INTRUSION DETECTION SYSTEM
A NETWORK TRAFFIC, SYSTEM ACTIVITY MONITOR

A project report submitted in partial

Fulfillment of the requirements for the award of the Degree

BACHELOR OF TECHNOLOGY
IN
CSE-CYBER SECURITY

Submitted by

G. NAGA MALLESWARI	21HT1A4619
V. NARENDRA	21HT1A4659
Y. ASHOK REDDY	21HT1A4655
P. KOTESWARA RAO	21HT1A4662

Under the esteemed guidance of

Mr. T. YEDU KONDALU
Assistant Professor

DEPARTMENT OF CSE-CYBERSECURITY

CHALAPATHI INSTITUTE OF TECHNOLOGY

(Approved by A.I.C.T.E, Affiliated To JNTUK, Kakinada)

A.R.Nagar, Mothadaka, Guntur, Andhra Pradesh, India 522016

2021-2025

CITY CHALAPATHI Institute of Technology

AUTONOMOUS INSTITUTION | NBA | NAAC GRADE A | Recognised under 2(f) & 12(B) of UGC ACT1956 | Permanently Affiliated to JNTUK

DEPARTMENT OF CSE- CYBERSECURITY

CERTIFICATE

This is to certify that G. NAGA MALLESWARI (21HT1A4619), V. NARENDRA (21HT1A4659), P. KOTESWARA RAO (21HT1A4662), Y. ASHOK REDDY (21HT1A4655) completed a project entitled "INTRUSION DETECTION SYSTEM" for the partial fulfilment of the requirements for the award of BACHELOR OF TECHNOLOGY in CSE – CYBER SECURITY by JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, KAKINADA.

T. YEDU KONDALU	Dr. D. KALYAN KUMAR
Assistant Professor	HEAD OF THE DEPARTMENT
PROJECT GUIDE	

Submitted for Viva Voice Examination held on

External Examiner

DECLARATION

We hereby declare that the project entitled **"INTRUSION DETECTION SYSTEM"** submitted in partial fulfillment of the requirements for the award of bachelor of technology in computer science and engineering, to CHALAPATHI INSTITUTE OF TECHNOLOGY (CITY-HT), permanently affiliated to JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA (JNTUK) is an authentic work and has not been submitted to university or institute for the award of the degree.

DATE**:**

PLACE**:**

SIGNATURE OF THE CANDIDATES

G. NAGA MALLESWARI	21HT1A4619
V. NARENDRA	21HT1A4659
Y. ASHOK REDDY	21HT1A4655
P. KOTESWARA RAO	21HT1A4662

ACKNOWLEDGMENT

We Consider it our privilege to express our gratitude to all those who guided and inspired us in the completion of this Project. We express our sincere thanks to our beloved **Chairman Sri Y.V. ANJANEYULU Garu** for providing support and stimulating environment for developing the project.

We express deep sense of reverence and profound gratitude **Dr. K. NAGA SREENIVASA RAO Garu, Principal, Chalapathi Institute of Technology** for providing us the great support in completing our resource for carrying out the project.

Our sincere thanks to **Dr. D. KALYAN KUMAR Garu**, **HOD DEPT.OF CSECYBER SECURITY** for his co-operation and guidance in helping us to make our project successful and complete in all aspects we are grateful to his precious guidance and suggestions.

It is with immense pleasure that we would like to express our indebted gratitude to our guide **Mr. T. YEDU KONDALU Garu** has guided us a lot and encouraged us in every step of the project work. He has given moral support and guidance throughout the project helped us to a great extent.

We also place our floral gratitude to all other Teaching Staff and Lab programmers for their constant support, advice throughout the project. Last but not least; we thank our PARENTS and FRIENDS who directly or indirectly helped us in the successful completion of our project.

PROJECT ASSOCIATES

G. NAGA MALLESWARI	(21HT1A4619)
V. NARENDRA	(21HT1A4659)
Y. ASHOK REDDY	(21HT1A4655)
P. KOTESWARA RAO	(21HT1A4662)

ABSTRACT

The Intrusion Detection System (IDS) Implementation project addresses critical challenges in modern network security by developing a robust and adaptable tool for identifying and responding to malicious activities. An IDS acts as a crucial defence mechanism, monitoring network traffic and system behaviour for suspicious patterns that may indicate an intrusion attempt. While traditional IDS solutions offer basic threat detection, they often struggle with evolving attack vectors, high false positive rates, and difficulties in real-time analysis and response. This project offers an innovative solution by designing a Python-based platform that leverages machine learning algorithms and advanced anomaly detection techniques to enhance intrusion detection accuracy and efficiency.

The primary objective of the IDS Implementation project is to provide a user-friendly and intelligent tool that overcomes the limitations of conventional IDS systems. This tool empowers security administrators with real-time threat identification, automated alert generation, and adaptive learning capabilities to stay ahead of emerging threats. It promotes a proactive approach to network security by integrating threat intelligence feeds and visualization tools, enhancing understanding of attack patterns and enabling faster incident response.

