Semi-Supervised Machine Learning Approach For DDOS Detection

A project Report submitted

in partial fulfillment for the award of the Degree of

Bachelor of Technology
in
Computer Science and Engineering by

A. YASHWANTH REDDY (U19CS067)

CH. BHARGAVA SIVA KUMAR REDDY (U19CS218)

D. DINESH GOUD (U19CS241)

R. MANI RAGHAVENDRA (U19CN330)

Under the guidance of

Mrs. R. jagadeeswari, Asst Prof/CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

BHARATH INSTITUTE OF HIGHER EDUCATION AND RESEARCH

(Deemed to be University Estd u/s 3 of UGC Act, 1956)

CHENNAI 600 073, TAMILNADU, INDIA

April, 2023

Bharath
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Declared as Deemed - to - be - University under section 3 of UGC Act 1956)
BHARATH INSTITUTE OF SCIENCE AND TECHNOLOGY
No.173, Agharam Road, Selaiyur, Chennai - 600 073.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

BONAFIDE CERTIFICATE

This is to Certify that this Project Report Titled "**Semi-Supervised Machine Learning Approach For DDOS Detection**" is the Bonafide Work of A. Yashwanth Reddy (U19CS067), Ch. Bhargava Siva Kumar Reddy (U19CS218), D. Dinesh Goud(U19CS241), R. Mani Raghavendra (U19CN330) of Final Year B.Tech. (CSE) who carried out the major project work under my supervision Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on basis of which a degree or awardconferred on an earlier occasion by any other candidate.

PROJECT GUIDE

Mrs. R. Jagadeeswari

Assistant Professor

Department of CSE

BIHER

HEAD OF THE DEPARTMENT

Dr. S. Maruthuperumal

Professor

Department of CSE

BIHER

Submitted for the Project Viva-Voce held on ……

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We declare that this project report titled **Semi-Supervised Machine Learning Approach For DDOS Detection** submitted in partial fulfillment of the degree of **B. Tech in (Computer Science and Engineering)** is a record of original work carried out by us under the supervision of **Mrs. R. Jagadeeswari,** and has not formed the basis for the award of any other degree or diploma, in this or any other Institution or University. In keeping with the ethical practice in reporting scientific information, due acknowledgementshave been made wherever the findings of others have been cited.

A. Yashwanth Reddy
(U19CS067)

Ch. Bhargava Siva Kumar Reddy
(U19CS218)

D. Dinesh Goud
(U19CS241)

R. Mani Raghavendra
(U19CN330)

Chennai
Date

ACKNOWLEDGMENTS

First, we wish to thank the almighty who gave us good health and success throughout our project work.

We express our deepest gratitude to our beloved President **Dr. J. Sundeep Aanand,** andManaging Director **Dr. E. Swetha Sundeep Aanand** for providing us the necessary facilities for the completion of our project.

We take great pleasure in expressing sincere thanks to Vice Chancellor **Dr. K. Vijaya Baskar Raju**, Pro Vice Chancellor (Academic) **Dr. M. Sundararajan**, Registrar **Dr. S. Bhuminathan** and Additional Registrar **Dr. R. Hari Prakash** for backing us in this project. We thank our Dean Engineering **Dr. J. Hameed Hussain** for providing sufficient facilities for the completion of this project.

We express our immense gratitude to our Academic Coordinator **Mr. G. Krishna Chaitanya** for his eternal support in completing this project.

We thank our Dean, School of Computing **Dr. S. Neduncheliyan** for his encouragementand the valuable guidance.

We record indebtedness to our Head, Department of Computer Science and Engineering **Dr. S. MaruthuPerumal** for his immense care and encouragement towards us throughout the course of this project.

We also take this opportunity to express a deep sense of gratitude to our Supervisor **Mrs. R. Jagadeeswari** and our Project Co-Ordinator **Dr. L. Nalini Joseph** for their cordial support, valuable information and guidance, They helped us in completing this project through various stages.

We thank our department faculty, supporting staff and friends for their help andguidance to complete this project.

A. YASHWANTH REDDY **(U19CS067)**

CH. BHARGAVA SIVA KUMAR REDDY **(U19CS218)**

D. DINESH GOUD **(U19CS241)**

R. MANI RAGHVENDRA **(U19CN330)**

ABSTRACT

The fast propagation of computer networks has changed the viewpoint of network security.An easy accessibility conditions cause computer network as susceptible against several threats from hackers. Threats to networks are numerous and potentially devastating. Up to the moment, researchers have developed Intrusion Detection Systems (IDS) capable of detecting attacks in several available environments. A boundlessness of methods for misuse detection as well as anomaly detection has been applied. Many of the technologies proposed are complementary to each other, since for different kind of environments some approaches perform better than others. This project presents a new intrusion detection system that is then used to survey and classify them. The taxonomy consists of the detection principle, and second of certain operational aspects of the intrusion detection system. In our project we have used algorithms like Naïve Bayes (NB) as existing system and Random Forest (RF) as proposed system. All are measured in terms of accuracy. From the results its proved that proposed Random Forest (RF) works better than existing Naïve Bayes (NB).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER-1

1. INTRODUCTION

1.1 Sensor network:

A network comprised of interconnected sensor nodes exchanging sensed data by wired or wireless communication.

1.1.1 Sensor node:

A device consisting of sensor(s) and optional actuator(s) with capabilities of sensed data processing and networking. Sensor node consists of a great number of nodes of the same type (sensor nodes), which are spatially distributed and cooperate with each other. Each such node has a sensing element (sensor), a microprocessor (microcontroller), which process sensor signals, a transceiver and an energy source. Distributed over the object, sensor nodes with the necessary sensors make it possible to gather information about the object and control processes which take place on this object.

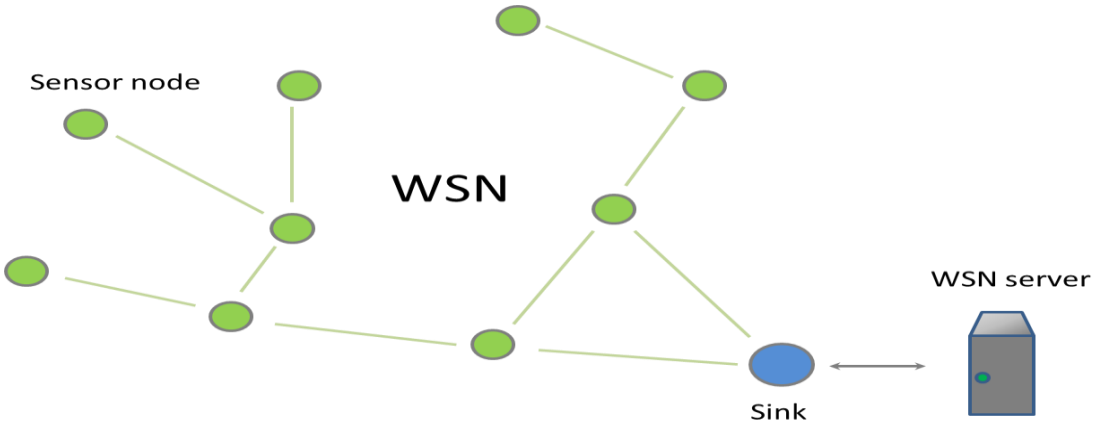


Fig 1.1.1.1 Overview of the network architecture

The above figure represents an example of a WSN. Here we can see a WSN which consists of twelve sensor nodes and a *network sink*, which also functions as a *gate*. Each sensor node is a device which has a transceiver, a microcontroller, and a sensitive element. Usually sensor node is an autonomous device. Each sensor node in WSN measures some physical conditions, such as temperature, humidity, pressure, vibration, and converts them into digital data. Sensor node can also process and store measured data before transmission. Network sink is a kind of a sensor node which aggregates useful data from other sensor nodes. As a rule, network sink has a stationary power source and is connected to a *server* which is processing data received from WSN. Such connection is implemented directly, if server and WSN are placed on the same object. If it is necessary to provide a remote access to WSN,

network sink also functions as a gate, and it is possible to interact with WSN through global network such as the Internet.

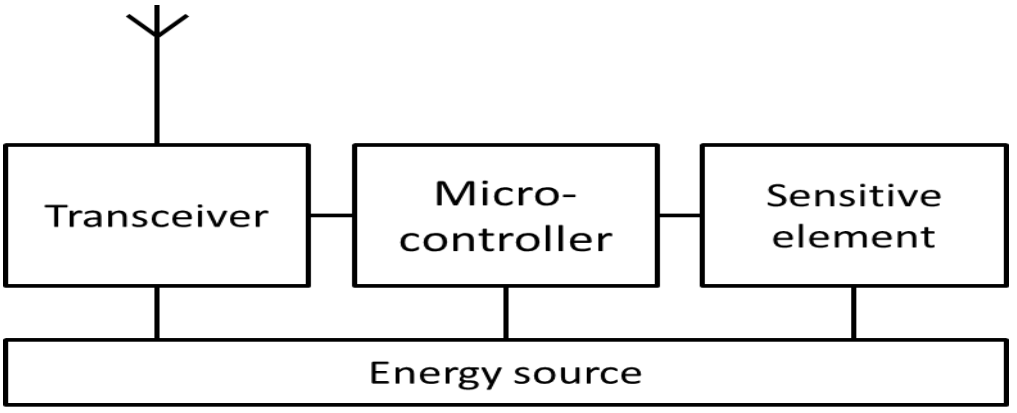


Fig 1.1.1.2 Sensor node inner structure

Some Sample Applications

• Habitat and Ecosystem Monitoring
• Seismic Monitoring
• Civil Structural Health Monitoring
• Monitoring Groundwater Contamination
• Rapid Emergency Response
• Industrial Process Monitoring
• Perimeter Security and Surveillance
• Automated Building Climate Control

1.2 WSN Structure

WSN sink

Sensor nodes are the basis of a WSN. They collect and exchange data necessary for WSN functionality. Data collected by sensor nodes are the raw information and require processing. Depending on application, such data can be averaged statistical information or detailed measurements of parameters which define the condition of some object. A separate group of WSN applications is detecting and tracking of targets, for examples, vehicles, animals etc. Each of these cases requiresprocessing of data provided by WSN. Usually it is impossible to perform this processing on sensor nodes themselves, by reason of energy saving and low computing power of sensor nodes. That is why in WSNs the final part of data processing is usually made beyond sensor nodes, on WSN

server. WSN server is connected to only one sensor node which is called *sink* or *base station*. Sink is a collecting point of all data in the WSN and interact both with the sensor nodes and the WSN server.

WSNS With The Cluster Structure

Since energy content of sensor nodes is limited and non-renewable, it is important to use it in the most economical way. The figure illustrates the data streaming from sensor nodes to sink. Sink collects data from all the sensor nodes periodically. On the figure, every arrow between sensor nodes shows a transfer of a portion of measurements for a single period of data collection.

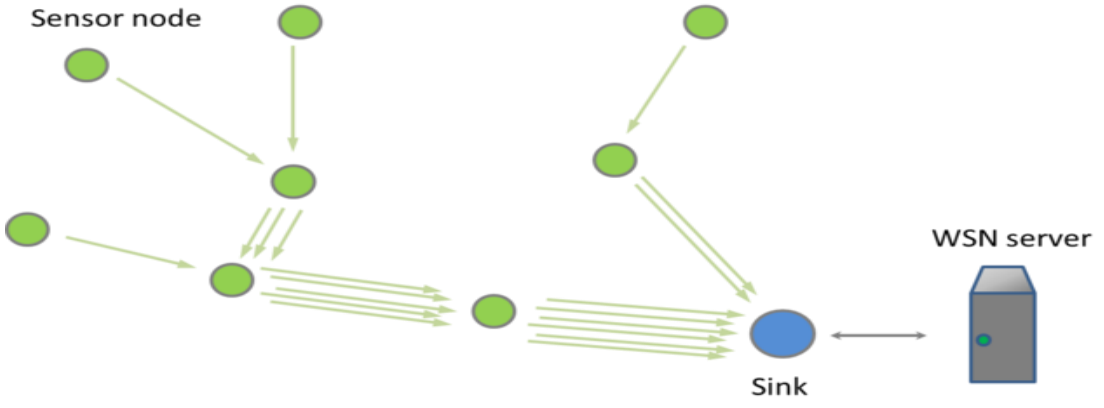


Fig 1.2.1 Data streaming from sensor nodes

Data is collected from all the sensor nodes; in result, the sensor nodes located closer to sink have to receive and transmit not only their own measurements, but also measurements from other sensor nodes which are further from sink. So, transceivers of the nearest sensor nodes retransmit much more information, and hence they consume more energy than remote sensor nodes. And since sensor nodes are usually all of the same type and have equal energy content, it leads to the fact that the nearest sensor nodes fail much earlier than remote ones, and so the former disrupt the work of the rest of WSN.

So, if WSN application provides periodical data collecting (and it happens in the most cases), it turns out that time of autonomous operating of sensor nodes which are the nearest to sink is much reduced because of more frequent retransmitting. In the long run, traffic from all the sensor nodes is going through one sensor node that is nearest to sink. And the more sensor nodes are in a WSN, the higher is this traffic. From the point of view of energy saving, big WSNs with only one sink cannot consume resources effectively.

To solve this problem it is necessary to divide the WSN into clusters. Each cluster has its own sink and, in fact, is a separate, but smaller, WSN. And each sink communicates with the server directly. The figure represents a network with two sinks. On the figure the arrows also used to represent the amount of transmitted data. As we can see, the number of retransmissions is significantly decreased, and it reduces the load on the nearest to sinks sensor nodes.

