

# **THREAT MONITORING USING MACHINE LEARNING**

Submitted in partial fulfillment of the requirements for the award of  
Bachelor of Engineering degree in Computer Science and Engineering

By:

**VEERAMALLI BHANUKIRAN(Reg.No-39111069)**  
**YENUMULA ROHITHKUMAR(Reg.No-39111128)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**SCHOOL OF COMPUTING**

# **SATHYABAMA**

**INSTITUTE OF SCIENCE AND TECHNOLOGY  
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE  
JEPPIAAR NAGAR, RAJIV GANDHISALAI,  
CHENNAI - 600119**

**APRIL - 2023**



# SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY  
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

[www.sathyabama.ac.in](http://www.sathyabama.ac.in)

## DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

### BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Veeramalli Bhanukiran(39111069)** and **Yenumula Rohithkumar(39111128)** who carried out the project entitled "**THREAT MONITORING USING MACHINE LEARNING**" under my supervision from January 2023 to April 2023.

Internal Guide

**Dr. A. C. SANTHA SHEELA.M.E., Ph.D.,**

Head of the Department

**Dr. L. LAKSHMANAN, M.E., Ph.D.,**



Submitted for Viva-voice Examination held on \_\_\_\_\_

Internal Examiner

External Examiner

## DECLARATION

“

I, **Veeramalli Bhanukiran(39111069)**, hereby declare that the project report entitled **”THREAT MONITORING USING MACHINE LEARNING”** done by me under the guidance of **Dr. A. C. SANTHA SHEELA.M.E.,Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in **Computer Science and Engineering**.

**DATE:**

**V Bhanu Kiran**

**PLACE: Chennai**

**SIGNATURE OF CANDIDATE**

## ACKNOWLEDGMENT

I am pleased to acknowledge my sincere thanks to the **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computing, **Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department of Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.A.C. SANTHA SHEELA .M.E.,Ph.D.**, for his valuable guidance, suggestions, and constant encouragement that paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project

## ABSTRACT

In the present-day scenario cybercrimes and data breaches have become increasingly prevalent and are causing huge losses and hazards to companies and netizens. For a recent example, we have seen how a 17year old allegedly hacked the social media accounts of famous personalities like Jeff Bezos, and Barack Obama. Late detection of such attacks increase the possibility of irreparable damage. Such attacks are causing high financial losses to the organizations , and also affecting the lives of individuals. Recent studies show that cybercrime results in a loss of over US\$6 trillion by 2021 and the size, complexity, frequency of these attacks are growing exponentially. Also considering the pandemic worldwide, the data being generated is increasing rapidly along with the equal chances of breaches. So, network security is one of the very essential features that organizations and individuals should consider. Though there are firewalls and Intrusion Detection Systems(IDS) to detect harmful packets we still see that they fail at times and cause severe damages. Building an Intelligent Intrusion Detection System(IIDS) using past data of attacks and ML algorithms to differentiate a legitimate packet from a harmful one helps us in mitigating threats. In this project, we are building an ML model that classifies any network packet into two categories legitimate and malicious once deployed in network devices forwards only legitimate packets to the end system. To build the model we are us the NSL\_KDD data set and implementing “Random forest” and “XGBoost” algorithms for the classification purpose. The data set is built considering various fields in the packet header.

## TABLE OF CONTENTS

Chapter No	TITLE	Page No.
	<b>ABSTRACT</b>	v
	<b>LIST OF ABBRIVATIONS</b>	viii
1	<b>INTRODUCTION</b>	1
2	<b>LITERATURE SURVEY</b>	4
	2.1 Inferences from Literature Survey	8-11
	2.2 Open problems in Existing System	12-13
3	<b>REQUIREMENTS ANALYSIS</b>	14
	3.1 Feasibility Studies/Risk Analysis of the Project	14-15
	3.2 Software Requirements Specification Document	16
4	<b>DESCRIPTION OF PROPOSED SYSTEM</b>	17
	4.1 Selected Methodology or process model	20
	4.2 Architecture / Overall Design of Proposed System	21
	4.3 Description of Software for Implementation	22-23
	4.4 Project Management Plan	24-25
	4.5 Transition/ Software to Operations Plan	26-27
5	<b>IMPLEMENTATION DETAILS</b>	28-39
	5.1 Development and Deployment Setup	32
	5.2 Algorithms	33-38
	5.3 Testing	39
6	<b>RESULTS AND DISCUSSION</b>	40
7	<b>CONCLUSION</b>	41
	7.1 Conclusion	41

7.2	Future work	42-43
7.3	Research Issues	44-45
7.4	Implementation Issues	46
<b>REFERENCES</b>		47
<b>APPENDIX</b>		49-65
<b>A.SOURCE CODE</b>		49-53
<b>B.SCREENSHOTS</b>		54-59
<b>C.RESEARCH PAPER</b>		60-65

## LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
IP	Internet Protocol



# **CHAPTER 1**

## **INTRODUCTION**

Cybercrime is one of the key challenges that's been raised due to exponential growth in the usage of internet across the globe. One of the research studies made by a cybersecurity venture states that the damage due to these invasions is close to US 6 trillion dollars. Also the tremendous growth in fields such as IOT and BigData has become a significant reason for more challenges in the domain of cybersecurity. The long time in detection of these is due to human intervention which significantly affects the efficiency of dealing with threats. Though there are firewalls and intrusion detection systems (IDS) existing recent attacks prove that changes have to be brought in security along with changing trends of attacks.

In this paper the authors have tried to bring in better security solutions where they have developed a tool that performs real time threat detection using machine learning algorithms. This work is an advancement over the previous foundation established by a group of scholars at UFRJ which was called as CATRACA. This focuses on scalability, performance and correctness to identify diversity of new attacks. The novelty of the paper lies in using distributed processing completely using Scala and the Resilient Distributed Datasets, Data Frame and Dataset data structures available in Apache spark. The tool proposed works in two different modes online and offline. The major problem lies in nonavailability of public datasets for network based attacks, of the existing ones the NSL\_KDD dataset was chosen as it is the most efficient one with no redundant data. The dataset consists of over 40 attributes which are derived from TCP/IP packet header. The dataset has attributes such as protocol type, flag, source bytes, logged in etc and all the different 24 types of attacks are broadly classified into 4 which are Dos, Probe, U2R, R2L. The packet is classified as normal or anomaly based on its attributes. Various machine learning algorithms were used to build and train the model. The classification algorithms that were used include Decision Tree, Support Vector Machine, Gradient Boost, Logistic Regression,

Random Forest. The model which results in best performance is taken considering various performance measures evaluated such as F1\_score, accuracy\_score, recall\_score and precision. The architecture considered works in 2 different modes which are online and offline. The online mode does the classification of packets in real time whereas offline mode is used to take a note of performance of multiple classifiers subjected to be implemented on the dataset. The three main modules of the architecture includes data collection, processing and visualization. The famous big data handling tool Hadoop Distributed File System(HDFS) was used to collect and retain the data which is used to train the model and then perform the testing part. Processing is carried using Apache Spark cluster as it processes the data at much faster rates in real time. The visualization module is implemented using Elasticsearch and Kibana softwares. The large volume of data to be analyzed in real-time also increases the complexity of classifying network traffic and detecting threats. Finally, the average time to detect an attack is a crucial factor in the impact of cyber threats. More than a quarter of cyberattacks take a long time before being discovered, with this time often ranging from weeks to months. The late detection of an attack exponentially increases the risk of financial losses and the risk of irreparable damage. The long time is due to the need for human intervention in these situations, significantly affecting the efficiency of dealing with threats. In this scenario in which security is a fundamental aspect, the need for systems capable of guaranteeing safe and reliable network use is increasing. Solutions based on Security Information and Event Management (SIEM) tools partially mitigate the problem by providing real-time network monitoring. This type of solution, however, is still highly dependent on the intervention of experts and is based on threat signature databases, therefore being inefficient in the detection of new attacks. Using machine learning algorithms for threat detection, on the other hand, automates the detection process and meets the required agility to prevent and mitigate network attacks. It is of utmost importance to select algorithms that perform well in the classification process, without harming accuracy and other evaluation metrics. Previously, our research group (GTA/UFRJ) proposed CATRACA [7], a tool that uses machine learning to detect threats in real-time. Intelligent Monitoring and Analysis of Network Traffic Threats, an intelligent

threat monitoring and detection system based on machine learning and distributed processing in clusters. TeMIA-NT proposes and develops an entirely new distributed processing system with significant improvements in machine learning processing optimization. Our proposal focuses on the intelligence, scalability, and performance required to process large volumes of data while optimizing multiple machine learning algorithms to meet the diversity of new attacks. To increase performance, TeMIA-NT implements distributed processing entirely in Scala language and uses dataframe structures, instead of the standard Resilient Distributed Datasets (RDD) structure on the open- source Apache Spark platform. TeMIA-NT offers many options for machine learning algorithms and the possibility of optimizing hyper parameters, allowing testing, selecting, and adjusting the parameters of the best algorithm for each type of scenario. We implement the offline threat detection using the structured streaming library, which allows flow processing in micro-batches, with fault tolerance and reduced intervals. Online threat detection uses the continuous processing mode, which enables our proposal to perform similar to a native stream processing tool.

The rest of the article is organized as follows. Section II presents papers with themes related to the article. Section III introduces the Apache Spark platform, as well as its data structures and its machine learning library. Section IV presents the network traffic dataset used to test the proposed tool, while Section V presents the tool's architecture and features. Section VI presents and analyzes the performance tests

## **CHAPTER-2**

### **LITRATURE SURVEY**

This section presents the set of inference strategies obtained from the SLR. The resulting set is which includes an index column, the name of the inference strategy category and the sources describing the inference strategy the processes associated with the inference strategies, and an example inference strategy for each category. The inference strategies have been divided into the following The literature was searched through Web of Science (WoS) by topic from 1950 to 2015 with the key words of “insider threat”, “insider threat detection”, and “insider threat prediction” We also searched Google Scholar (GS) database using “insider threat detection and prediction” as search terms. We screened the first 100 search records within GS. Additionally, we searched the reference lists of all potentially relevant articles and book chapters. We removed duplicate papers. The abstracts of the identified articles were scrutinised for relevance based on the following questions—if the answers to all of these questions are ‘Yes’, then we shortlisted the article for further consideration. Only a small percentage of studies use original real-world data – in our online systems, and social media websites. Obtaining security data for research purposes is difficult, if not impossible, due to financial, business and national security concerns. Furthermore, even if one acquires real data, data on those conducting insider attacks are not publicly available for privacy reasons . Often, insider attacks are not even caught as attacker leaves no traces. Consequently, research with real-world data proves challenging. For example, Kandias et al. have conducted a content analysis of user comments on YouTube videos looking for any negative comments on law enforcement. Theoretically, these negative comments posted by employees are likely to reflect their intent to commit malicious acts. However, there is no survey 4 out of 16 studies . The major sources of real data are the log files of the ground-truth whether the individual was involved in any malicious activity.

## **Practical Threat Intelligence and Monitoring**

This is an introduction for those who don't know much about cyber threat intelligence (CTI) and the TH world and a guide for those with more advanced knowledge of other cybersecurity fields looking to implement a TH program from scratch. You will start by exploring what threat intelligence is and how it can detect and prevent cyber threats. As you progress, you'll learn how to collect data, along with understanding it by developing data models. The book will also show you how to use open-source tools to set up an environment for TH. Later, you will focus on how to plan a hunt with practical examples before exploring the MITRE ATT&CK framework.

## **Cyber Security Threat**

Nadean Tanner's vast array of experience from teaching at a University to working for the Department of Defense, the Cybersecurity Blue Team Toolkit strikes the perfect balance of substantive and accessible, making it equally valuable for those in IT or management positions across a variety of industries. This handy guide takes a strategic and straightforward look at best practices and tools available to cybersecurity management and hands-on professionals, whether they are new to the field or looking to expand their expertise.

## **Threat Hunting (V1.02): A Condensed Guide for the Security Operations Team and Threat Hunter**

Don Murdoch has implemented five powerful platforms, integrated over one hundred data sources into various platforms, and ran an MSSP practice for two years. This book covers the topics below using a "zero fluff" approach as if you hired him as a security consultant and were sitting across the table with him (or her). The book begins with a discussion for professionals to help them build a successful business case and a project plan, decide on SOC tier models, anticipate and answer tough questions you need to consider when proposing a SOC, and considerations in building a logging infrastructure. The book goes through numerous data sources that feed a SOC and SIEM and provides specific real-world guidance on using those data sources to the best possible effect.

## **Malware Analysis and Detection Engineering: A Comprehensive Approach to Detect and Analyze Modern Malware**

Malware analysis and reverse engineers to provide insight into the different types of malware and the terminology used in the anti-malware industry. You will know how to set up an isolated lab environment to execute and analyze malware safely. You will learn about malware packing, code injection, and process hollowing, plus how to analyze, reverse, classify, and categorize malware using static and dynamic tools. You will be able to automate your malware analysis process by exploring detection tools to modify and trace malware programs, including sandboxes, IDS/IPS, anti-virus, and Windows binary instrumentation.

