T.C.

ALTINBAS UNIVERSITY

Institute of Graduate Studies

Electrical and Computer Engineering

CLUSTER BASED INTRUSION DETECTION SYSTEM IN WIRELESS SENSOR NETWORKS

ALAA ZEYAD TAHSEEN TAHSEEN

Master of Science

Supervisor

Asst. Prof. Dr. Sefer KURNAZ

Istanbul, 2021

CLUSTER BASED INTRUSION DETECTION SYSTEM IN WIRELESS SENSOR NETWORKS

by

ALAA ZEYAD TAHSEEN TAHSEEN

Electrical and Computer Engineering

Submitted to the Institute of Graduate Studies

In partial fulfillment of the requirements for the degree of

Master of Science

ALTINBAŞ UNIVERSITY

2021

The thesis titled "CLUSTER BASED INTRUSION DETECTION SYSTEM IN WIRELESS SENSOR NETWORKS" prepared and presented by ALAA ZEYAD TAHSEEN TAHSEEN was accepted as a Master of Science Thesis in Electrical and Computer Engineering.

_____

Asst. Prof. Dr. Sefer KURNAZ

Supervisor

Thesis Defense Jury Members:

Asst. Prof. Dr. Sefer KURNAZ	School of Engineering and Natural Sciences, Altinbas University	_____
Prof. Dr. Osman Nuri UCAN	School of Engineering and Natural Sciences, Altinbas University	_____
Prof. Dr. A.Mesut RAZBONYALI	Faculty of Engineering and Architecture, Maltepe University	_____

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Approval Date of Institute of Graduate Studies: ____/____/____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

ALAA ZEYAD TAHSEEN TAHSEEN

DEDICATION

I devote and pledge this research work to my supervisor who is salient for guiding me through whole research work as well as my family for always assisting me in my hard time.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisor Asst. Prof. Dr. Sefer KURNAZ for guiding and helping me along the way in writing this dissertation. Discussing my progress, problems, and ideas with my supervisor Asst. Prof. Dr. Sefer KURNAZ a couple of times every week helped me tremendously in understanding the logic behind the research. It made me better realize the technical need for this research work.

ABSTRACT

TITLE OF THE THESIS/DISSERTATION

TAHSEEN, Alaa Zeyad Tahseen,

M.Sc., Electrical and Computer Engineering, Altınbaş University,

Supervisor: Asst. Prof. Dr. Sefer KURNAZ

Date: July / 2021

Pages: 61

The aim of this thesis is to perform cluster-based intrusion detection system in Wireless Sensor Network (WSN). WSNs are composed of several sensors which are randomly or deterministically distributed for data acquisition and to forward the data to the gateway for further analysis. WSNs are used in many applications such as in health care for in communication; in utilities such as in industries to monitor the state of equipment and detect any malfunction during normal production activity. In general, WSNs take measurements of the desired application and send this information to a gateway, whereby the user is able to interpret the information to achieve the desired purpose. The main importance of WSNs in area of intrusion detection is that they can be trained to detect the intrusion and real time attacks in the CIC IDS 2017 Dataset. WSNs routing protocols are designed to establish routes between the source and destination nodes. What these routing protocols do is that they decompose the network into more manageable pieces and provide ways of sharing information among its neighbors first and then throughout the whole network. The landscape of interesting libraries and techniques is constantly evolving, and so are the possibilities and options for experiments. Implementing the framework in python helps in reducing syntactic complexity, increases performance compared to implementations in scripting languages, and provides memory safety.

The fact that memory safety is a big problem for today's intrusion detection frameworks has been recognized by this advance research, which have begun to port parts of their protocol decoding logic to achieve an accuracy of 99.21% for detecting all the intrusions within the WSN that use a parser generation framework to accomplish safe protocol parsing. Although, these countermeasures are a good step forward, they only reduce the attack surface partially, since the remaining parts of the frameworks are still written in a language that is affected by this category of vulnerabilities.

Keywords: Cluster, Nodes, Wireless, Sensor, Network.

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

LIST OF ABBREVIATIONS

IOT	:	Internet of things
TLS	:	Transport Layer Protocols
WSN	:	Wireless Sensor Networks
DoS	:	Denial Of Service
TADAP	:	Task Assignment And Data Advertisement Protocol
SMP	:	Sensor Management Protocol
PA	:	Power Available
IOS	:	International Organization for Standardization
SVM	:	Support Vector Machine
RBF	:	Radial Basis Function
SIEM	:	Security Information and Event Management
PKI	:	Transmis-Sion Control Protocol
UDP	:	User Datagram Protocol

1. INTRODUCTION

Fueled by the invention of the TCP protocol (IP) in 1974, modern life relies heavily on complex computer networks and applications as mentioned in [1]. This ranges from everyday personal use of computers, to distributed business networks, medical and industrial applications and self-driving cars. Due to the increasing amount of valuable information stored in such systems, they are becoming a more and more popular target for attackers as mentioned in [2]. Successful attacks against computer systems often have a direct negative economic impact. Adversaries range from amateur enthusiasts, to organized criminals and state-sponsored organizations. Due to the rising availability and sophistication of internet technology and the increased number of network attacks, network defense has become an important area of research as mentioned in [3]. Network defense requires in-depth knowledge about all protocols flowing across the network. Capture software must therefore implement many complex parsers in order to interpret the collected data the right way. Keeping up with the increasing amount of data that has to be processed can be challenging, and requires effective implementations and algorithms as mentioned in [4]. Real time data processing is necessary, to ensure a fast uncovering of an ongoing or past attack and to prevent or lighten economic damage. New vulnerabilities appear every day and are quickly exploited with so called zero-day attacks. Signature based detection strategies suffer from the inability to detect previously unknown threats as mentioned in [5]. However, although having been a popular area of research over the last 20 years, machine learning techniques are rarely seen in commonly available tools. High amounts of false positives and the lack of proper training and evaluation data are well known problems of anomaly-based intrusion detection as mentioned in [6]. Traffic patterns change a lot and so does the software stack inside network environments. To reflect this, modern datasets must be used for the evaluation of anomaly-based intrusion detection strategies. Network intrusion detection greatly helps in identifying network breaches, tracing them back to the responsible parties and then taking action to isolate and retrieve any damage that occurred as mentioned in [7]. The attack recognition and event monitoring capabilities of intrusion detection systems also have a deterrent effect on attackers, who face a greater risk of being discovered and prosecuted. The presence of an intrusion detection might convince an attacker to search for another target, that is easier to penetrate. Being blocked by the monitor or by an analyst due to an alert, creates unwanted

attention for the intruder and slows down his operations as mentioned in [8]. However, in order to benefit from intrusion detection, there is a need for a reliable and extensive data source in order to make accurate predictions.

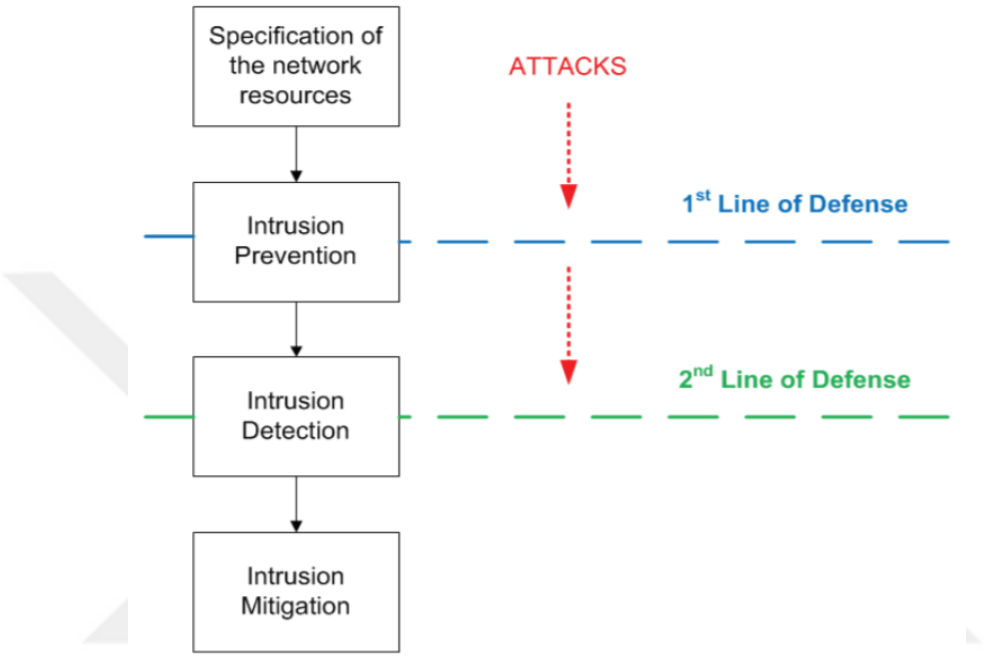


Figure 1.1: Provisioning schema for wireless sensor networks.

Preventative techniques like authentication and access control can fail, intrusion detection provides a second line of defense and serves as the base for incident response after a system was compromised as mentioned in [9]. After gaining a foothold on a network, time passes for the attacker to perform reconnaissance and identify assets of interest. Although the initial infection might not be detected, it is possible to prevent further damage when detecting the lateral movement of the attacker as mentioned in [10]. Intrusion detection plays a critical role in uncovering such behavior and greatly aids in reconstructing it for further forensic investigation. Only one third of organizations discovered the intrusion themselves. Their latest report from 2018 states a median detection time of 99 days in 2016 and 101 days in 2017 as mentioned in [11]. Even though the decreasing detection time is a positive development, it is still too high. 100 days are a lot of time for an attacker to identify assets of interest and take action. Besides that, even the most sophisticated and secure systems are still vulnerable to insiders, who misuse their privileges. When inspecting vulnerability visualizations, an alarming trend becomes visible: the

overall number of reported vulnerabilities each year is growing and the vast amount of them is scored as medium or high severity as mentioned in [12]. Intrusion detection is a key component for network defense and information security, it helps identifying attempts to compromise information systems and protect their availability, integrity, and confidentiality. Security information and event management (SIEM) systems often struggle with organizing the huge amount of data and provide interoperability between various data formats and input sources as mentioned in [13]. Data fed into a SIEM can be divided into 4 categories: network traffic, host data from the monitored endpoints, logs generated by various systems and threat intelligence feeds as mentioned in [14]. Network data is the most valuable of these information's, since every attack has to go through the network and thus leaves traces, even if an attacker tries to hide or obfuscate his presence. On the contrary for example, host data is not always reliable, because a compromised machine can be manipulated by an attacker as mentioned in [15]. A recent study from the American Consumer Institute Center for Citizen Research has shown that 83% of routers contain vulnerable code. Infrequent updates cause the devices to be vulnerable for a long period of time, even after the vulnerabilities have been published in [16]. Although recent claims about hardware attacks by implants in the supply chain of the server manufacturer super micro have not been proven at the time of this writing, they describe a possible scenario that should be considered when securing a network environment. Network security monitoring can detect malicious attempts to contact the outside world, from both compromised software or hardware components, and allows to search the gathered data for indicators of compromise afterwards, to determine if there were other attempts in the past as mentioned in [17]. In order to infiltrate assets or information, the data has to be transferred to a server that is controlled by the intruder. This makes network data a very powerful piece of evidence. A recent study on the history of Remote Access Tools (RATs) shows that the development of Trojans has not stopped, instead it seems to be increasing. The study analyzed the 300 most important RATs of the last 20 years as mentioned in [18]. It shows the importance of behavior based detection strategies, since signatures can only identify known malicious programs, and new malware variants appear frequently.

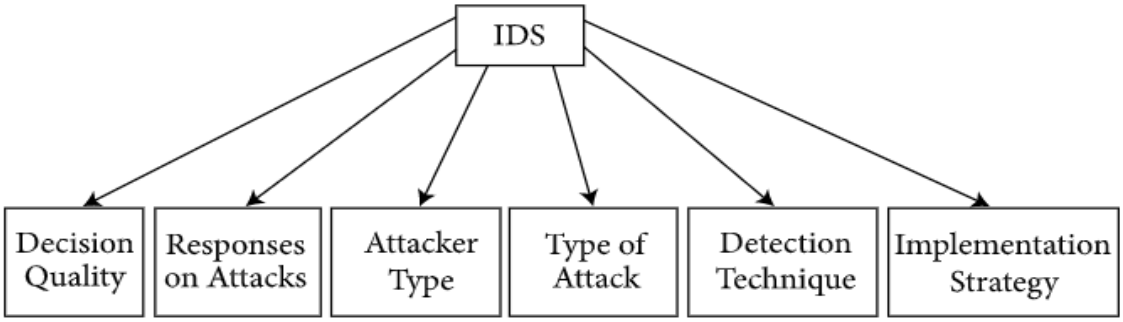


Figure 1.2: Classification of intrusion detection system into different categories [19].

Feature collection for the purpose of network intrusion detection faces several problems. First of all, the increasing volume of data that has to be processed: high resolution video streaming, large file transfers and video conferences are common scenarios in corporate environments, and generate huge amounts of network packets that have to be processed as mentioned in [20]. Another problem are attacks against the monitoring system, which can result in crashes and therefore loss of audit data, or in the worst case even to remote compromise of the network monitor. Current solutions for collecting features from traffic are written in low level system programming languages. This increases the attack surface tremendously, since these languages do not provide automated memory management as mentioned in [21]. This can lead to memory corruption vulnerabilities that enable denial of service or remote code execution attacks. Even if the state of an implementation is considered audited and secure, future updates to protocol specifications or new protocols will require continuous modifications, which can possibly (re-) introduce exploitable security vulnerabilities. Furthermore, the collection sensor should not be dependent on a specific architecture and be functional across all major platforms, in order to qualify for a widespread area of applications. The output data format should be consumable by as many systems as possible, to enable easy and quick integration into an existing network monitoring stack. The most commonly used data format for big data analysis and machine learning are comma separated values (CSV), which do not preserve data types or provide structure to the data. Deployment and configuration of existing solutions is complex and time intensive for large environments as mentioned in [22]. A growing share of internet traffic is encrypted, denying access to content and requiring analysis strategies that do not rely on clear text packet payloads for classification. The current state of tooling is insufficient to meet requirements for convenient and effective experiments, due to the fact that most of those tools

were not designed specifically with a research task in mind. There might be fundamental differences in architecture or in client behavior as mentioned in [23]. Collecting network traffic during a penetration test inside the network and using this as a dataset to verify the deployed countermeasures work as expected, would create a much more realistic scenario and therefore more meaningful results. Currently, no publicly available tooling exists to accomplish this kind of independent verification in an effective and reproducible manner.

1.1 MOTIVATION

Many software systems are exploitable, due to logic errors or memory corruption vulnerabilities. An example are buffer overflows, in which a section of memory is being overfilled with data and the following section overwritten, which can lead to a crash or even remote code execution. Another common example is use after free vulnerabilities, which refer to the process of accessing memory after it has been freed as mentioned in [24]. This can also cause a program to crash or result in the execution of arbitrary code. The programming language and its derivatives without automated memory management, suffer from these kinds of vulnerabilities, due to human errors in bounds checking or even because of compiler optimizations that eliminate bound checks. However, also languages that provide such automated memory management are not prune to attacks. For example, the Java virtual machine and object serialization have been exploited in the past. Web browsers are a well-known target for attacks, as they implement a huge protocol stack and have many potentially vulnerable components, such as plugins and extensions as mentioned in [25]. A commonly exploited component of web browsers is Adobe Flash, a programming language to create animated web contents.

Even big companies that invest large amounts of money in a security team and offer bug bounties for submitted vulnerabilities, fail to protect themselves or their users from attacks on a regular basis. In the recent Facebook incident, attackers exploited a technical vulnerability to steal access tokens that would allow them to log into about 50 million people's accounts. Apple recently fixed a vulnerability in the ICMP packet error handling code of their XNU kernel, which resulted in a heap overflow and could potentially be exploited for remote code execution (CVE-2018-4407). The vulnerability exists in such a fundamental part of the operating systems networking code that anti-virus software is unable to detect exploitation attempts as mentioned in

[26]. Also, even companies run by engineers aware of computer security implications, such as the spyware manufacturer Hacking Team, have been compromised with a zero-day exploit against an embedded device in their network in the past as mentioned in [27]. A new zero day in Cisco's Adaptive Security Appliance and Firepower Threat Defense Software has recently been discovered, due to an error in handling Session Initiation Protocol (SIP) packets. The vulnerability allows an attacker to remotely reload or crash the affected systems. The attack surface is further increased by the tools used for forensic investigation. Protocol dissectors have a huge attack surface, due to the amount of supported protocols and implemented code that has be to maintained. Wireshark for example, has 1400 authors, supports over 2400 protocols and is made of a total of at least 2.5 million lines of code. Due to its complexity and dynamic nature, current software stacks are potentially vulnerable to exploitation and compromise as mentioned in [28]. Monitoring them is therefore necessary to ensure rapid uncovering of breaches and to increase the attacker's efforts and costs. It is important to keep in mind that also the network monitor might be the target of an attack.

