduction
- Nonlinear dimensionality reduction
- Semidefinite embedding
- Autoencoder

An overlapping set of packets can be used to reduce the possibility of missing important patterns. An overlapping set of captured packets slows down an algorithm by managing redundant information, but it also improves the algorithm's ability to detect patterns that would otherwise be missed.

Example of procedure for the performance analysis of the unsupervised algorithm:

- Time-series data from real-life examples are collected and anonymised.

- Anomalies are manually labelled before performance evaluation — this is stored as the ground truth dataset, based on the analyst's assessment, independent of results from the unsupervised algorithm.
- The labelled dataset is used to evaluate the outcome of the unsupervised algorithm.

Chapter 3: Desiderata for a centralised NIDS

This chapter explains the decisions that underpin the development of a centralized NIDS versus other possible configurations in an IoT environment.

Because IoT devices have limited resources it is not recommended to implement a host-based IDS. Using a HIDS would result in performance degradation, as well as a total degradation of performance in a network of a certain size with several IoT devices. NIDS, on the other hand, excel at securing hundreds of IoT device systems from a single network location. This makes a NIDS less expensive and easier to deploy. NIDS also provide a broader examination of a network.

Implementing a NIPS would have required significantly more resources because it is necessary to configure a firewall to activate rules that block specific attacks. Furthermore, real-time data must be analysed to determine if an attack is in progress in the shortest amount of time, which necessitates a significant amount of computing power.

Because almost all open-source software employs a signature-based detection method and IDS based on anomaly detection algorithms are still uncommon. Using a signature base detection system and an anomaly-based detection system is one of the best choices to obtain the advantages of both methods. Its advantages and disadvantages are listed in *Table 1*.

Since the IDS that is being developed is centralized, it's a single point of failure. To reduce recovery times in the event of a failure, consider performing a deployment via Docker and NoSQL database with the backup.

Finally, various open-source software applications were evaluated for their compatibility with NIDS. The data relating to the various software have been collected on an excel sheet and attached to this report. Evaluation of open-source software was based on application scope, detection method and validation strategy. Moreover, a specific proprietary IPS for the IoT / OT field was also analysed. The intent of the "application scope" is to assess whether the software was suitable for use as a Network IDS, as a Host IDS or as hybrid IDS. The evaluation concerning the "detection method" instead focuses on analysing the methods implemented by the software to determine threats. Evaluations of the "validation strategy" instead focus on analysing the methods that can be used to test the IDS's efficiency.

3.4.1 Open Source IDS Tools

One way to understand how to integrate the evaluated open-source tools is to look at how they handle PCAP files. These files contain network packet data and are used to analyse network characteristics. They also help to control network traffic and determine network status. PCAP files can be used to detect network problems and resolve data communications using various programmes.

Ideally, all the tools allow for network analysis with PCAP, so it's possible to incorporate them into any project. However, each tool has distinct capabilities and modes of operation.

There are different ways to use snort. Snort can be used in packet logger mode, in this way it's possible to record packets on disk, when Snort runs in this packet logger mode, it collects every packet it sees and places it in a directory hierarchy based upon the IP address of one of the hosts in the datagram.

In other words, a new directory is created for each address captured and the data about that address is stored in that directory. Snort places the packets into ASCII files, with filenames generated from the protocol and port number. This hierarchical organization is most useful if a limited number of hosts are involved or if you wish to view the IP addresses of captured hosts briefly. However, the logging directory can become very congested over time due to the ever-increasing number of directories and files. If you're logging all the traffic on a very active network, it is even possible to run out of inodes (a Unix data structure that limits the total number of files in a filesystem) long before you run out of storage. If someone were to do a full scan of your network and map all 65,536 TCP ports as well as all 65,536 UDP ports, you would suddenly have 131,000 or more files, all possibly in a single directory. This file explosion can severely tax the limits of almost any machine, and could easily turn into a full-blown denial-of-service attack.