## **Practice of Network Security Monitoring**

Network security is not simply about building impenetrable walls—determined attackers will eventually overcome traditional defenses. The most effective computer security strategies integrate network security monitoring (NSM): the collection and analysis of data to help you detect and respond to intrusions...In The Practice of Network Security Monitoring, Mandiant CSO Richard Bejtlich shows you how to use NSM to add a robust layer of protection around your networks—no prior experience required. The dataset has attributes such as protocol type, flag, source bytes, logged in etc and all the different 24 types of attacks are broadly classified into 4 which are Dos, Probe, U2R, R2L. The packet is classified as normal or anomaly based on its attributes. Various machine learning algorithms were used to build and train the model. The classification algorithms that were used include Decision Tree, Support Vector Machine, Gradient Boost, Logistic Regression, Random Forest. The model which results in best performance is taken considering various performance measures evaluated such as F1\_score, accuracy\_score, recall\_score and precision. The architecture considered works in 2 different modes which are online and offline. The online mode does the classification of packets in real time whereas offline mode is used to take a note of performance of multiple classifiers subjected to be implemented on the dataset. The three main modules of the architecture includes

data collection, processing and visualization. The famous big data handling tool Hadoop Distributed File System(HDFS) was used to collect and retain the data which is used to train the model and then perform the testing part. Processing is carried using Apache Spark cluster as it processes the data at much faster rates in real time. The visualization module is implemented using Elastic search and Kibana softwares.

## **2.1. Inferences from Literature Survey**

This section presents the set of inference strategies obtained from the SLR. The resulting set is which includes an index column, the name of the inference strategy category and the sources describing the inference strategy the processes associated with the inference strategies, and an example inference strategy for each category. The inference strategies have been divided into the following

The literature was searched through Web of Science (WoS) by topic from 1950 to 2015 with the key words of “insider threat”, “insider threat detection”, and “insider threat prediction” We also searched Google Scholar (GS) database using “insider threat detection and prediction” as search terms. We screened the first 100 search records within GS. Additionally, we searched the reference lists of all potentially relevant articles and book chapters. We removed duplicate papers. The abstracts of the identified articles were scrutinised for relevance based on the following questions—if the answers to all of these questions are ‘Yes’, then we shortlisted the article for further consideration. Only a small percentage of studies use original real-world data – in our online systems, and social media websites. Obtaining security data for research purposes is difficult, if not impossible, due to financial, business and national security concerns. Furthermore, even if one acquires real data, data on those conducting insider attacks are not publicly available for privacy reasons . Often, insider attacks are not even caught as attacker leaves no traces. Consequently, research with real-world data proves challenging. For example, Kandias et al. have conducted a content analysis of user comments on YouTube videos looking for any negative comments on law enforcement. Theoretically, these negative comments posted by employees are likely to reflect their intent to commit malicious acts. However, there is no survey 4 out of 16 studies . The major sources of real data are the log files of the ground-truth whether the individual was involved in any malicious activity. New challenges in the intrusion detection area arise due to the high volume of traffic, a large number of IoT devices, distributed denial of service attacks, and zero-day attacks. To meet these challenges, the use of machine learning techniques to classify flows in real-time became popular. The classification of large volumes of data



at high speeds available employs three main distributed processing platforms: Apache Spark, Apache Storm, and Apache Flink. The fundamental difference between the platforms is that Spark performs batch processing while the Storm and Flink platforms perform native flow processing. The Open Security Operations Center (OpenSOC) [14] is an analytical security framework for monitoring large amounts of data. OpenSOC originated a new project, Apache Metron [15], that is a tool that comprises the acquisition of different types of data, distributed processing, enrichment, storage, and visualization of results. Metron allows the correlation of security events from various sources, such as logs of applications and network packages. For this purpose, the framework uses distributed data sources, such as sensors on the network, logs of security element events, and enriched data called telemetry sources. The tool also provides a historical base of Cisco network threats. Based on the Apache Spark Platform, there are the Apache Spot, Stream4Flow, and Hogzilla. Apache Spot is a project still in the incubation stage that uses telemetry and machine learning techniques for analyzing packages to detect threats. The Stream4Flow prototype uses the Elastic stack to view network parameters, however, it lacks the intelligence to perform anomaly detection. The Hogzilla tool<sup>1</sup> provides support for Snort, S Flows, Gray Log, Apache Spark, H Base, and libn DPI, offering network anomaly detection. Hogzilla also allows visualizing network traffic, using Snort to capture packets, and obtaining features through deep packet inspection. Stream4Flow captures packets using IPFIXcol and only considers header information. In our work, we use the flowtbag2 software, which captures various flow statistics. In addition, our offline processing mode allows updates to the machine learning model, further promoting the detection of new threats. We select the Apache Spark platform to develop the TeMIA- NT because it is the most adopted among the examined Big Data processing platforms. Spark offers more possibilities for machine learning algorithms and is the one with the largest active community. Nevertheless, to the best of our knowledge, TeMIA-NT is the only available system to use the recent structured streaming technology in batch and continuous modes in Apache Spark, allowing the selection among several machine learning algorithms, and operating in offline and online modes. Despite the advances made in threat monitoring systems, there are still

several open problems in the existing system that need to be addressed. Some of the open problems are

One of the biggest challenges in threat monitoring systems is the high rate of false positives and false negatives. False positives can lead to wasted resources and unnecessary alerts, while false negatives can result in missed threats. The accuracy and completeness of the data used in threat monitoring systems can impact the effectiveness of the system. Ensuring the quality of the data and identifying gaps in the data is a critical challenge. Many existing threat monitoring systems struggle to detect threats in real-time, especially when dealing with large and complex data sets. Identifying anomalies and patterns in the data is crucial for effective threat monitoring. However, existing anomaly detection algorithms are still limited in their ability to detect subtle anomalies and may generate false positives. Integrating threat monitoring systems with other security tools such as firewalls and intrusion prevention systems can be challenging, especially when dealing with different data formats and protocols. Threat actors are becoming more sophisticated in their attacks, and they are finding ways to evade detection by threat monitoring systems. Addressing adversarial attacks is a critical challenge in threat monitoring. As organizations grow, their data volumes increase, making it challenging to scale threat monitoring systems to keep pace with the growth. Data privacy and compliance regulations impose additional challenges in threat monitoring systems, such as data retention and data protection. Addressing these open problems is critical for improving the effectiveness of threat monitoring systems and providing better protection against cybersecurity threats. Researchers and practitioners must continue to work on developing innovative solutions to these challenges. A literature survey on threat monitoring provides insights into the current state of the art, research gaps, and emerging trends in the Threat monitoring is a critical component of a comprehensive cybersecurity strategy, and the demand for threat monitoring solutions is increasing due to the growing number and sophistication of cybersecurity threats. Machine learning techniques, such as deep learning and anomaly detection, are widely used in threat monitoring systems to detect and respond to threats. However, there are still challenges in developing accurate and effective machine learning models that can detect new and emerging threats. Threat intelligence is an essential component of

threat monitoring systems, and integrating external threat intelligence feeds with internal security data can enhance the effectiveness of the system. Real-time threat detection and response are critical for reducing the time to detect and respond to threats. Emerging trends such as automated threat hunting and security orchestration, automation, and response (SOAR) are focused on improving real-time threat detection and response. Data quality and data integration are crucial for effective threat monitoring. Many organizations struggle with data quality issues, and integrating data from disparate sources can be challenging. Privacy and compliance are important considerations in threat monitoring, especially in light of increasing data protection regulations such as GDPR and CCPA. Threat monitoring is a rapidly evolving field, and there is a need for continued research and development to address emerging threats and challenges. Overall, the literature survey highlights the importance of threat monitoring in ensuring the security of organizations' digital assets and the need for continued innovation and research to address the challenges and emerging threats in this field.

## **2.2 Open problems in Existing System**

### **Modernization of Applications**

Finding hidden threats has grown exponentially more difficult due to the increasing complexity involved with monitoring modern applications (cloud, containers, serverless and composable infrastructure) and the associated cloud-scale volume of data that needs to be processed and correlated.

### **Fluid Threat Environment**

The attack surface that cyber criminals can target has expanded significantly with cloud, mobile, IOT / OT / medical IOT, and work-from-anywhere environments. It is increasingly difficult to defend against threat actors that can quickly launch automated, complex, multi stage attacks. Their weapons are more sophisticated and effective, and they increasingly leverage machine learning (ML) and artificial intelligence (AI) to achieve their objectives.

### **Increasing Security Complexity**

Fully leveraging the data generated from numerous and disparate security solutions that have already been deployed, and correlating and monitoring this data 24/7 to find indicators of an attack requires specialized tools that are expensive and difficult to integrate and operate effectively.

Overcoming these challenges requires a range of specialized security skills that are difficult to find and expensive to hire, manage and retain. All of these factors have created the need for a modern, more effective approach to threat detection and response. Vendors are overpromising and underdelivering. When it comes to threat detection, too many vendors falsely claim or exaggerate that they have machine learning (ML), artificial intelligence (AI), multicloud support, and/or apply risk metrics. CISOs are barraged with vendors claiming to offer a silver bullet at worst or using questionable marketing claims at best. Neither delivers what's promised. Automation can drive efficiency and speed threat detection. This can free up security team members to focus their attention on more intensive tasks. When done effectively, this

provides OPEX savings – which means less time and resources spent on simple and manual tasks of low value, while also shrinking the time for high-value tasks. It can also provide better experience for junior analysts, especially when your analytics and automation are transparent, allowing them to learn and improve.

Viruses and worms are malicious software programs (malware) aimed at destroying an organization's systems, data and network. A computer virus is a malicious code that replicates by copying itself to another program, system or host file. It remains dormant until someone knowingly or inadvertently activates it, spreading the infection without the knowledge or permission of a user or system administration. A computer worm is a self-replicating program that doesn't have to copy itself to a host program or require human interaction to spread. Its main function is to infect other computers while remaining active on the infected system. Worms often spread using parts of an operating system that are automatic and invisible to the user. Once a worm enters a system, it immediately starts replicating itself, infecting computers and networks that aren't adequately protected.

Phishing attacks are a type of information security threat that employs trick engineering to social users into breaking normal security practices and giving up confidential information, including names, addresses, login credentials, Social Security numbers, credit card information and other financial information. In most cases, hackers send out fake emails that look as if they're coming from legitimate sources, such as financial institutions, eBay, PayPal -- and even friends and colleagues. In phishing attacks, hackers attempt to get users to take some recommended action, such as clicking on links in emails that take them to fraudulent websites that ask for personal information or install malware on their devices. Opening attachments in emails can also install malware on users' devices that are designed to harvest sensitive information, send out emails to their contacts or provide remote access to their devices.

## **CHAPTER-3**

### **REQUIREMENT ANALYSIS**

#### **3.1 Feasibility Studies/Risk Analysis of the Project**

To motivate more effective in-process cyber-attack detection techniques, this section summarizes the following open issues associated with capturing execution behaviors from web-based applications via program transformations and detecting potential cyber-attacks at runtime using machine learning. Capturing features representative of application behavior without creating a significant burden on developers: To augment human cybersecurity experts, automated mechanisms are needed to capture and analyze application execution information. A promising approach uses machine learning to build models of expected application execution behavior. We define execution behavior as the invocation of methods, the ordering of method invocation, and the inputs/outputs of method invocations. Key open issues facing researchers are how to instrument an application unobtrusively and (2) how much knowledge of the underlying code base and access to the code is required to collect execution features and train accurate machine learning attack detection models. Determining the performance overhead of execution feature vector collection.: Monitoring instructions executed along the critical path of a program can degrade its performance. To evaluate this degradation, Section V-A describes the results of experiments that evaluated the average and worst-case performance overhead of RSMT program transformation in web-based applications. In particular, we identified trade-offs between lightweight online detection techniques that identify obviously dangerous behaviors versus fine-grained offline detection. Feasibility studies and risk analysis are important steps in evaluating the viability of a project like threat monitoring. Here are some potential feasibility studies and risk analysis considerations for a project on threat monitoring

Assess the technical feasibility of implementing the proposed system for threat monitoring. This includes evaluating the availability of suitable technology, expertise, and infrastructure required for the system. Conduct a cost-benefit analysis to evaluate the economic feasibility of the project. This includes estimating the costs associated with hardware, software, personnel, and maintenance and comparing them to the benefits, such as improved security and reduced risks. Evaluate the operational feasibility of the project by assessing the impact on existing processes and systems. This includes analyzing how the proposed system will integrate with existing security controls and processes. Evaluate the legal and regulatory feasibility of the project by identifying any legal or regulatory requirements that must be met. This includes assessing the impact of data protection laws, privacy regulations, and other legal and regulatory requirements.