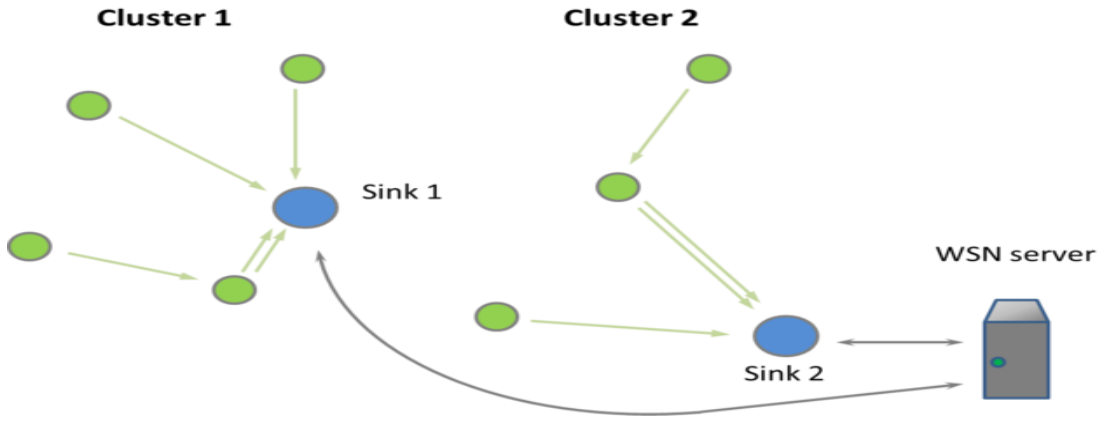


Fig 1.2.2 Multiple-sink WSN

Multiple-sink WSN is not a random division of one WSN into parts. In the most cases such division is made automatically when WSN is deployed and used. Sensor nodes automatically choose the sink to which they send data. This choice is made according to the algorithm of WSN protocol. Depending on requirements of the application, different criteria may be used, for example, the minimum time for data delivery, the minimum number of retransmissions, achieving the optimal traffic distribution in WSN and others.

1.3 WSN Gate

WSN organization schemes considered above suppose placement of all WSN elements in the same location. In practice, there is often necessary to have a remote access to WSN data. For example, WSN can be deployed in woodland in suburbs, and collecting and processing of WSN data have to be done in the office in a city. To organize data transmission from WSN to a remote server one uses specialized gates which receive sensor network data from sink and retransmit them using other (i. e. non-WSN) communication standard, wired or wireless. The figure represents such a network, which transmit collected data to server through the Internet using a gate.

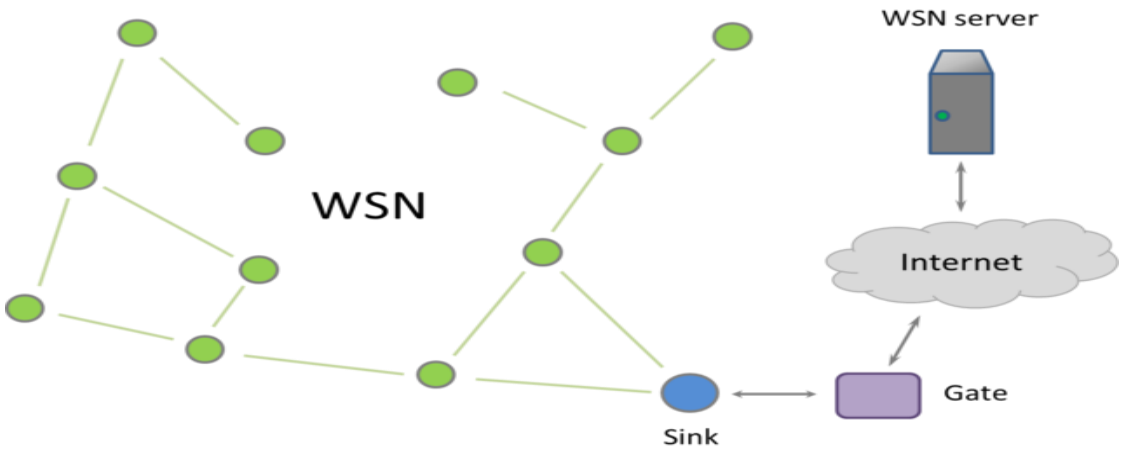


Fig 1.3.1 WSN server is connected via the Internet

Gates also provide the possibility to organize service provision. Nowadays, when access to the Internet via cellular, cable and satellite networks is available almost in any place in the world, connection of WSNs to the Internet in most cases is easy to implement. The figure represents the scheme of possible interaction between a user and a WSN.

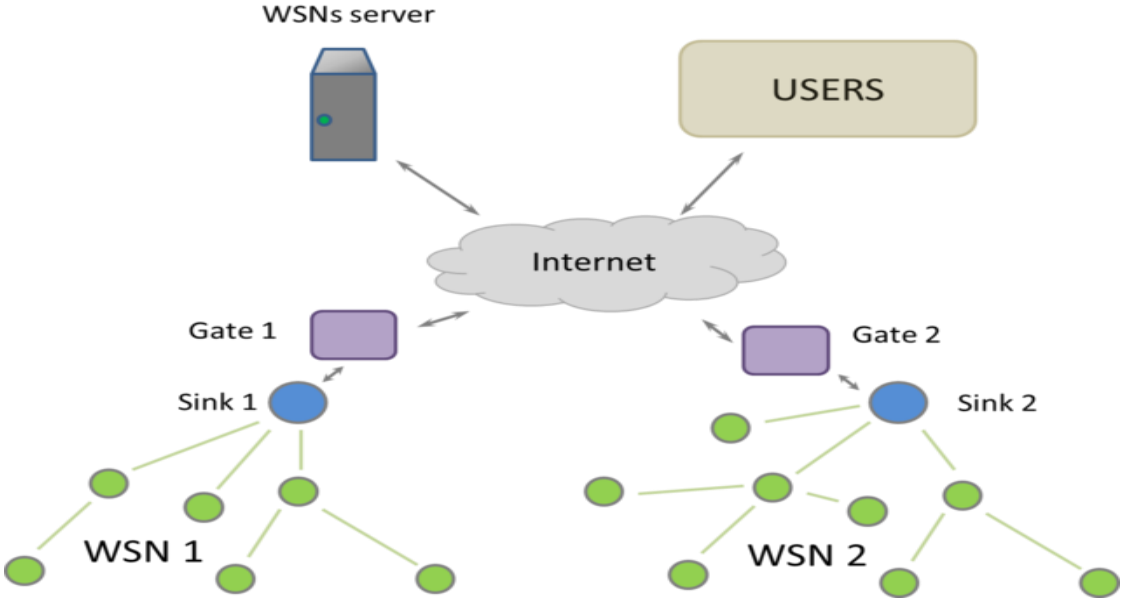


Fig 1.3.2 Scheme of provision of WSN services

1.4 Network Topology

Previously we have described traditional applications of WSN for data collecting and processing. Such applications have a special feature: they have one data collecting point, namely sink. But there are also applications where sensor nodes have not only to send information to sink, but to exchange data between themselves. That is why there are different schemes of organization of

interaction between sensor nodes within WSN. These schemes are called network topologies. The main types of network topologies for WSNs are: *star*, *tree* and *mesh*. Different WSN standards support different types of network topologies.

Star

The star topology is widely used in computer networks, so when WSN appeared, it started being used also for organization of interaction between sensor nodes. The main characteristic of the star topology is connecting of all the sensor nodes to sink directly. The figure schematically represents this topology. In such cases sensor nodes are not connected between themselves, and all interactions between sensor nodes are taking place only via sink. Disadvantage of this topology is limited number of sensor nodes in such WSN. This limitation appears because all the sensor nodes have to be placed in the vicinity of sink, in order to connect to it directly. Another limiting factor is sink's performance, i. e. the maximum number of supported connections.

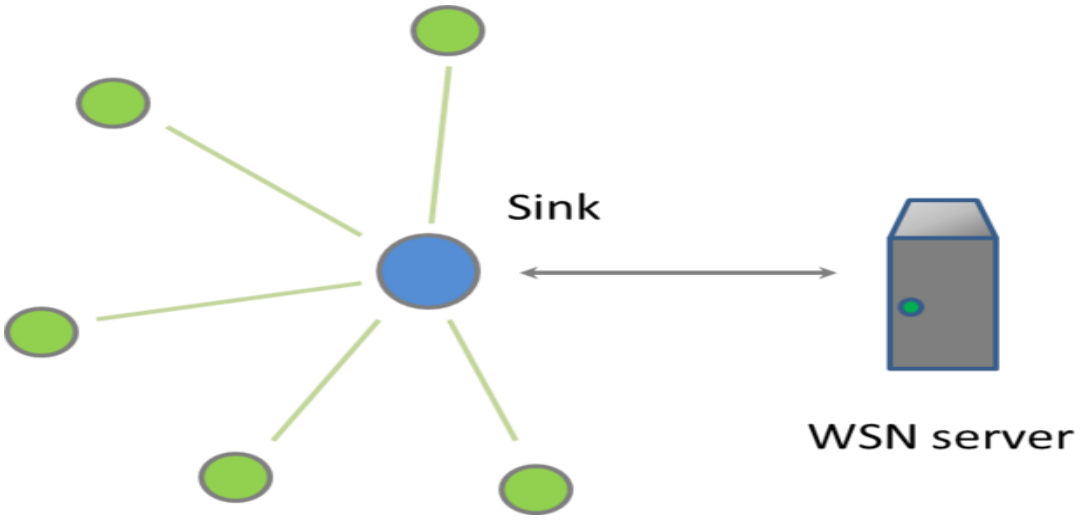


Fig 1.4.1 The star topology

Tree

The tree topology, in contradiction to the star topology, is much better suitable for WSN with the large number of sensor nodes. It has a hierarchical structure, as it is illustrated on The figure .Sensor nodes which are the nearest to sink interact with the sink directly. And more remote sensor nodes interact with the nearest ones according to the rules of the star topology. The tree topology also does not provide direct data exchange between all the sensor nodes. Data transmissions only from any sensor node to the sink and in the opposite direction are allowed. Also, in this topology data flow from the levels with greater numbers (i. e. "leaves") can be delivered only through the levels with smaller

numbers (i. e. “root” and “branches”). So, if on the first level there are only two sensor nodes, and the whole WSN consists of eleven sensor nodes, traffic will be delivered through these two sensor nodes much longer, because of data retransmission from nine sensor nodes on lower levels. Such network can fail quickly, because of energy consuming by the nearest to sink sensor nodes.

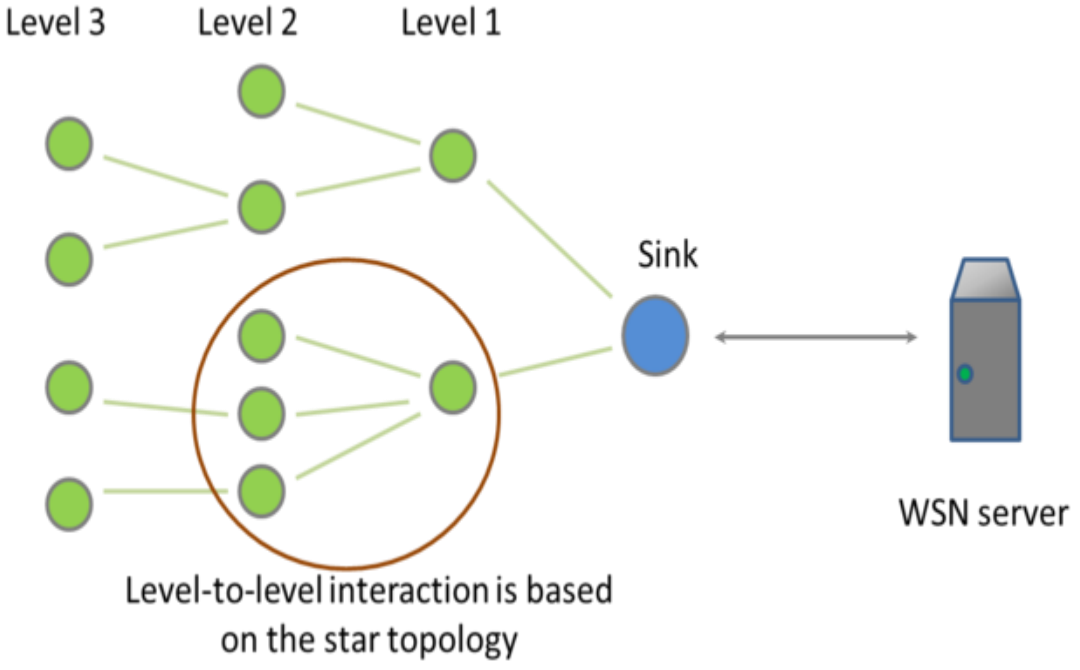


Fig 1.4.2 The tree topology

Mesh

The mesh topology is the most difficult one for implementation, but it provides much more opportunities for data exchange between sensor nodes. In WSN with the mesh topology interaction between sensor nodes is taking place according to the principle “with every nearest one”, as shown on The figure. It means that every sensor node cooperates with other sensor nodes, which are in its transceiver’s proximity. In such WSN data exchange between sensor nodes goes through the shortest ways and with the smallest number of retransmissions, what has a positive effect on the energy consumption of the sensor nodes.