1.2 PROBLEM STATEMENT

Bugs or vulnerabilities exist across the whole technology stack that is being used in co- operate or personal environments. The risk that attackers will find a way in is high, and depends on the amount of time and money adversaries are willing to spend. Vulnerabilities can potentially be abused to disable, disrupt or circumvent the monitoring system. The network monitor might even serve as an entry point to the network it was supposed to protect, and thus weakens the overall system security, instead of increasing it. To shed some light on the situation regarding the most popular open source projects, various searches have been conducted and will be summarized below. If configurable in advanced search, the exact match option has been selected to filter the results. Without this, a search query for bro would also deliver results for entries containing the keyword browser. For the product name field was used to improve search accuracy. The estimated number of unrecorded cases is probably much higher, since many bugs get discovered and fixed by the authors without being assigned a CVE identification as mentioned in [29]. Additionally, software libraries that are commonly used are at risk of being exploited and increase the attack surface, they will be included in the search as well. Many of these vulnerabilities occur due to memory corruptions, such as buffer under- or overflows and heap

corruptions. With proper auditing and unit testing, logic errors can be detected early in the development cycle. It should be noted that the authors of those frameworks are highly experienced software engineers, who have a strong sense for security and their code is audited by several similar experienced developers. Nevertheless, mistakes happen and can lead to compromise or denial-of-service attacks if not discovered and patched in time. Many of the vulnerabilities could have been avoided by choosing a memory safe language instead of using python programming for better performance. Code is often written in a hurry, as developers need to get things done on time. This introduces programming mistakes and bad design decisions, which can lead to exploitable vulnerabilities. Source code audits are expensive and time intensive and therefore often omitted. Even more problematic is the sanitation of a production infrastructure as mentioned in [30]. Changes can lead to new bugs or malfunctions that negatively impact the systems availability. Penetration testing on the live environment is often not possible, which increases costs because a testing environment has to be created for this purpose. The risk that an intruder, who invests sufficient amounts of time and money, will always be able to successfully penetrate an IT infrastructure, has to be taken into account. Therefore, secure and efficient strategies for detecting system compromise are necessary to protect todays network environments. Core questions of anomaly based intrusion detection haven't been solved completely, after more than two decades of research:

A. Which are the most powerful features, which do require extraction and which don't?

B. Which feature combination delivers the best results across all available datasets or algorithms for wireless sensor network?

C. Which features are delivering the best results for a specific algorithm in wireless sensor network?

D. What is the most efficient summary structure for the cluster based intrusion detection task with classification types of attacks?

E. What delivers better results, observations for each protocol or summary structures such as flows and connections?

F. Which technique is the most efficient in terms of computing resources?

Solving these questions entirely requires efficient tooling for feature collection, selection and extraction. Unfortunately, many researchers don't publish their tools, which makes it hard to reproduce and validate their results, and increases time needed for future research on the topic. Furthermore, relying on crafted datasets for verification of network security monitoring, does not reflect the actual situation inside the protected network environment.

1.3 RESEARCH CONTRIBUTIONS

Goal of this thesis is the implementation of a modern, systematic approach to access network protocol specific information, for the purpose of research on anomaly based intrusion detection strategies in wireless sensor network. In this context, the term modern refers to the following attributes of the implemented framework: A suitable programming language must be chosen for the implementation, which supports all major platforms, provides memory safety and has a concurrent programming model in wireless sensor network. A suitable output format must be selected that preserves type safety and structure of the data, while being platform neutral. To make use of modern computer architecture with multiple processor cores and allow scaling for large workloads, the implementation must have a concurrent design in wireless sensor network. Also it must be well documented to support other researchers working with it in the future. The implemented approach should be suitable for usage with dump files and live traffic from a network interface and provide a command-line and library interface in wireless sensor network. Furthermore, unlike many solutions that try to achieve maximum data reduction, this approach shall focus on making the data more accessible and provide maximum flexibility, in order to improve the workflow and efficiency for experiments in wireless sensor network. This follows the philosophy, that a network feature collection system should provide data at the lowest level of its internals, prior to generating any abstractions on top of it. The implemented approach shall be evaluated with an up-to-date dataset and an anomaly based classification strategy in wireless sensor network.

1.4 THESIS ORGANIZATION

This introduction explains the motivation for network security monitoring and anomaly based detection strategies with clusters, and outlines the impact of memory corruptions on the security of today's networking infrastructure within wireless sensor networks. Common terminology will

be explained, along with a problem definition and a task description to define the scope of this thesis. After a brief discussion of related work, requirements for network feature collection mechanisms are analyzed, followed by an evaluation of existing solutions. Afterwards, concept and implementation of the research project will be outlined in-depth, as well as ideas for future improvements and several use cases beyond network intrusion detection. In the final evaluation, several practical appliances of the developed tool are shown, which includes a series of experiments on classifying malicious behavior with a wireless sensor network. After the conclusion, a look ahead on possible future developments is given.

2. RELATED WORK

This thesis work is motivated by the goal of overcoming the challenges and difficulties that intrusion detection system units in wireless sensor network faces with the help clusters techniques in real time with their structure and the environmental hostility surrounds them, that is by creating a better environment solution, a solution that will avoid the shortcomings of the traditional wireless networks for intrusion detection system and controlling operations. For the sake of explaining the motivation of this thesis work, a detailed review of all these difficulties and constraints is discussed in depth in the sections below; furthermore, a comprehensive review of wireless sensor network system and their advantages that this thesis work aims to overcome.

2.1 LITERATURE REVIEW

Researchers in [31] presents several network traffic flow capture formats, tools and techniques to visualize, filter and analyze network flow data.

Researchers in [32] covers the general practice of network security monitoring and incident response, and provides lots of background knowledge on network topologies and sensor placement. Common tools are explained and their usage is demonstrated.

Researchers in [33] present several feature selection algorithms among classification techniques to identify network anomalies and vulnerabilities at various layers. They also cover the assessment of network anomaly detection systems, present different tools and discuss research challenges.

Researchers in [34] presents techniques for experimental data analysis, traffic behavior analysis, data collection using sensors and network mapping using python. He also deals with application identification and provides various ways to visualize network data.

Researchers in [35] describes requirements for a network intrusion detection system and explained the structure and functionality of the bro network monitor. Although this paper is more than 20 years old at the time of this writing, it contains fundamental aspects of network security monitoring, and inspired this research project with its philosophy of separating mechanism from policy.

Researchers in [36] reduce the 41 features widely adopted from the KDD dataset, to only 16 features, while still achieving similar detection results as with the full feature set. They used several filter algorithms including Weight by Maximum Relevance (WMR), Stepwise Regression and Stability Selection. For validation the Bayes Classifier was used, besides Support Vector Machines (SVM). This research paper served as a motivation for questioning the need for overly complex extracted features, and served as a hint to conduct experiments with a more basic feature set.

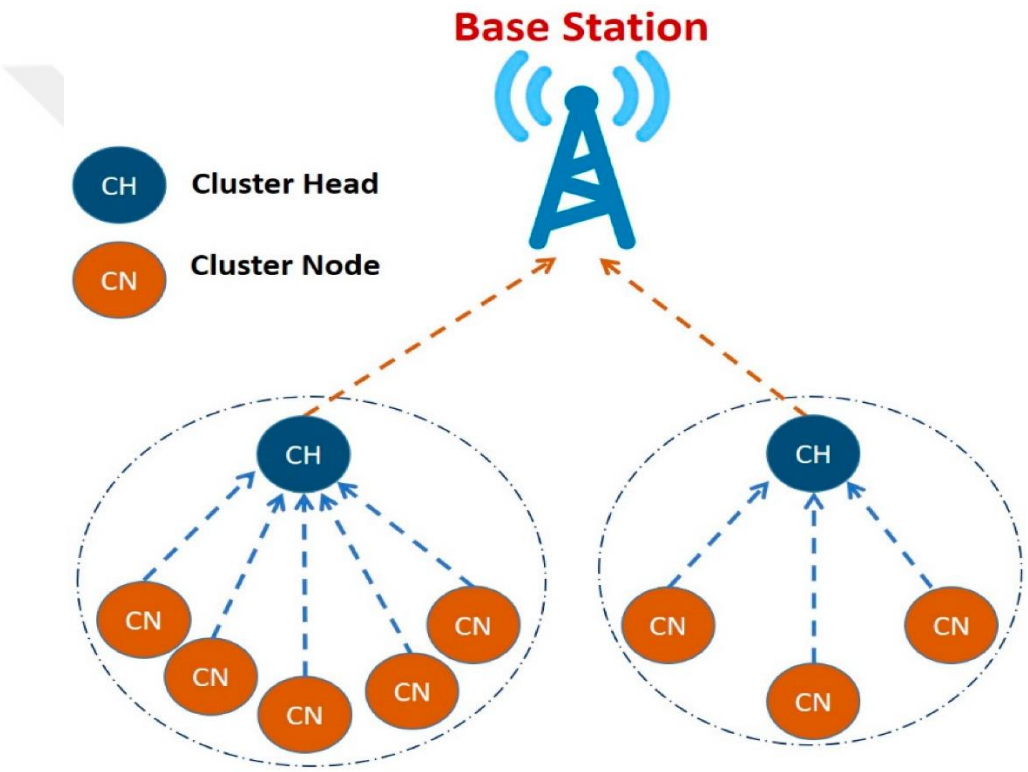


Figure 2.1: The basic schema of base station with two-cluster head and several cluster nodes [37].

Researchers in [38] presented their packet extraction tool for large volume network traces and compared the performance to existing solutions. Unfortunately, while their implementation could have been of great interest to the research community, they do not seem to have released it. The fact that their results could therefore not be evaluated, and that future research will need to continue with inefficient tooling, greatly encouraged me to publish my whole tool chain, in order to assist future researchers and allow independent verification of my results.

Researchers in [39] analyzed the state of the art for traffic flow analysis, and provided an overview and comparison of common tools and capture formats. It served as a useful introduction to flow-based collection methodology.

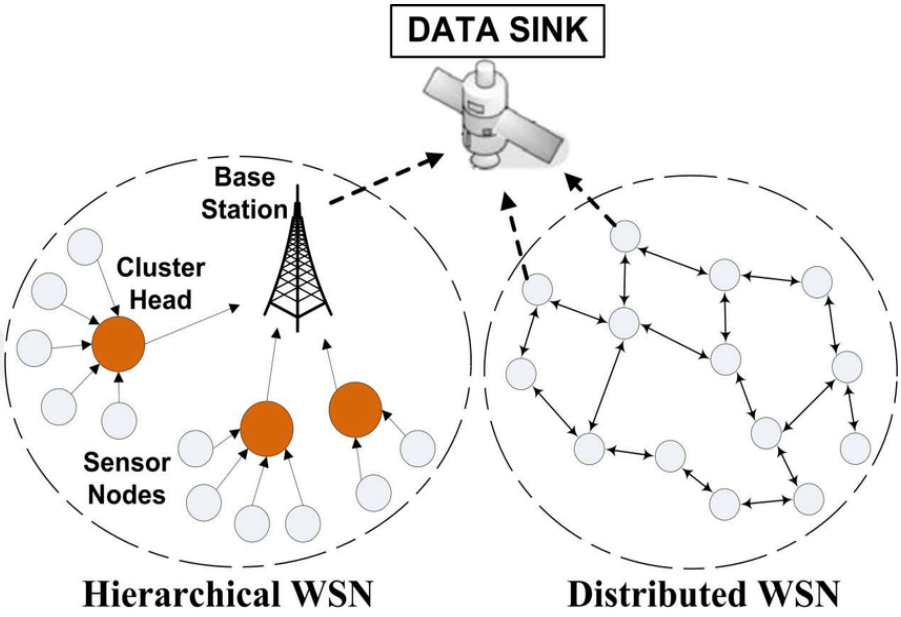


Figure 2.2: Conceptual model comparison between Hierarchical WSN and Distributed WSN [40].

Table 2.1: Signature based IDSs.

IDS	Mechanism	Attacks	Evaluation metrics
[25]	Collaborative	Black hole, selective forwarding	Window length, false negative rates
[26]	Local and cooperative detection	Sink hole	Detection rate, false negative rates
[27]	Hierarchical	N/A	N/A
[28]	Genetic programming	DoS, unauthorized access	Classification accuracy
[29]	Soft computing	Unauthorized access, probing	Classification accuracy
[30]	Specification based	Repetition attack, delay attack, worm hole, alteration attack, black hole, selective forwarding	Detection rate, false positives
[31]	Spontaneous watchdog	N/A	N/A
[32]	Ant colony	Abnormal transmission	N/A

2.2 FUNCTIONAL REQUIREMENT

The feature collector must be able to parse as many protocols as possible, in order to get the most detailed view on activity on the network. Additionally, the system must report on traffic that it does not understand.

2.2.1 Data Availability

For research purposes, the network monitor should make the collected data available at the lowest possible level, prior to generating summary structures. That means, protocol specific information should be offered for every protocol that can be interpreted by the monitor.

2.2.2 Abstraction Capabilities

Using the information that has been gathered in the collection stage, abstractions, summary structures, aggregate values and statistics can be generated from the collected data. A popular example for summary structures are unidirectional network flows and bidirectional connections.

2.2.3 Concurrent Design

With the evolution of multi-core processors and their decreasing cost, more and more systems are equipped with this technology. When designing a monitoring system, availability of several processing cores should be taken into account from the beginning. The term concurrency refers to the process of splitting a big problem into many small sub-problems, which unveils opportunities to execute independent operations in parallel as mentioned in [41]. Concurrency describes programming as the composition of independently executing processes. Parallelism refers to programming as the simultaneous execution of, possibly related, computations. While concurrency could also be expressed as dealing with lots of things at once, parallelism can be described as doing lots of things at once. While concurrency refers to structure, parallelism refers to execution. Although not identical, both are related. In order to enable concurrency, independent executions must be coordinated through communication. Since receiving network packets always happens sequentially, no matter if reading them from a dump file or live from the network card, parallelizing their decoding process is an option to utilize the power of multi-core

systems, and speed up the overall processing. Many systems haven't been designed with concurrent processing in mind from the ground up.

2.2.4 File Extraction Capabilities

Various network protocols allow the transfer of files or have been specifically created for this purpose. Examples include HTTP, HTTPS, FTP, SFTP and Bit-Torrent. Other protocols can be abused to carry file payloads as well. A prominent example for covert data exfiltration is the DNS protocol as mentioned in [42]. Encrypted communication such as HTTPS also makes inspection impossible, without access to the used certificate or cryptographic key. From a security monitoring perspective, files sent across a network need to be extracted to perform further analysis, like matching executables against anti-virus (AV) databases. Breaking encryption for this purpose should not be considered, because this weakens the overall system security, which is not a desirable effect.

2.2.5 Supported Input Formats

Suitable formats for offline network data input need to be supported, for this purpose the industry standards PCAP and the improved version PCAPNG are the best candidates as mentioned in [43]. PCAP preserves every single packet and the full payloads, and thus is a suitable input format for data collection. However, this comes at the cost of increased size of the dump files, which is especially problematic when storing traffic over long periods of time. For this reason, other formats such as Cisco's NetFlow have been developed, that summarize traffic on a per flow basis without payload data, which greatly reduces size.

2.2.6 Suitable Output Formats

The output format of the collected data should not only be consumable by many different systems, libraries and frameworks, but also displayable in a human readable format. Comma separated values (CSV) is a commonly chosen output format for this, it is not limited to a particular character set, and work just as well with Unicode as with ASCII as mentioned in [44]. However, CSV do not preserve the character set currently in use, this must be communicated separately. Although human readable in its original form, CSV is not a space efficient mechanism of storing or exchanging information. Additionally, CSV cannot represent

hierarchical or object-oriented structured data, without separating sub structures from its parent context.

2.2.7 Real-Time Operation

Live capture from a network interface should be supported, to allow operation and monitoring in real time. Since dumping the collected data to disk is not always an option due to disk space constraints, e.g. on embedded devices, capture to disk should be optional and an alternate data collection mechanism should be provided. Uncovering successful breaches in real time and raising alerts, puts pressure on attackers to move quicker in a compromised network. This increases the possibility for mistakes and leaves less time for cleaning up traces, which is beneficial for damage reduction, incident response and digital forensics as mentioned in [45]. A system that is too slow might generate alerts with a high delay, thus giving attackers more time to accomplish their goals. Identifying attempted breaches in real time gives network security operators valuable information about targeted systems and used techniques, and helps to put counter measures into place.