Evaluate the potential security risks associated with implementing the threat monitoring system. This includes assessing the risk of data breaches, system failures, unauthorized access, and other security threats. Assess the potential operational risks associated with the threat monitoring system. This includes evaluating the risk of false positives and false negatives, as well as the risk of system downtime or failure. Evaluate the potential compliance risks associated with implementing the threat monitoring system. This includes assessing the risk of non-compliance with legal and regulatory requirements such as GDPR, CCPA, and HIPAA. Assess the potential financial risks associated with the threat monitoring system. This includes evaluating the risk of cost overruns, budget constraints, and the potential impact of the system on the organization's financial performance. Assess the potential reputational risks associated with the threat monitoring system. This includes evaluating the risk of negative publicity, loss of customer trust, and damage to the organization's reputation. By conducting feasibility studies and risk analysis, an organization can identify potential risks and challenges associated with implementing a threat monitoring system and take steps to mitigate them. This can help ensure the success of the project and minimize the potential negative impact on the organization.

## **3.2 Software Requirements Specification Document**

### **HARDWARE REQUIREMENTS**

System : Pentium Dual Core.  
Hard Disk : 120 GB.  
Monitor : 15"LED  
Input Devices : Keyboard, Mouse  
Ram : 1GB.

### **SOFTWARE REQUIREMENTS**

Operating system : Windows 7.  
Coding Language : python  
Toolkit : Jupitor Notebook  
DATABASE : EXCEL



## CHAPTER-4

### DESCRIPTION OF PROPOSED SYSTEM

Looking at the disadvantages of all the above methodologies used in previous systems statement using only a predefined data set of faces with closed eyes and opened eyes. Using threat monitoring enables organizations to identify previously undetected threats such as outsiders connecting to or exploring networks and compromised or unauthorized intern accounts. It can be difficult to detect these activities otherwise, but threat monitoring solutions correlate information about network and endpoint activity with contextual factors such as IP addresses , URLs, and file and application details to provide more accurate identification of anomalies indicative of threat activity have full visibility into data access and usage an can enforce data protection policies to prevent or SQL injection.In addition,remote services like SSH (Secure Shell), RDP (Remote Desktop Protocol), SNMP (Simple Network Management Protocol), and SMB (Server Message Block) can be leveraged to gain unauthorized remote access sensitive data loss.Specifically,threat monitoring brings several benefits by helping security professionals.

Threat monitoring reduces insider threat risks and maximizes data protection capabilities. Organizations are in a better position to defend against insider and outsider threats when they. After the development phase, threat modeling is still an important activity.Let's take an example of the initial access tactic from the MITRE ATT&CK framework, which addresses methods attackers use to gain access to a target network or systems. Customers may have internet-facing web applications or servers hosted in AWS cloud, which may be vulnerable to attacks like DDOS (Distributed Denial of Service), XSS (Cross-Site Scripting).

The proposed tool has two operation modes: online and offline. The online mode performs classification in real time, whilst the offline mode allows to observe the performance of multiple classifiers for a given dataset, making the resulting metrics

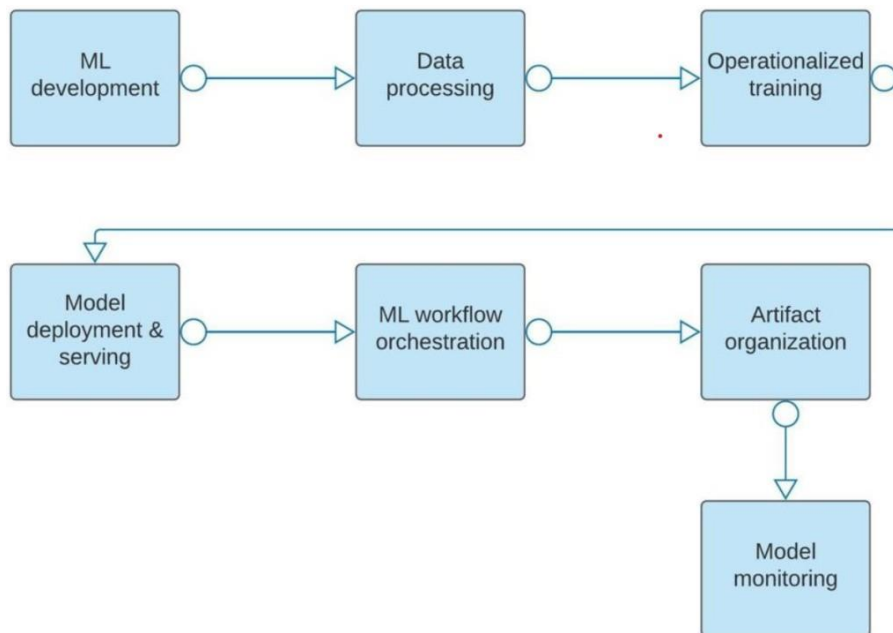
available in the visualization module. The proposed architecture, shown in Figure 1, is modular and consists of three main modules: data collection, processing and visualization. The data collection module captures and abstracts network traffic flows. It also stores the datasets used in offline processing. The capture process reflects network traffic through the libpcap library. Then, the flowtag tool abstracts the sequence of packets in the flows and their 40 features, including the flows length and the total number of packages for each flow. We use the five fields of the TCP/IP packet header, source IP address, destination IP address, source port, destination port, and protocol to abstract packets into flows. A channel on the Apache Kafka platform, which acts as a data buffer, receives the streams of data. We use the Hadoop Distributed File System (HDFS), a distributed database, to store the datasets used to train and test the classification models. The processing module carries out the process of classifying these flows. The processing module is implemented in an Apache Spark cluster. This platform presents advantages to the development of the tool, as it has libraries aimed at the implementation of machine learning algorithms and the fast processing of data in real-time, using the micro-batch method with the structured streaming engine. The training module extracts the classification model using a dataset labeled from HDFS. In the online mode of operation, packages are collected and added to an Apache Kafka channel and flows are then classified as legitimate or malicious by the classification model obtained previously. To allow flows to be analyzed and classified in real-time in the online processing mode, we developed a classification module responsible for collecting this data as they are added to a certain Apache Kafka channel. The flows are then classified as legitimate or malicious by the classification model obtained previously. For execution in the offline mode, the classification module runs tests on various algorithms and datasets, obtaining performance metrics for each combination. The results of the classification, both online and offline, are then sent to an Elasticsearch server using the Apache Spark integration library. The visualization module allows the network administrator to visualize the classifications history and the current state of the network, as well as the results of tested algorithms. We implement the visualization module using the Elasticsearch3 and Kibana software, both developed by Elastic. Elasticsearch

implements a distributed and efficient search server, based on JSON documents. It receives and stores the data as the processing module sends it after the classification process is finished. Kibana is responsible for providing a user interface through dashboards, displaying to the network administrator the data received by Elasticsearch in real-time for both execution modes. It also allows consultation by historical data, using the search server features Elastic search.

## **4.1 Selected Methodology Or Process Model**

Business Assets Data, various components, and functions (applications) that are necessary for the continued operations of your business. Those targeting business assets can be better seen as malicious actors committing sabotage intended to disrupt continued activity. Data Assets These are data, components, and functions of particular use to the hacker, who can gain access to certain functions to perform further reprehensible deeds. For example, cyber criminals may exploit data assets to help their crypto-mining operations, or look for It's a myth that today's hackers perform custom attacks on their victims. Rather than the ultra-competent cyber sleuths we see in the media, they're instead opportunists who look for the most direct entry points and exploit already-known vulnerabilities. This makes up your attack surface – essentially the totality of your exposed components that may connect a bad actor to one to your assets. Within threat modeling, teams outline all elements of the attack surface, and demonstrates how data flows to and from these components.

## 4.2 Architecture/Overall Design Of Proposed System



**Fig 4.1: System Architecture**

#### **4.3 Description of Software for Implementation and Testing plan of the Proposed Model/System**

The purpose of this project is to incorporate a mechanism that labels the severity of cyber security threats in a comprehensive framework that addresses cyber crime incidents. This formal procedure aims to assess cyber crime incidents serving as essential input to an expert system that triggers the assignment of countermeasures to the respective stakeholders. The employed methodology involved consecutive steps for a qualitative analysis of cybersecurity threats, deriving from annual reports on cybercrime incident frequency and evaluates their threat severity based on time progression. The authors evaluated a series of certified sources from ENISA, the European Union Agency for Network and Information Security, to prioritize threats by means of frequency combination of: (appearance/reference, and number of incidents. Research findings highlight a clustering of severity regarding the top 15 threats to cyber-security in the last years. A set of highly trending threats are continually labeled as hyper critical threats and/or maintain an exponential rate, while other threats appear to be less severe as they present declining rates in terms of incident frequency. The severity labeling system generates assessment information that can be combined with incident characteristics and features to direct an expert system to efficient distribution of both general and customized preventive measures. The necessity of an inclusive approach towards cybercrime can be actualized through different steps of the framework in prospect, each one designated to a different perspective. The main contribution of the current paper lies in the resolution of a cybercrime incident depending on its severity and frequency and the sure assigned consolidation with the next step for automated. To implement and test the proposed model/system for threat monitoring, a software program will be needed. The software program will need to perform several functions, including data

collection, data processing, alerting, and response. The software program will need to collect data from various sources such as network devices, servers, applications, and other systems. The program should be able to handle different data formats and protocols and store the data securely. The software program will need to process the collected data to detect potential threats. This can be done using machine learning algorithms, anomaly detection techniques, or other methods. The program should be able to analyze the data in real-time and provide alerts when a potential threat is detected. When a potential threat is detected, the software program should generate an alert. The alert should be sent to the appropriate personnel via email, SMS, or other means. The program should also include a dashboard that displays the alerts and provides additional information about the potential threat. The software program should be able to initiate a response when a potential threat is detected. This can include blocking network traffic, shutting down a system, or other actions. The program should also provide a way for the response to be documented and tracked. The software program should have a testing plan in place that includes unit testing, integration testing, and system testing. The testing plan should ensure that the program is working correctly, and that the results are accurate and reliable. The implementation and testing plan for the proposed model/system for threat monitoring should include the following steps:

1. Define the requirements and objectives of the system.
2. Select the appropriate hardware and software platforms for the system.
3. Design and develop the software program based on the requirements and objectives.
4. Test the software program using a variety of scenarios and data sets.
5. Evaluate the results and make any necessary adjustments to the software program.
6. Deploy the software program in a production environment and monitor its performance.
7. Maintain the software program by regularly updating it with new threat intelligence data and security patches.

#### **4.4 Project Management Plan**

Developing a project management plan for a threat monitoring project is critical to ensuring the project's success. Here are some key elements that should be included in a project management plan for a threat monitoring project

**Step 1:** Project scope The project management plan should define the project's scope, including its goals, objectives, and deliverables. This should also include a definition of the project's boundaries and any constraints or limitations that may impact the project's success.

**Step 2:** Project timeline: The project management plan should include a detailed timeline for the project, outlining key milestones and deliverables. This should also include a breakdown of the project into smaller phases, with specific timelines for each phase.

**Step 3:** Project budget: The project management plan should include a detailed budget for the project, outlining the cost of all resources required, including personnel, equipment, software licenses, and other expenses. The budget should also include a contingency plan for unexpected expenses.

**Step 4:** Project team: The project management plan should identify the members of the project team, including their roles and responsibilities. This should also include a plan for managing team communication and collaboration.

**Step 5:** Project risks: The project management plan should identify the risks associated with the project, including technical, operational, and financial risks. The plan should also include a risk management strategy that outlines how risks will be mitigated or addressed.

**Step 6:** Quality management: The project management plan should include a quality management strategy that outlines how the project's deliverables will be reviewed, tested, and validated to ensure they meet the project's requirements.



**Step 7:** Change management: The project management plan should include a change management strategy that outlines how changes to the project scope, timeline, or budget will be managed and communicated to stakeholders.

**Step 8:** Stakeholder management: The project management plan should include a stakeholder management strategy that outlines how stakeholders will be identified, engaged, and managed throughout the project.

## 4.5 Transition/ Software to Operations Plan

Security Testing (DAST), and Infrastructure as a Code (IaC) scanning tools In any agile development methodology, when business teams start creating a user story, they should include security as a key requirement and appoint a security champion. Some planning factors to consider are the presence of private data, business-critical assets, confidential information, users, and critical functions. Integrating security tools in the continuous integration/continuous development (CI/CD) pipeline automates the security code review process that examines the application's attack surface. This code review might include Static Application Security Testing (SAST), Dynamic can build in the Application. All these inputs should be shared with the security champion, who would then identify the potential security threats and their mitigations and add them to the user story. With this information, the developers right security controls.deployed and managed over time. Here are some key elements that should be included in an S2O plan for a threat monitoring project

Configuration management The S2O plan should include details on how the software and its components will be configured and managed over time. This should include details on version control, change management, and configuration management.Deployment strategy S2O plan should outline the strategy for deploying the software and its components into the production environment. This should include details on how the deployment will be tested and validated, and how it will be rolled back in case of issues.Monitoring and alerting the S2O plan should include details on how the monitoring and alerting system will be set up and managed over time. This should include details on how the system will be configured, what metrics will be monitored, and how alerts will be triggered and handled.Maintenance and support the S2O plan should outline the maintenance and support plan for the software and its components over time. This should include details on how the software will be

updated and maintained, and how support issues will be handled. Disaster recovery and business continuity the S2O plan should include details on how the software and its components will be recovered in case of a disaster, and how business continuity will be ensured. Security the S2O plan should include details on how security will be managed over time. This should include details on access control, vulnerability management, and incident response. Performance and scalability the S2O plan should outline the strategy for managing performance and scalability over time. This should include details on how the system will be optimized, and how it will be scaled up or down as needed.