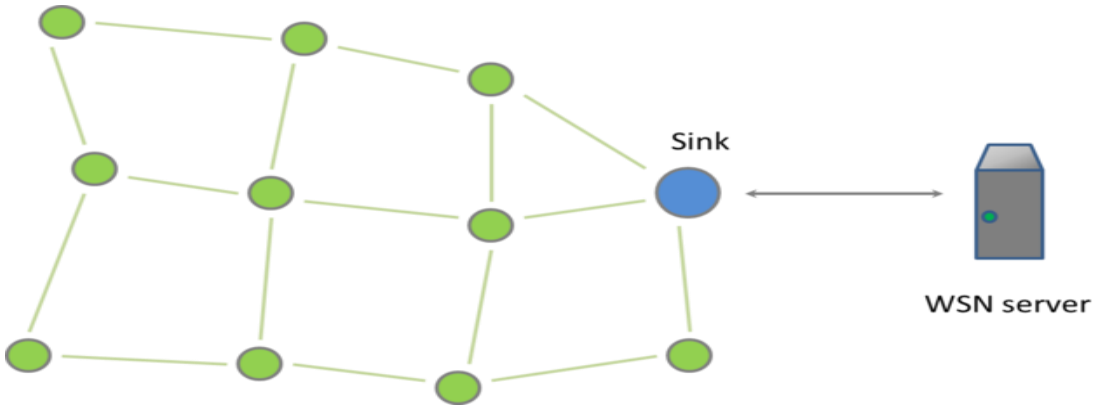


Fig 1.4.3 The mesh topology

1.5 General Design Issues

One of the most important tasks which have to be solved when working out a WSN for a specific application is the choice of the hardware platform which will serve as a basis for creating a sensor node. There are a lot of sensor nodes implementations from different vendors, but all the platforms have common elements. Choosing one or other existing platform or development of a new one from scratch have to be made in order to meet WSN functional requirements. Any hardware platform provides its own set of sensor node parameters. Variety of available platforms is caused by a wide range of WSN applications, and each existing platform has its own features according to the set of the tasks it is meant for. Also we have to understand that the current level of technology makes it necessary for the researchers to constantly find a balance between such parameters as size, productivity, battery lifetime, communication range, coverage, reliability, functionality, cost etc. The figure illustrates the correlation between the primary sensor node parameters. The arrows link directly related parameters, so improving a parameter in one end of the arrow will lead to worsening of the parameter in the other end of the arrow. For example, refinement of functionality (such as increasing the number of controlled parameters , improving the microcontroller performance) will inevitably cause an increase in cost, a decrease in battery lifetime and/or an increase in the size of a sensor node. So, the task of developing new hardware/software platforms which would support new technologies, expand the application scope, facilitate the deployment of WSNs is still relevant.

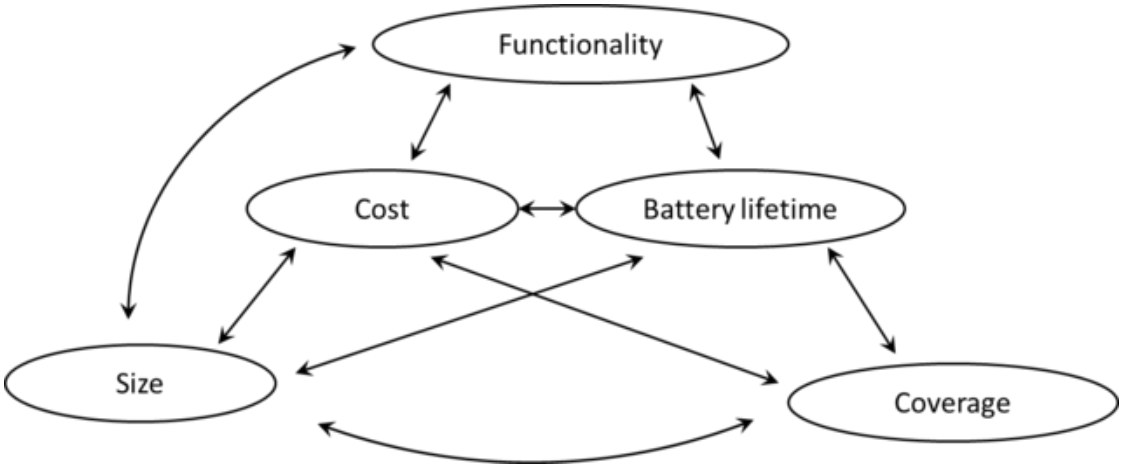


Fig 1.5.1 Relationships Of The primary sensor node parameters

In the next sections we are going to consider the internal structure of a sensor node as well as the main problems of sensor node development more precisely.

1.6 The Key Features Of Sensor Nodes

Before starting the consideration of the sensor node's structure in more detail, we should pay more attention to the main features of sensor nodes. They directly affect the capabilities of the whole WSN. That is why the requirements made to a WSN by a concrete application can always be converted to requirements for sensor nodes.

Energy Efficiency (Autonomy)

Unlike other common battery-powered mobile devices, sensor nodes deal with much more stringent requirements on energy efficiency, and it imposes restrictions on the all sensor node components. For example, for a mobile phone it is acceptable to keep working autonomously for a few days because the user usually have the possibility to charge the battery if necessary. But with sensor nodes we have another situation. The WSN parts may be spatially distributed on the area of many kilometers, especially if a WSN user is managing it via the Internet. At the same time, sensor nodes can be located in the inaccessible places, or the concrete location of each sensor node can be unknown. Also, a WSN may consist of dozens, hundreds or even thousands of sensor nodes. Under these conditions charging of sensor nodes by the user is out of question. That is why a sensor node must have high energy efficiency in order to keep working on small and inexpensive battery for a few months and even years. This ultra-

low-power operation can only be achieved by using low-power hardware components. Also, one of the key techniques extending the sensor node battery life is reduction of duty-cycle. This parameter is defined as the ratio of the sensor node active functioning time and the time when it is in the low power mode (the sleep mode). In WSNs with long lifetime sensor nodes most of the time are in the sleep mode, where sensor node power consumption is reducing in 3-4 times due to switching off all the main components excepting the part which is responsible for returning from the sleep mode when needed. After returning from the sleep mode the sensor node exchanges data with surrounding sensor nodes, takes readings from sensing elements, and then the sleep mode in turned on again.

Platform Flexibility

The majority of real applications require flexibility and adaptability of the WSN platform. In one application a user may need a WSN able to keep working for a few years, and herewith data update speed and data transmission delay won't play a significant role. For example, to monitor the soil temperature and humidity there is no need of frequent readings update and fast data transmission (because the soil temperature cannot change quickly), but it is very important that WSN which performs these functions keeps working as long as possible. In other applications such as monitoring of the spread of forest fires, fast detecting and fast data transmission will be more important, and the WSN lifetime will be less important parameter. So, each sensor node platform must have ability to be adjusted to meet the requirements of a specific application.

Reliability

Certainly, every WSN developer and manufacturer is interested in cost reduction of sensor nodes taking into consideration that every WSN has a great number of sensor nodes. Nevertheless, each concrete sensor node has to be reliable to such extent that it could work without breaking from the moment of turning on until the complete using of battery supply. In addition to increasing reliability of each sensor node, to provide the whole WSN reliability one may use adaptive protocols of data transmission management (*adaptive routing*). They are meant for providing WSN general robustness when certain sensor nodes are failing. For example, if traffic from one or a few sensor nodes is going through the other sensor node and it suddenly fails, as it is illustrated on The figure, the WSN will change its structure and reconnect the "lost" node through the others nearest to it. It is worth mentioning that

the main modern WSN platforms support this function.

There is also another common threat to WSN reliability which doesn't deal with reliability of any concrete sensor node. It is interference with the signals of other wireless networks and household or industrial devices' radiation. WSNs are often fully or partially located in places with significant electromagnetic fields of other wireless connection systems and appliances. In such cases these electromagnetic fields interfere with low-power transmitters in sensor node. This interference can be significant if it falls on radio spectrum in operation frequency range of sensor nodes' transmitters. In this case connection between nodes in the interference area can get much worse or even break down, and here even operable sensor nodes cannot transmit collected data. In such situations to increase the system's robustness to a node failure, a wireless sensor network must also be robust to external interference. The robustness of wireless links can be greatly increased through the use of *multi-channel and spread spectrum radios*. The figure represents principal of operation of the sensor nodes which support multi-channel radios. So, the possibility to change frequency channel for data transmitting is a necessary function for WSNs that are supposed to be deployed in a harsh electromagnetic environment.

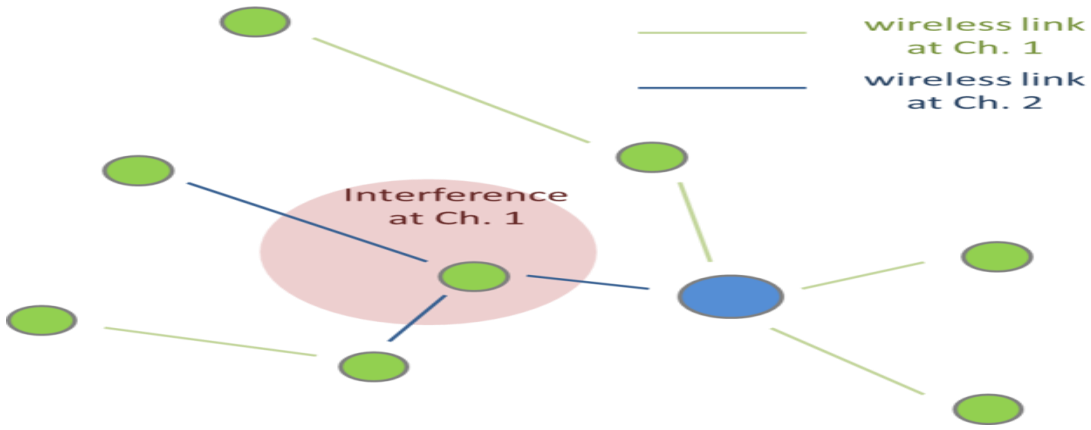


Fig 1.5.2 WSN working under conditions of strong interference on communication channel1

Information Security

Certain WSN applications make stringent requirements to information security. And this requirement becomes increasingly important, by reason of growth of cybernetic threats when WSNs are connected to the Internet. In order to meet the security requirements, sensor nodes must be capable of performing complex encrypting and authentication algorithms. In fact, radio communication channels can be easily tapped and become available for intruders. The only way to avoid it is encrypting of all

data transmitted in the WSN. Many modern sensor nodes make it possible to flexibly set traffic encryption in the network. In some platforms it is made by means of software, but some sensor nodes include special hardware encryption blocks. But in any case, encryption requires additional expenditure of energy, and it has negative impact on WSN lifetime.

Another aspect of information security in WSNs is protection of sensor nodes' internal memory. Sensor node internal memory includes not only information meant to be transmitted in the WSN, but also private keys for traffic encryption. So it must be reliably protected from external intervention.

These information security aspects have to be taken into account simultaneously. On the one hand, weak protection of internal memory will make WSN private keys available making it possible to "crack" the WSN no matter how complex encryption algorithm is. On the other hand, weak traffic encryption will make data transmitted in the network available for sniffing and alteration by the intruder, even if internal memory of each sensor node is well protected.

Transceiver Performance

One of the key sensor node characteristics is transceiver performance. The main parameters of transceiver performance which affect the sensor node characteristics are maximum data transfer rate, frequency range, modulation method, receiver sensitivity and transmitter power.

All these sensor node technical parameters affect such main WSN characteristic as reliability, the minimum spatial density of sensor nodes, the maximum readings update rate and lifetime. So, sensor node transceiver parameters are one of the main characteristics of WSN platform.

Above we have considered how the interference affects WSN reliability on a qualitative level. It is possible to estimate quantitatively how interference affects wireless link between sensor nodes with the help of the mentioned transceiver characteristics. For estimating the impact of noise on the quality of signal reception, in information theory the *signal-to-noise ratio* (SNR) is used. This ratio shows in how many times the wanted signal (signal from other sensor node) received by the sensor node receiver exceeds the power level of interference. The higher the SNR is, the more powerful is useful signal as compared with noise, and the higher is probability to receive the signal correctly. For every method of signal modulation there is a special SNR value at which or above which communication between receiver and transceiver is possible. Also, in information theory there is a fundamental principle, which can be expressed (in a simplified form) by the following statement: the higher SNR is, the higher is maximum data transfer rate.

Now it is obvious that the SNR affects reliability and quality of wireless link between two sensor nodes. And the higher SNR is, the better is the quality of communication. So, the more powerful is emission of the first sensor node's transmitter, the higher is SNR of receiving sensor node, and the higher is wireless link quality, hence the whole WSN reliability. In addition, the closer the sensor nodes to each other are located, the better is the SNR for both of them. It means that the maximum distance between sensor nodes in WSN is inextricably linked with power of sensor node transmitter, and the maximum distance between sensor nodes specifies the minimum number of sensor nodes necessary for covering the given space by WSN.

Sensor node receiver *sensitivity* represents the ability to receive weak signals, for example, at a great distance from other sensor nodes, and it, as well as transmitter power, affects the maximum distance between the sensor nodes. It is worth mentioning that increasing the power and sensitivity of sensor node transmitter and receiver leads to higher energy consumption and cost of the sensor nodes. But this dependence is not linear, and the benefit in increasing the range of sensor node is not so great. That is why the most common characteristics of transceivers measure up with ones mW of power, which is acceptable in terms of energy consumption and provides reliable wireless connection between sensor nodes at the distance of about 10 meters.

Frequency range of transceiver affects the maximum possible rate of data exchange and the maximum possible distance between the sensor nodes. At the heart of this dependence are the physical laws of the radio signal. According to these laws, the higher is frequency used as carrier, the stronger is the signal attenuation with the distance. That is why the sensor nodes platforms which operate in lower frequency ranges allow having higher value of the maximum distance between sensor nodes in WSN. But the basic physical laws don't allow using very low frequencies for connecting sensor nodes in the majority of WSNs, because the size of the transceiver's antenna has to be the bigger, the lower is the frequency, and it affects the size of the sensor nodes.

The maximum speed of data transmission and reception by sensor node transceiver restrains the maximum speed of data gathering in the WSN. In addition, the higher is the maximum speed of data transmission, the higher is the energy consumption of transceiver during transmission and reception. On the other hand, the higher is speed of transmission, the less time is necessary for transmitting the same data; hence, transceiver will be switched on for less time. But high speed of transmitting also requires more computing power and energy for this computing, which is not always acceptable.

characteristics of the wsn which utilizes such sensor nodes computing power

Sensor node's microcontroller (and hence, consumes battery energy) uses its computing powers for two kinds of tasks. First kind of these tasks deals with supporting WSN functioning, the second task is reading and processing measurements of sensing element. Both kinds of tasks require certain computing power and take the time of the microcontroller. When the micro controller is busy, its energy consumption becomes significant.