2.3 NON-FUNCTIONAL REQUIREMENTS

When parsing untrusted input, it must be guaranteed that there are no exploitable vulnerabilities in the parser code. Since modern network communication is based on a huge stack of protocols, many different and potentially very complex parsers have to be implemented and maintained, which tremendously increases the chances of programming errors in the corresponding parser code.

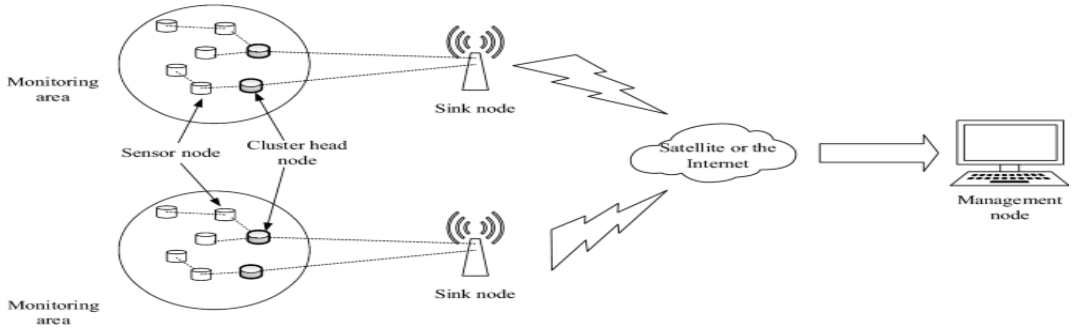


Figure 2.3: Equality constrained optimization in wireless sensor network with management node and sensor nodes [46].

2.3.1 Memory Safety

Code audits are often omitted due to high cost, or only performed on a fraction of the available source code. Additionally, many vendors violate the RFC specifications for protocols, which requires parsers that are fault tolerant and have been tested with fuzzing extensively. Choosing a language with a memory safe garbage collected runtime for the design of a feature collection system, allows to focus on creating robust parsing logic rather than secure memory handling.

2.3.2 Open Source Codebase

The source code should be accessible and well documented, to enable other researchers to audit or extend the code. Although this makes the code also available to potential attackers, the security model of the network monitor must not rely on secret code.

2.3.3 Scalability

The system should be scalable to meet requirements for large scale distributed deployments, such as receiving input data from many sensors across the network. This should happen in an efficient and secure way, that means data should be compressed when being transferred, and the network monitor must encrypt its communication to prevent eavesdropping.

2.3.4 Performance

The number packets of that has to be captured and processed, is rising continuously. The network monitor must be efficient enough to keep up with all the data, without losing anything. Although an alert does not automatically prevent an intrusion, it is a first step towards detection, and increases the risk for an attacker to be discovered. An overloaded system that starts to drop packets, can lead to missing the discovery of an ongoing attack. Performance depends heavily on the programming language used for the implementation. As mentioned before, low level system programming languages do not provide memory safety. Because of this, the usage higher level languages such as python programming for the design of security critical infrastructure, should be considered in the future. This introduces a performance penalty; however, this is a trade-off that comes with great benefits for the overall application security. A recently published paper implemented a POSIX compliant kernel in python, and measured a performance penalty ranging

from 5-15% in comparison to the python implementation. Today's networks transport lots of huge traffic streams, for example video streaming such as netflix, torrent downloads and more. Because these flows cause a huge performance degradation for the monitoring system, a common practice is to detect those so called 'elephant flows' and discard them at the earliest stage possible as mentioned in [47]. However, this imposes the security risk of missing relevant information, an attacker could use the knowledge of certain traffic types being discarded to hide in a big data stream. Since audio and video players and their codecs are a common source for security vulnerabilities, discarding these flows may lead to missing an actual intrusion, due to an exploited vulnerability in the media player. Searching the MITRE CVE database for the VLC media player for example returns 92 results as mentioned in [48]. An optimal implementation should not discard any data to meet performance goals.

2.3.5 Configurable Design

The monitor application needs to be configurable, to allow performance tweaking and adaption to special requirements, such as for example privacy regulations. Automated capture and analysis of network traffic can result in severe privacy violations, and may violate local, state and national laws. Therefore, it is necessary to obtain the required permissions for the target network as well as qualified legal advice, prior to installing any monitoring software. How to protect the user's privacy during a network security monitoring operation has become an important question for many network administrators. Since the 28th of May 2018, the General Data Protection Regulation (GDPR) further increases protection of personal data from citizens of the European Union, by drastically increasing the fines for violations as well as the scope of protected information as mentioned in [49].

2.3.6 Extensibility

Computer networks are highly dynamic environments and thus require the network monitor to be easily extensible, to integrate new protocols as fast as possible and keep up with the extremely fast development of technology. For this reason, the monitoring system should allow to add parsers for new protocols in a fast and secure manner, and should provide an interface to further extend functionality.

2.3.7 Reliability

The monitor must always be available and deliver meaningful results, even under attack or in high load scenarios. Also, critical errors must not lead to a shutdown of the monitoring system, but should notify the administrator and rollback the system to working state as mentioned in [50]. Recovering from system failure, no matter if due to logic errors or due to a fault produced by an attacker, it is important to ensure continuous monitoring. Reporting and recovering on parsing errors or un-decodable traffic is mandatory, because the monitor will be subject to attacks.

2.3.8 Usability

The user interface of the network monitor should be powerful enough to accomplish complex tasks, but also be simple enough to be efficiently used by a human, without confusion or waste of time. Since network data is very abstract for humans, the software should provide proper visualization options, to assist humans in understanding, correlating and analyzing huge amounts of high dimensional data in a short period of time.

2.3.9 Storage Efficiency

Storage is limited and the continuously increasing amount of data that modern networks are transporting increases the amount of collected audit data. Therefore, the generated output should be as small as possible, while preserving all information relevant for the detection task as mentioned in [51]. Additionally, it should be configurable what data will be collected. In some scenarios there might be data that is not relevant for the detection task or a concrete monitoring system. If this is the case, no resources should be wasted on collecting or storing irrelevant data.

2.4 FEATURE COLLECTION IN IDS SYSTEM

Feature collection systems for intrusion detection should run continually without or with minimal human supervision, and should be reliable enough to run in the background of the observed system and impose minimal overhead. The system must be fault tolerant and able to recover from errors automatically, and should preserve its own integrity in order to detect attacks against the monitor itself. Also important are aspects like re-configurability and extensibility, since

dynamic environments such as computer networks often require adaptations. Deployment should be easy and fast and the system must be able to observe deviations from normal behavior. Additionally, they must provide an efficient user interface and provide visualization to assist human analysts. The named requirements for the underlying hardware also play a vital role in ensuring the monitoring systems integrity and performance. For future reference, all software requirements for modern network intrusion detection systems have been listed in table 2.1.

Table 2.2: Summary of feature collection requirements [52].

Requirement	Short Description
Protocol Support Coverage	Range of supported protocols
Data Availability	Access to data on the lowest level
Abstraction Capabilities	Generation of abstractions such as flows
Concurrent Design	Efficient use of multi-core architectures
File Extraction	Extraction of files from network traffic
Suitable Output Format	Widely supported and efficient output format(s)
Real Time Operation	Live acquisition of traffic from a network interface
Supported Input Formats	Support for industry standard input formats
Memory Safety	Secure memory management
Open Source Codebase	Publicly available source code
Scalability	Operation in large scale, possibly distributed networks
Performance	Efficient processing for high volume traffic
Configurable Design	Configuration and adaption options for special requirements
Extensibility	Ease of extension for new protocols
Reliability	Reliable implementation
Usability	Ease of use for the analyst / researcher
Storage Efficiency	Low storage overhead

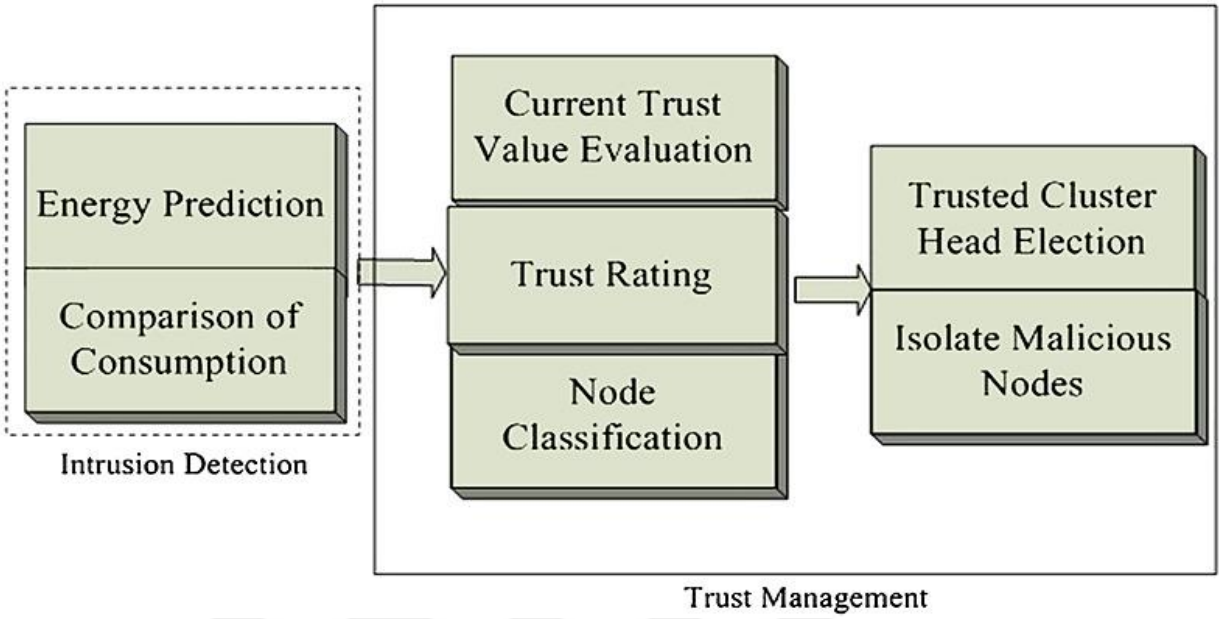


Figure 2.4: Network re-configurability and extensibility with two modules in the security scheme for intrusion detection [53].

3. METHODOLOGY

Wireless sensor networks (WSN) are normally used to evaluate one or more physical criterion in a vast area. A WSN consists of hundreds of small nodes (sensors). These sensors measure the physical attributes of the area such as temperature, pressure, humidity, air pollution and etc. They are widely used in different fields such as natural disasters monitoring like floods, earthquakes, and volcanic and other geological accidents. As well, they are used in military, robotics, automated warehouses, chemical and medical applications. One of the most popular usage of the WSN in the nature to measure temperature, pressure, humidity and disasters in the mountains. We assume that each sensor is a small device with a limited energy source and memory, a system to send and receive the data to other nodes and some component to measure the characteristics of the area. In most WSN applications, knowing the sensor locations is of crucial importance for the system to work. Indeed, without the exact position of the sensors, there is limited use to their data. In many applications, the exact position of only a few nodes may be in the hand a-priori. Thus, the topology of the rest of the network can be completely random and ad-hoc. WSN is an active area for researchers and there are many surveys studying WSN-related problems. In most of these studies the problem of providing per-node location information has been addressed.

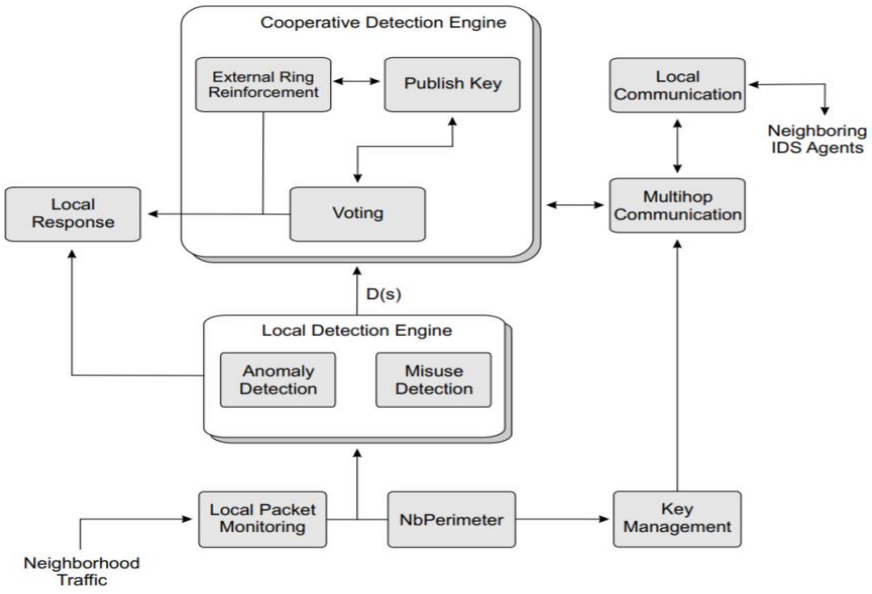


Figure 3.1: The architecture and flow of intrusion detection system (IDS) in Wireless Sensor Network (WSN).

3.1 WIRELESS SENSOR NETWORK

We distinguish WSNs from other conventional or wired even wireless networks through sensor based interaction with the environment. WSNs typically possess little (or sometimes no) infrastructure. They consist of a number of sensor nodes which can range from few tens to thousands. These sensors work together to monitor and measure a specific region to obtain information about the environment. In most cases, we are interested in the usage of inexpensive sensors, which intrinsically have limited processing components. This chapter focuses on classification and evaluation of the approaches proposed in the literature for the localization processes for intrusion detection system within wireless sensor network. They can be classified in different categories based on various criteria. In the thesis, we consider the problem as distributed systems problem. We also assume the direct connection may not be established across the whole network. Thus, one of the most significant categorization with regard to hardware limitation.

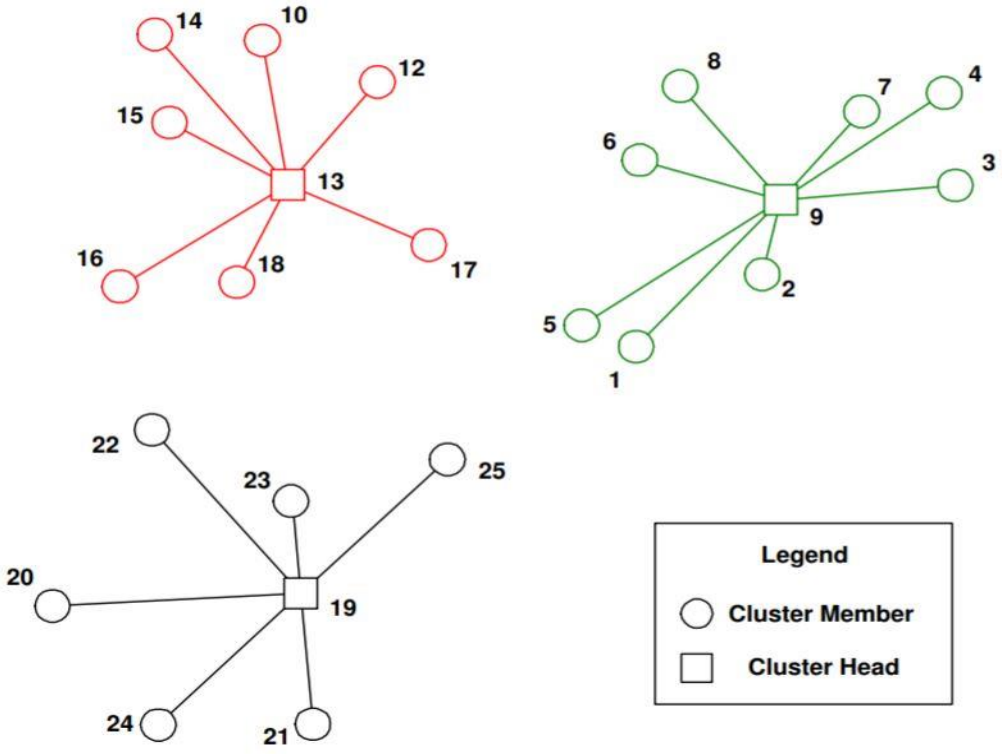


Figure 3.2: Topology of wireless sensor network after organization into cluster.