## **CHAPTER 5**

### **IMPLEMENTATION DETAILS**

Cyber security has been a critical concern for organizations, especially with the ever-increasing number of cyber attacks. Moreover, according to Statista's report, in 2021, network intrusion was the most common type of cyber attack, with 56% of incidents. Because of this, it's essential to have a proactive approach to identifying and mitigating threats. One way you can achieve this is by implementing a threat intelligence program. A well-designed and executed threat intelligence program can help you identify and respond to threats more quickly and effectively. In addition, cyber threat intelligence is a vital component of this program. It aims to provide actionable information on the latest cyber threats, vulnerabilities, and attack trends. Overall, implementing a threat intelligence program can help improve your security posture and minimize the risks of cyber attacks.

Defining the scope and objectives of your threat intelligence program is vital as it helps set the entire program's foundation. To begin, you need to identify the specific threats and vulnerabilities the program will focus on. This step includes understanding the types of threats relevant to your organization. For example, you can specify threats like phishing attacks, malware, or advanced persistent threats. Once you clearly understand the threat your organization faces, you can set specific objectives for the program. Establishing measurable objectives and aligning them with the organization's overall goals is best. Your targets may include increasing the speed at which you detect and mitigate new threats or reducing the number of successful phishing attacks. Defining clear objectives helps keep the program focused on the most critical threats. As a result, it can make a meaningful impact on the organization's overall security. In developing a plan, you must ensure the program can gather the information crucial to your organization's protection. To do so, identify

first the various threat intelligence sources. These sources can include open-source information, industry reports, and intelligence from other organizations. With such data, you can have a broad view of the threat landscape and help you identify new and emerging threats.

After identifying the sources, establish a process for analyzing and disseminating the intelligence gathered. It would be best if this is efficient and effective. Such a process can allow you to identify and respond to threats quickly. In addition, you should establish a system that prioritizes intelligence based on relevance and potential impact. This step ensures you can take appropriate action to protect your organization effectively. implement the relevant security controls to protect against identified threats. You can start by updating security policies and procedures to reflect the latest threats and vulnerabilities. Such sections as updated incident response plans and security awareness training can fall here.

After that, the technical security controls come next. This section focuses on implementing firewalls, intrusion detection and prevention, endpoint security software, and other cyber security features. You can also consider implementing threat-hunting techniques. In short, it's essential to configure and implement your security controls correctly and regularly for effective threat prevention.

## **Modules**

### **Vulnerability Intelligence**

Threat actors are closely monitoring vulnerability trends as much as you public-facing services and technology to find the easy way in. To prevent them disrupt your business, get alerted whenever there's a new critical vulnerability or exploit for the pre-defined product components and technologies associated with your auto-discovered digital footprint. See which vulnerabilities are being leveraged by threat actors. Get actionable insights and context on potentially vulnerable technologies to speed up the assessment and verification processes.

### **Threat Actor Tracking**

Threat actors and APT groups often integrate a variety of tools and tactics to achieve their objectives. Understanding and tracking these adversaries in a dynamic manner provides insights on current TTPs which may be way more important than easily-changed IOCs. Through automated data collection, classification and AI-powered analysis of hundreds of sources across surface, deep and dark web, SOCRadar keeps you alerted on APT groups' activities, helping you define use cases to more effectively detect and prevent malicious activities.

### **Big threat share data**

SOC Radar opens the door of the deep web in a secure and easy way. Search on the deep web hacker forums, digital assets (IP addresses, domains, etc.), hashes, or any keyword mentioned in the deep web sites without any subscription or providing personal data in a secure environment. SOCRadar's ThreatHose is the peeping hole to the deep web.

### **Threat Feed**

SOC Radar's Threat Feed & IOC Management module helps cybersecurity teams to research cyber threats with enriched data backed up by easy-to-use dashboards. Cyber security professionals can customize the feeds and stay up-to-date with recent threats, search for indicators-of-compromise (IoCs), and integrate with the company systems with TAXII protocol.

### **Cyber News**

Staying up to date about the latest news and advancements is vitally important for threat intelligence teams. To prevent you being bombarded with irrelevant, time-wasting headlines and losing focus, SOCRadar CyberSec News module features the latest cyber security news processed and sorted based on the products and technologies auto-discovered in your external-facing digital assets. Auto-aggregated from credible RSS, Twitter and Telegram channels to bring you the most relevant news.

## **Threat Intelligence Sharing**

ThreatShare scraps the deep web and darknet to create a searchable platform for the users. The hacker chatters on forums, social media, communication channels such as Telegram, ICQ, IRC, etc., ransomware groups' websites are continuously monitored and important information is shared in ThreatShare, including the screenshots and texts. With its smart tagging feature, users can filter the results based on the country, industry, attack type, and content.

## **5.1 Development and Deployment Setup**

System design is the first step in developing a threat monitoring system is to design the system architecture, which involves identifying the key components of the system and their interactions. The design should also consider factors such as scalability, performance, and reliability. Data collection is the next step is to identify the sources of data that the system will use to identify threats. This may include network logs, system logs, and other sources of data. The data should be collected in a standardized format that can be easily analyzed. Once the data has been collected, it must be processed to identify potential threats. This may involve using machine learning algorithms or other techniques to identify patterns in the data that may indicate a threat. Once potential threats have been identified, the system should take appropriate action to mitigate the threat. This may involve sending alerts to security personnel or taking automated actions to block network traffic or shut down systems. Once the system has been developed, it must be deployed to the production environment. This involves configuring the system and its components, testing the system to ensure that it is functioning correctly, and monitoring the system to ensure that it is detecting threats effectively. Once the system has been deployed, it must be maintained to ensure that it continues to operate effectively over time. To facilitate the development and deployment of a threat monitoring system, it is important to have a well-defined development and deployment setup. This may involve using a continuous integration/continuous deployment pipeline to automate the deployment process, using version control systems to manage code changes, and using configuration management tools to manage system configurations.



## 5.2 Algorithms

### 5.2.1 *Random forest*

Random forests or random decision forests is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of over fitting to their training. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees.[citation needed] However, data characteristics can affect their performance. The first algorithm for random decision forests was created in 1995 by Tin Kam Ho using the random subspace method, which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg. An extension of the algorithm was developed by Leo Breiman and Adele Cutler, who registered "Random Forests" as a trademark in 2006 (as of 2019, owned by Minitab, Inc.)] The extension combines Breiman's "bagging" idea and random selection of features, introduced first by Ho and later independently by Amit and Geman in order to construct a collection of decision trees with controlled variance. Random forests are frequently used as black box models in businesses, as they generate reasonable predictions across a wide range of data while requiring little configuration. The general method of random decision forests was first proposed by Ho in 1995. Ho established that forests of trees splitting with oblique hyper planes can gain accuracy as they grow without suffering from over training, as long as the forests are randomly restricted to be sensitive to only selected feature dimensions. A subsequent work along the same lines concluded that other splitting methods behave similarly, as long as they are randomly forced to be insensitive to some feature dimensions. Note that this observation of a more complex classifier (a larger forest) getting more accurate nearly monotonically is in sharp contrast to the common belief that the complexity of a classifier can only grow to a certain level of accuracy before being hurt by over fitting.

The explanation of the forest method's resistance to over training can be found in Klein berg's theory of stochastic discrimination. The early development of Breiman's notion of random forests was influenced by the work of Amit and Geman who introduced the idea of searching over a random subset of the available decisions when splitting a node, in the context of growing a single tree. The idea of random subspace selection from Ho was also influential in the design of random forests. In this method a forest of trees is grown, and variation among the trees is introduced by projecting the training data into a randomly chosen subspace before fitting each tree or each node. Finally, the idea of randomized node optimization, where the decision at each node is selected by a randomized procedure, rather than a deterministic optimization was first introduced by Thomas G. Dietterich. The proper introduction of random forests was made in a paper by Leo Breiman. This paper describes a method of building a forest of uncorrelated trees using a CART like procedure, combined with randomized node optimization and bagging. Decision trees are a popular method for various machine learning tasks. Tree learning "comes 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 over fit 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. As part of their construction, random forest predictors naturally lead to a dissimilarity measure among the observations. One can also define a random forest dissimilarity measure between unlabeled data: the idea is to construct a random forest predictor that distinguishes the "observed" data from suitably generated synthetic data. The

observed data are the original unlabeled data and the synthetic data are drawn from a reference distribution. A random forest dissimilarity can be attractive because it handles mixed variable types very well, is invariant to monotonic transformations of the input variables, and is robust to outlying observations. The random forest dissimilarity easily deals with a large number of semi-continuous variables due to its intrinsic variable selection; for example, the random forest dissimilarity weighs the contribution of each variable according to how dependent it is on other variables. The decision tree algorithm builds a tree in which each internal node evaluates a data feature. Each branch represents a decision around a possible value for the selected feature, and each final node in a branch indicates the class the element is most likely to belong to. Thus, the algorithm traverses the tree branches and evaluates the features of each node to estimate the sample probability to belong to a particular class. A great advantage of the decision tree algorithm is its ease of understanding and interpretation, being composed exclusively by rules in the "if-then-else" format. is a probabilistic classifier that works through the application of Bayes' theorem. This theorem, shown in the Equation 1, indicates the probability that an event  $c$  will occur knowing that a given event  $x$  has happened. The parameters used by the equation are the a priori probabilities of  $c$  and  $x$ , as well as their likelihood. Since the algorithm performs the classification through a simple mathematical calculation, resulting in linear execution time, it is easily scalable for large datasets and several features. However, the accuracy obtained by this classifier may be lower than that obtained by other algorithms, since it assumes that analyzed elements are statistically independent, which may not be valid depending on the chosen dataset. The random forest is an ensemble learning algorithm that works by creating multiple decision trees and whose classification result is defined as the mode of the classification results obtained by each tree. This method usually presents better results than those obtained by working with only one decision tree, while also offering a lower risk of overfitting; however, it has a considerably longer processing time and is not scalable for large datasets without sacrificing. However, these and other datasets are often not recent, and the model's performance. As with the random forest algorithm, the gradient-boosted tree is an ensemble learning algorithm based on decision tree

models. Unlike random forest, where each tree is trained individually, trees in this algorithm are trained iteratively, with new trees using the prediction of previous trees to offer a more accurate model. This algorithm also has a high processing time and is not ideal for large datasets. The support vector machines (SVM) method initially works by mapping the input features into a larger space, called the feature space; this change allows a model that is difficult to separate in the original space to be separable by a hyperplane in the feature space. The method performs the separation between classes by defining this hyperplane, aiming to maximize the separation margin between the points closest to each of the classes.

An important aspect of this method is the definition of the chosen kernel function, responsible for the mapping done in the feature space. Multiple kernel functions exist, with some being: RBF, polynomial, hyperbolic tangent, among others. is a statistical model that approximates the a posteriori probability of the positive class, using the sigmoid function as a discriminant function, whose formula can be seen in the Binary classification problems usually employ this method.