The task of supporting WSN functioning, in the first place, is implementation data reception and further transmission algorithms that are part of the WSN communication protocol. Every sensor node is permanently receiving data from other surrounding sensor nodes. Microcontroller identifies these data and depending on the content transmits to the nearest sensor nodes, ignores them or saves to internal memory for further processing. All it happens in accordance with the WSN communication protocol. Computing power of sensor node microcontroller has to be the higher, the higher is the maximum rate of data exchange, so that to have time for data decoding.

We can see the same situation with computing powers necessary for reading and processing of sensor measurements. Sensitive elements can produce a plenty of data which have to be timely processed. And the types of necessary processing can vary a lot, from simple averaging, digital filtration, tracking of some threshold exceeding to calculation of autocorrelation and spectral analysis. The last two operations are the example of the especially resources-consuming ones.

Size And Cost

Miniaturization, price reduction, and improvement of other parameters are the most important priorities from the very first researches in WSNs. The good example is the SmartDust project which took place in the end of 1990s and the beginning of 2000s. Miniaturization and price reduction of sensor were constantly expanding the possibilities of WSN applications, and in future they can lead to the widespread use of WSNs and to uprising of ubiquitous WSNs.

Above we have already considered the dependence between different WSN characteristics, and now, after considering the additional characteristics, it is possible to imagine how difficult is to find balance between them when developing the sensor nodes.

1.7 Introduction of Domain

Machine Learning:

In the statistical context, Machine Learning is defined as an application of artificial intelligence where available information is used through algorithms to process or assist the processing of statistical data. While Machine Learning involves concepts of automation, it requires human guidance. Machine Learning involves a high level of generalization in order to get a system that performs well on yet unseen data instances.

Machine learning is a relatively new discipline within Computer Science that provides a collection of data analysis techniques. Some of these techniques are based on well established statistical methods (e.g. logistic regression and principal component analysis) while many others are not.

Most statistical techniques follow the paradigm of determining a particular probabilistic model that best describes observed data among a class of related models. Similarly, most machine learning techniques are designed to find models that best fit data (i.e. they solve certain optimization problems), except that these machine learning models are no longer restricted to probabilistic ones.

Therefore, an advantage of machine learning techniques over statistical ones is that the latter require underlying probabilistic models while the former do not. Even though some machine learning techniques use probabilistic models, the classical statistical techniques are most often too stringent for the oncoming Big Data era, because data sources are increasingly complex and multi-faceted. Prescribing probabilistic models relating variables from disparate data sources that are plausible and amenable to statistical analysis might be extremely difficult if not impossible.

Machine learning might be able to provide a broader class of more flexible alternative analysis methods better suited to modern sources of data. It is imperative for statistical agencies to explore the possible use of machine learning techniques to determine whether their future needs might be better met with such techniques than with traditional ones.

1.8 Objective of the Problem

The goal of intrusion detection is to monitor the network assets to detect anomalous behavior and misuse in network. Intrusion detection concept was introduced in early 1980’s after the evolution of internet with surveillance end monitoring the threat. There was a sudden rise in reputation and incorporation in security infrastructure. Since then, several events in IDS technology have advanced intrusion detection to its current state. James Anderson's wrote a paper for a government organization and imported an approach that audit trails contained important information that could be valuable in tracking misuse and understanding of user behavior. Then the detection appeared and audit data and its importance led to terrific improvements in the subsystems of every operating system. IDS and Host Based Intrusion Detection System (HIDS) were first defined. In 1983, SRI International and Dorothy Denning began working on a government project that launched a new effort into intrusion detection system development. Around 1990s the revenues are generated and intrusion detection market has been raised. Real secure is an intrusion detection network developed by ISS. After a year, Cisco recognized the priority for network intrusion detection and purchased the Wheel Group for attaining the security solutions. The government actions like Federal Intrusion Detection Networks (FID Net) were designed under Presidential Decision Directive 63 is also adding impulse to the IDS .

The main objectives of the problem are as follows:

1. Less throughput.
2. No energy checkup after each communication.
3. Time delay is high.
4. High packet loss.
5. Delivery ratio is less.
6. Attacking is very high.

1.9 Scope of the project

We propose a method to detect intrusions in computer network security. The main idea is to reuse the already available system information that is generated at various layers of a network stack. To the best of our knowledge, this is the first such approach for intrusion detection in computer network security. Wireless Sensor Network consists of large number of nodes, which will be in distributed nature. Security is a very important consideration while designing a Wireless Sensor Network. So, an Advanced Intrusion Detection System has been proposed where the Heterogeneous Hybrid Intrusion Detection System (H-HIDS), Intrusion Detection Prediction based are implemented in various stages in order to assure maximum possible security from the Intrusions.

The main scope of this project is to provide communication continuously by checking the energy at every communication and also to find out the intrusion that is happening inside the network. The main objectives of the projects are as follows:

1. To find out early attacks.
2. To minimize packet loss.
3. More throughputs.
4. To reduce time consumption.
5. Continuous energy check up of all nodes to avoid node failure.

CHAPTER-2

2. SURVEY OF THE LITERATURE

2.1 General

A literature survey is the overall description of the reference papers, which identifies the problem of existing methodologies. Also, the methods to overcome such issues can be identified.

2.2 Literature survey

NL-IDS: Trust Based Intrusion Detection System for Network layer in Wireless Sensor Networks Umashankar Ghugar, Jayaram Pradhan IEEE 2022.

From the last few years, security in wireless sensor network (WSN) is essential because WSN application uses important information sharing between the nodes. There are large number of issues raised related to security due to open deployment of network. The attackers disturb the security system by attacking the different protocol layers in WSN. The standard AODV routing protocol faces security issues when the route discovery process takes place. The data should be transmitted in a secure path to the destination. Therefore, to support the process we have proposed a trust based intrusion detection system (NL-IDS) for network layer in WSN to detect the Black hole attackers in the network. The sensor node trust is calculated as per the deviation of key factor at the network layer based on the Black hole attack. We use the watchdog technique where a sensor node continuously monitors the neighbor node by calculating a periodic trust value. Finally, the overall trust value of the sensor node is evaluated by the gathered values of trust metrics of the network layer (past and previous trust values). This NL-IDS scheme is efficient to identify the malicious node with respect to Black hole attack at the network layer. To analyze the performance of NL-IDS, we have simulated the model in MATLAB R2015a, and the result shows that NL-IDS is better than Wang et al. [11] as compare of detection accuracy and false alarm rate.

Analyzing the Vulnerability of Wireless Sensor Networks to a Malicious Matched Protocol Attack George D. O'Mahon, Philip J. Harris, Colin C. Murphy IEEE 2022.

Safety critical, Internet of Things (IoT) and space-based applications have recently begun to adopt wireless networks based on commercial off the shelf (COTS) devices and standardized protocols, which inherently establishes the security challenge of malicious intrusions. Malicious intrusions can cause severe consequences if undetected, including, complete denial of services. Particularly, any safety

critical application requires all services to operate correctly, as any loss can be detrimental to safety and/or privacy. Therefore, in order for these safety critical services to remain operational and available, any and all intrusions need to be detected and mitigated. Whilst intrusion detection is not a new research area, new vulnerabilities in wireless networks, especially wireless sensor networks (WSNs), can be identified. In this paper, a specific vulnerability of WSNs is explored, termed here the matched protocol attack. This malicious attack uses protocol-specific structures to compromise a network using that protocol. Through attack exploration, this paper provides evidence that traditional spectral techniques are not sufficient to detect an intrusion using this style of attack. Furthermore, a ZigBee cluster head network, which co-exists with ISM band services, consisting of XBee COTS devices is utilized, along with a real time spectrum analyzer, to experimentally evaluate the effect of matched protocol interference on a realistic network model. Results of this evaluation are provided in terms of device errors and spectrum use. This malicious challenge is also examined through Monte-Carlo simulations. A potential detection technique, based on coarse inter-node distance measurements, which can theoretically be used to detect matched protocol interference and localize the origin of the source, is also suggested as a future progression of this work. Insights into how this attack style preys on some of the main security risks of any WSN (interoperability, device

Abnormal-Node Detection Based on Spatio-Temporal and Multivariate-Attribute Correlation in Wireless Sensor Networks Nesrine Berjab, Hieu Hanh Le, Chia-Mu Yu, Sy-Yen Kuo, Haruo Yokota IEEE 2022.

In wireless sensor networks (WSNs), data can be subject to malicious attacks and failures, leading to unreliability. This vulnerability poses a challenge to environmental monitoring applications by creating false alarms. To guarantee a trustworthy system, we therefore need to detect abnormal nodes. In this paper, we propose a new framework for detecting abnormal nodes in clustered heterogeneous WSNs. It makes use of observed spatiotemporal (ST) and multivariate-attribute (MVA) sensor correlations, while considering the background knowledge of the monitored environment. Based on the ST correlations, the collected data is analyzed by computing the crosscorrelation between sensor streams. A new method is proposed for evaluating the intensity of the correlation between two sensor streams. The crosscorrelation value obtained is compared against two thresholds, the lag threshold and the correlation threshold. Based on available background knowledge and the observed MVA correlations, a number of rules are presented to detect abnormal nodes while identifying real events. Our experiments on real-world sensor data demonstrate that our approach captures the correlation and

discovers abnormal nodes efficiently.

Secure Knowledge and Cluster-Based Intrusion Detection Mechanism for Smart WirelessSensor Networks Amjad Mehmood, Akbar Khanan, Muhammad Muneer Umar, Salwani Abdulla, Khairul Akram Zainol Ariffin, Houbing Song IEEE 2022.

Wireless sensor networks, due to their nature, are more prone to security threats than other networks. Developments in WSNs have led to the introduction of many protocols specially developed for security purposes. Most of these protocols are not efficient in terms of putting an excessive computational and energy consumption burden on small nodes in WSNs. This paper proposes a knowledge-based context-aware approach for handling the intrusions generated by malicious nodes. The system operates on a knowledge base, located at the base station, which is used to store the events generated by the nodes inside the network. The events are categorized and the cluster heads (CHs) are acknowledged to block maliciously repeated activities generated. The CHs can also get informational records about the maliciousness of intruder nodes by using their inference engines. The mechanism of events logging and analysis by the base station greatly affects the performance of nodes in the network by reducing the extra security-related load on them.

Threshold Tuning-Based Wearable Sensor Fault Detection for Reliable Medical Monitoring Using Bayesian Network Model Haibin Zhang, Jiajia Liu, Nei Kato IEEE 2022.

As the medical body sensor network (BSN) is usually resource limited and vulnerable to environmental effects and malicious attacks, faulty sensor data arise inevitably which may result in false alarms, faulty medical diagnosis, and even serious misjudgment. Thus, faulty sensory data should be detected and removed as much as possible before being utilized for medical diagnosis-making. Most available works directly employed fault detection schemes developed in traditional wireless sensor network (WSN) for body sensor fault detection. However, BSNs adopt a very limited number of sensors for vital information collection, lacking the information redundancy provided by densely deployed sensor nodes in traditional WSNs. In light of this, a Bayesian network model-based sensor fault detection scheme is proposed in this paper, which relies on historical training data for establishing the conditional probability distribution of body sensor readings, rather than the redundant information collected from a large number of sensors. Furthermore, the Bayesian network-based scheme enables us to minimize the inaccuracy rate by optimally tuning the threshold for fault detection. Extensive online dataset has been adopted to evaluate the performance of our fault detection scheme, which shows that our scheme

possesses a good fault detection accuracy and a low false alarm rate.

A Light-Weight Countermeasure to Forwarding Misbehavior in Wireless Sensor Networks: Design, Analysis, and Evaluation Cong Pu, Sunho Lim IEEE 2022.

Due to the lack of centralized coordination, physical protection, and security requirements of inherent network protocols, wireless sensor networks (WSNs) are vulnerable to diverse denial-of-service (DoS) attacks that primarily target service availability by disrupting network routing protocols or interfering with on-going communications. In this paper, we propose a light-weight countermeasure to a selective forwarding attack, called SCAD, where a randomly selected single checkpoint node is deployed to detect the forwarding misbehavior of malicious node. The proposed countermeasure is integrated with timeout and hop-by-hop retransmission techniques to quickly recover unexpected packet losses due to the forwarding misbehavior or bad channel quality. We also present a simple analytical model and its numerical result in terms of false detection rate. We conduct extensive simulation experiments for performance evaluation and comparison with the existing CHEMAS and CAD schemes. The simulation results show that the proposed countermeasure can improve the detection rate and packet delivery ratio (PDR) as well as reduce the energy consumption, false detection rate, and successful drop rate.

Energy Efficient Detection-Removal Algorithm for Selective Forwarding Attack In Wireless Sensor Networks T.R Sreelakshmi, G.S Binu IEEE 2022.

Wireless sensor networks (WSNs) propose the promise of a flexible, low cost solution for monitoring critical infrastructure. Sensor networks have been recommended for applications such as traffic monitoring, military and battlefield surveillance. Wireless sensor networks are more prone to security attacks due to their broadcasting nature of the transmission medium and unattended deployment of nodes in hostile and unfriendly areas where they are not protected as compared to wired networks. Attackers can deploy various types of security attacks to obstruct the security of WSNs. Network layer attacks are more severe since if the routing information is disregarded, disturbances may bring about routing loops, changing of routes etc. Selective forwarding attack is a type of active attack affecting network layers that selectively drops or refuses to forward the data packets. This paper discusses about an energy efficient detection-removal algorithm for effective detection of selective forwarding attack in a clustered WSN scenario. The impact of the malicious node in network parameters like packet delivery ratio, throughput, residual energy of network and end to end delay are analyzed.

False Data Injection Prevention in Wireless Sensor Networks using Node-level Trust Value Computation B. Sreevidya, M. Rajesh IEEE 2022.