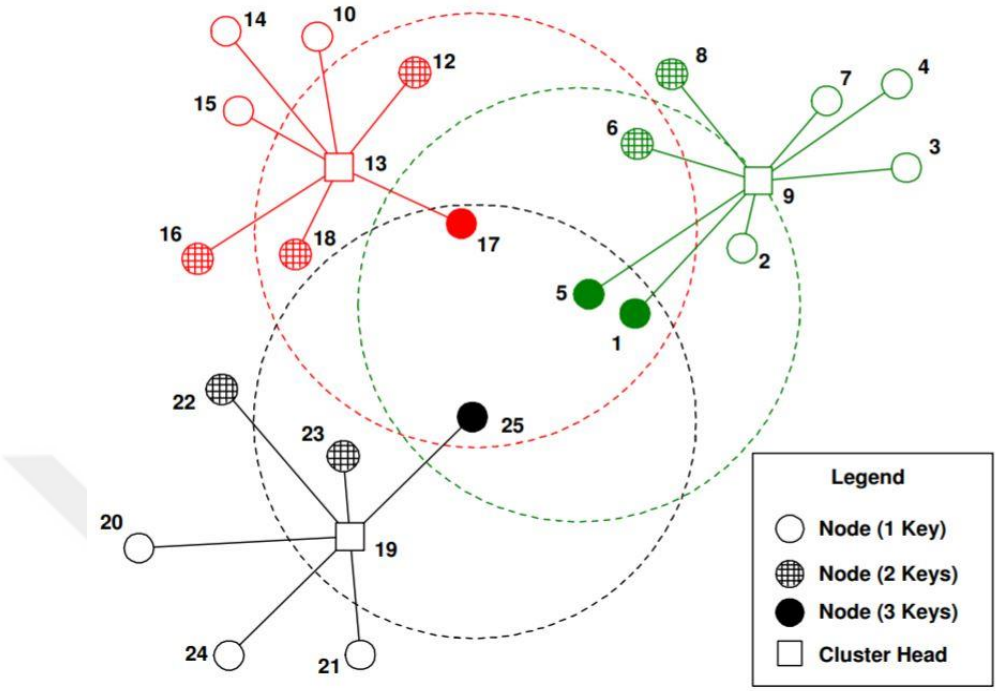


Figure 3.3: Topology of wireless sensor network during the key setup phase.

3.2 DATASET DESCRIPTION

Anomaly-based intrusion detection approaches suffer from the lack of reliable datasets for testing and verification. Evaluations of the eleven datasets that exist since 2017, conducted by the Canadian Institute of Cybersecurity, have shown that most datasets are out of date and unreliable for evaluation purposes. While some of these datasets fail to provide sufficient diversity and volume of network traffic, some do not contain latest attack patterns or anonymize packet payload data, which limits research capabilities. Additionally, some are also lacking feature set and metadata information. An interesting approach to- wards the creation of realistic datasets was presented in 2015, when researchers released their framework for injecting attacks into existing datasets. Although their tool was not used in this research, it is a promising option for future experiments. The CIC-IDS-2017 dataset has been chosen as the most up to date and most extensively documented dataset, and will be used for the experiments [54]. The dataset consists of CSV flow records generated with CICFlowMeter, and the PCAP files used for creation. Data was captured for a total of five days and no anonymization has been performed on the dataset. The total size of all included PCAP dumpfiles is 48.9 GB. Attacks have been carried

out on the working days Tuesday, Wednesday, Thursday and Friday, in both morning and afternoon. The implemented attacks include DoS & DDoS (Wednesday), Heartbleed (Wednesday), Web Attacks (Thursday), FTP Brute Force & SSH Brute Force (Tuesday), Infiltration (Thursday), Botnet traffic and Port Scans (Friday). Monday is the normal day and includes only legitimate traffic. Detailed information on the dataset and the testbed architecture can be found in the corresponding research paper and on the CIC website.

3.3 RANGE-BASED LOCALIZATION

These methods use the point-to-point distance or angle information. The most popular techniques for estimating the distance and the angle between two nodes are:

3.3.1. Received Signal Strength Indicators (RSSI)

This technique relies on comparing the strength or the power of the received signal with the one sent at the transmitter node. This approach works with RF signals, it is relatively cheaper (no additional devices) and used more than the other techniques that will be described. It is also simple and inexpensive in the configuration and calibration process. However, one of its most important drawbacks is its sensitivity to noise. The distance estimation accuracy can be very bad specially for large distances.

3.3.2. Angle of Arrival (AoA)

As the name suggests, the technique is used to find the angle of the emitted signal from the source. This approach normally needs at least two antennas for each node. This requirement limits its usability in the networks with small and low-cost sensors.

3.3.3. Time of Arrival (ToA)

One of the classical methods to localize the indoor environment is ToA. The speed of the radio signal is equal to the speed of light (in the cases where audio signals are used for localization, we consider the sound speed instead of the light speed). If we are able to find the time that the signal has traveled, by dividing it with the speed of light we can estimate the distance between the sender and the receiver.

One of the most important issues to be considered in this technique is the time synchronization between the nodes. One of the ways to overcome synchronization problems in acoustic ToA estimation is to use two types of signals. The sender emits the radio signal and ultrasound propagation at the same time. The receiver receives the radio signal and starts the timer. When it receives the ultrasound signal it stops the timer. Thus it can compute the difference between them. This method achieves the high ranging precision but it needs additional hardware and consumes more energy.

3.3.4. Time Difference of Arrival (TDoA)

The measurement procedure for this technique is technically the same as the ToA method, but the relative time of arrivals between nodes are used to estimate the locations. By using the TDoA, we eliminate the need to know the exact signal emission time at the source.

3.4 RANGE-FREE LOCALIZATION

These schemes are applied as cost-effective alternative to the more expensive Range-Based approaches. In these approaches, there is no information about the distance and the angle of paths between the nodes. From an algorithm architecture perspective, we divide the available methods in two blocks:

3.4.1. Centralized Architectures

We select a node as a monitor for several nodes. It should be a powerful central base station for others. The selected node works as a coordinator for neighbors. The advantages of these algorithms are the easy implementation and the elimination of the computation process in each node. But if we lose the monitor node, we will lose a part of the region.

3.4.2. Decentralized Architectures

In the traditional centralized algorithms, we need some hubs and the data may be difficult to obtain. In decentralized algorithms the region is divided in many clusters and in each of them the transmission is possible between all nodes together. Thus, compared to the former structure, we have more reliability.

Adding a GPS device to each node in large-scale ad-hoc networks, may increases the price and the size of the sensors. As well, it affects the consumption of the power and reduce the network lifetime and its performance. Thus it may be not a good idea to add a space-based satellite navigation system that can provide the location to each node. This consideration also divides the approaches in WSN into two main categories

3.4.3. Anchor Based

In many approaches we can obtain the exact position of only a few nodes. Sometimes it is possible by using GPS. As well, we may know the ab- solute position of a node in the region. These nodes are called anchor or beacon node.

3.4.4. Anchor Free

Based on the cost, consumption and environment constraint we might not be able (or not interested) to have sensors with known exact geographical location. In other words, there are no anchor nodes in the network. In these cases, localization schemes based on distance information are only able to find the relative positions of the sensors in the network.

In the real environment, we may get incorrect data from the nodes or we may lose some nodes. Thus there is a random and unwanted variable for each node that can affect the solution. The random variable modeling the error is independent for each node.

There is also another category for network localization algorithm. This category is divided the methods in two main groups of methods based on the granularity of data gotten by the sensors.

3.4.5. Fine-Grained Algorithms

These methods use accurate information during the communication such as the distance from a reference node determined using RSSI or ToA measurements. In general, this type of algorithms uses several technologies, such as ultrasound, infrared, or radio frequency signals. In the absence of measurement errors, fine-grained algorithms provide exact network nodes positioning.

3.4.6. Coarse-Grained Algorithms

Basically, these algorithms use less accurate in- formation for the localization of unknown nodes. These algorithms always, estimate the distances between the sensors in the network with some techniques such as hop-count. Figure 3.4 shows the possible categories to divide the localization algorithms in WSN.

Especially in the range-based proposals the noise is typically dependent on the devices and interfaces. Thus many of them work only in isolated environment.

i) They try to estimate the distance or angle of two nodes.

ii) They try to combine the achieved distances or angles to estimate the locations of the sensors.

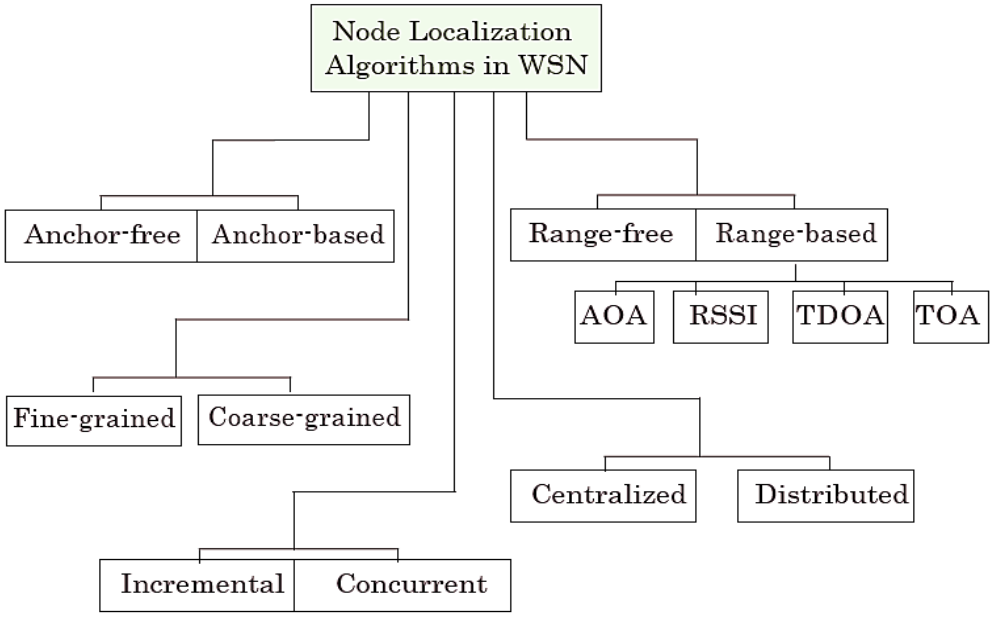


Figure 3.4: Classification of the approaches in WSN area.

There are six metrics which determine the high level of performance in localization algorithms:

a. Accuracy: The algorithms try to find the closest value of measurement to the actual value. Thus, accuracy or precision is the most important factor in the localization techniques.

b. Coverage: Each node has several neighbors. It can have transmission with them. Of course, the nodes can exchange data on short distances. Each connection technology operates in a special range. The effective coverage varies due to many criteria such as propagation conditions, antenna, battery power and etc. So for achieving the better coverage and better performance we need an adequate density of nodes and the number of anchor nodes.

c. Complexity: In networks where the ranging information is sent between the sensors (range-based) the devices need to have more complex hardware com- pared to networks which rely only on the connectivity information (range-free). The former architecture of the network is more complex.

d. Scalability: The ideal algorithm has the ability to be enlarged. It is one of the properties of each localization algorithm. It should be suitable to apply for larger situations. In other words, when the network gets larger, the algorithm does not need to change.

e. Robustness: In the outdoor environment all signals are noisy with a random error, and we expect to not unduly affect the performance of algorithm. Perhaps, we lost some nodes and some parts of the networks. In the range-based algorithm, we always work with the angles, distances, and time. Thus, if we do not apply the reliable and robust algorithm, we will face a lot of errors at the end.

f. Cost: One of the most important criteria in the wireless sensor networks is the price of each sensor. With a small change in hardware and software of nodes we may save or lose a lot of money. One of the factors that effects on the price is energy consummation. If the sensors consume less amount of energy, we could cut the cost. On the other hand, if our algorithm can find the position of sensors rapidly, we do not need long life battery.

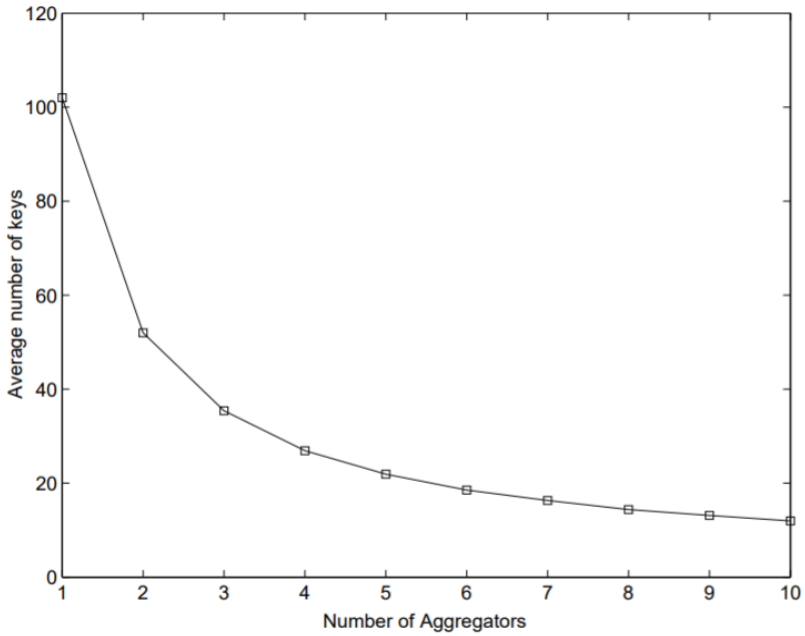


Figure 3.5: The 100 keys used as an aggregator for nodes in WSN.

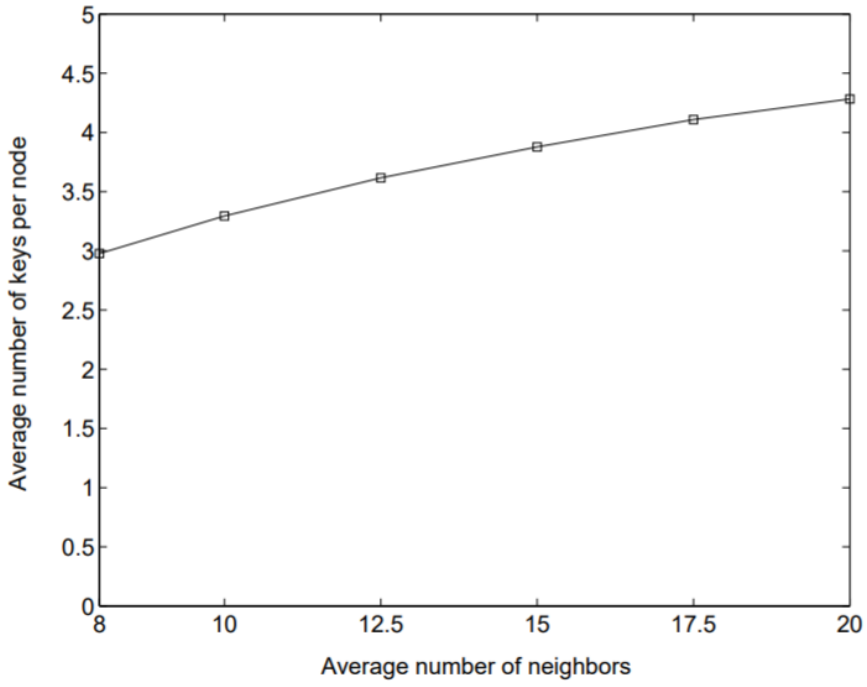


Figure 3.6: The keys held by sensor nodes for 20 neighbors in the WSN network.

3.5 IMPLEMENTATION

Python is a statically typed and compiled imperative systems programming language released by python software foundation in 1991. It is syntactically similar to the Go programming language, but adopted lots of ideas from other languages, in order to improve readability and productivity. Commonly used for network programming and backend implementation, Python is known for its extremely fast compile time and generation of statically linked binaries, which are independent of any libraries on the execution environment. Python is currently available in version 3.8 and provides a built-in model for concurrent execution and coordination of asynchronous processes, which was inspired by the Communicating Sequential Processes (CSP) paper. An asynchronous process is called a go routine, which should not be confused with an operating system (OS) thread. WSN in python are multiplexed onto threads of the OS as required. In case a WSN blocks, the corresponding OS thread blocks as well, but no other block is affected. WSNs are less computationally expensive compared to a thread and allocate resources dynamically as needed. For synchronization and messaging, python offers channels as a lightweight way to communicate between WSN. This design decision was inspired by the idea of sharing memory by communicating, instead of communicating by sharing memory.

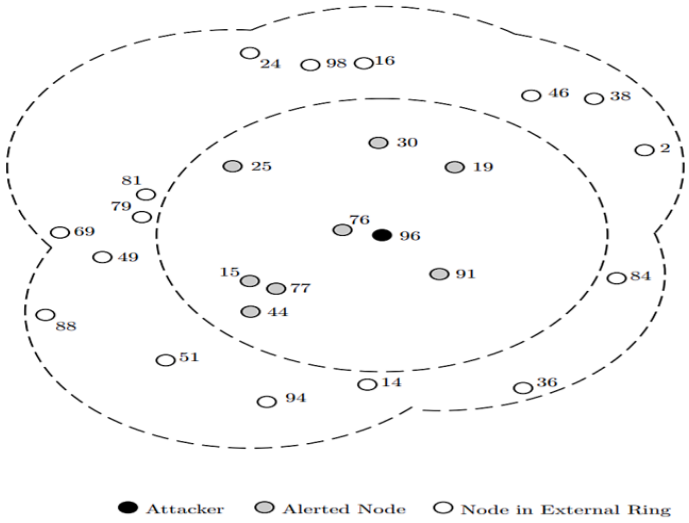


Figure 3.7: The ringing concept in wireless sensor network for attacker identification.