### 5.2.2 XG Boost

XG Boost (extreme Gradient Boosting) is an open-source software library which provides a regularizing gradient boosting framework for C++, Java, Python, R, Julia, Perl, and Scala. It works on Linux, Windows, and macOS.[8] From the project description, it aims to provide a "Scalable, Portable and Distributed Gradient Boosting (GBM, GBRT, GBDT) Library". It runs on a single machine, as well as the distributed processing frameworks Apache Hadoop, Apache Spark, Apache Flink, and Dask. It has gained much popularity and attention recently[when?] as the algorithm of choice for many winning teams of machine learning competitions. XG Boost initially started as a research project by Tianqi Chen as part of the Distributed (Deep) Machine Learning Community (DMLC) group. Initially, it began as a terminal application which could be configured using a libsvm configuration file. It became well known in the ML competition circles after its use in the winning solution of the Higgs Machine Learning Challenge. Soon after, the Python and R packages were built, and XGBoost now has package implementations for Java, Scala, Julia, Perl, and other languages. This brought the library to more developers and contributed to its popularity among the Kaggle community, where it has been used for a large number of competitions. It was soon integrated with a number of other packages making it easier to use in their respective communities. It has now been integrated with scikit-learn for Python users and with the caret package for R users. It can also be integrated into Data Flow frameworks like Apache Spark, Apache Hadoop, and Apache Flink using the abstracted Rabbit and XGBoost4J. XGBoost is also available on OpenCL for FPGAs. An efficient, scalable implementation of XGBoost has been published by Tianqi Chen and Carlos Guestrin. While the XGBoost model often achieves higher accuracy than a single decision tree, it sacrifices the intrinsic interpretability of decision trees. For example, following the path that a decision tree takes to make its decision is trivial and self-explained, but following the paths of hundreds or thousands of trees is much harder. To achieve both performance and interpretability, some model compression techniques allow transforming an XGBoost into a single "born-again" decision tree that approximates the same decision function. Gradient boosting is a machine learning technique used in regression and classification tasks, among others. It gives a

prediction model in the form of an ensemble of weak prediction models, which are typically decision trees. When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted trees; it usually outperforms random forest. A gradient-boosted trees model is built in a stage-wise fashion as in other boosting methods, but it generalizes the other methods by allowing optimization of an arbitrary differentiable loss function. The idea of gradient boosting originated in the observation by Leo Breiman that boosting can be interpreted as an optimization algorithm on a suitable cost function. Explicit regression gradient boosting algorithms were subsequently developed, by Jerome H. Friedman, simultaneously with the more general functional gradient boosting perspective of Llew Mason, Jonathan Baxter, Peter Bartlett and Marcus Frean. The latter two papers introduced the view of boosting algorithms as iterative functional gradient descent algorithms. That is, algorithms that optimize a cost function over function space by iteratively choosing a function (weak hypothesis) that points in the negative gradient direction. This functional gradient view of boosting has led to the development of boosting algorithms in many areas of machine learning and statistics beyond regression and classification. A large and growing list of data scientists globally that are actively contributing to XGBoost open source development Usage on a wide range of applications, including solving problems in regression, classification, ranking, and user-defined prediction challenges. A library that's highly portable and currently runs on OS X, Windows, and Linux platforms. Cloud integration that supports AWS, Azure, Yarn clusters, and other ecosystems Active production use in multiple organizations across various vertical market areas. A library that was built from the ground up to be efficient, flexible, and portable.

### 5.3 Testing

Testing is a crucial part of the development process for any threat monitoring system. The purpose of testing is to ensure that the system is functioning as intended, and that it is capable of detecting and responding to threats effectively. Here are some key types of testing that should be performed for a threat monitoring system.

Functional testing ensures that the system is performing its intended functions correctly. This involves testing the system's features, such as its ability to collect and analyze data, identify threats, and generate alerts. Performance testing ensures that the system is capable of processing data quickly and efficiently, even under heavy loads. This involves testing the system's ability to handle large amounts of data, and to respond quickly to threats. Security testing ensures that the system is secure, and that it cannot be compromised by attackers. This involves testing the system's ability to detect and respond to attacks, as well as testing the system's security controls, such as access controls and encryption. Usability testing: Usability testing ensures that the system is easy to use and understand for end-users. This involves testing the system's user interface, as well as the user documentation and help system. Integration testing ensures that the threat monitoring system integrates correctly with other systems and applications. This involves testing the system's ability to communicate with other systems, and to exchange data with them. Regression testing ensures that changes to the system do not cause unintended consequences or affect existing functionality. This involves testing the system after changes have been made to ensure that everything still works as expected.

## CHAPTER- 6

### RESULTS AND DISCUSSION

The model convergence and training time plus the processing speed must be considered in the context of real-time analysis. To verify the impact of the data structure used during model training, we compared the model training time of Dataframe-based TeMIA-NT with the RDD-based CATRACA, and IDS previously proposed by our research group. Figure 2 shows that the DataFrame data structure several performance optimizations have a significant impact on the latency. Training the model with DataFrame is ten times faster than the same operation made with RDD. We split the dataset into 70% for the training set and the other 30% for the test set to perform the classification score and obtain the models in the offline processing mode. Also, we used K-fold cross-validation, with  $k = 10$ , to guarantee the model's generalizability. Finally, we use the grid search method to tune the hyperparameters in each algorithm. Figure 3 shows the results of each algorithm, with random forest, decision tree, and gradient-boosted tree offering the best performances. The online mode of operation uses the model with the highest processing capacity and good accuracy. Table I shows that the decision tree algorithm presented the maximum flow volume rate of 50 GB/s. The random forest classification model has a lower performance due to the need to process multiple trees, and it is necessary to obtain the result for all trees to achieve the final classification result. As the decision tree model presents the best results both in accuracy and in classification capacity, this is the model used by default in the execution of the tool. However, other models can also be used according to the user's needs. The last test observed the impact of parallelism on the tool's performance, observing how variations in the number of processing nodes affect the results. The processing time required to classify 10 million network flows was measured while varying the processing nodes number between 1 and 4 in the Spark environment; the results can be seen in Figure 4. As can be seen from the Figure, increasing the number of processing nodes reduces the total time required to process flows for most algorithms. This impact is more significant in the case of the Random Forest, as this algorithm works by creating multiple models of trees that can be executed in parallel during the classification process. The results of the Decision Tree algorithm are negatively affected by the increase in the number of node.



## CHAPTER-7

### 7.1 Conclusion

Machine Learning approaches towards intrusion detection, malware detection, and vulnerability have been discussed. Among all the fields, deep learning techniques considerably optimize the accuracy comparing to the conventional method by introducing feature extraction and representation. At the cost of acceptable extra complexity and training periods, the large number of expert interactions can be reduced. The combination of machine learning and deep learning also brings new possibilities to threat detection in web security. More and more classification methods and neural network architecture have been designed and put into practice. Some of them have achieved productive results and are ready for additional examination in the future. The others may need to be re-designed based on their drawbacks but the overall trend is optimistic. Similar measures can be taken into account to help overcome the obstacles. First, about the feature generation method, more mechanisms should be developed to boost up precision and efficiency. Improper feature representations may lead to a huge rise in the cost of the training section. Second, about the training and testing database, the extremely limited dataset is available to the researchers in the open environment. This burning issue results from a large number of demands in the feature pattern generation. I hope most of the researchers can share their private datasets for investigation use to push the development of machine learning in all kinds of application fields, besides threat detection.

## 7.2 Future Work

To identify the future work of a threat monitoring project, it's important to first understand the current state of the project and what goals it has achieved so far. Assuming that the threat monitoring project is currently operational and fulfilling its objectives, some potential future work could include

- Expansion of monitored systems:** The project could be expanded to include more systems and networks to increase its scope and effectiveness. This would require the development of new monitoring tools and techniques to accommodate the additional data.
- Improved threat detection algorithms:** The project could benefit from more advanced algorithms for detecting threats. This could involve integrating machine learning or artificial intelligence techniques to identify patterns and anomalies that may be indicative of a threat.
- Better response protocols:** The project could develop improved response protocols for dealing with detected threats. This could involve more automation of responses or the development of a centralized system for managing responses.
- Integration with other security systems:** The project could be integrated with other security systems, such as firewalls, intrusion detection systems, and antivirus software, to provide a more comprehensive approach to threat monitoring.
- Enhanced reporting and analytics:** The project could develop more robust reporting and analytics capabilities to better understand trends and patterns in the data, and to provide more actionable insights to security teams.

Threat intelligence has evolved in very short period and there is hundreds of threat data feed available whether from open source, closed source or free to use. To defend against cyber-attack, it is very important for customer to have timely access to relevant, actionable threat intelligence and the ability to act on that intelligence. However, many of them still struggle with an overwhelming amount of threat data and a lack of staff expertise to make the most of their threat intelligence programs. According to a survey conducted by Ponemon on 1000 IT practitioners in 2016, 70% of the respondent said threat intelligence is too voluminous and/or complex to provide actionable intelligence. To address this issue many organizations have successfully identified a variety of resources and techniques to help maximize the effectiveness of their threat intelligence. This is support by the result of survey conducted by Ponemon that show 80% of respondent agree that deploying threat intelligence platform can help the organization to automate threat intelligence. While 54% urge to have a qualified threat analyst staff to fully utilize threat intelligence potential.

Vazquez raised an interoperability issue that face by existing threat sharing platform.

The various standard and format use by threat sharing platform hindered the producer and receiver speak seamlessly to each other due to data extension is not supported by the used application. As an initiative, MITRE group has developed three specifications/standards namely CYBOX (Cyber Observable Expression), STIX (Structured Threat Information Expression) and TAXII (Trusted Automated Exchange of Indicator Information). By adopting to the standard introduced by MITRE, interoperability issue between threat sharing peers can be solved. However, if there is no data standard can be established between peers due some constraint, data transformation can come in handy.

### **7.3 Research Issues**

Threat monitoring is an important component of information security, and there are many research issues related to this area. Develop and evaluate automated systems that can detect and respond to threats in real-time. These systems should use machine learning and other artificial intelligence techniques to analyze vast amounts of data and detect anomalies or patterns that indicate potential threats. Investigate the effectiveness of multi-layered defense strategies that combine different security controls such as firewalls, intrusion detection and prevention systems, antivirus software, and data loss prevention systems to protect against a variety of threats. Explore methods for collecting, analyzing, and sharing threat intelligence data to improve threat monitoring capabilities. This can include the use of open-source intelligence, dark web monitoring, and threat feeds from security vendors. Investigate methods for detecting and mitigating insider threats, which are often more difficult to detect than external threats. This can include the use of behavior analysis and monitoring tools to identify abnormal activity by employees or contractors. Study the challenges and opportunities of monitoring threats in cloud environments, where data and applications are distributed across multiple platforms and providers. This can include the use of cloud-based security services and technologies such as virtual firewalls and intrusion prevention systems. Investigate the impact of regulatory requirements such as GDPR, CCPA, and HIPAA on threat monitoring and develop methods for ensuring compliance while maintaining effective threat monitoring. Explore methods for visualizing threat data and alerts to help security teams quickly identify and respond to threats. This can include the use of dashboards, heat maps, and other graphical representations of threat data. Investigate the effectiveness of proactive threat hunting methods, which involve actively searching for threats rather than waiting for alerts to be triggered. This can include the use of threat hunting tools and techniques such as honeypots and deception technologies. Develop methods for threat monitoring that preserve the privacy of users and organizations by minimizing the amount of sensitive data that is collected and analyzed. This can include the use of differential privacy techniques and other privacy-preserving data analytics methods.

Investigate methods for attributing threats to specific actors or groups, which can be useful for determining the appropriate response and for sharing threat intelligence with other organizations. This can include the use of digital forensics and incident response techniques to trace the origin of attacks.

## **7.4 Implementation Issues**

Implementing a threat monitoring system can be challenging and there are several issues that need to be considered. One of the first implementation issues for threat monitoring is collecting the data that will be used to detect potential threats. This can include logs from network devices, servers, applications, and other systems. The challenge here is to ensure that the data is accurate, complete, and consistent. Once the data has been collected, it needs to be stored securely and efficiently. This can be a challenge when dealing with large amounts of data, especially if it needs to be retained for a long period of time for compliance or legal reasons. The next step is to process the data to detect potential threats. This can involve using machine learning algorithms, anomaly detection techniques, or other methods to analyze the data and identify suspicious activity. The challenge here is to ensure that the processing is efficient and accurate, and that it can handle large volumes of data in real-time. One of the biggest challenges of threat monitoring is dealing with false positives and false negatives. False positives occur when the system identifies an activity as a potential threat when it is actually harmless, while false negatives occur when the system fails to detect a real threat. Reducing the number of false positives and false negatives is critical to the effectiveness of the system. When a potential threat is detected, the system needs to alert the appropriate personnel and initiate a response. The challenge here is to ensure that the alerts are timely, relevant, and actionable, and that the response is appropriate to the severity of the threat. Threat monitoring systems need to be integrated with other security controls such as firewalls, intrusion detection and prevention systems, and antivirus software. The challenge here is to ensure that the integration is seamless and that the different controls work together effectively. Threat monitoring can be resource-intensive, and organizations need to allocate sufficient resources to ensure that the system is effective. This can include hardware, software, and personnel resources. Organizations may be subject to regulatory requirements such as GDPR, CCPA, and HIPAA that impact how threat monitoring is implemented and operated. The challenge here is to ensure that the system is compliant with these requirements while still being effective in detecting and responding to potential threats.

## REFERENCES

1. CyberSecurity Market Report - <https://cybersecurityventures.com> 2020
2. E. Bertino and N.Islam, "Botnets and Internet of things security", computer, vol. 50, no.2, pp.76-79,2020
- 3.M.Tavallaee, E.Bhageri, W.Lu "A detailed analysis of KDD Cup99 dataset", in 2009 IEEE symposium on CSIDA, July 2019
- 4.Symantec, "Internet Security Threat Report" - <https://docs.broadcom.com/doc/istr-21-2019-en>
- 5.Verzion Enterprise, "Data breach investigation report"- <https://enterprise.verzion.com/resources/reports/2020-data-breach-investigation-report.pdf>
- 6.M. Pelloso, A. Vergutz, A.Santos "A self adaptable system for DDos attack prediction based on metastability theory" 2018 IEEE Global Communications Conference.
- 7.M. A. Lopez, D. M. F. Mattos, O. C. M. B. Duarte "Towards a monitoring and threat detection system based on stream processing as a virtual network function for big data" Concurrency and Computation: Practice and experience, vol. 31, n0. 20, p. e5344, 2019
- 8.M. Zaharia, R. S. Xin, P. Wendell, T. Das, A. Dev et.al.," Apache Spark: a unified engine for big data processing" Communications of the ACM, vol. 59, n0. 11, pp 56-65, 2018.
- 9.R. Xin and J. Rosen "Project Tungsten: Bringing Apache spark closer to Bare Metal" - [https://databricks.com/blog/2015/04/28/project-tungsten-bringing-spark-closer-to-bare\[1\]metal.html](https://databricks.com/blog/2015/04/28/project-tungsten-bringing-spark-closer-to-bare[1]metal.html)

10. X. Meng, J. Bardley, B. Yavuz, E. Sparks, S. Venkataraman et al., "Mllib-Machine Learning in apache spark," The Journal of Machine Learning Research, vol. 17, n0. 1, pp. 1235-1241, 2017.