Wireless Sensor Networks are extensively used in developing applications for surveillance, habitat monitoring, border security, intrusion detection etc. Most of these applications require secure data transmission among the nodes of the network. Out of the different types of attacks a data critical application faces, False Data Injection attacks are the most damaging one. So prevention of False Data Injection attacks is a crucial aspect while building data critical wireless sensor network applications. Researchers have suggested cryptographic schemes like RSA, ECC for the prevention of False Data Injection Attacks. Use of cryptographic techniques increases the computation complexity on all the nodes and the energy constraints on WSN demands an alternate solution for False Data Injection attacks prevention. The proposed work aims on using trust parameter of every nodes to distinguish malicious and non-malicious nodes and use only trusted nodes to forward the packet to destination thus by prevention FDI attacks. Simulation is carried out with the help of Network Simulator 2 (NS2). The results shows the energy consumption is less in the proposed scheme compared to the cryptographic technique

A Comprehensive Trust-Aware Routing Protocol With Multi-Attributes for WSNs Boyuan Sun, Donghui Li IEEE 2022.

Due to the impact of an open deployment environment, severe restrictions in power with poor hardware equipment, and a lack of centralized administration in management, wireless sensor networks (WSNs) are extremely vulnerable to malicious attacks aimed at routing and other aspects. To face this problem, we propose a novel trust-aware routing protocol for WSNs which incorporates multiattributes (TRPM) of sensor nodes in terms of communication, data, energy, and recommendation. The proposed trust model relies on an improved sliding time window considering attack frequency to facilitate the discovery of malicious behaviors of attackers. Combined with effective routing detection and maintenance protocol, the performance of our solution is tested through a wide set of simulation experiments. Extensive results reveal that an average packet transfer rate of TRPM is increased by about 19% and time consumption on the routing update is shortened by about 11%.

Heterogeneous statistical QoS provisioning over 5G mobile wireless networks Author: Xi Zhang ;Wenchi Cheng ; Hailin Zhang Published in: IEEE Network (Volume: 28, Issue: 6, Nov.-Dec. 2022)

In this article we propose a novel heterogeneous statistical QoS provisioning architecture for 5G mobile wireless networks. First, we develop and analyze the new heterogeneous statistical QoS system model by applying and extending the effective capacity theory. Then, through the wireless coupling channels, we apply our proposed heterogeneous statistical QoS architecture to efficiently implement the following powerful 5G-candidate wireless techniques: 1) device-to-device networks; 2) full-duplex networks; and 3) cognitive radio networks, respectively, for providing heterogeneous statistical delay-bounded QoS guarantees. Finally, using the simulation experiments we show that our proposed architecture and schemes significantly outperform the existing traditional statistical delay-bounded QoS provisioning schemes in terms of satisfying the heterogeneous delay-bounded QoS requirements while maximizing the aggregate system throughput over 5G mobile wireless networks.

CHAPTER-3

3. PROBLEM STATEMENT AND METHODOLOGY

3.1 PROBLEM DEFINITION

An IDS is referred as burglar alarm. For example the lock system in the house protects the house from theft. But if somebody breaks the lock system and tries to enter into the house, it is the burglar alarm that detects that the lock has been broken and alerts the owner by raising an alarm. Moreover, Firewalls do a very good job of filtering the incoming traffic from the Internet to circumvent the firewall. For example, external users can connect to the Intranet by dialing through a modem installed in the private network of the organization; this kind of access cannot be detected by the firewall. An Intrusion Prevention System (IPS) is a network security/threat prevention technology that audits network traffic flows to detect and prevent vulnerability exploits. There are two types of prevention system they are Network (NIPS) and Host (HIPS). These systems watch the network traffic and automatically take actions to protect networks and systems. IPS issue is false positives and negatives. False positive is defined to be an event which produces an alarm in IDS where there is no attack. False negative is defined to be an event which does not produces an alarm when there is an attacks takes place. Inline operation can create bottlenecks such as single point of failure, signature updates and encrypted traffic. The actions occurring in a system or network is measured by IDS.

3.2 METHODOLOGY

The IDS have been implemented in organizations to collect and analyze various types of attacks within a host system or a network. In addition, to identify and detect possible threats violations, which involve both intrusions, which are the attacks from outside the organizations and misuses that are known as the attacks within the organizations. In this paper, we proposed the integrated model which involves a combination of the two systems Intrusion Detection (ID) and Intrusion Prevention (IP) adding to those getting benefits from well-known techniques: intruder Detection (ID) which is totally different from most of the recent works that focused only on using one system, either detection or prevention and also using either Intruder detection or Signature based detection. Some works even used a hybrid method which is a combination of both such as the work presented where the researchers used ID based on Signature but even then, their method was not provided with prevention capabilities. On the other hand, in our case, we proposed to use our approach IDPS, which not only can detect the attack but also can

further stop it using the capabilities of prevention system, which has not been utilized in the previous works. Therefore, the proposed system can outperform the hybrid system in terms of preventing the attack from conducting any bad action through blocking the event and saving that threat with the other signatures in order to be observed by Signature Based Intrusion Detection for next time so that it can be detected earlier. Finally, deploying such integrated model in the Wireless environment will reduce the probability of risks than the normal system or even than other systems, which are just provided with Intrusion Detection methods.

CHAPTER-4

4 Existing System

4.1 Naïve Bayes (NB):

In statistics, **naive Bayes classifiers** are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naïve) independence assumptions between the features. They are among the simplest Bayesian network models, but coupled with kernel density estimation, they can achieve higher accuracy levels.

Naïve Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

In the statistics and computer science literature, naive Bayes models are known under a variety of names, including **simple Bayes** and **independence Bayes**. All these names reference the use of Bayes' theorem in the classifier's decision rule, but naïve Bayes is not (necessarily) a Bayesian method.

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. There is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.

For some types of probability models, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods.

Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In 2004, an analysis of the Bayesian

classification problem showed that there are sound theoretical reasons for the apparently implausible efficacy of naive Bayes classifiers. Still, a comprehensive comparison with other classification algorithms in 2006 showed that Bayes classification is outperformed by other approaches, such as boosted trees or random forests.

An advantage of naive Bayes is that it only requires a small number of training data to estimate the parameters necessary for classification.

Existing System Disadvantages:

1. Less Accuracy

2. Attack can happen at any time.

3. More time consuming process

4. Higher Computational Cost

5. Lack of standards

6. Cannot be implemented in all datasets

4.2 Proposed System

4.2.1 Random Forest (RF):

Random forests or **random decision forests** are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of over fitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

Decision trees are a popular method for various machine learning tasks. Tree learning "come[s] closest to meeting the requirements for serving as an off-the-shelf procedure for data mining", say Hastie *et al.*,

"because it is invariant under scaling and various other transformations of feature values, is robust to inclusion of irrelevant features, and produces inspectable models. However, they are seldom accurate".

In particular, trees that are grown very deep tend to learn highly irregular patterns: they overfit their training sets, i.e. have low bias, but very high variance. Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance. This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance in the final model.

Forests are like the pulling together of decision tree algorithm efforts. Taking the teamwork of many trees thus improving the performance of a single random tree. Though not quite similar, forests give the effects of a K-fold cross validation.

Proposed System Advantages:

1. High output efficiency.
2. User friendly.
3. Less time consumption.
4. Can be implemented in all datasets.
5. Early prediction of attack is possible.

4.3 SYSTEM ARCHITECTURE

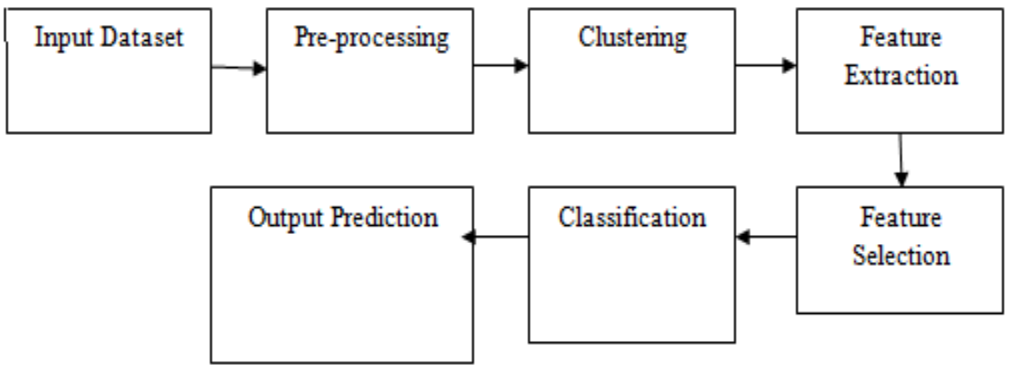


Fig 4.3.1 Architecture diagram

4.4 UML Diagram

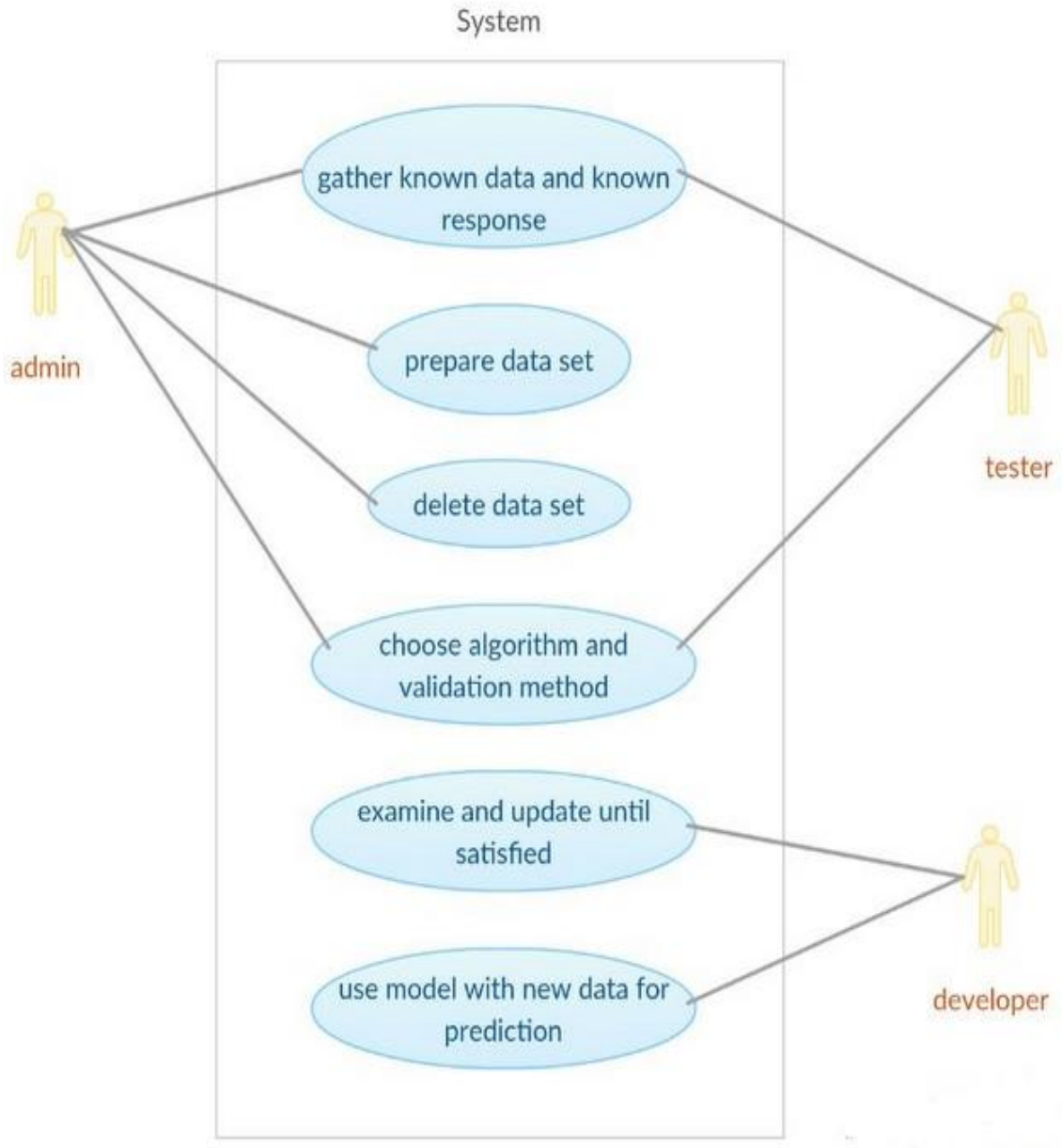
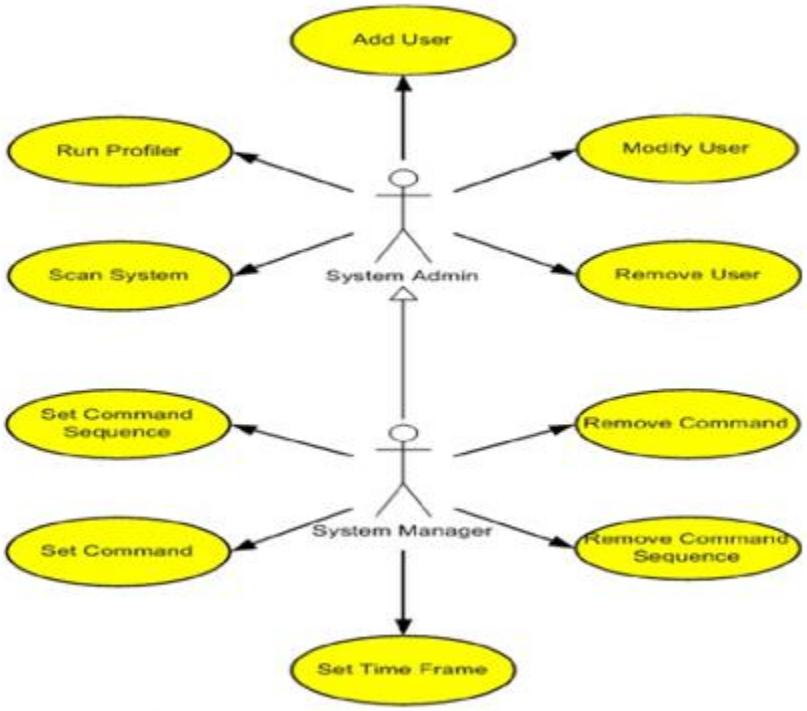