Additionally, multi-way concurrent control is provided through select statements, which are used to receive data from channels. Another important aspect of python is its rich platform and architecture support, which will be described in detail below. Besides that, the language offers a

garbage collected runtime, and stack management, in order to combat memory corruptions. However, this should not be confused with the virtual machine approach of the python runtime. Although python is an object oriented programming language, it provides interfaces and the ability to define methods on structures. Python also enforces a uniform programming style for formatting the source code, which greatly helps increasing readability across code from different authors. It is noteworthy that python provides functionality to circumvent the type system and runtime checks, by reading and writing to arbitrary memory addresses via the unsafe package.

Table 3.1: WSN protocol sub structures

Name	**Description**
Dot11QOS	IEEE 802.11 Quality Of Service
Dot11HTControl	IEEE 802.11 HTC information
Dot11HTControlVHT	IEEE 802.11 HTC information
Dot11HTControlHT	IEEE 802.11 HTC information
Dot11HTControlMFB	IEEE 802.11 HTC information
Dot11LinkAdapationControl	IEEE 802.11 HTC information
Dot11ASEL	IEEE 802.11 HTC information
LLDPChassisID	Link Layer Discovery Protocol information
LLDPPortID	Link Layer Discovery Protocol information
LinkLayerDiscoveryValue	Link Layer Discovery Protocol information
LLDPSysCapabilities	Link Layer Discovery Protocol information
LLDPCapabilities	Link Layer Discovery Protocol information
LLDPMgmtAddress	Link Layer Discovery Protocol information
LLDPOrgSpecificTLV	Link Layer Discovery Protocol information
IPv4Option	IPv4 option
ICMPv6Option	ICMPv6 option
TCPOption	TCP option
DNSResourceRecord	Domain Name System resource record
DNSSOA	Domain Name System start of authority record
DNSSRV	Domain Name System service record
DNSMX	Mail exchange record
DNSQuestion	Domain Name System request for a single domain
DHCPOption	DHCP v4 option
DHCPv6Option	DHCP v6 option
IGMPv3GroupRecord	IGMPv3 group records for a membership report
IPv6HopByHopOption	IPv6 hop by hop extension TLV option
IPv6HopByHopOptionAlignment	Hop By Hop Option Alignment

Table 3.2: WSN Layer Encoder Fields.

Layer	NumFields	Fields
TCP	22	Timestamp, SrcPort, DstPort, SeqNum, AckNum, DataOffset, FIN, SYN, RST, PSH, ACK, URG, ECE, CWR, NS, Window, Checksum, Urgent, Padding, Options, PayloadEntropy, PayloadSize
UDP	7	Timestamp, SrcPort, DstPort, Length, Checksum, PayloadEntropy, PayloadSize
IPv4	17	Timestamp, Version, IHL, TOS, Length, Id, Flags, FragOffset, TTL, Protocol, Checksum, SrcIP, DstIP, Padding, Options, PayloadEntropy, PayloadSize
IPv6	12	Timestamp, Version, TrafficClass, FlowLabel, Length, NextHeader, HopLimit, SrcIP, DstIP, PayloadEntropy, PayloadSize, HopByHop
DHCPv4	16	Timestamp, Operation, HardwareType, HardwareLen, HardwareOpts, Xid, Secs, Flags, ClientIP, YourClientIP, NextServerIP, RelayAgentIP, ClientHWAddr, ServerName, File, Options
DHCPv6	7	Timestamp, MsgType, HopCount, LinkAddr, PeerAddr, TransactionID, Options
ICMPv4	5	Timestamp, TypeCode, Checksum, Id, Seq
ICMPv6	3	Timestamp, TypeCode, Checksum

ICMPv6Echo	3	Timestamp, Identifier, SeqNumber
ICMPv6NeighborSolicitation	3	Timestamp, TargetAddress, Options
ICMPv6RouterSolicitation	2	Timestamp, Options
DNS	18	Timestamp, ID, QR, OpCode, AA, TC, RD, RA, Z, ResponseCode, QDCount, ANCount, NSCount, ARCount, Questions, Answers, Authorities, Additional
ARP	10	Timestamp, AddrType, Protocol, HwAddressSize, ProtAddressSize, Operation, SrcHwAddress, SrcProtAddress, DstHwAddress, DstProtAddress
Ethernet	6	Timestamp, SrcMAC, DstMAC, Ethernet-Type PayloadEntropy, PayloadSize

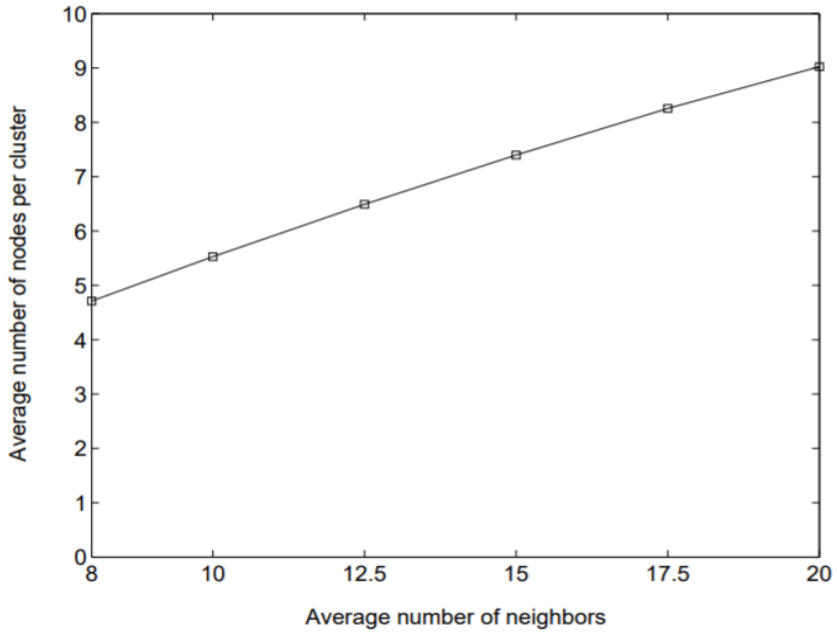


Figure 3.8: Wireless sensor network has average number of nodes in clusters.

3.6 FLOWS AND CONNECTIONS IN WSN

Flow audit records in WSN are unidirectional, whereas Connections and Layer Flows are bidirectional. Flows and Connections are continuously flushed, to keep the memory usage as low as possible. Timeout intervals can be configured separately for both types. The first packet seen decides about source and destination of a Connection or Flow.

Table 3.3: Diagram that describes the different networks with data transfer rate.

Name	**Layer**	**Description**
Flow	All	Unidirectional Multi-Layer Flow
Connection	All	Bidirectional Multi-Layer Flow
LinkFlow	Link	Bidirectional Link Layer Flow
NetworkFlow	Network	Bidirectional Network Layer Flow
TransportFlow	Transport	Bidirectional Transport Layer Flow

For Flows and Connections, the following identical information is preserved:

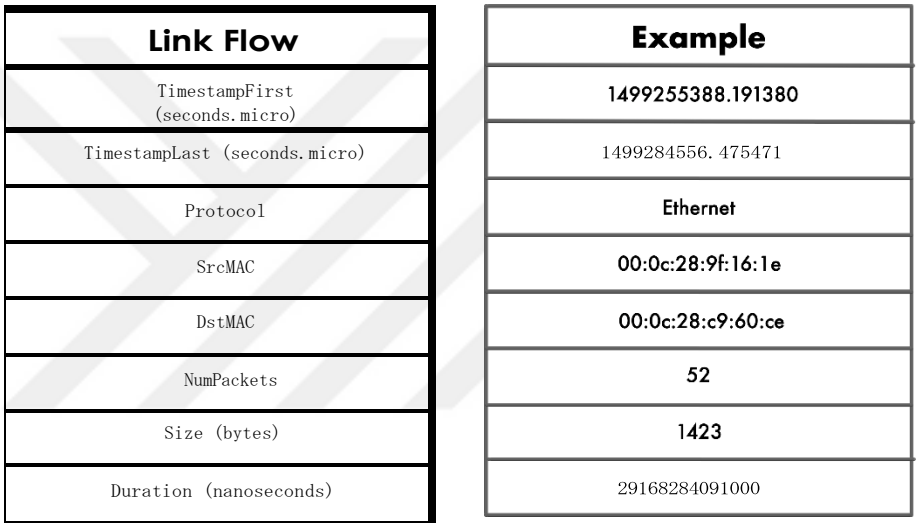
Flow / Connection	Example
Timestamp First Seen (seconds.micro)	1499257434.003136
Link Layer Protocol	Ethernet
Network Layer Protocol	IPv4
Transport Layer Protocol	TCP
Application Layer Protocol	HTTP
Source Mac Address	00:0c:28:9f:16:1e
Destination Mac Address	00:0c:28:c9:60:ce
SrcIP	173.15.11.103
SrcPort	1873
DstIP	95.100.238.75
DstPort	80
Size in bytes	922
Number of Packets	6
Timestamp Last Seen (seconds.micro)	1499257551.553088
Duration (nanoseconds)	117549952000

Figure 3.9: WSN flows and connections for intrusion detection.

3.6.1. LinkFlow in WSN

In order to further decouple Flows, they are also exported for each layer, e.g: LinkFlow, NetworkFlow, TransportFlow. As mentioned previously, Layer Flows are bidirectional, their UID is the same for both directions. Note that in the following graphics the UID field was omitted. A LinkFlow describes communication on layer 1 of the Internet Protocol Suite. It contains the hardware adresses of the communicating devices, and connection statistics.

Figure 3.10: WSN Link Layer Flow for intrusion detection.



Link Flow	Example
TimestampFirst (seconds.micro)	1499255388.191380
TimestampLast (seconds.micro)	1499284556.475471
Protocol	Ethernet
SrcMAC	00:0c:28:9f:16:1e
DstMAC	00:0c:28:c9:60:ce
NumPackets	52
Size (bytes)	1423
Duration (nanoseconds)	29168284091000

3.6.2. Network Flow in WSN

A NetworkFlow describes communication on layer 2 of the Internet Protocol Suite. It contains the IP addresses of the communicating devices, and connection statistics.

Network Flow	Example
TimestampFirst (seconds.micro)	1499283872.749692
TimestampLast (seconds.micro)	1499284166.267369
Protocol	IPv4
SrcIP	192.168.10.8
DstIP	185.11.128.203
NumPackets	137
Size (bytes)	30160
Duration (nanoseconds)	293517677000

Figure 3.11: WSN Network Layer Flow for intrusion detection.

3.6.3. Transport Flow in WSN

A TransportFlow describes communication on layer 3 of the Internet Protocol Suite. It contains the ports of the communicating devices, and connection statistics.

Transport Flow	Example
TimestampFirst (seconds.micro)	1499279129.560185
TimestampLast (seconds.micro)	1499282267.760859
Protocol	TCP
SrcPort	60846
DstPort	443
NumPackets	156
Size (bytes)	118942
Duration (nanoseconds)	3138200674000

Figure 3.12: WSN Transport Layer Flow for intrusion detection.

3.7 CLUSTER BUILD INTRUSION DETECTION

In concurrent programming, shared resources need to be synchronized, in order to guarantee their state when modifying or reading them. If access is not synchronized, race conditions occur, which will lead to faulty program behavior. To avoid this and detect race conditions early in the development cycle, the go toolchain offers compiling the program with the race detector enabled. This will let the application crash with stack traces to assist the developer in debugging, if a data race occurs. Programs with active race detection are slower by the factor of 10 to 100. To compile a python program with the race detection enabled the **-race** flag must be added to the compilation command.

To add support for a new protocol or custom abstraction the following steps need to be performed. First, a type definition of the new audit record type must be added to the *netcap.proto* protocol buffers definitions, as well as a *Type* enumeration following the naming convention with the *NC* prefix. After recompiling the protocol buffers, a file for the new encoder named after the protocol must be created in the *encoder* package. The new file must contain a variable created with *CreateLayerEncoder* or *CreateCustomEncoder* depending on the desired encoder type. Depending on the choice of the encoder type, the new variable must be added to the *customEncoderSlice* in *encoder/customEncoder* or *layerEncoderSlice* in *encoder/layerEncoder*. Next, the interface for conversion to CSV must be implemented in the *types* package, by creating a new file with the protocol name and implementing the *CSVHeader() []string, CSVRecord() []string, NetcapTimestamp() string* functions of the *types.CSV* interface. If the new protocol contains sub-structures, functions to convert them to strings need to be implemented as well. Finally, the *InitRecord(typ types.Type) (record proto.Message)* function in *netcap.go* needs to be updated, to initialize the structure for the new type.

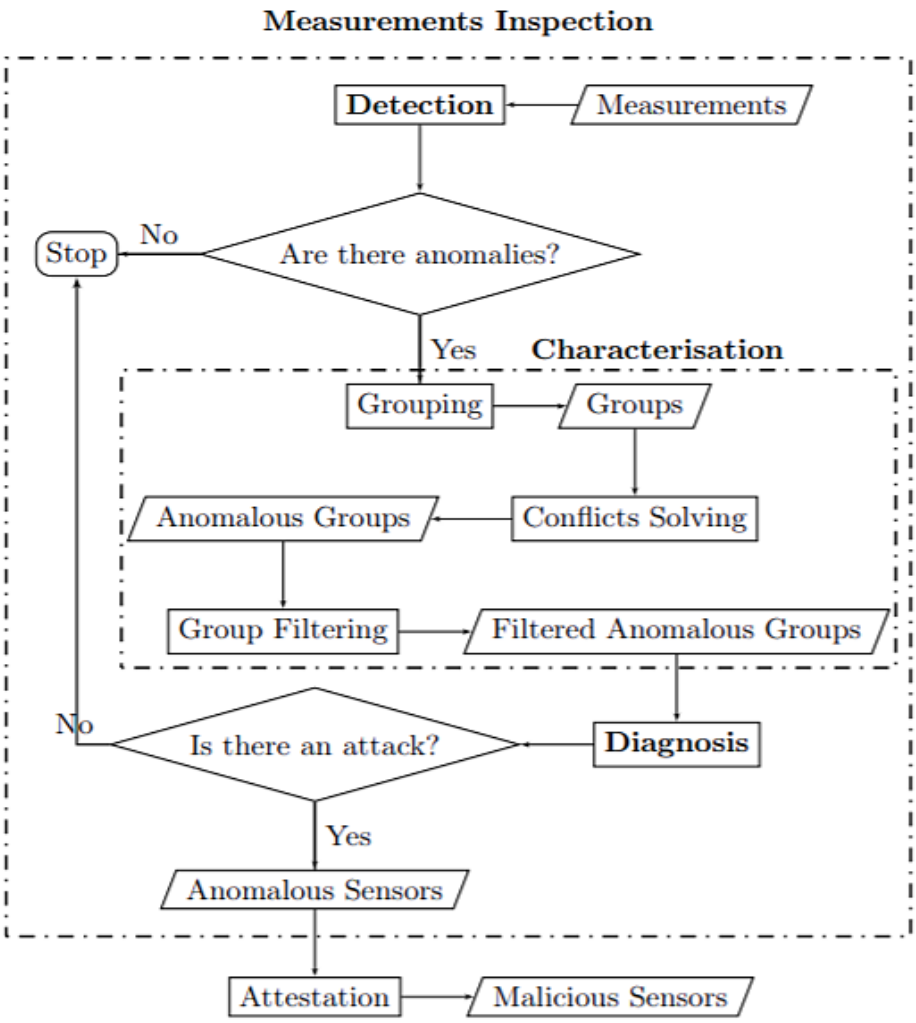


Figure 3.13: The flowchart of approach being followed.

Processing time depends heavily on the type of traffic, configuration and system performance. Processing times on the development machine (a recent MacBook Pro with 32 GB 2400 MHz DDR4, 2,9 GHz Intel Core i9, running Windows 10), using the default configuration with 4096 byte buffers, compression, all 40+ encoders, 1000 workers and a packet buffer size of 100, can be found in the experiments. The project may contain bugs, that have not yet been discovered. Error handling is not very graceful, in many cases that could have been handled otherwise, the program panics in order to assist in debugging with a stack trace. Until there are further unit tests and the error handling is more robust, using python for other purposes than research is recommended. The following shows a summary of statistics regarding lines of code (LoC) in python version 3.8 (excluding generated code for the protocol buffers):

Table 3.4: Python code statistics for intrusion detection within files.