Instead of Snort, *tcpdump or tshark* are recommended as packet loggers. *Tcpdump/Tshark* are command-line network sniffers, used to capture network packets. These tools are very helpful to sniff network packets because it offers many options and is more efficient than Snort.

The possible alternatives, between Snort Suricata and Zeek, should be considered based on the comparative test. Because Fortigate is proprietary software no comparative tests were carried out.

3.2 Comparative analysis of Snort and Suricata

In this chapter, Snort and Suricata are compared based on their resource consumption and threat detection accuracy.

Snort, the de-facto industry standard open-source solution, is a mature product that has been available for over a decade. Instead, Suricata offers a new approach to signature-based intrusion detection and takes advantage of current technology such as process multi-threading to improve processing speed. Researchers [2] ran each product on a multi-core computer and evaluated several hours of network traffic on the NPS backbone. The speed, memory requirements, and accuracy of the detection engines have been evaluated in a variety of experiments. The paper concludes that Suricata will be able to handle larger

volumes of traffic than Snort with similar accuracy, and thus recommend it for future needs at NPS since the Snort installation is approaching its bandwidth limits.

The experiments tested and compared the Suricata and Snort intrusion-detection engines in performance and accuracy in a busy virtual environment. The experiments evaluated performance by measuring the percentage of CPU use, memory use, and network use. Accuracy was measured by subjecting both detection engines to malicious traffic in controlled tests, and comparing the alerts generated by each application.

Suricata consumed more computational resources than Snort while monitoring the same amount of network traffic.

Network performance remained an issue for Snort. While it's possible to reduce the packet drop rate in Tcpdump to less than 1%, the output log from Snort reported a packet drop rate of 53%. Suricata, on the other hand, had a drop rate of 7%.

The multi-threaded architecture of Suricata necessitates more memory and CPU resources than Snort. Suricata used nearly twice as much CPU as Snort, and it used more than twice as much RAM as Snort. This could be attributed to the overhead required to manage Suricata's multiple detection threads. Suricata has the advantage of being able to scale to accommodate increased network traffic without the need for multiple instances. Snort is small and fast, but it cannot scale beyond 200-300 Mbps network bandwidth per instance. While Snort's processing overhead is lower than Suricata's, the need for multiple instances to achieve what Suricata can with its multi-threaded design raises the cost of running and managing a Snort environment.

After the analyses carried out I have evaluated not to use Snort in my thesis project in packet logger mode since the Packet Capturing Module plays a fundamental role in the performance, which if selected without proper evaluation, could result in a large packet drop. Snort has packet-dropping issues leading to degraded system performance. I believe I will use *tcpdump* as a network packet acquisition system.

3.3 Comparative analysis of Suricata and Zeek

Suricata is the standard when it comes to signature-based threat detection engines. It was introduced to quickly identify known threats and allow for the deployment of additional rules when new exploits are discovered. Suricata, which is based on a multi-threaded architecture that takes advantage of modern hardware, enables high-performance traffic inspection and quickly processes many rules against large volumes of network traffic. Suricata is compatible with Snort rule repositories and supports the LUA scripting language, allowing users to create rules to detect complex threats.

Zeek, on the other hand, monitors traffic streams and generates logs that record everything it knows about network activity as well as other metadata that is useful for analysing and understanding the context of network behaviour. Much of the metadata produced by Zeek is only available from packet capture (PCAP) data. Metadata can also be searched, indexed, queried, and reported in new ways that PCAP did not allow. The Zeek programming language, which is structured similarly to C++, can be used to calculate numerical statistics, perform regular expression pattern matching, and tailor metadata interpretation to an organization's specific needs.

Suricata outperforms Zeek in terms of monitoring traffic for known threats and producing alerts when they are detected. Another advantage is that new threat intelligence is frequently available first in a Suricata-compatible format.

Zeek provides the large volumes of high-quality data required to enable network baselining, host and service profiling, passive inventory collection, policy enforcement, anomaly detection, and threat-hunting efforts.