11. Apache Software Foundation - " Apache Metron . <https://metron.apache.org/>" .



## APPENDIX

### A. Source Code

```
# Importing Libraries
import numpy as np
import pandas as pd
import seaborn as sns

# Importing Dataset
train=pd.read_csv('NSL_KDD_Train.csv')

# Visualising the Dataset
train.shape
train.head()
train.info()

# draw a graph using seaborn showing the null values present in the dataset
sns.heatmap(train.isnull(),yticklabels=False,cbar=False)

# Preprocessing the Dataset
result=train[["class"]]
train.drop(["class"],axis=1,inplace=True)
train.drop(["id", "duration", "protocol_type","land","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","srv_count","error_rate",
"srv_error_rate","error_rate","srv_error_rate","srv_diff_host_rate",
"dst_host_count",
"dst_host_srv_count","dst_host_same_srv_rate",
"dst_host_diff_srv_rate","dst_host_srv_error_rate", "dst_host_error_rate",
"dst_host_srv_error_rate"],axis=1,inplace=True)
train.head()

train.info()

# Converting categorical data to numerical data
```

```

categoricalfeat=["service","flag"]
def onehotencoding(features):
    final=train
    i=0
    for feat in features:
        f1=pd.get_dummies(train[feat],drop_first=True)
        train.drop([feat],axis=1,inplace=True)
        if i==0:
            final=f1.copy()
        else:
            final=pd.concat([final,f1],axis=1)
        i=i+1
    final=pd.concat([train,final],axis=1)
    return final
encoded=onehotencoding(categoricalfeat)
encoded.shape
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
result=le.fit_transform(result)
result=pd.DataFrame(result)
result.shape
result.head()

# Splitting the Dataset into Train and Test sets
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(encoded,result,test_size=0.2,
random_state=1)
X_train.shape
X_test.shape
y_train.shape
y_test.shape

```

```

# Applying RandomForest Classifier Algorithm
from sklearn.ensemble import RandomForestClassifier
cls=RandomForestClassifier()
cls.fit(X_train,y_train)
y_pred=cls.predict(X_test)
print(y_pred)
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test,y_pred))
print('True Negative Percentage')
print('percentage of legitimate packets correctly identified is:',(2339/(2339+14))*100)
print('False Positive Percentage')
print('percentage of legitimate packets wrongly identified as malicious
is:',(14/(2339+14))*100)
print('False Negative Percentage')
print('percentage of malicious packets wrongly identified as legitimate
is:',(3/(2683+3))*100)
print('True Positive Percentage')
print('percentage of malicious packets correctly identified are:',(2683/(2683+3))*100)

# Evaluation Metrics
from sklearn.metrics import accuracy_score,f1_score,recall_score
print('Accuracy score:',accuracy_score(y_test,y_pred)*100)
print('F1 score:',f1_score(y_test,y_pred)*100)
print('Recall score:',recall_score(y_test,y_pred)*100)
from sklearn.metrics import mean_squared_error
from math import sqrt
rmse=sqrt(mean_squared_error(y_test,y_pred))
print(rmse)
# By doing k-fold cross validation
from sklearn.model_selection import cross_val_score
accuracies=cross_val_score(estimator=cls,X=X_test,y=y_test,cv=5)

```

```

print('Accuracy scores:', accuracies*100)
print('Average Accuracy score:', accuracies.mean()*100)
print('Average standard deviation:', accuracies.std()*100)
# Applying XGBoost Classifier Algorithm
from xgboost import XGBClassifier
xgb_clf=XGBClassifier()
xgb_clf.fit(X_train,y_train)
y_pred_xgb=xgb_clf.predict(X_test)
print(y_pred_xgb)
print(confusion_matrix(y_test,y_pred_xgb))
print('True Negative Percentage')
print('percentage of legitimize packets correctly identified is:',(2331/(2331+22))*100)

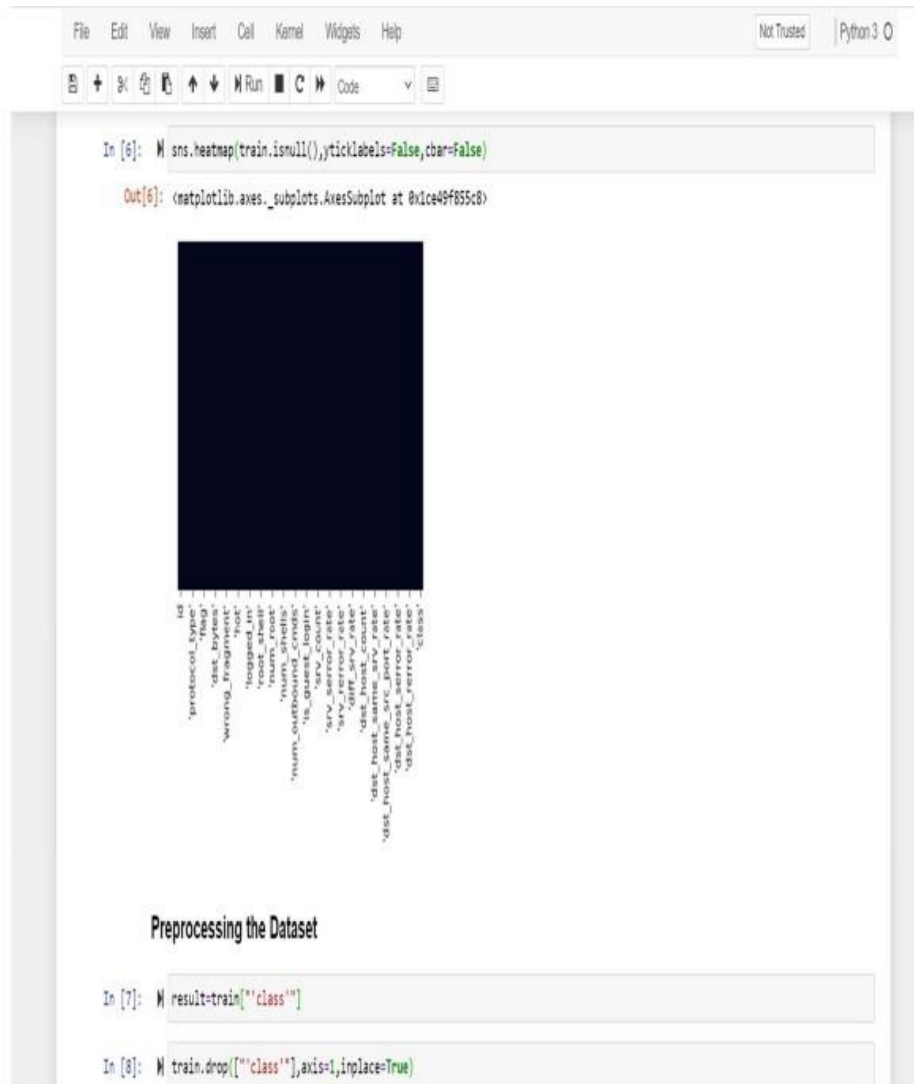
print('False Positive Percentage')
print('percentage of legitimize packets wrongly identified as malicious
is:',(22/(2331+22))*100)
print('False Negative Percentage')
print('percentage of malicious packets wrongly identified as legitimize
is:',(13/(2673+13))*100)
print('True Positive Percentage')
print('percentage of malicious packets correctly identified are:',(2673/(2673+13))*100)
# Evaluation Metrics
print('Accuracy score:', accuracy_score(y_test,y_pred_xgb)*100)
print('F1 score:', f1_score(y_test,y_pred_xgb)*100)
print('Recall score:', recall_score(y_test,y_pred_xgb)*100)
rmse=sqrt(mean_squared_error(y_test,y_pred_xgb))
print(rmse)
## By doing k-fold cross validation
from sklearn.model_selection import cross_val_score
accuracies=cross_val_score(estimator=xgb_clf,X=X_test,y=y_test,cv=5)
print('Accuracy scores:', accuracies*100)

```

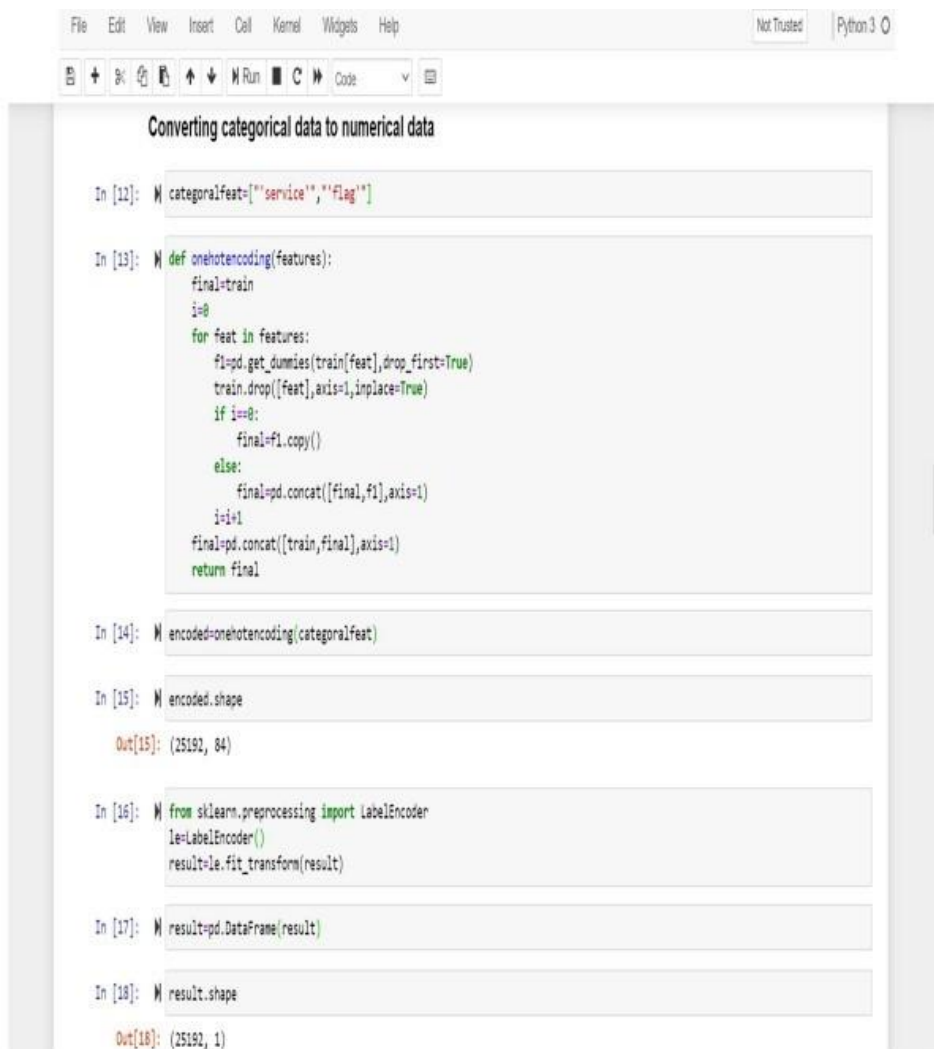
```
print('Average Accuracy score:', accuracies.mean()*100)
print('Average standard deviation:', accuracies.std()*100)
```