Package	Language	Files	Blank	Comment	Code
collector	Python	13	273	371	882
types	Python	38	232	465	1721
label	Python	15	365	413	1266
encoder	Python	48	559	957	3299
cmd	Python	3	57	64	282
utils	Python	2	39	44	156
netcap.proto	XML	1	93	100	660
total	Python	126	1693	2520	8296

Python source code contains many descriptive comments (80-100 lines of comments on several lines of code as of version 3.8), in order to ease future development by other researchers and to make implementation details comprehensible.

Table 3.5: Statistics for processing input packets for intrusion detection.

File	Num Packets	Size	Processing Time	Extracted Data
Tuesday-WorkingHours.pcap	11551954	11 GB	11m11.918614271s	440 MB
Wednesday-WorkingHours.pcap	13788878	13 GB	12m58.13498869s	536 MB
Thursday-WorkingHours.pcap	9322025	8.3 GB	8m39.274062535s	369 MB
Friday-WorkingHours.pcap	9997874	8.8 GB	9m0.653867719s	410 MB

.

4. RESULTS

The research on WSN network and application layer data for several types of real-time attack. Moreover, the previous works on the application layer data in the paper titled "A Model for Anomalies Detection in Wireless Sensor Network" used only a 2-node clustering algorithm for comparison whereas we are comparing with 10-node WSN network with clusters and evaluating the accuracy. Furthermore, the research worked on application layer data. However, the data was collected from open-source repository and it was synthetic data set. Our data is non-synthetic and attained from an open-source repository dedicated for wireless sensor network based systems. Another important addition to our research is the analysis of the time taken by each algorithm to execute itself as we have seen that our own algorithm took the least execution time for the execution. As we discussed earlier in literature review that in wireless sensor networrk, the difficulties were discovering calculations which can deal with huge datasets. Numerous calculations can deal with huge dataset with higher measurements.

The research saw support wireless sensor calculation with a massive precision of 99.21% and the least execution time which is exceptionally high accuracy as it works on multiple classes and does not have any classification of attacks in real-time, whereas it classifies the 'Worms' in a wireless sensor network.

Moreover, this research portrays the procedure of interruption identification and classification with the assistance of a few techniques which are somewhat sure in deciding the classification procedure and characterizing them so as to accomplish higher precision utilizing logistic processing with the clusters situated in WSN network with low execution time. This WSN technique classifies the real-time attacks in a wireless sensor based network.

Anyway in the research, the random clusters are processed for intrusions and accomplished a level of 99.21% so as to detect and classify the attack on a sensor based system with an execution time of practically low. In this research, we accommodate to detect and classify the intrusions and real-time attack types with higher accuracy in detailed comparison with other work and contribution of detecting and classifying the attacks in a sensor based network.

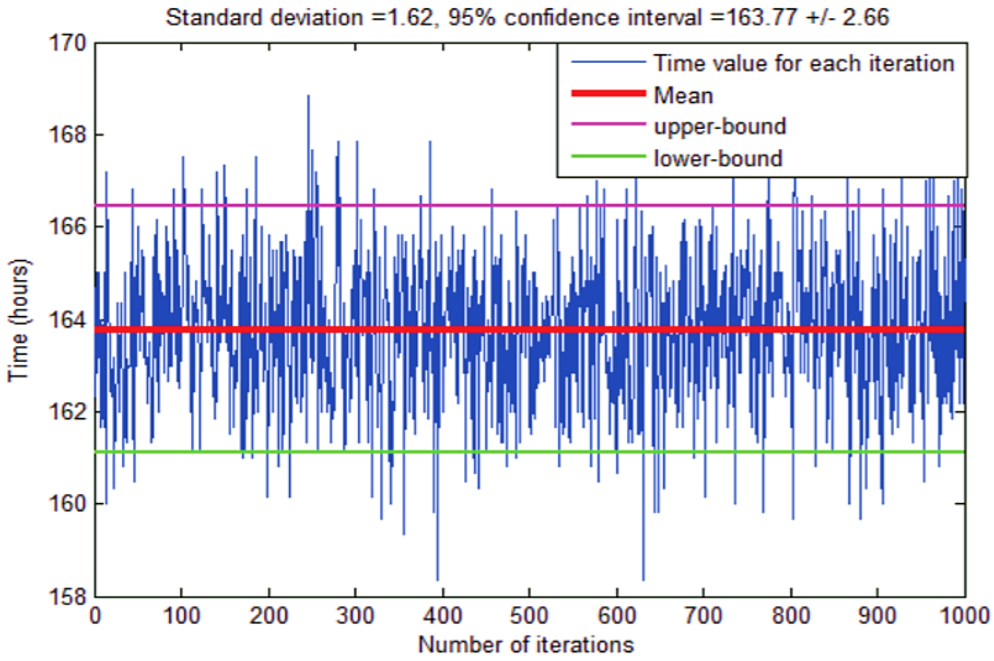


Figure 4.1: Conveyance plot of the recreation time in the simulation for intrusion detection.

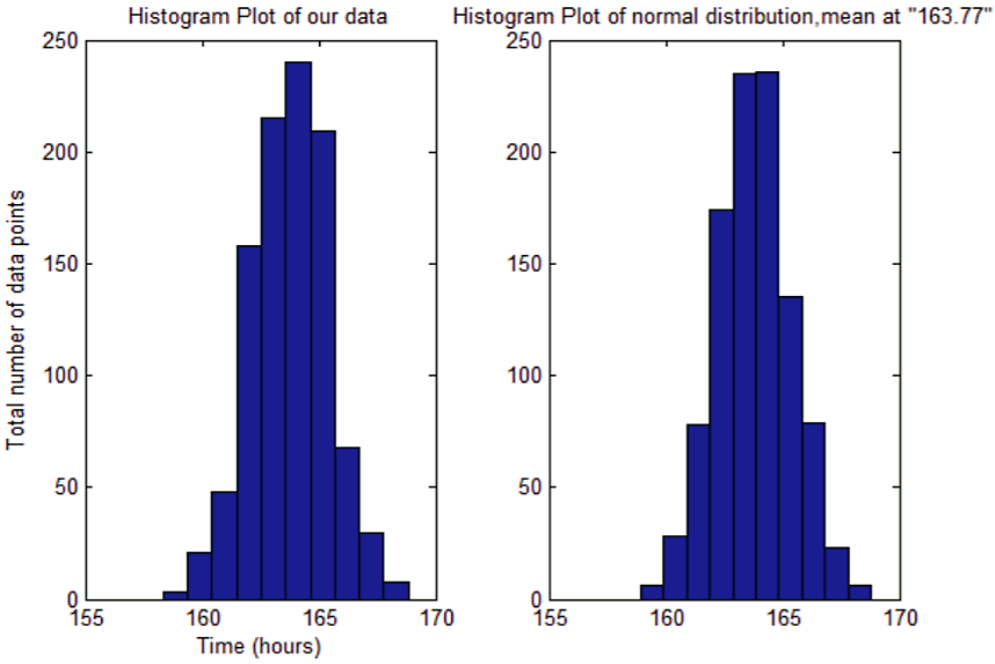


Figure 4.2: Histogram plot of the reproduction time contrasted with the ordinary conveyance for data points in the WSN.

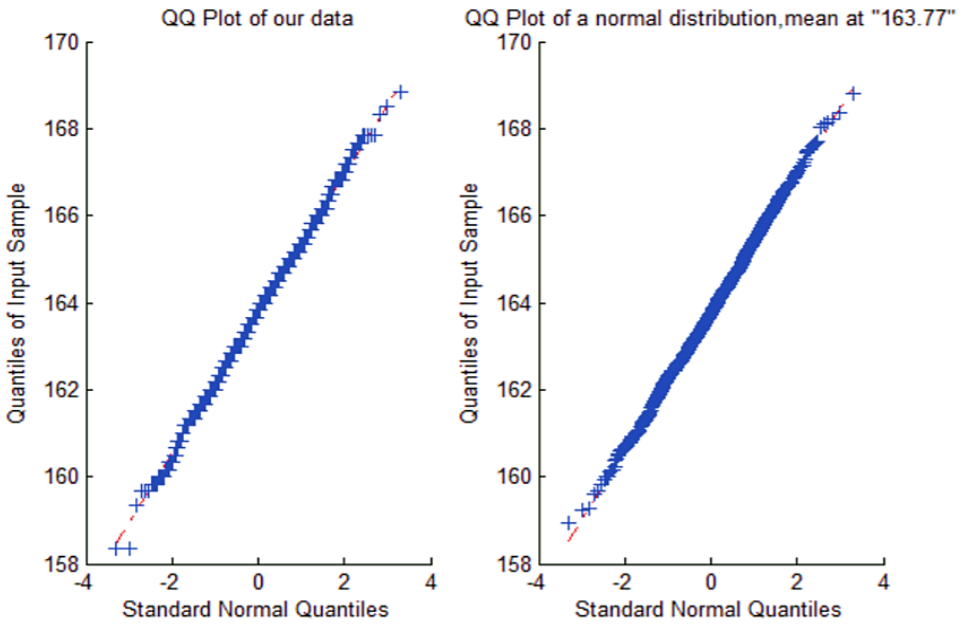


Figure 4.3: Standard normal quantiles represented in the simulation with time in contrast with the normal distribution.

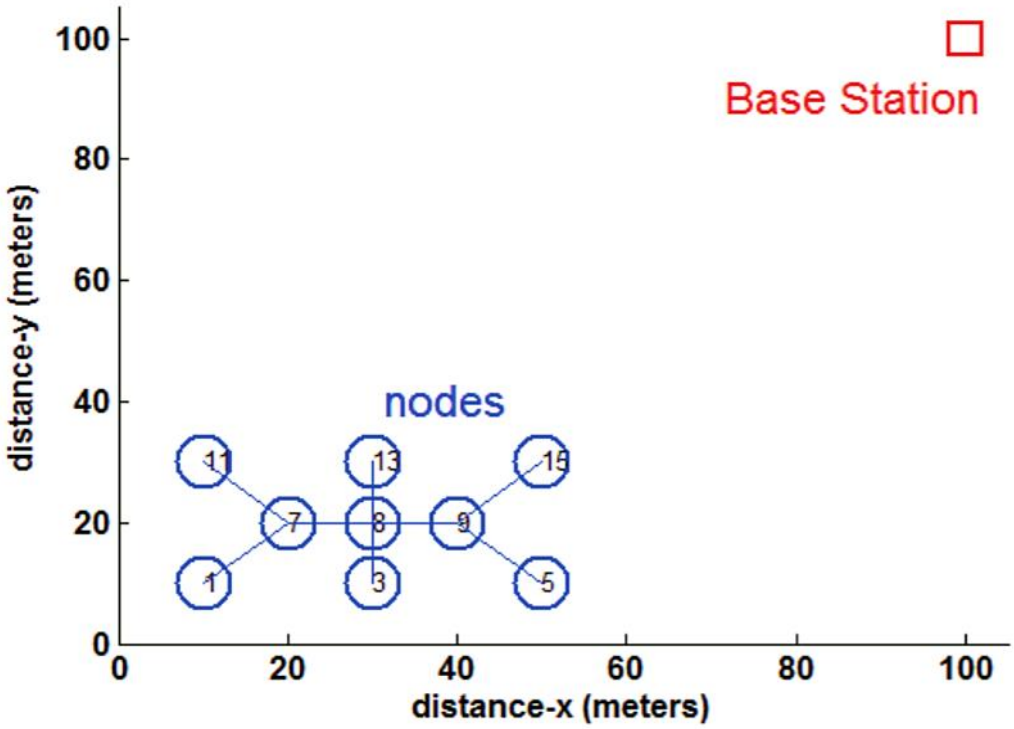


Figure 4.4: Plot of directions of the number of nodes with distance and the BS all through the reproductions of nodes in WSN.

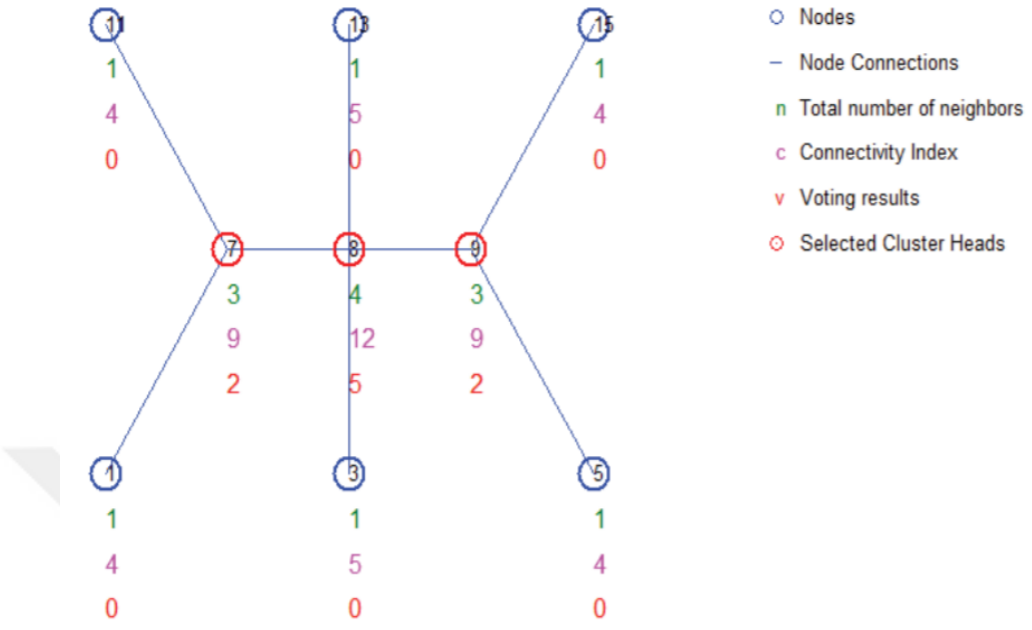


Figure 4.5: The nodes, nodes connection, total number of neighbors, connectivity index, voting results and cluster head selection of a 9-nodes in WSN.

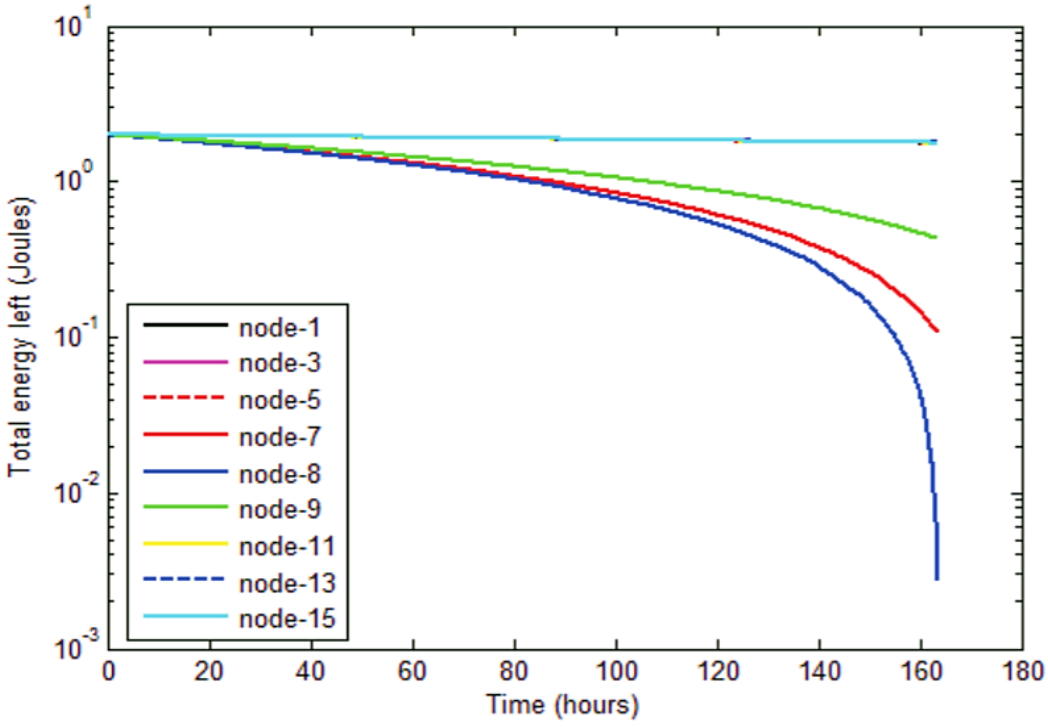


Figure 4.6: The detection of total energy in the WSN nodes representing attacks in terms of time.