Organisations would rely on Suricata to quickly identify attacks where signatures are readily available, and Zeek would provide the metadata and context required to successfully triage Suricata alerts and create comprehensive timelines of the entire threat landscape. Suricata and Zeek work well together for threat detection as well. Suricata, for example, may send an alert indicating that a system has been compromised, and the incident and connections preceding and following the incident are recorded by Zeek and can be analysed to determine whether other network communications strengthen or help explain the incident.

3.4 Design of a NIDS

This chapter presents a possible design for a NIDS based on open-source software.

There are a total of six different modules in a NIPS:

Figure 4 NIDS modules

1. Packet Capturing Module: It is the first building block in any NIPS solution which is used for packet acquisition from the network. The module can be used in passive mode (port mirroring based) for NIDS. In passive mode (NIDS) packets could be temporarily stored in buffers while awaiting processing.

   Given the proper analysis, the tool *tcpdump* or *tshark* with an interface in promiscuous mode should be used for this phase. [In promiscuous mode, a network device, such as an adapter on a host system, can intercept and read in its entirety each network packet that arrives]

2. Packet Decoder Module: Packet Decoder applies protocol-specific rules to decode contents and further establish packets' conformance with standards. With *tcpdump/tshark* it's possible to capture directly in PCAP format. [The de facto standard network packet capture format is libpcap (PCAP), which is used in packet analyzers such as tcpdump/WinDump and Wireshark. The pcap file format is a binary format, with support for nanosecond-precision timestamps]

3. Pre-processor Module: This module provides support for both low-level and high-level protocols-based packet pre-processing. At the lower-level packets are processed for defragmentation, reassembly, and session conformance; while, the

high-level plugins validate packets concerning the various application protocols (HTTP, FTP, etc.).

It is possible to perform pre-processing with several Python scripts, however, the PCAP format already provides a high level of formatting. Ingesting a large dataset and efficiently pre-processing it can be difficult to implement, but TensorFlow makes it simple thanks to the Data API: you simply create a data set object, tell it where to get the data, and then transform it in any way you want, and TensorFlow handles all the implementation details, such as multithreading, queuing, batching, prefetching, and so on.

4. Detection Engine Module: The Detection Engine is the most crucial part of any NIDPS solution where rules are applied to compare incoming packets against defined malicious patterns. There are three types of detection engines: signature-based which uses a static rule set to identify malicious flows (misuse detection), anomaly-based which aims to identify deviations from normal behaviour, and hybrid systems which are based on the combination of the prior two. Open-Source solutions by default support signature-based detection engines.

   Implementing Suricata with an anomaly detection algorithm is a good thing to get the benefits of both methods. As discussed in previous chapters *Comparison of K-Means and Gaussian Mixture Model*, k-means and Gaussian Mixture Models are two possible algorithms for detecting anomalies. K-means is a straightforward and quick clustering method, but it may fail to capture the heterogeneity inherent in the data. Instead, Gaussian Mixture Models can discover complex patterns and group them into cohesive, homogeneous components that are close representations of real patterns in the data set.

   In this manner, the advantages of both anomaly-based and signature-based methods can be utilised. To evaluate the performance of the algorithm, refer to the methods illustrated in chapter 2 *Strategies for IDS validation*.

5. Alerting Module: The detection engine is followed by Alerting module that generates an alert when a rule matches the content of a packet. In the case of IDS, this module only generates an alert whereas in the IPS mode it actively blocks the malicious packets after generating alerts. This module could be implemented using a web framework like Django

6. Output Module: It is the last module in the building block of the NIDS solution. The output module generates statistics regarding the packets traversing the NIDS solution. For the open-source solutions, the output module is normally based on Command Line Interface instead of using a web framework to provide a better interface.

3.4.1 Testing a NIDS

This chapter summarizes possible methods for testing the efficiency of the NIDS engine module.

To generate network traffic, it could be used a live Node-RED system that simulates an always-on IoT network. Other possible IoT network simulators could be IoTIFY, NS3, MIMIC IoT Simulator, IoTNetSim and Cooja IoT.