## B. Screenshots

### Processing the Dataset



## Converting categorical data to numerical data



The image shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The notebook is titled "Converting categorical data to numerical data". The code is as follows:

```
In [12]: categorical_feat=["service","flag"]

In [13]: def onehotencoding(features):
    final=train
    i=0
    for feat in features:
        f1=pd.get_dummies(train[feat],drop_first=True)
        train.drop([feat],axis=1,inplace=True)
        if i==0:
            final=f1.copy()
        else:
            final=pd.concat([final,f1],axis=1)
        i=i+1
    final=pd.concat([train,final],axis=1)
    return final

In [14]: encoded=onehotencoding(categorical_feat)

In [15]: encoded.shape

Out[15]: (25192, 84)

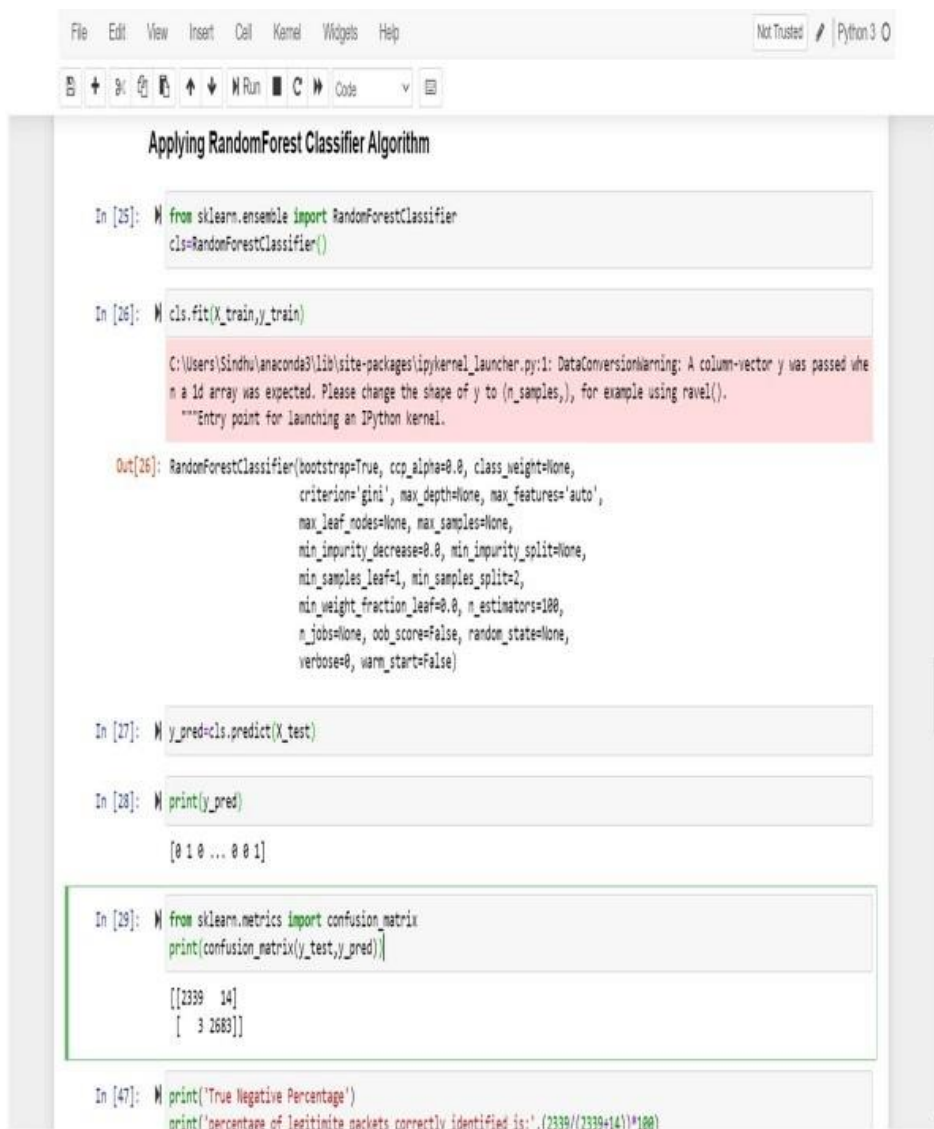
In [16]: from sklearn.preprocessing import LabelEncoder
    le=LabelEncoder()
    result=le.fit_transform(result)

In [17]: result=pd.DataFrame(result)

In [18]: result.shape

Out[18]: (25192, 1)
```

## Applying Random Forest Classifier Algorithm



```
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3
```

Applying RandomForest Classifier Algorithm

```
In [25]: from sklearn.ensemble import RandomForestClassifier
        cls=RandomForestClassifier()

In [26]: cls.fit(X_train,y_train)
```

C:\Users\Sindhu\anaconda3\lib\site-packages\ipykernel\_launcher.py:1: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples,), for example using ravel().  
"""Entry point for launching an IPython kernel.

```
Out[26]: RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                                criterion='gini', max_depth=None, max_features='auto',
                                max_leaf_nodes=None, max_samples=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, n_estimators=100,
                                n_jobs=None, oob_score=False, random_state=None,
                                verbose=0, warm_start=False)
```

```
In [27]: y_pred=cls.predict(X_test)

In [28]: print(y_pred)

[0 1 0 ... 0 0 1]
```

```
In [29]: from sklearn.metrics import confusion_matrix
        print(confusion_matrix(y_test,y_pred))

[[2339  14]
 [   3 2683]]
```

```
In [47]: print("True Negative Percentage")
        print('percentage of legitimate packets correctly identified is:'.(2339/(2339+14))*100)
```



```
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 O
In [47]: print('True Negative Percentage')
print('percentage of legitimize packets correctly identified is:', (2339/(2339+14))*100)

True Negative Percentage
percentage of legitimize packets correctly identified is: 99.40581487462814

In [48]: print('False Positive Percentage')
print('percentage of legitimize packets wrongly identified as malicious is:', (14/(2339+14))*100)

False Positive Percentage
percentage of legitimize packets wrongly identified as malicious is: 0.5949851253718658

In [49]: print('False Negative Percentage')
print('percentage of malicious packets wrongly identified as legitimize is:', (3/(2683+3))*100)

False Negative Percentage
percentage of malicious packets wrongly identified as legitimize is: 0.11169824571854058

In [50]: print('True Positive Percentage')
print('percentage of malicious packets correctly identified are:', (2683/(2683+3))*100)

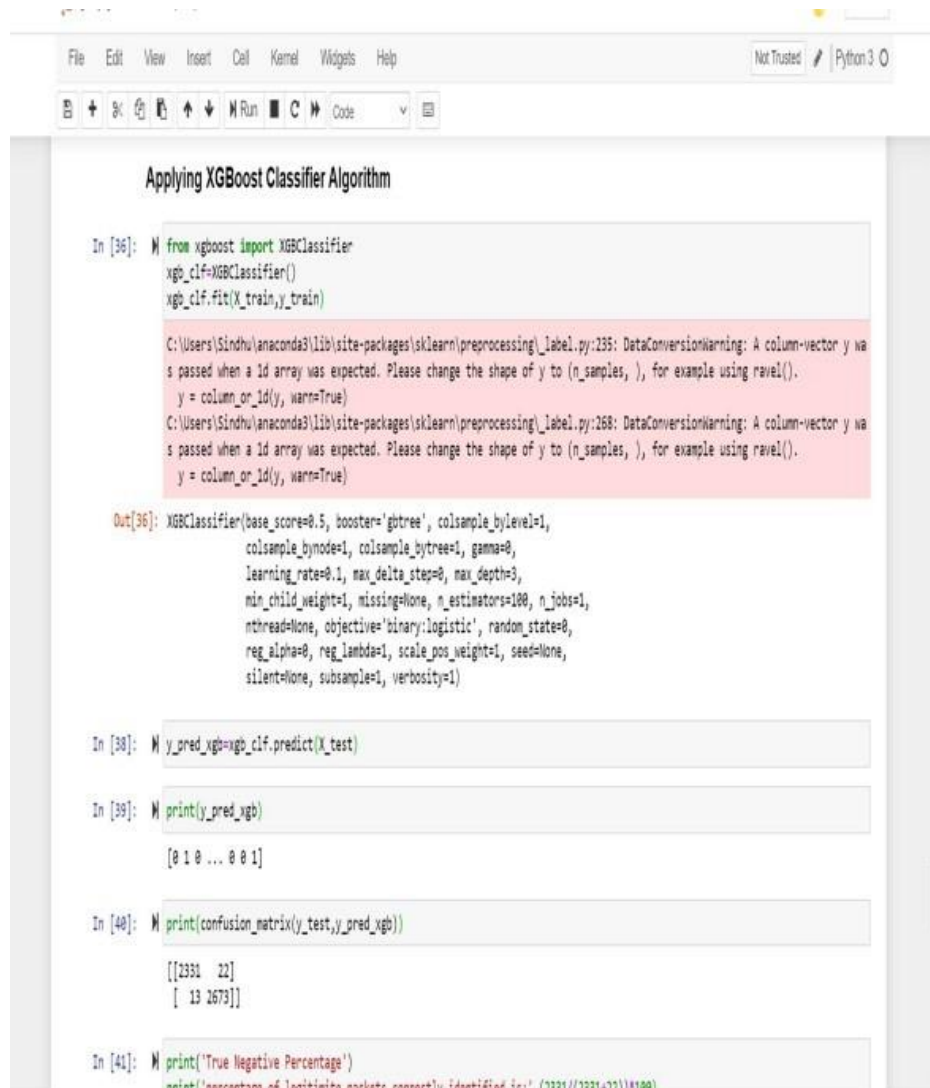
True Positive Percentage
percentage of malicious packets correctly identified are: 99.88838975428145

Evaluation Metrics

In [57]: from sklearn.metrics import accuracy_score, f1_score, recall_score
print('Accuracy score:', accuracy_score(y_test, y_pred)*100)
print('F1 score:', f1_score(y_test, y_pred)*100)
print('Recall score:', recall_score(y_test, y_pred)*100)

Accuracy score: 99.66263147449891
F1 score: 99.68419897157718
Recall score: 99.88838975428145
```

## Applying XGBoost Classifier Algorithm



```
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 O
+ - < > Run C Code v
Applying XGBoost Classifier Algorithm

In [36]: from xgboost import XGBClassifier
xgb_clf=XGBClassifier()
xgb_clf.fit(X_train,y_train)

C:\Users\Sindhu\anaconda3\lib\site-packages\sklearn\preprocessing\_label.py:235: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
y = column_or_1d(y, warn=True)
C:\Users\Sindhu\anaconda3\lib\site-packages\sklearn\preprocessing\_label.py:268: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
y = column_or_1d(y, warn=True)

Out[36]: XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
colsample_bynode=1, colsample_bytree=1, gamma=0,
learning_rate=0.1, max_delta_step=0, max_depth=3,
min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
nthread=None, objective='binary:logistic', random_state=0,
reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
silent=None, subsample=1, verbosity=1)

In [38]: y_pred_xgb=xgb_clf.predict(X_test)

In [39]: print(y_pred_xgb)

[0 1 0 ... 0 0 1]

In [40]: print(confusion_matrix(y_test,y_pred_xgb))

[[2331  22]
 [ 13 2673]]

In [41]: print('True Negative Percentage')
print('percentage of legitimize packets correctly identified is:',(2331/(2331+22))*100)
```

## Evaluation Metrics

```
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3
+ - % < > Run C Code v
In [41]: print('True Negative Percentage')
print('percentage of legitimize packets correctly identified is:', (2331/(2331+22))*100)

True Negative Percentage
percentage of legitimize packets correctly identified is: 99.06502337441565

In [42]: print('False Positive Percentage')
print('percentage of legitimize packets wrongly identified as malicious is:', (22/(2331+22))*100)

False Positive Percentage
percentage of legitimize packets wrongly identified as malicious is: 0.9349766255843603

In [43]: print('False Negative Percentage')
print('percentage of malicious packets wrongly identified as legitimize is:', (13/(2673+13))*100)

False Negative Percentage
percentage of malicious packets wrongly identified as legitimize is: 0.4839910647803425

In [44]: print('True Positive Percentage')
print('percentage of malicious packets correctly identified are:', (2673/(2673+13))*100)

True Positive Percentage
percentage of malicious packets correctly identified are: 99.51600893521966

Evaluation Metrics

In [58]: print('Accuracy score:', accuracy_score(y_test, y_pred_xgb)*100)
print('F1 score:', f1_score(y_test, y_pred_xgb)*100)
print('Recall score:', recall_score(y_test, y_pred_xgb)*100)

Accuracy score: 99.3854177416154
F1 score: 99.34956327820107
Recall score: 99.51600893521966
```

## C.Research Paper

# THREAT MONITORING USING MACHINE LEARNING

V.Bhanukiran<sup>1</sup>  
*Student*  
*Department of CSE,*  
*Sathyabama Institute of Science*  
*and Technology.*  
bhanuveeramalli123@gmail.com

Y.Rohithkumar<sup>2</sup>  
*Student*  
*Department of CSE,*  
*Sathyabama Institute of Science*  
*and Technology.*  
rohithkumar.y710@gmail.com

DR. A.C.Santha Sheela<sup>3</sup>  
*Associate Professor*  
*Department of CSE,*  
*Sathyabama Institute of Science*  
*and Technology.*  
santhasheela.cse@sathyabama.ac.in

## ABSTRACT:

In today's world, instances of cybercrime and data breaches are on the rise, causing significant losses and risks for businesses and internet users. A recent example is the alleged hacking of social media accounts belonging to prominent figures such as Jeff Bezos and Barack Obama by a 17-year-old. Delayed detection of such attacks can lead to irreparable harm. These attacks are causing substantial financial losses to organizations and negatively impacting individuals. Recent research indicates that cybercrime will result in more than \$6 trillion in losses by 2021, with the size, frequency, and complexity of these attacks growing exponentially. Additionally, the ongoing pandemic is generating a vast amount of data, which increases the risk of data breaches. Therefore, network security is a critical feature that both individuals and organizations must prioritize. In spite of the existence of firewalls and intrusion detection systems (IDS) that are meant to identify malicious packets, there may be instances where these security measures prove to be ineffective, leading to significant damage. To tackle this issue, we suggest the development of an Intelligent Intrusion Detection System (IIDS) that takes advantage of past attack information and machine learning algorithms to distinguish between authentic and harmful packets, thereby reducing potential hazards. Our aim is to create an ML model that can accurately categorize network packets as either genuine or malicious. Once integrated into network devices, this model will transmit only genuine packets to the end system. To achieve this, we will utilize the NSL\_KDD dataset and apply the most advanced machine learning techniques to build a highly efficient and precise IIDS "Random forest" and "XG Boost" techniques for classification objectives. The dataset is created by taking into account multiple fields in the packet header.