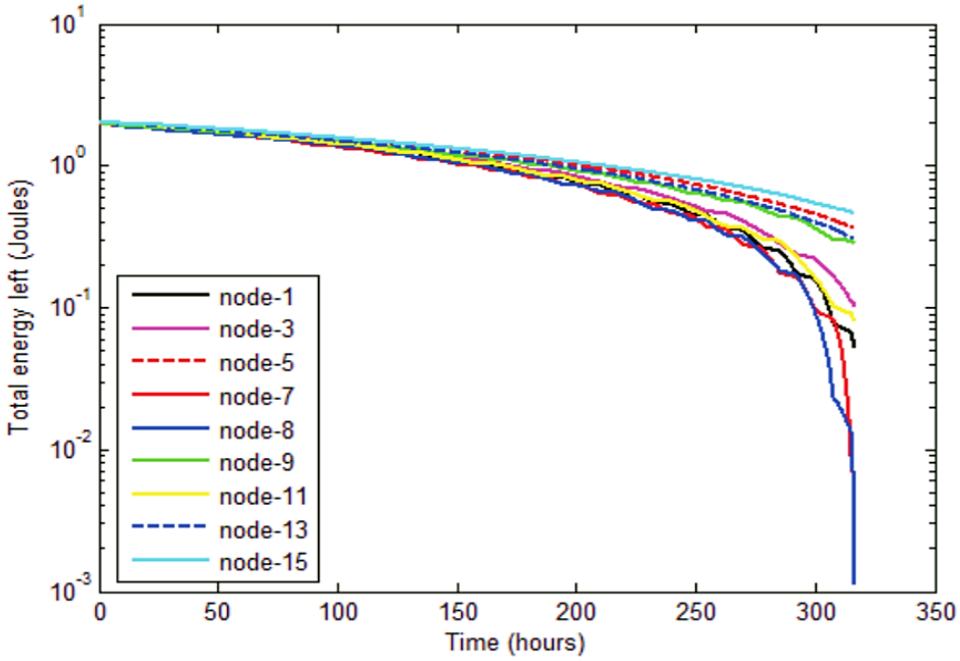


Figure 4.7: Predicted energy utilization diagram of node energy and cluster availability bunching calculation for 1-hop network case in WSN.

The theoretically good false positive rate of 0.0015 in practice would still mean that more than 1 in 1000 normal data points would be falsely labeled as an attack. If a data point is generated every 100 milliseconds, that would give us a false alarm every 100 seconds, and this is not practical. One data point by itself, however, could hardly be considered an attack, where a continuous stream of packets is usually sent. To that regard, this WSN approach could be implemented, where an alarm only goes off if there is a majority of attack data points within a window of some specified size.

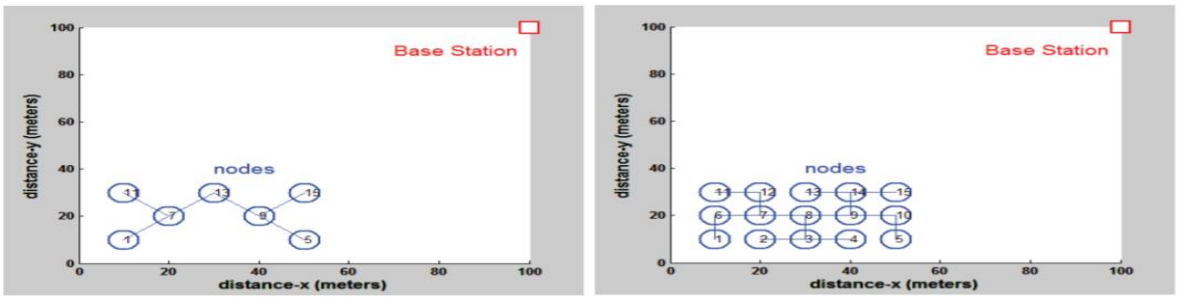


Figure 4.8: Diverse organization geographies of nodes in wireless sensor network with 7 and 15 nodes in base station.

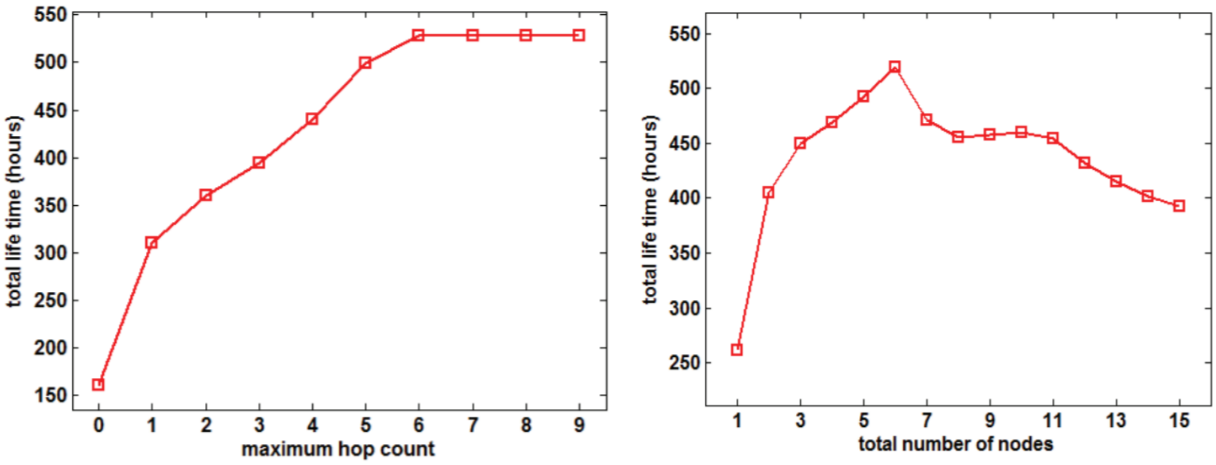


Figure 4.9: Maximum hops and total number of nodes in a network vs. total life-time of the network.

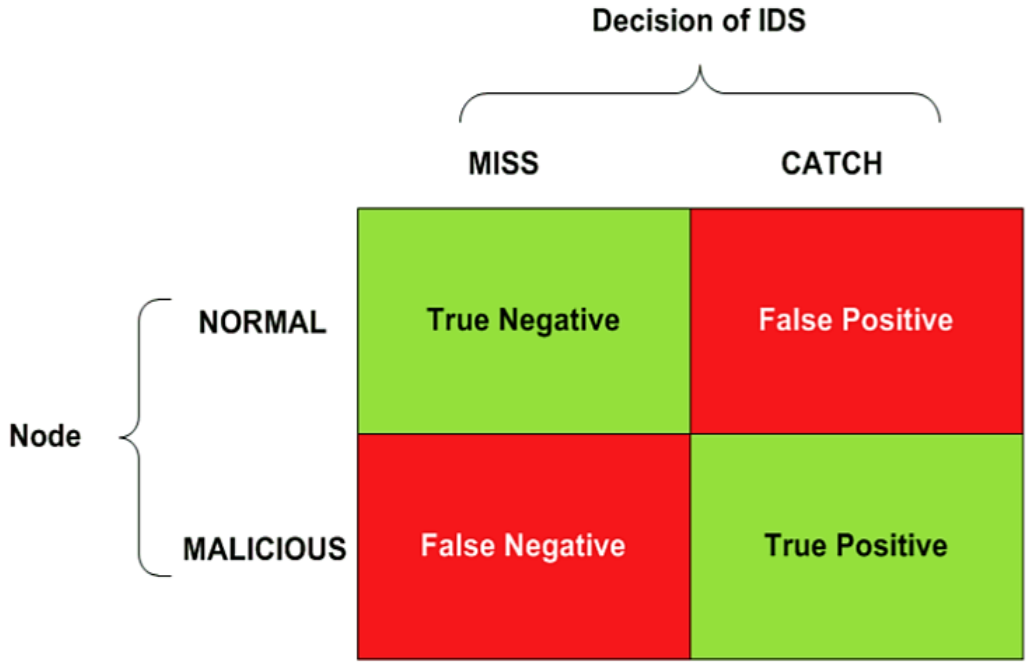


Figure 4.10: All conceivable recognition after-effects and prediction of an intrusion detection system.

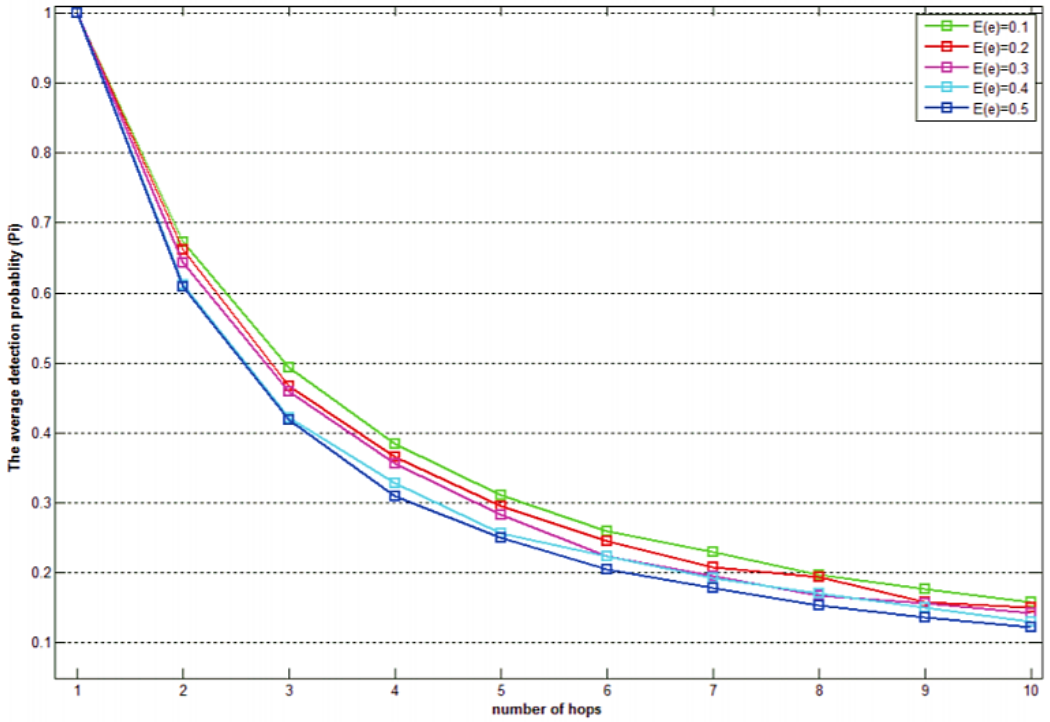


Figure 4.11: Impact of cluster size on the location likelihood of the intrusion detection system for different packet misfortune rates while the rest rate is 60% in WSN.

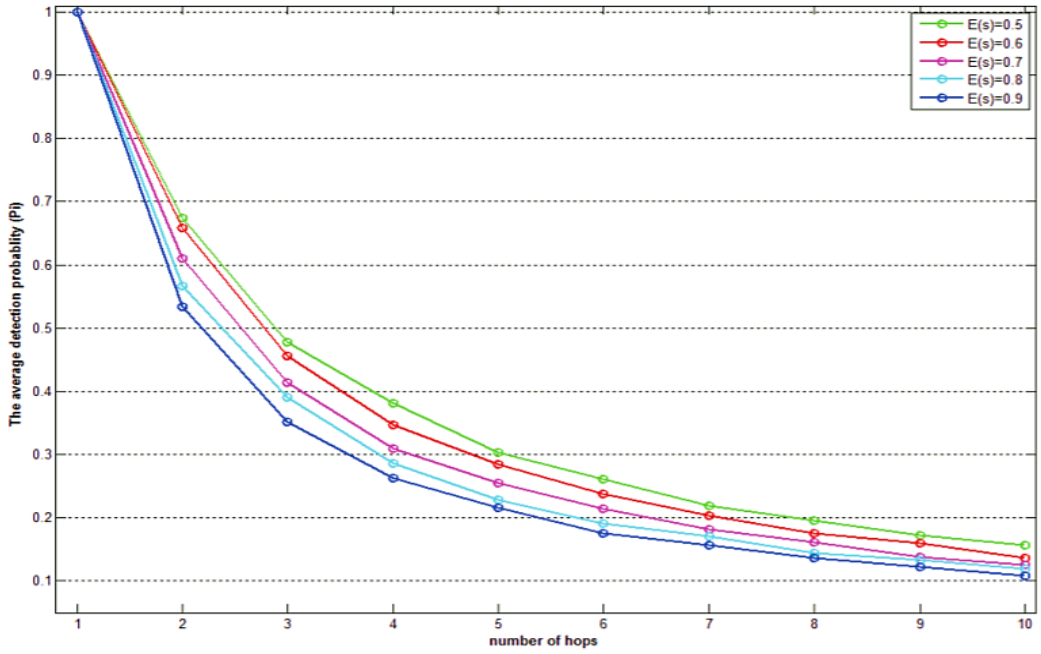


Figure 4.12: Impact of cluster size on the location likelihood of the intrusion detection system for different rest rates while the packet misfortune rate is 30% in WSN.

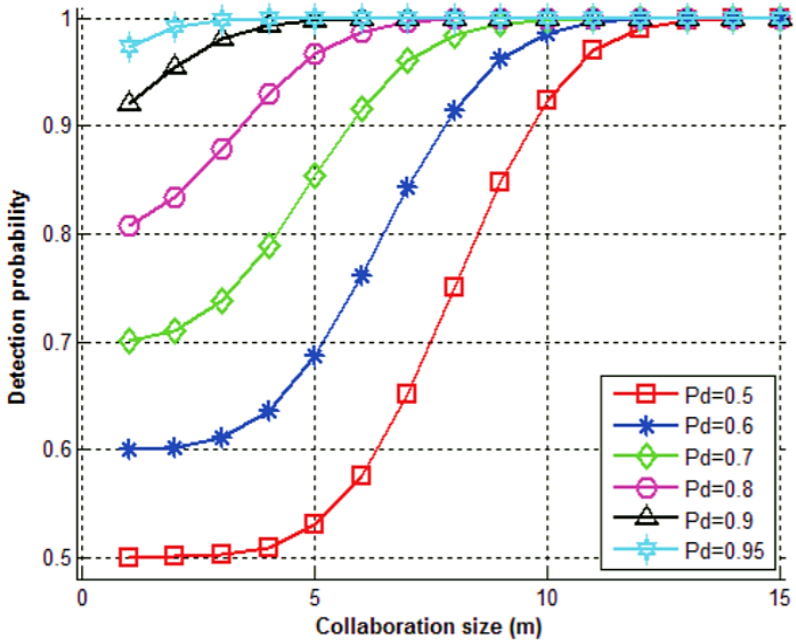


Figure 4.13: Compiling the WSN network with the race detector enabled for detection probability (P_D) vs. collaboration size(m) for malicious attacks.

5. DISCUSSION

For experiments performed in this research, all intrusions in the WSN have been detection within clusters and nodes. Nodes will be encoded as dummy variables, which creates a new column for each unique entry and deletes the original column from the dataset afterwards. Booleans are encoded to numeric values, 0 for false, 1 for true. The first run of experiment 1 was stopped after 12 hours, after hanging for more than 7 hours at the dummy variable encoding of the Answers columns from Tuesday-WorkingHours-1/DNS labeled.csv. In order to reduce the amount of data that has to be processed, the sample size was dropped from 1.0 to 0.5. This means only 50% of the available data will be used. However, the next run also did not complete in a reasonable amount of time, and was stopped after 8 hours, hanging at the same place like before. After starting a third run, while sampling only 20% of the available data, the experiment finished after over 29 hours. Experiment one was conducted with an early version of clusters, and was not repeated. It is shown here for completeness and because following experiments were adapted after observations from this one. Entries marked with"-" contained only one label type after sampling, thus no prediction was possible and the Keras library aborted.

Table 5.1: The classification of different attacks with respective count.

Classification	Count
Generic Protocol Command Decode	6449
Attempted Information Leak	3
Not Suspicious Traffic	80
Potential Corporate Privacy Violation	12
Potentially Bad Traffic	25
Misc Attack	18

Table 5.2: Classification results experiment of intrusions in WSN network.