Using NodeRED, I simulated an IoT network and came to the following conclusions:

Node-red is not suitable for testing a NIDS that works with the multilayer functionality of the network stack. With Node-RED, it is not possible to change the MAC address of the network card or the IP address of the packets generated by the simulated IoT devices. The only anomaly factor that can be detected is at the application level, which is not sufficient to assess the NIDS.

It is recommended to use other types of software to have a better testing network, such as MIMIC IoT and NS-3. They are very complex, so it is not easy to carry out a multitude of tests able to cover all use cases.

Another possibility is to use PCAP file generators, in this way it is possible to generate random PCAP files related to a certain structure to test borderline cases. Free software used for these tests is *Randpkt*, *Ostinato* and *Trex,* while for a fee *IxNetwork.*

With PCAP file generators we could use several different approaches:

- Mutation-based: Anomalies are added to existing good input. Using random or heuristic methods.
- Smart: Generate inputs given the file format or protocol specification.
- Responsive: Inputs are generated based on program response and structure.

Reference list

1. Alyami, Hashem, Md Tarique Jamal Ansari, Abdullah Alharbi, Wael Alosaimi, Majid Alshammari, Dhirendra Pandey, Alka Agrawal, Rajeev Kumar, and Raees Ahmad Khan. 'Effectiveness Evaluation of Different IDSs Using Integrated Fuzzy MCDM Model'. *Electronics* 11, no. 6 (January 2022): 859. https://doi.org/10.3390/electronics11060859.

2. Albin, Eugene. 'A Comparative Analysis of the Snort and Suricata Intrusion-Detection Systems'. Thesis, Monterey, California. Naval Postgraduate School, 2011. https://calhoun.nps.edu/handle/10945/5480.

3. Bohutska, Julia. 'Anomaly Detection — How to Tell Good Performance from Bad'. Medium, 19 August 2021. https://towardsdatascience.com/anomaly-detection-how-to-tell-good-performance-from-bad-b57116d71a10.

4. Patel, Eva, and Dharmender Singh Kushwaha. 'Clustering Cloud Workloads: K-Means vs Gaussian Mixture Model'. *Procedia Computer Science*, Third International Conference on Computing and Network Communications (CoCoNet'19), 171 (1 January 2020): 158–67. https://doi.org/10.1016/j.procs.2020.04.017.

5. Sai Kiran, K. V. V. N. L., R. N. Kamakshi Devisetty, N. Pavan Kalyan, K. Mukundini, and R. Karthi. 'Building an Intrusion Detection System for IoT Environment Using Machine Learning Techniques'. *Procedia Computer Science*, Third International Conference on Computing and Network Communications (CoCoNet'19), 171 (1 January 2020): 2372–79. https://doi.org/10.1016/j.procs.2020.04.257.

6. Spadaccino, Pietro, and Francesca Cuomo. 'Intrusion Detection Systems for IoT: Opportunities and Challenges Offered by Edge Computing and Machine Learning. arXiv, 14 April 2022. https://doi.org/10.48550/arXiv.2012.01174.

7. Vasilomanolakis, Emmanouil, Shankar Karuppayah, Max Mühlhäuser, and Mathias Fischer. 'Taxonomy and Survey of Collaborative Intrusion Detection. *ACM Computing Surveys* 47 (1 May 2015). https://doi.org/10.1145/2716260.

8. ‘12 Tcpdump Commands - A Network Sniffer Tool’, 1 September 2021. https://www.tecmint.com/12-tcpdump-commands-a-network-sniffer-tool/.

9. Fortinet. ‘FortiGuard IPS Security Services’. Accessed 13 July 2022. https://www.fortinet.com/products/ips?utm_source=%253Cwhs&utm_campaign=2018-q2-ips-landing-page.

10. ‘SNORT Users Manual 2.9.16’. Accessed 13 July 2022. http://manual-snort-org.s3-website-us-east-1.amazonaws.com/.

11. ‘TCPDUMP & Save Capture to Remote Server Using Ssh, Tcpdump’. Accessed 13 July 2022. https://www.commandlinefu.com/commands/view/12379/tcpdump-save-capture-to-remote-server.