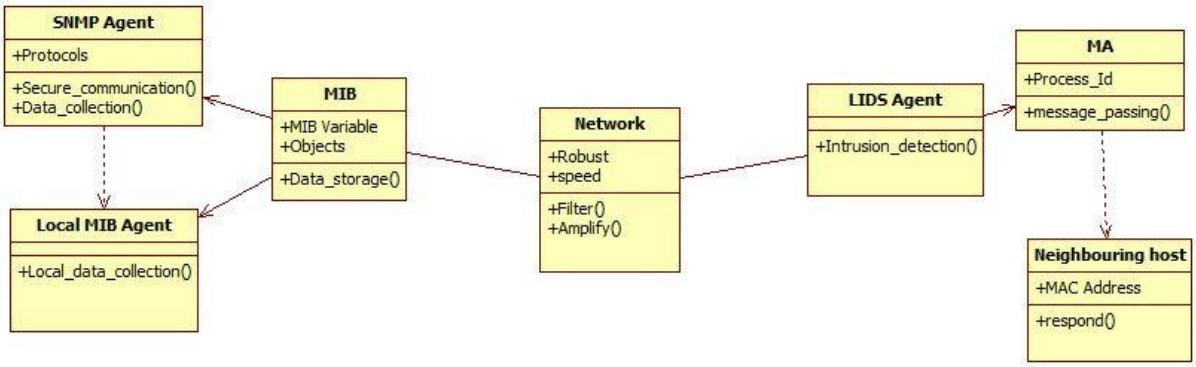
Fig.4.4 Uml Diagram

4.4.1 Use Case Diagram



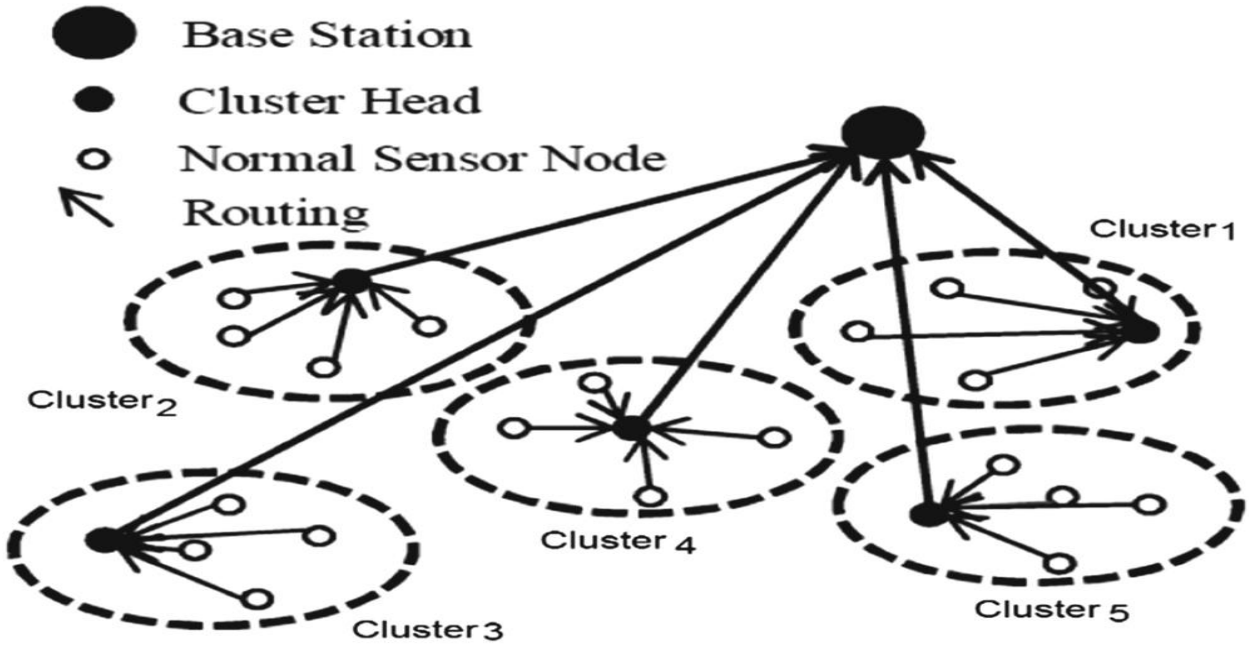
4.4.1 Use Case Diagram

4.4.2 Class Diagram



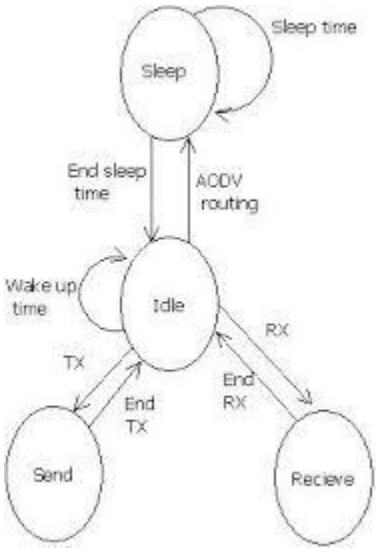
4.4.2 Class diagram

4.4.3 Component Diagram



4.4.3 Component diagram

4.4.4 Deployment Diagram



4.4.4 Deployment diagram

CHAPTER-5

5. SYSTEM IMPLEMENTATION

5.1 MODULE DESCRIPTION

There are 8 components in the system. They are

- Incoming packet
- Packet Capture
- Packet scanner
- Packet analyzer
- Labeling
- Training model
- Prediction:
- Output

Incoming packet: In our project directly connected to network (we capture online packet) and transfer to packet scanner.

Packet Capture: In Real time networking many packets are transferred from source to destination. In this module live packets from the network are captured and passed to the pre-processor module. According to the concept of machine learning the data is grouped or clustered based on its features or characteristics for e.g. protocols used by the packets.

Packet scanner: After catching the packet we use packet scanner for the purpose of scanning the packet. Packet scanning is the important part of our system.

Packet analyzer: Packet analyzer is also known as Packet sniffer. As data streams flow across the network sniffer capture each packet and if needed decodes the packet, raw data showing the values of various fields in the packet and analyzes its content accord- ing to specification.

Labeling: Labeling is used for defining the corresponding packet.

Training model: Training model contain set of trained data set which used to detect attack in the packet.

Prediction: Based on the training model we make the prediction that the packet is normal packet or

abnormal packet.

Output: Based on the prediction (i.e. normal or abnormal) we generate output in the form for abnormal packet

5.2 ALGORITHM OF PROPOSED WORK

i. Dataset is divided into small overlapping or either non-overlapping blocks.

ii. Extract the features using traditional techniques.

iii. Extracted feature values corresponding to each key point are stored in matrix.

iv. Apply sorting techniques to get similar features that lie in nearness.

v. Introduce shift vector concept to find key point with similar shifting.

vi. Use the counter vector to count the occurrence same shifting key point and set the counter to 1.

vii. Similar regions are identified with the help of threshold value above steps are used.

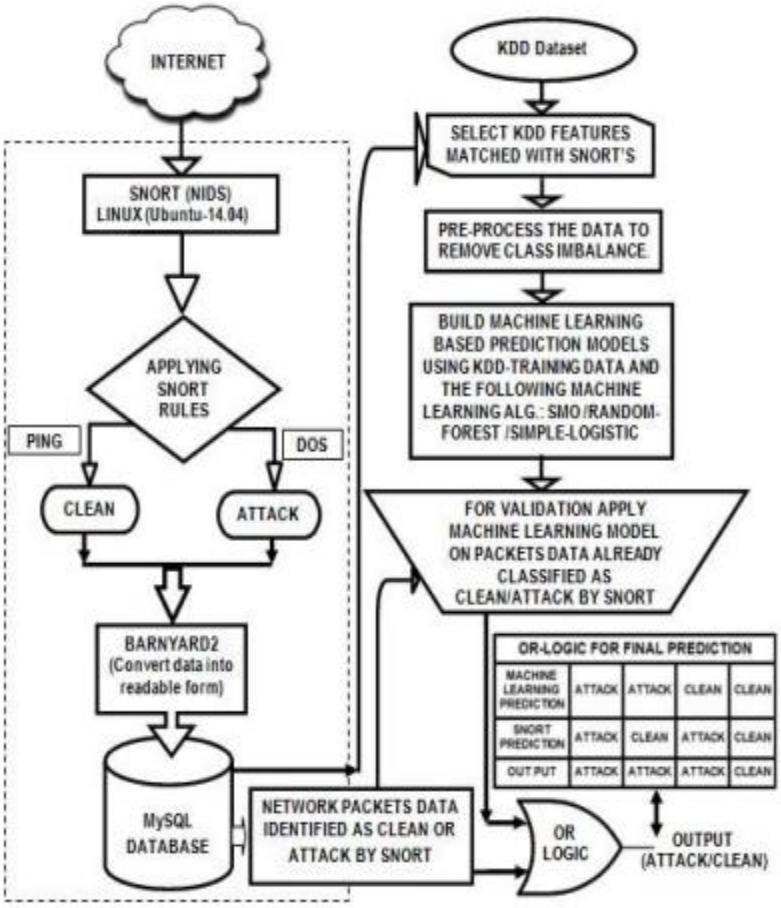


Fig 5.2.1 Flow chat

5.3 COMPARATIVE STUDY OF EXISTING AND PROPOSED SYSTEM

In our project we have used algorithms like Naïve Bayes (NB) as existing system and Random Forest (RF) as proposed system. All are measured in terms of accuracy. From the results its proved that proposed Random Forest (RF) works better than existing Naïve Bayes (NB). From this we are getting good accuracy so we can state that our proposed system works better than the existing system.

CHAPTER-6

6. SYSTEM STUDY

6.1 FEASIBILTY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

The feasibility study investigates the problem and the information needs of the stakeholders. It seeks to determine the resources required to provide an information systems solution, the cost and benefits of such a solution, and the feasibility of such a solution. The analyst conducting the study gathers information using a variety of methods, the most popular of which are:

- Developing and administering questionnaires to interested stakeholders, such as potential users of the information system.
- Observing or monitoring users of the current system to determine their needs as well as their satisfaction and dissatisfaction with the current system.
- Collecting, examining, and analyzing documents, reports, layouts, procedures, manuals, and any other documentation relating to the operations of the current system.
- Modeling, observing, and simulating the work activities of the current system.

The goal of the feasibility study is to consider alternative information systems solutions, evaluate their feasibility, and propose the alternative most suitable to the organization. The feasibility of a proposed solution is evaluated in terms of its components. These components are:

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

6.2 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

6.3 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client.

6.4 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity.

The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism.

6.5 OPERATIONAL FEASIBILITY

The ability, desire, and willingness of the stakeholders to use, support, and operate the proposed computer information system. The stakeholders include management, employees, customers, and suppliers. The stakeholders are interested in systems that are easy to operate, make few, if any, errors, produce the desired information, and fall within the objectives of the organization.

CHAPTER-7

7. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

7.1 TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER-8

8. SOFTWARE DESCRIPTION

A. Python

Python is a remarkably powerful dynamic, object-oriented programming language that is used in a wide variety of application domains. It offers strong support for integration with other languages and tools, and comes with extensive standard libraries. To be precise, the following are some distinguishing features of Python:

- Very clear, readable syntax.
- Strong introspection capabilities.
- Full modularity.
- Exception-based error handling.
- High level dynamic data types.
- Supports object oriented, imperative and functional programming styles.
- Embeddable.
- Scalable
- Mature

With so much of freedom, Python helps the user to think problem centric rather than language centric as in other cases. These features makes Python a best option for scientific computing.

B. Open CV

Open CV is a library of programming functions for real time computer vision originally developed by Intel and now supported by Willogarage. It is free for use under the open source BSD license. The library has more than five hundred optimized algorithms. It is used around the world, with forty thousand people in the user group. Uses range from interactive art, to mine inspection, and advanced robotics. The library is mainly written in C, which makes it portable to some specific platforms such as Digital Signal Processor. Wrappers for languages such as C, Python, Ruby and Java (using Java CV) have been developed to encourage adoption by a wider audience. The recent releases have interfaces for C++. It focuses mainly on real-time image processing. Open CV is a cross-platform library, which can run on Linux, Mac OS and Windows. To date, Open CV is the best open source computer vision library that developers and researchers can think of.

C. Tesseract

Tesseract is a free software OCR engine that was developed at HP between 1984 and 1994. HP released it to the community in 2005. Tesseract was introduced at the 1995 UNLV Annual Test OCR Accuracy and is currently developed by Google released under the Apache License. It can now recognize 6 languages, and is fully UTF8 capable. Developers can train Tesseract with their own fonts and character mapping to obtain perfect efficiency.

8.1 ABOUT THE SOFTWARE "PYTHON"

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include −

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below –

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.

- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python - Environment Setup

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

Local Environment Setup

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

- Unix (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)
- Win 9x/NT/2000
- Macintosh (Intel, PPC, 68K)
- OS/2
- DOS (multiple versions)
- PalmOS
- Nokia mobile phones
- Windows CE
- Acorn/RISC OS
- BeOS
- Amiga
- VMS/OpenVMS
- QNX
- VxWorks
- Psion
- Python has also been ported to the Java and .NET virtual machines

Getting Python

The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python https://www.python.org/

You can download Python documentation https://www.python.org/doc/The documentation is available in HTML, PDF, and PostScript formats.

Installing Python

Python distribution is available for a wide variety of platforms. You need to download only the binary code applicable for your platform and install Python.

If the binary code for your platform is not available, you need a C compiler to compile the source code manually. Compiling the source code offers more flexibility in terms of choice of features that you require in your installation.

Here is a quick overview of installing Python on various platforms −

Unix and Linux Installation

Here are the simple steps to install Python on Unix/Linux machine.