Keywords: IIDS, Network security, Ensemble classifiers, Threat detection.

## I. INTRODUCTION

Due to the rapid growth of internet usage worldwide, cybercrime has

become a significant challenge. According to a cybersecurity study, the cost of damages caused by cyber invasions is nearly 6 trillion US dollars.

Additionally, the rise of Internet of Things (IoT) and Big Data has further

complicated cybersecurity. The delay in detecting these threats is often due to human intervention, which reduces the effectiveness of response. Although firewalls and intrusion detection systems (IDS) exist, recent attacks have demonstrated the need for changes in security measures to keep up with evolving attack trends. In this research, the authors propose a tool that utilizes machine learning algorithms for real-time threat detection, aiming to improve security solutions. This work builds upon the foundation established by UFRJ scholars in a previous project called CATRACA, with a focus the primary focus of this document is on the capacity to expand, operate proficiently, and precisely identify fresh assaults. The distinctive feature of this research is the deployment of dispersed processing by means of Scala and Resilient Distributed Datasets, along with the usage of Data Frame and Dataset data arrangements presented by Apache Spark. The instrument suggested in this article is competent to work in either real-time or offline modes. The main challenge in this research is the lack of public datasets for network-based attacks. Among the existing options, the NSL\_KDD dataset was chosen for its efficiency and lack of redundant data. The dataset includes over 40 attributes derived from TCP/IP packet headers, such as protocol type, flag, and source bytes authenticated and so on, as well as all the distinct 24 forms of assaults are broadly sorted into 4 distinct groups, namely Denial of Service (DoS), Scanning, User to Administrator (U2A), and Remote to Local (R2L). The

qualities of the package determine whether it is classified as typical or atypical. Diverse artificial intelligence algorithms were employed to construct and educate the model, comprising Decision Tree, Support Vector Machine, Gradient Boost, Logistic Regression, and Random Forest. The model that exhibits the most optimal performance is picked based on several performance metrics, such as F1\_score, accuracy\_score, recall\_score, and precision. The structure can function in two distinct modes: real-time and offline.

## II. LITERATURE SURVEY

**AUTHOR NAME:** Valantina Palacin

This is a primer for individuals unfamiliar with the realm of cyber threat intelligence (CTI) and a manual for those proficient in other cybersecurity domains seeking to create a TH initiative from the ground up. You will commence by examining the definition of threat intelligence and its ability to identify and forestall cyber hazards. As you advance, you will acquire the knowledge to gather data and interpret it by constructing data models. The book will also demonstrate how to utilize open-source tools to construct a TH environment. Subsequently, you will concentrate on planning a hunt with practical examples before exploring the MITRE ATT&K framework.

**AUTHOR NAME:** Nadean H Tanner

Nadean Tanner's extensive background, ranging from teaching at a university to serving in the Department of Defense, results in the Cybersecurity Blue Team Toolkit being a perfect blend of informative and comprehensible, rendering it equally advantageous for IT

or management professionals throughout various sectors. This convenient guide provides a strategic and uncomplicated examination of the best practices and tools accessible to cybersecurity management and hands-on practitioners, whether they are neophytes in the field or seeking to broaden their expertise.

**AUTHOR NAME:** Don Murdoch

Don Murdoch has established five sturdy platforms, amalgamated more than one hundred data sources into diverse platforms, and administered an MSSP practice for two years. This book presents the following topics with a straightforward approach as if you have appointed him as a security advisor and were sitting across the table with him (or her). The book begins with a discussion for experts to assist them in developing a successful business case and a project plan, identify SOC tier models, anticipate and respond to challenging questions that you need to consider when proposing a SOC, and factors to consider while building a logging infrastructure. The book explores numerous data sources that fuel a SOC and SIEM and provides comprehensive practical guidance on using those data sources to the best possible effect.

### **III. SYSTEM REQUIREMENTS**

-Intel Xeon E2630 v4 - A processor with ten cores and a clock speed of 2.2 GHz, which can be boosted up to 3.1 GHz using Turbo boost technology. It is equipped with a 25 MB cache.

Motherboard - ASRock EPC612D8A

-RAM - 128 GB DDR4 with a clock speed of 2133 MHz

Storage - A 2 TB hard disk with a spin rate of 7200 RPM, combined with a 512

GB solid-state drive

Graphics Processing Unit - NVidia Titan X Pascal (12 GB Video Random Access Memory)

Intel Cooling System to regulate temperature

Storm Trooper Chassis

## **IV.METHODOLOGY**

### **EXISTING SYSTEM**

The intricacy of overseeing modern applications (cloud, containers, serverless and composable infrastructure) has made it progressively challenging to identify concealed threats. This is intensified by the immense amount of data that must be processed and linked at cloud-scale. Wrongdoers in cyberspace currently have a greater scope of attack to exploit, including cloud, mobile, IoT/OT/medical IoT, and work-from-anywhere environments. It is becoming more difficult to safeguard against these attackers, who can implement automated, multi-stage attacks using advanced and potent weapons, frequently leveraging machine learning (ML) and artificial intelligence (AI). Integrating and monitoring data produced from multiple security solutions is costly and demanding, necessitating specialized tools that are hard to incorporate and operate efficiently. To surmount these difficulties, specialized security skills are required, but they are expensive to recruit, manage, and retain. Providers often overcommit and underdeliver, deceitfully claiming or exaggerating their capabilities in machine learning (ML), artificial intelligence (AI), multi cloud support, and risk metrics. Automation can boost efficiency and hasten threat detection, liberating security team members to focus on more challenging tasks. Harmful software applications like malware and Trojans are created to harm an

organization's network, data, and systems. A malware duplicates itself by copying to another application, host file, or system and remains inactive until activated. A Trojan is a self-duplicating program that spreads and contaminates other computers and networks that lack adequate security without any human involvement.

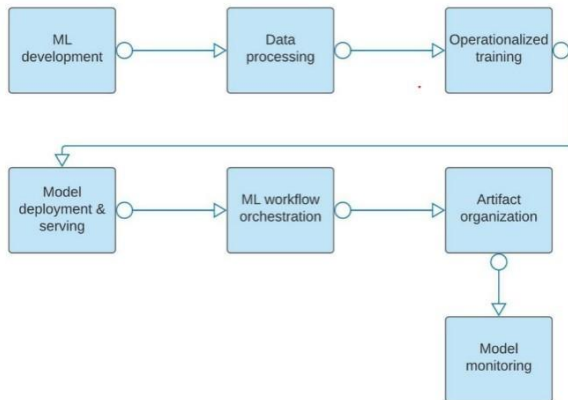
## **PROPOSED SYSTEM**

Upon examination of the limitations of prior systems that relied solely on a predetermined dataset of faces with open and closed eyes, it becomes apparent that implementing threat monitoring is a more effective approach to detecting previously undetected threats, such as external parties attempting to connect to or explore networks, and compromised or unauthorized internal accounts. Entities can protect their data from SQL injection by gaining comprehensive knowledge about data access and usage. Moreover, malicious entities can misuse remote tools like SSH (Secure Shell), RDP (Remote Desktop Protocol), SNMP (Simple Network Management Protocol), and SMB (Server Message Block) to gain unauthorized remote access, which can result in the compromise of sensitive information. Observation of potential dangers offers numerous benefits for security professionals, including reducing the likelihood of internal risks and improving the efficacy of protecting information. When organizations have implemented threat surveillance, their ability to counter both internal and external threats is significantly increased. Even after the development stage, risk evaluation remains an essential task. For example, the MITRE ATT&CK framework's initial access strategy, which deals with the methods that hackers use to infiltrate a target network or system, can be utilized

for customers with web applications or servers hosted in AWS cloud that are exposed to attacks such as DDoS (Distributed Denial of Service) and XSS (Cross-Site Scripting). Malicious actors may aim at business resources such as data, various components, and functions (applications) that are crucial for

business continuity to commit sabotage and disrupt ongoing operations. Data resources, however, are components and functions that hackers find particularly valuable, as they can exploit them to carry out additional malicious activities. For instance, cybercriminals may exploit data assets to assist their crypto-mining operations. Contrary to popular belief, today's hackers do not necessarily perform custom attacks on their victims. Instead, they are opportunistic and search for the most direct entry points and exploit already-known vulnerabilities. This constitutes your attackable area - fundamentally the entirety of your vulnerable constituents that might link a malicious actor to one of your resources. In threat modelling, teams delineate all aspects of the attackable area and illustrate how information moves to and from these constituents.

## V. SYSTEM ARCHITECTURE



The underlying system design establishes the fundamental framework for the system, and we propose creating each architectural diagram based on our specific needs. The objective of this project is to incorporate a mechanism that utilizes a comprehensive framework to classify the severity of cybersecurity threats in response to incidents of cybercrime. The main goal of this official process is to evaluate occurrences of online criminal activity and produce essential information for an advanced system that assigns appropriate measures to relevant parties. The utilized approach comprises a series of successive phases for a qualitative examination of hazards to cybersecurity, which is founded on annual records of cybercrime incidents and an assessment of their level of risk over time. The writers assessed various authorized sources from ENISA, the Agency for Network and Information Security of the European Union, to rank menaces by combining frequency, references to occurrence, and the number of incidents. The results of the study show that the most significant 15 cyber-security risks have intensified in severity in recent years. A group of highly prevalent risks is consistently

categorized as hyper-critical threats and/or maintains an exponential growth rate, while other risks are less severe, demonstrating in the frequency incidents.

## VI REQUIREMENT ANALYSIS

This section aims to encourage the development of more efficient methods for identifying cyber-attacks during the application process. It outlines the challenges associated with capturing execution behaviors of web-based applications through program modifications and using artificial intelligence to promptly recognize potential cyber-attacks. A crucial obstacle is to capture application behavioral traits without overburdening developers, as automated mechanisms are necessary for cybersecurity specialists to collect and scrutinize application execution data. One promising approach involves utilizing machine learning to create patterns of anticipated application operation behavior. By operation behavior, we refer to the activation of procedures, the series of method invocations, and the inputs/outputs of method invocations.

## VII CONCLUSION

After finishing the assignment, I want to declare that it has effectively accomplished its desired objective. The outcomes acquired from a trustworthy gathering of data centre on conveyance exhibit a remarkable capacity to handle streams each second. Additionally, the efficacy of each AI methodology used is evident, as both the decision tree and random forest models have achieved praiseworthy precision and fl-score evaluations. In addition, the findings of the study also highlight how most algorithms can be expanded and optimized as more nodes are added to



the cluster. Thus, taking into account the evaluation criteria of diverse classifiers, it is quite apparent that the Random Forest classifier yields the most superior outcomes with increased precision. Therefore, by employing the trained model generated through this categorization technique, we can establish an intelligent system for detecting intrusions. By installing such a model in any network layer appliance, we can guarantee that exclusively genuine network traffic is transmitted to the ultimate user, thereby furnishing enhanced security measures for both entities and individuals.

## REFERENCES

1. Report on cybersecurity request-  
<https://cybersecurityventures.com>
- 2.M. Tavallaee,E. Bhageri,W. Lu," In-depth analysis of KDD Cup99 dataset," in 2009 IEEE Symposium on CSIDA, July 2009.
3. Symantec," Internet Security trouble Report"  
<https://docs.broadcom.com/doc/istr-21-2019-en>
4. Verizon Enterprise," Report on Data Breach examinations"  
<https://enterprise.verizon.com/resources/reports/2020-data-breach-investigation-report.pdf>
- 5.M. Zaharia,R.S. Xin,P. Wendell,T. Das,A. Dev etal.," Apache Spark A unified machine for big data processing," *Dispatches of the ACM*,vol. 59,no. 11,pp. 56- 65, 2016.
- 6.X. Meng,J. Bradley,B. Yavuz,E. Sparks,S. Venkataraman etal.," MLlib-Machine Learning in Apache Spark," *The Journal of Machine Learning Research*,vol. 17,no. 1,pp. 1235- 1241, 2016.
7. Apache Software Foundation-" Apache Metron."  
<https://metron.apache.org/>