File	Num Records	Size	Labels	Exec Time	Validation Score
Connection labeled.csv	284166	51 MB	2119	50s	0.9919065381096488
DNS labeled.csv	700447	144 MB	33	3h 27m	0.9999428946692174
Ethernet labeled.csv	11551954	880 MB	3556	6m 49s	0.9996693201846267
Flow labeled.csv	647294	112 MB	4852	4m 2s	0.9904835470415573
HTTP labeled.csv	45852	14 MB	5609	21s	0.9637554585152839
IPv4 labeled.csv	11469736	1.0 GB	3555	-	-
NTP labeled.csv	15507	2.1 MB	18	2s	0.9987113402061856
TCP labeled.csv	10710230	1.5 GB	3504	-	-
TransportFlow labeled.csv	91861	5.8 MB	11	8s	0.9999059354717336
UDP labeled.csv	787015	44 MB	51	27s	0.9999491753703845

Using the dummy variable method for encoding strings delivered a good detection accuracy of around 99%, however it is impractical for operating on large datasets, due to the very high time for encoding of features. Because each unique string will lead to the creation of a new column, the dataset grows in width tremendously. For the evaluation of CIC IDS 2017 this strategy of encoding strings might have been possible, but regarding the size of modern datasets, another approach is needed. Dropping columns with a high amount of unique entries (such as DNS Answers, IP addresses) was not considered, because this first series of experiments aims to create a baseline of detection accuracy with the complete data that cluster delivers. Entries with"-" have failed when entering the training phase, because after sampling and encoding, the subset did only contain normal labels and none of the malicious types. For experiments, encoding will be slightly modified in order to reduce the time needed for the encoding stage. All numeric values will be clustered like before. Booleans are encoded to numeric values, 0 for false, 1 for true. Instead of choosing dummy variables, strings will be encoded as indices this time. This keeps the original column in the dataset and creates a new one containing a unique integer for each unique string. Originally this was used to encode only the result column, the following experiment evaluates

the use of this string encoding method for the whole CIC IDS 2017 dataset. Experiments have a total processing time of over 8 hours. Processing time for encoding is now much better. Detection accuracy on HTTP traffic declined, most notably on the Working-Hours dump. Others slightly increased, for example the Working-Hours/Connection labeled.csv detection accuracy increased from 99.19% to 99.21%. Accuracy for detecting flows stayed almost the same. Classification results are comparable to the ones from the previous run and in this case do not seem to be influenced by the higher amount of unique labels. This is quite interesting, because classifying specific attacks could generate more specific alerts, which would provide more value to the analyst.

Results show that accurate classification of malicious or unwanted behavior is possible within wireless sensor network, with a feature set consisting mainly of collected and only a handful of generated features. Furthermore, classification achieved better results when applied to specific protocols, compared to classification of summary structures such as flows and connections. The time needed to encode the data to a numeric feature vector, heavily depends on the strategy used for encoding alphanumeric values (strings). Encoding strings as indices performed better in the encoding phase, but required more time for training the deep neural network, compared to encoding them as dummy variables. Using attack descriptions instead of attack classes as labels delivered comparable results and should be subject to further experiments. Due to the fact that there might be several alerts for the same audit records, a strategy for handling this must be chosen. Collecting all classifications and merging them into a final label has shown great results, and should be further investigated in combination with using specific attack descriptions instead of attack classes.

6. CONCLUSION

The implemented solution is referred to as secure, because parsing the network traffic is implemented in a memory-safe language. It is scalable because WSN can increase throughput with more processing cores and can be used in a distributed monitoring setup. Python programming can take advantage of multi-core architectures as well as GPUs, in order to accelerate the computations for the Deep Neural Network. Results from the series of experiments show that classification of labeled audit records is possible with a high accuracy, while only using a reduced feature subset compared to previous research in the area. WSN is meant to be a useful technique in the process of feature collection, selection and generation for research on anomaly-based intrusion detection strategies. By providing the generated data in separate files, WSN enables other applications that consume its output to implement a concurrent design as well. Choosing a platform neutral format for serialized structured data, greatly in- creases accessibility of the generated audit records for experiments across all major programming languages. The landscape of interesting libraries and techniques is constantly evolving, and so are the possibilities and options for experiments. Implementing the framework in python helps in reducing syntactic complexity, increases performance compared to implementations in scripting languages, and provides memory safety. The fact that memory safety is a big problem for today's intrusion detection frameworks has been recognized by this advance research, which have begun to port parts of their protocol decoding logic to achieve an accuracy of 99.21% for detecting all the intrusions within the WSN that use a parser generation framework to accomplish safe protocol parsing. Although, these countermeasures are a good step forward, they only reduce the attack surface partially, since the remaining parts of the frameworks are still written in a language that is affected by this category of vulnerabilities. A key takeaway from the series of experiments is the absence of many computationally expensive extracted features, while still achieving very high classification results. Questioning the need for those features comes with the possibility to improve the performance of network monitoring systems. The fewer data is needed for classification of malicious behavior, the more efficient implementations for feature collection and extraction can become. By publishing the developed tooling, the experiments are made reproducible for other researchers, which will hopefully lead to more academic

publications and novel research in the area. However, the problem is not and will never be solved simply by deploying an intrusion detection system - there is always the need for an analyst that monitors the alerts and events and takes appropriate action. It is important to keep in mind, that network monitoring systems can be evaded as well and may even provide an increased attack surface to the system or network they were supposed to protect. To counter this development, frequent updates to defensive strategies and tools are necessary.

6.1 FUTURE RECOMMANDATION

Future development on WSN will target increasing the unit test coverage, as well as adding more benchmarks for performance critical operations. Additionally, the output of WSN network will be compared to those of other tools, to ensure no data is missed or misinterpreted. Clusters will be extended with further functionality in the future, for example by adding support for extracted features that have proven value in other academic publications. Furthermore, WSN should be used for experiments on other datasets, to explore its capabilities in delivering the necessary intelligence for accurate predictions on network data. Encoding of the feature vectors could also be implemented as part of the WSN framework, as this would tremendously speed up processing compared to conducting encoding in a scripting language. An interface for adding additional application layer encoders that require stream reassembly is also planned. The project will be evaluated for monitoring industrial control systems communication, as recent efforts have been made by contributors to packet to implement support for Modbus, Ethernet/IP (ENIP), and the Common Industrial Protocol (CIP). A DPI module could help identify app layer protocols, before discarding the packet payloads. Another interesting candidate for future integration, is the recently open sourced fingerprinting strategy for SSH handshakes (HASSH) by salesforce. An in-depth performance analysis of WSN will be carried out, as many parts of the framework can be further optimized. Robustness tests need to be performed and error handling improved to be more graceful and prevent the monitor from crashing in edge cases. Several exporters for the most common SIEM and data analytics solutions will be added, in order to visualize and organize data produced by WSN for human analysts. Protocol coverage will be increased, as there are still protocols offered by packet that are not yet integrated into Netcap, i.e: IPSec, MLDv2MulticastListenerReport, Geneve, VXLAN, USB and more. To satisfy the need for large scale monitoring, tooling needs to be created to manage a multi instance architecture and deliver

specific traffic to selected instances. The Canadian Institute for Cybersecurity meanwhile has published their newest dataset from 2018 (CSE-CIC-IDS2018). This dataset has an overall size of almost 500GB and will be evaluated with WSN in the future as well.

REFERENCES

[1] Bamakan, S.M.H.; Wang, H.; Yingjie, T.; Shi, Y. An effective intrusion detection framework based on MCLP/SVM optimized by timevarying chaos particle swarm optimization. Neurocomputing 2016, 199, 90–102.

[2] Ambusaidi, M.A.; He, X.J.; Nanda, P.; Tan, Z.Y. Building an intrusion detection system using a filter-based feature selection algorithm. IEEE Trans. Comput. 2016, 65, 2986–2998.

[3] Ghazy, R.A.; El-Rabaie, E.M.; Dessouky, M.I.; El-Fishawy, N.A.; Abd El-Samie, F.E. Efficient techniques for attack detection using different features selection algorithms and classifiers. Wirel. Pers. Commun. 2018, 100, 1689–1706.

[4] Aljawarneh, S.; Aldwairi, M.; Yassein, M.B. Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. J. Comput. Sci. 2018, 25, 152–160.

[5] Kang, S.H.; Kim, K.J. A feature selection approach to find optimal feature subsets for the network intrusion detection system. Cluster Comput. 2016, 19, 325–333.

[6] Salo, F.; Nassif, A.B.; Essex, A. Dimensionality reduction with IG-PCA and ensemble classifier for network intrusion detection. Comput. Netw. 2019, 148, 164–175.

[7] Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the 2019 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009; pp. 1–6.

[8] Beulah, J.R.; Punithavathani, D.S. A hybrid feature selection method for improved detection of wired/wireless network intrusions. Wirel. Pers. Commun. 2018, 98, 1853–1869.

[9] Bostani, H.; Sheikhan, M. Hybrid of binary gravitational search algorithm and mutual information for feature selection in intrusion detection systems. Soft Comput. 2017, 21,

2307–2324.

[10] Acharya, N.; Singh, S. An IWD-based feature selection method for intrusion detection system. Soft Comput. 2017, 22, 4407–4416.

[11] Alabdallah, A.; Awad, M. Using weighted support vector machine to address the imbalanced classes problem of intrusion detection system. KSII Trans. Internet Inf. Syst. 2018, 12, 5143–5158.

[12] Akashdeep, S.; Manzoor, I.; Kumar, N. A feature reduced intrusion detection system using ANN classifier. Expert Syst. Appl. 2017, 88, 249–257.

[13] Akyol, A.; Hacibeyoglu, M.; Karlik, B. Design of multilevel hybrid classifier with variant feature sets for intrusion detection system. IEICE Trans. Inf. Syst. 2016, ED99, 1810–1821.

[14] Bhattacharya, S.; Selvakumar, S. LAWRA: A layered wrapper feature selection approach for network attack detection. Secur. Commun. Netw. 2015, 8, 3459–3468.

[15] Panda, M.; Abraham, A.; Patra, M.R. Hybrid intelligent systems for detecting network intrusions. Secur. Commun. Netw. 2015, 8, 2741–2749.

[16] Ahmad, I.; Basheri, M.; Iqbal, M.J.; Rahim, A. Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection. IEEE Access 2018, 6, 33789–33795.

[17] Aburomman, A.A.; Reaz, M.B. A survey of intrusion detection systems based on ensemble and hybrid classifiers. Comput. Secur. 2017, 65, 135–152.

[18] Lilakiatsakun, W.; Somwang, P. Anomaly traffic detection based on PCA and SFAM. Int. Arab J. Inf. Technol. 2017, 12, 253–260.

[19] Zeeshan Ali Khan, Peter Herrmann, "Recent Advancements in Intrusion Detection Systems for the Internet of Things", Security and Communication Networks, vol. 2019, Article ID

4301409, 19 pages, 2019. https://doi.org/10.1155/2019/4301409.

[20] Li, L.J.; Yu, Y.; Bai, S.S.; Hou, Y.; Chen, X.Y. An effective two-step intrusion detection approach based on binary classification and kNN. IEEE Access 2018, 6, 12060–12073.

[21] Demir, N.; Dalkilic, G. Modified stacking ensemble approach to detect network intrusion. Turk. J. Electr. Eng. Comput. Sci. 2018, 26, 418–433.

[22] Kamarudin, M.H.; Maple, C.; Watson, T.; Safa, N.S. A LogitBoost-based algorithm for detecting known and unknown web attacks. IEEE Access 2017, 5, 26190–26200.

[23] Tian, Y.J.; Mirzabagheri, M.; Bamakan, S.M.H.; Wang, H.D.; Qu, Q. Ramp loss one-class support vector machine: A robust and effective approach to anomaly detection problems. Neurocomputing 2018, 310, 223–235.

[24] Kabir, E.; Hu, J.K.; Wang, H.; Zhuo, G.P. A novel statistical technique for intrusion detection systems. Future Gener. Comput. Syst. 2018, 79, 303–318.

[25] Ahmim, A.; Derdour, M.; Ferrag, M.A. An intrusion detection system based on combining probability predictions of a tree of classifiers. Int. J. Commun. Syst. 2018, 31, 1–14.

[26] Aburomman, A.A.; Reaz, M.B. A novel weighted support vector machines multiclass classifier based on differential evolution for intrusion detection systems. Inf. Sci. 2017, 414, 225–246.

[27] Yan, B.H.; Han, G.D. LA-GRU: Building combined intrusion detection model based on imbalanced learning and gated recurrent unit neural network. Secur. Commun. Netw. 2018, 1, 1–13.

[28] Idhammad, M.; Afdel, K.; Belouch, M. Semi-supervised machine learning approach for DDoS detection. Appl. Intell. 2018, 48, 3193–3208.

[29] Mohammadi, S.; Namadchian, A. A new deep learning approach for anomaly base IDS

using memetic classifier. Int. J. Comput. Commun. 2017, 12, 677–688.

[30] Imamverdiyev, Y.; Abdullayeva, F. Deep learning method for denial of service attack detection based on restricted boltzmann machine. Big Data-US 2018, 6, 159–169.

[31] Ma, T.; Wang, F.; Cheng, J.; Yu, Y.; Chen, X. A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks. Sensors (Basel) 2016, 16, 1701.

[32] Shamshirband, S.; Daghighi, B.; Anuar, N.B.; Kiah, M.L.M.; Patel, A.; Abraham, A. Co-FQL: Anomaly detection using cooperative fuzzy Q-learning in network. J. Intell. Fuzzy Syst. 2015, 28, 1345–1357

[33] Al-Qatf, M.; Yu, L.S.; Al-Habib, M.; Al-Sabahi, K. Deep learning approach combining sparse autoencoder with SVM for network intrusion detection. IEEE Access 2018, 6, 52843–52856.

[34] Hussain, J.; Lalmuanawma, S.; Chhakchhuak, L. A two-stage hybrid classification technique for network intrusion detection system. Int. J. Comput. Int. Syst. 2016, 9, 863–875.

[35] Li, L.J.; Yu, Y.; Bai, S.S.; Cheng, J.J.; Chen, X.Y. Towards effective network intrusion detection: A hybrid model integrating Gini index and GBDT with PSO. J. Sens. 2018, 6, 1–9.

[36] He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), PIEAS, Islamabad, Pakistan, 26–27 August 2016; pp. 770–778.

[37] Almomani, I.; Alromi, A. Integrating Software Engineering Processes in the Development of Efficient Intrusion Detection Systems in Wireless Sensor Networks. Sensors 2020, 20,

1375. https://doi.org/10.3390/s20051375.

[38] Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein.; et al. Imagenet large scale visual recognition challenge. Int. J. Comput. Vis. 2016, 115, 1–37.

[39] Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. Comput. Sci. 2016, 9, 1–14.

[40] A survey on key management mechanisms for distributed Wireless Sensor Networks - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Hierarchical-and-Distributed-WSNs_fig1_222021115 [accessed 21 Mar, 2021]

[41] He, K.; Zhang, X.; Ren, S.; Sun, J. Identity mappings in deep residual networks. In Proceedings of the 2017 European Conference on Computer Vision (ECCV), Amsterdam, The Netherlands, 11–14 October 2015; pp. 630–645.

[42] Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic Minority Over-Sampling Technique. J. Artif. Intell. Res. 2017, 16, 321–357.

[43] Wu, K.H.; Chen, Z.G.; Li, W. A novel intrusion detection model for a massive network using convolutional neural networks. IEEE Access 2018, 6, 50850–50859.

[44] Le, T.T.H.; Kim, Y.; Kim, H. Network intrusion detection based on novel feature selection model and various recurrent neural networks. Appl. Sci.-Basel 2019, 9, 1392.

[45] Panda, M.; Abraham, A.; Patra, M.R. Discriminative multinomial naive Bayes for network intrusion detection. In Proceedings of the 2017 Sixth International Conference on Information Assurance and Security, Atlanta, GA, USA, 23–25 August 2017; pp. 5–10.

[46] Zhang, W., Han, D., Li, KC. et al. Wireless sensor network intrusion detection system based

on MK-ELM. Soft Comput 24, 12361–12374 (2020). https://doi.org/10.1007/s00500-020-04678-1.

[47] Gogoi, P.; Bhuyan, M.H.; Bhattacharyya, D.; Kalita, J.K. Packet and flow based network intrusion dataset. Contemp. Comput. 2017, 306, 322–334.

[48] Yin, C.; Zhu, Y.; Fei, J.; He, X. A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access 2017, 5, 21954–21961.

[49] Singh, R.; Kumar, H.; Singla, R.K. An intrusion detection system using network traffic profiling and online sequential extreme learning machine. Expert Syst. Appl. 2015, 42, 8609–8624.

[50] Yang, Y.Q.; Zheng, K.F.; Wu, C.H.; Niu, X.X.; Yang, Y.X. Building an effective intrusion detection system using the modified density peak clustering algorithm and deep belief networks. Appl. Sci.-Basel 2019, 9, 238.

[51] Kayacik, H.G.; Zincir-Heywood, A.N.; Heywood, M.I. Ahierarchical SOM-based intrusion detection system. Eng. Appl. Artif. Intell. 2017, 20, 439–451.

[52] Tsang, C.H.; Kwong, S.; Wang, H. Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection. Pattern Recognit. 2017, 40, 2373–2391.

[53] Guangjie, H., Shen, W., Duong, T. Q., Guizani, M., and Hara, T. (2014), A proposed security scheme against Denial of Service attacks in cluster- based wireless sensor networks, Security Comm. Networks, 7, 2542– 2554, doi: 10.1002/sec.373.

[54] CIC IDS 2017. Available online: https://www.unb.ca/cic/datasets/ids-2017.html