- Open a Web browser and go to https://www.python.org/downloads/.
- Follow the link to download zipped source code available for Unix/Linux.
- Download and extract files.
- Editing the *Modules/Setup* file if you want to customize some options.
- run ./configure script
- make
- make install

This installs Python at standard location */usr/local/bin* and its libraries at */usr/local/lib/pythonXX* where XX is the version of Python.

Windows Installation

Here are the steps to install Python on Windows machine.

- Open a Web browser and go to https://www.python.org/downloads/.
- Follow the link for the Windows installer *python-XYZ.msi* file where XYZ is the version you need to install.
- To use this installer *python-XYZ.msi*, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.
- Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

Macintosh Installation

Recent Macs come with Python installed, but it may be several years out of date. See http://www.python.org/download/mac/ for instructions on getting the current version along with extra tools to support development on the Mac. For older Mac OS's before Mac OS X 10.3 (released in 2003), MacPython is available.

Jack Jansen maintains it and you can have full access to the entire documentation at his website − http://www.cwi.nl/~jack/macpython.html. You can find complete installation details for Mac OS installation.

8.2 SETTING UP PATH

Programs and other executable files can be in many directories, so operating systems provide a search path that lists the directories that the OS searches for executables.

The path is stored in an environment variable, which is a named string maintained by the operating system. This variable contains information available to the command shell and other programs.

The **path** variable is named as PATH in Unix or Path in Windows (Unix is case sensitive; Windows is not).

In Mac OS, the installer handles the path details. To invoke the Python interpreter from any particular directory, you must add the Python directory to your path.

Setting path at Unix/Linux

To add the Python directory to the path for a particular session in Unix −

- **In the csh shell** − type setenv PATH "$PATH:/usr/local/bin/python" and press Enter.
- **In the bash shell (Linux)** − type export PATH="$PATH:/usr/local/bin/python" and press Enter.
- **In the sh or ksh shell** − type PATH="$PATH:/usr/local/bin/python" and press Enter.
- **Note** − /usr/local/bin/python is the path of the Python directory

Setting path at Windows

To add the Python directory to the path for a particular session in Windows −

At the command prompt − type path %path%;C:\Python and press Enter.

Note − C:\Python is the path of the Python directory

Python Environment Variables

Here are important environment variables, which can be recognized by Python −

Sr.No.	Variable & Description
1	**PYTHONPATH** It has a role similar to PATH. This variable tells the Python interpreter where to locate the module files imported into a program. It should include the Python source library directory and the directories containing Python source code. PYTHONPATH is sometimes preset by the Python installer.
2	**PYTHONSTARTUP** It contains the path of an initialization file containing Python source code. It is executed every time you start the interpreter. It is named as .pythonrc.py in Unix and it contains commands that load utilities or modify PYTHONPATH.

3	**PYTHONCASEOK** It is used in Windows to instruct Python to find the first case-insensitive match in an import statement. Set this variable to any value to activate it.
4	**PYTHONHOME** It is an alternative module search path. It is usually embedded in the PYTHONSTARTUP or PYTHONPATH directories to make switching module libraries easy.

Table 8.2.1 important environment variables

Running Python

There are three different ways to start Python −

Interactive Interpreter

You can start Python from Unix, DOS, or any other system that provides you a command-line interpreter or shell window.

Enter **python** the command line.

Start coding right away in the interactive interpreter.

```
$python # Unix/Linux
or
python% # Unix/Linux
or
C:> python # Windows/DOS
```

Here is the list of all the available command line options −

Sr.No.	Option & Description
1	**-d** It provides debug output.
2	**-O** It generates optimized bytecode (resulting in .pyo files).
3	**-S** Do not run import site to look for Python paths on startup.
4	**-v** verbose output (detailed trace on import statements).
5	**-X** disable class-based built-in exceptions (just use strings); obsolete starting with version 1.6.
6	**-c cmd** run Python script sent in as cmd string
7	**File** run Python script from given file

Table 8.2.2 command line

Script from the Command-line

A Python script can be executed at command line by invoking the interpreter on your application, as in

the following −

```
$python script.py # Unix/Linux

or

python% script.py # Unix/Linux

or

C: >python script.py # Windows/DOS
```

Note − Be sure the file permission mode allows execution.

Integrated Development Environment

You can run Python from a Graphical User Interface (GUI) environment as well, if you have a GUI application on your system that supports Python.

- **Unix** − IDLE is the very first Unix IDE for Python.
- **Windows** − PythonWin is the first Windows interface for Python and is an IDE with a GUI.
- **Macintosh** − The Macintosh version of Python along with the IDLE IDE is available from the main website, downloadable as either MacBinary or BinHex'd files.

If you are not able to set up the environment properly, then you can take help from your system admin. Make sure the Python environment is properly set up and working perfectly fine.

Note − All the examples given in subsequent chapters are executed with Python 2.4.3 version available on CentOS flavor of Linux.

We already have set up Python Programming environment online, so that you can execute all the available examples online at the same time when you are learning theory. Feel free to modify any example and execute it online.

PANDA

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc. In this tutorial, we will learn the various features of Python Pandas and how to use them in practice.

Audience

This tutorial has been prepared for those who seek to learn the basics and various functions of Pandas. It will be specifically useful for people working with data cleansing and analysis. After completing this tutorial, you will find yourself at a moderate level of expertise from where you can take yourself to higher levels of expertise.

Prerequisites

You should have a basic understanding of Computer Programming terminologies. A basic understanding of any of the programming languages is a plus. Pandas library uses most of the functionalities of NumPy. It is suggested that you go through our tutorial on NumPy before proceeding with this tutorial. You can access it from Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.

In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data.

Key Features of Pandas

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and subsetting of large data sets.

- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

Standard Python distribution doesn't come bundled with Pandas module.

NumPy

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc. An introduction to Matplotlib is also provided. All this is explained with the help of examples for better understanding.

Audience

This tutorial has been prepared for those who want to learn about the basics and various functions of NumPy. It is specifically useful for algorithm developers. After completing this tutorial, you will find yourself at a moderate level of expertise from where you can take yourself to higher levels of expertise.

Prerequisites

You should have a basic understanding of computer programming terminologies. A basic understanding of Python and any of the programming languages is a plus.umPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy

package by incorporating the features of Numarray into Numeric package. There are many contributors to this open source project.

Operations using NumPy

Using NumPy, a developer can perform the following operations −

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

NumPy – A Replacement for MatLab

NumPy is often used along with packages like **SciPy** (Scientific Python) and **Mat−plotlib** (plotting library). This combination is widely used as a replacement for MatLab, a popular platform for technical computing. However, Python alternative to MatLab is now seen as a more modern and complete programming language.

It is open source, which is an added advantage of NumPy.tandard Python distribution doesn't come bundled with NumPy module. A lightweight alternative is to install NumPy using popular Python package installer, **pip**.

```
pip install numpy
```

The best way to enable NumPy is to use an installable binary package specific to your operating system. These binaries contain full SciPy stack (inclusive of NumPy, SciPy, matplotlib, IPython, SymPy and nose packages along with core Python).

CHAPTER-9

9. CONCLUSION AND FUTURE SCOPE

9.1 Conclusion

Intrusion detection is currently attracting interest from both the research community and commercial companies. We have given background of the current state-of-the-art of IDS, based on a proposed taxonomy illustrated with examples of past and current projects. This taxonomy also highlights the recent work and covers the past and current developments adequately. Each of its technique has its own advantages and disadvantages. We believe that no single criterion can be used to completely defend against computer network intrusion. There is no single version of it that can be used as a standard solution against all possible attacks. It is both technically difficult and economically costly to build and maintain computer systems and networks that are not susceptible to attacks. The technique to be selected depends on the specifications of the type of anomalies that the system is supposed to face, the type and behavior of the data, the environment in which the system is working, the cost and computation limitations and the security level required.

9.2 Future Scope

Future research should consider other machine learning algorithms to ascertain more efficient ways to perform the classification technique on the datasets. It is recommended that further research should be carry out on other parameters that can improve the accuracy of detection

‘

CHAPTER-10

10.APPENDIX

10.1 SAMPLE CODING

```
import numpy as np
import pandas as pd

import numpy as np
import pandas as pd
from sklearn import metrics
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow import keras
import distutils
import cv2
from sklearn import metrics
from keras.models import Sequential
from keras.layers import Dense, BatchNormalization, Dropout

train = pd.read_csv('KDDTrain.csv')
test = pd.read_csv('KDDTest.csv')
train.head()

train.loc[train.attack == 'normal', 'is_attacked'] = 0
train.loc[train.attack != 'normal', 'is_attacked'] = 1
test.loc[test.attack == 'normal', 'is_attacked'] = 0
test.loc[test.attack != 'normal', 'is_attacked'] = 1

train_cnt = train.shape
test_cnt = test.shape
```

```
combined_data = pd.concat([train, test])
pd.options.display.max_columns = None
combined_data
for column in combined_data.columns:
    if len(np.unique(combined_data[column].dropna().values))<15:
        print('{}: {}'.format(column,np.unique(combined_data[column].dropna().values)))

from sklearn.preprocessing import OneHotEncoder
ohe = OneHotEncoder(sparse=False) # sparse=False categorical_features=...
new_ohe_features = ohe.fit_transform(combined_data.protocol_type.values.reshape(-1, 1))
tmp = pd.DataFrame(new_ohe_features, columns=['protocol_type=' + str(i) for i in
range(new_ohe_features.shape[1])])
tmp.reset_index(drop=True, inplace=True)
combined_data.reset_index(drop=True, inplace=True)
combined_data_ohe = pd.concat([combined_data, tmp], axis=1)
new_ohe_features = ohe.fit_transform(combined_data.service.values.reshape(-1, 1))
tmp = pd.DataFrame(new_ohe_features, columns=['service=' + str(i) for i in
range(new_ohe_features.shape[1])])
combined_data_ohe = pd.concat([combined_data_ohe, tmp], axis=1)
new_ohe_features = ohe.fit_transform(combined_data.flag.values.reshape(-1, 1))
tmp = pd.DataFrame(new_ohe_features, columns=['flag=' + str(i) for i in
range(new_ohe_features.shape[1])])
combined_data_ohe = pd.concat([combined_data_ohe, tmp], axis=1)
combined_data_ohe

train_ohe = combined_data_ohe[0:train_cnt[0]]
train_ohe

test_ohe = combined_data_ohe[0:test_cnt[0]]
test_ohe
```

10.2 SCREEN SHORTS

	0	tcp	ftp_data	SF	491	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.10	0.11	0.12	0.13	0.14	0.15	0.16	0.17	2	2.1	0.00	0.00.1	0.00.2	0.00.3	1.00	0.00.4	0.00.5
0	0	udp	other	SF	146	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	1	0.0	0.0	0.0	0.0	0.08	0.15	0.00
1	0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	123	6	1.0	1.0	0.0	0.0	0.05	0.07	0.00
2	0	tcp	http	SF	232	8153	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	5	5	0.2	0.2	0.0	0.0	1.00	0.00	0.00
3	0	tcp	http	SF	199	420	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	30	32	0.0	0.0	0.0	0.0	1.00	0.00	0.09
4	0	tcp	private	REJ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	121	19	0.0	0.0	1.0	1.0	0.16	0.06	0.00

150	25	0.17.1	0.03	0.17.2	0.00.6	0.00.7	0.00.8	0.05	0.00.9	normal	20
255	1	0.00	0.60	0.88	0.00	0.00	0.00	0.0	0.00	normal	15
255	26	0.10	0.05	0.00	0.00	1.00	1.00	0.0	0.00	neptune	19
30	255	1.00	0.00	0.03	0.04	0.03	0.01	0.0	0.01	normal	21
255	255	1.00	0.00	0.00	0.00	0.00	0.00	0.0	0.00	normal	21
255	19	0.07	0.07	0.00	0.00	0.00	0.00	1.0	1.00	neptune	21

Train data

```
(125972, 43)
```

Train Data Full shape

```
duration                     125972 non-null int64
protocol_type                125972 non-null object
service                      125972 non-null object
flag                         125972 non-null object
src_bytes                    125972 non-null int64
dst_bytes                    125972 non-null int64
land                         125972 non-null int64
wrong_fragment               125972 non-null int64
urgent                       125972 non-null int64
hot                          125972 non-null int64
num_failed_logins            125972 non-null int64
logged_in                    125972 non-null int64
num_compromised              125972 non-null int64
root_shell                   125972 non-null int64
su_attempted                 125972 non-null int64
num_root                     125972 non-null int64
num_file_creations           125972 non-null int64
num_shells                   125972 non-null int64
num_access_files             125972 non-null int64
num_outbound_cmds            125972 non-null int64
is_host_login                125972 non-null int64
is_guest_login               125972 non-null int64
...
class                        125972 non-null object
classnum                     125972 non-null int64
```

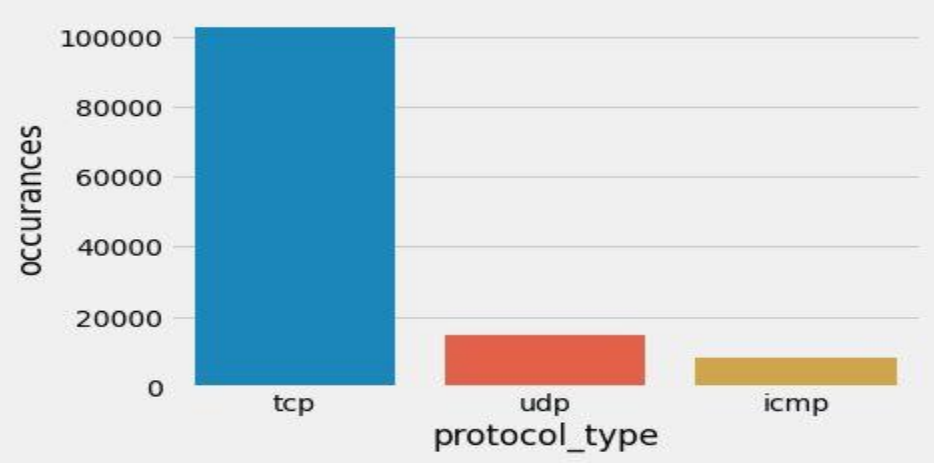
Train Full Info

```
duration                     22543 non-null int64
protocol_type                22543 non-null object
service                      22543 non-null object
flag                         22543 non-null object
src_bytes                    22543 non-null int64
dst_bytes                    22543 non-null int64
land                         22543 non-null int64
wrong_fragment               22543 non-null int64
urgent                       22543 non-null int64
hot                          22543 non-null int64
num_failed_logins            22543 non-null int64
logged_in                    22543 non-null int64
num_compromised              22543 non-null int64
root_shell                   22543 non-null int64
su_attempted                 22543 non-null int64
num_root                     22543 non-null int64
num_file_creations           22543 non-null int64
num_shells                   22543 non-null int64
num_access_files             22543 non-null int64
num_outbound_cmds            22543 non-null int64
is_host_login                22543 non-null int64
is_guest_login               22543 non-null int64
...
class                        22543 non-null object
classnum                     22543 non-null int64
```

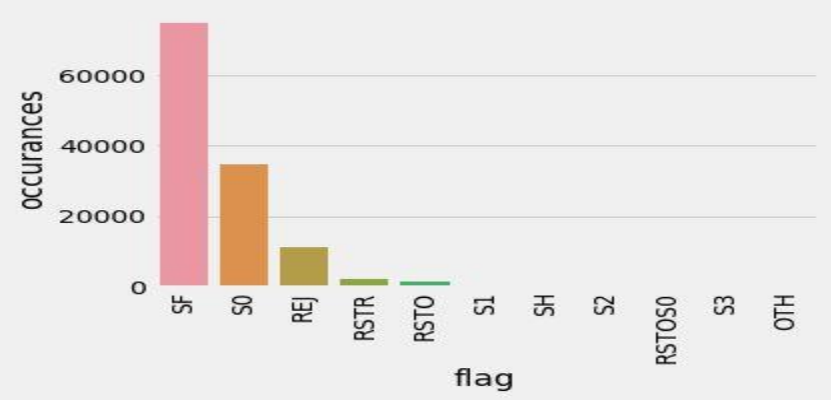
Train data Full Info

```
duration                     22543 non-null int64
protocol_type                22543 non-null object
service                      22543 non-null object
flag                         22543 non-null object
src_bytes                    22543 non-null int64
dst_bytes                    22543 non-null int64
land                         22543 non-null int64
wrong_fragment               22543 non-null int64
urgent                       22543 non-null int64
hot                          22543 non-null int64
num_failed_logins            22543 non-null int64
logged_in                    22543 non-null int64
num_compromised              22543 non-null int64
root_shell                   22543 non-null int64
su_attempted                 22543 non-null int64
num_root                     22543 non-null int64
num_file_creations           22543 non-null int64
num_shells                   22543 non-null int64
num_access_files             22543 non-null int64
num_outbound_cmds            22543 non-null int64
is_host_login                22543 non-null int64
is_guest_login               22543 non-null int64
...
class                        22543 non-null object
classnum                     22543 non-null int64
```

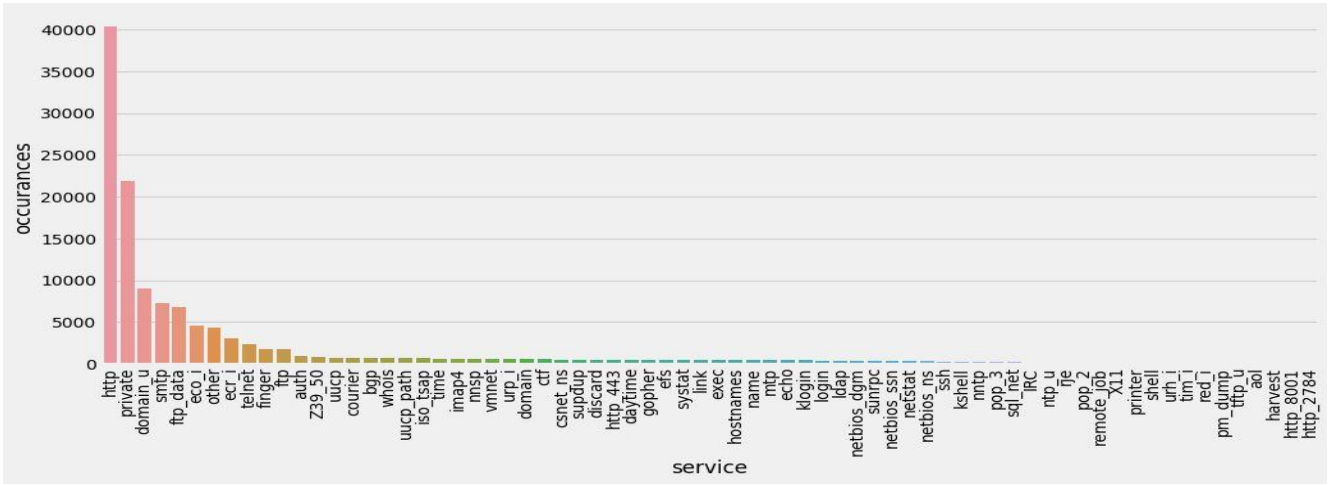
Train data Full unique



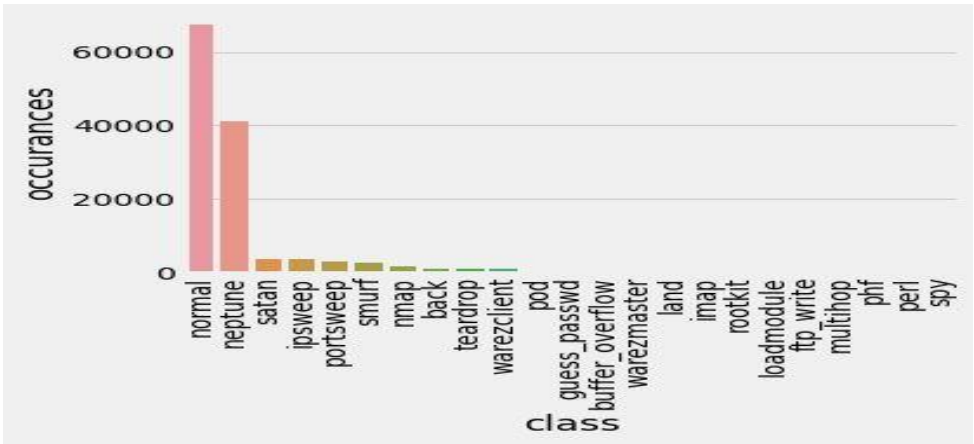
Protocol_Type



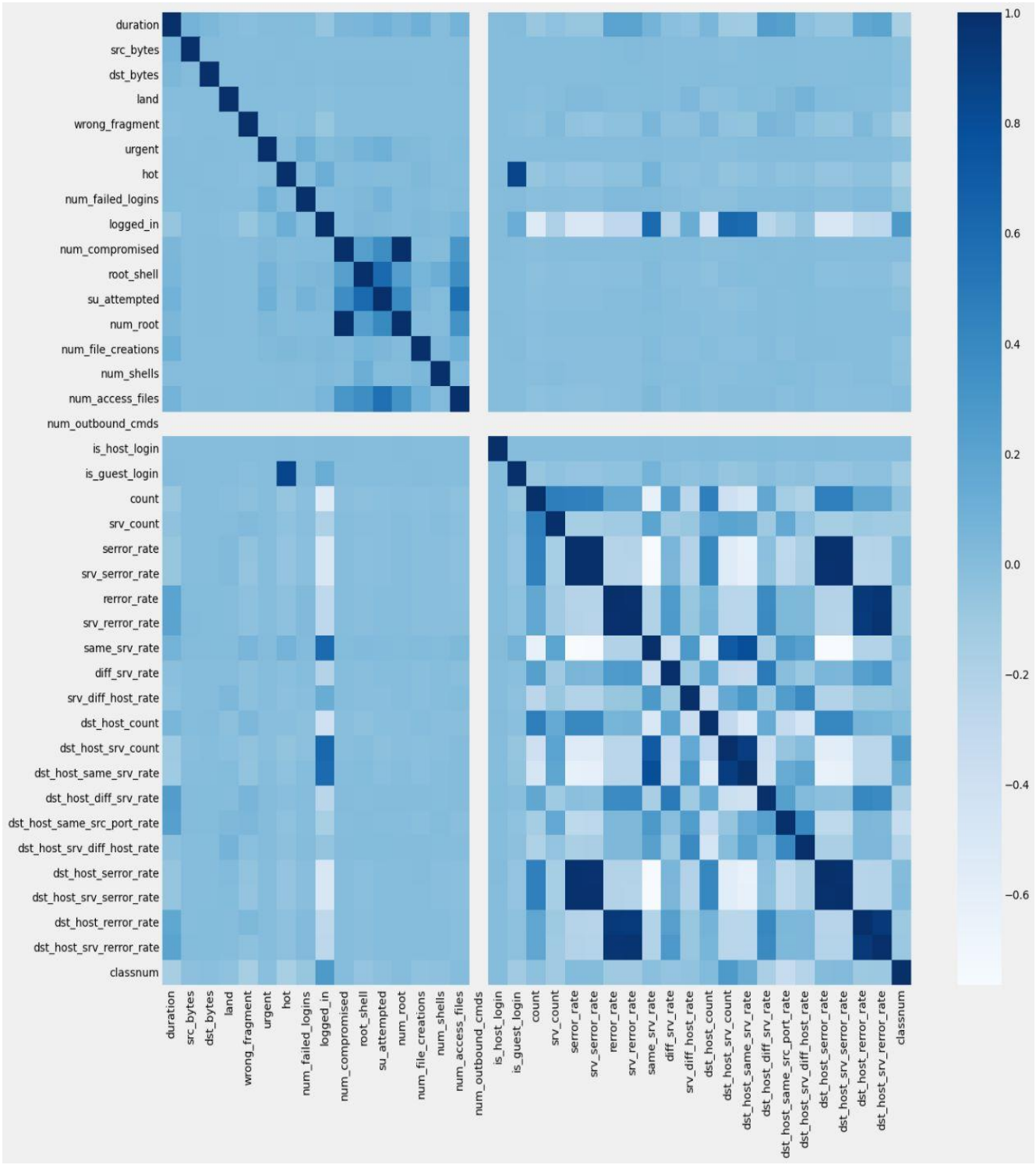
Occurances-Flag



Occurances-Flag



Occurances-Class


duration
src_bytes
dst_bytes
land
wrong_fragment
urgent
hot
num_failed_logins
logged_in
num_compromised
root_shell
su_attempted
num_root
num_file_creations
num_shells
num_access_files
num_outbound_cmds
is_host_login
is_guest_login
count
srv_count
serror_rate
srv_serror_rate
rerror_rate
srv_rerror_rate
same_srv_rate
diff_srv_rate
srv_diff_host_rate
dst_host_count
dst_host_srv_count
dst_host_same_srv_rate
dst_host_diff_srv_rate
dst_host_same_src_port_rate
dst_host_srv_diff_host_rate
dst_host_serror_rate
dst_host_srv_serror_rate
dst_host_rerror_rate
dst_host_srv_rerror_rate
classnum
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

Main Output

CHAPTER-11

11. REFERENCES

[1] I. F. Akyildiz et al., “Wireless Sensor Networks: A Survey, “Elsevier Comp. Networks, vol. 3, no. 2, 2019, pp. 393–422

[2] G.Li, J.He, Y. Fu. “Group-based intrusion detection system in wireless sensor networks” Computer Communications, Volume 31, Issue 18 (December 2019)

[3] Michael Brownfield, “Wireless Sensor Network Denial of Sleep Attack”, Proceedings of the 2019 IEEE Workshop on Information Assurance and Security United States Military Academy, West Point, NY.

[4] FarooqAnjum, DhanantSubhadrabandhu, SaswatiSarkar *, Rahul Shetty, “On Optimal Placement of Intrusion Detection Modules in Sensor Networks”, Proceedings of the First International Conference on Broadband Networks (BROADNETS19).

[5] Parveen Sadotra et al, International Journal of Computer Science and Mobile Computing, Vol.5 Issue.9, September- 2019, pg. 23-28

[6] K. Akkayaand M. Younis, ―A Survey of Routing Protocols in Wireless Sensor Networks, ‖ in the Elsevier Ad Hoc Network Journal, Vol. 3/3 pp. 325-349, 2019.

[7] A. Abduvaliyev, S. Lee, Y.K Lee, “Energy Efficient Hybrid Intrusion Detection System for Wireless Sensor Networks”, IEEE International Conference on Electronics and Information Engineering, Vol.2, pp. 25-29, August 2019.

[8] Parveen Sadotra and Chandrakant Sharma. A Survey: Intelligent Intrusion Detection System in Computer Security. International Journal of Computer Applications 151(3):18-22, October 2019.

[9] A. Araujo, J. Blesa, E. Romero, D. Villanueva, “Security in cognitive wireless sensor networks.

Challenges and open problems", EURASIP Journal on Wireless Communications and Networking, February 2019.

[10] A. Becher, Z. Benenson, and M. Dorsey, \Tampering with motes: Real-world physical attacks on wireless sensor networks." in SPC (J. A. Clark, R. F. Paige, F. Polack, and P. J.Brooke, eds.), vol. 3934 of Lecture Notes in Computer Science, pp. 104{118, Springer, 2019.

[11] I. Krontiris and T. Dimitriou, \A practical authentication scheme for in-network programming in wireless sensor networks," in ACM Workshop on Real- World Wireless Sensor Networks, 2019.

[12] M. Ali Aydın *, A. HalimZaim, K. GokhanCeylan "A hybrid intrusion detection system design for computer network security" Computers and Electrical Engineering 35 (2019) 517–526.