Operating within a framework that prioritizes accessibility and ethical use, the IDS Implementation project ensures adherence to legal boundaries and respects privacy considerations. By focusing on innovation and usability, this project sets a benchmark in intrusion detection, addressing the growing need for advanced security measures in today's interconnected world.

The IDS Implementation project tackles the evolving challenges of network security by creating a sophisticated, user-friendly tool that bridges the gap between traditional intrusion detection and modern network requirements. As cyberattacks become increasingly sophisticated and frequent, IDS solutions must evolve to provide real-time, adaptive, and intelligent protection against malicious intrusions and data breaches. This project delivers a solution designed with Python and machine learning, offering an intelligent platform for detecting, analysing, and responding to security threats tailored to individual network environments.

Traditional IDS solutions often suffer from signature-based detection limitations, high false positive rates, complexity in rule management, and a lack of real-time insights. These challenges hinder effective threat detection and response, leaving networks vulnerable to advanced persistent threats and zero-day exploits. This project aims to address these shortcomings by incorporating machine learning algorithms and anomaly detection techniques to improve detection accuracy and reduce false positives.

LIST OF CONTENTS

LIST OF FIGURES

CHAPTER-1: INTRODUCTION

INTRODUCTION

Overview of a Project:

In today's interconnected world, cybersecurity has become paramount. Organizations and individuals alike face constant threats from malicious actors seeking to exploit vulnerabilities and gain unauthorized access to sensitive data and systems. To combat these threats, Intrusion Detection Systems (IDS) have emerged as crucial tools in safeguarding digital assets.

An IDS is a security system that monitors network traffic and system activity for malicious activity or policy violations. It acts as a vigilant guardian, constantly analyzing data flows and user behaviour to identify any deviations from normal patterns. By detecting and alerting on suspicious activity, IDS empowers organizations to proactively respond to threats, minimize damage, and enhance their overall security posture.

This project focuses on the development and implementation of an (Specify your IDS type, e.g., "novel anomaly-based IDS," "signature-based IDS for IoT devices," "cloud-based IDS for critical infrastructure"). The system aims to (Clearly state the specific objectives and goals of your IDS project, e.g., "detect zero-day attacks with high accuracy," "protect IoT devices from common vulnerabilities," "provide real-time threat intelligence to security analysts").

The following sections will delve into the technical details of the proposed IDS, including its architecture, algorithms, implementation, and evaluation.

Types of Intrusion Detection Systems

IDS can be broadly categorized into two main types:

Network-Based Intrusion Detection Systems (NIDS): These systems monitor network traffic flowing in and out of a network segment. They analyze data packets for malicious patterns, such as unauthorized access attempts, port scans, and denial-of-service attacks. NIDS are typically deployed at strategic points within a network, such as at the perimeter or within critical segments.

Host-Based Intrusion Detection Systems (HIDS): These systems focus on monitoring the activity within individual hosts (computers or servers). They analyse system logs, audit trails, and other system-level data to detect suspicious activity, such as unauthorized file access, malware infections, and privilege escalations. HIDS provide a deeper level of visibility into the internal operations of a system.

IDS employ various techniques to detect malicious activity:

Signature-based Detection: This method relies on predefined patterns or signatures of known attacks. IDS systems compare incoming traffic or system events against these signatures. If a match is found, an alert is triggered. This approach is effective against known threats but may miss novel or zero-day attacks.

Anomaly-based Detection: This method focuses on identifying deviations from normal system behavior. IDS systems establish a baseline of normal activity and then analyze incoming traffic or system events for any unusual patterns. Anomalies, such as sudden spikes in traffic, unusual user activity, or unexpected system changes, can indicate potential threats. This approach is more effective at detecting novel attacks but may generate a higher number of false positives.

Behavior-based Detection: This method analyzes user behavior and system activity to identify suspicious patterns. IDS systems learn normal user behavior over time and then flag any deviations from this established baseline. This approach can effectively detect insider threats and other forms of malicious activity that may not be easily detectable using other methods.

Intrusion Detection Systems (IDS) serve a crucial role in safeguarding computer systems and networks from malicious activity. Here are some key usages:
Proactive Threat Detection: IDSs can identify and alert on suspicious activity in real-time, such as unauthorized access attempts, data breaches, and denial-of-service attacks. This allows for immediate response and mitigation of potential damage.

Improved Security Posture: By continuously monitoring network traffic and system activity, IDSs provide valuable insights into potential vulnerabilities and threats. This information can be used to enhance security measures, strengthen defenses, and improve overall network security.

Reduced Risk of Data Breaches: IDSs play a crucial role in preventing data breaches by detecting and blocking malicious activities that could compromise sensitive data.

Compliance with Regulations: Many industries and organizations are subject to regulatory requirements that mandate specific security measures. IDSs can help organizations demonstrate compliance with these regulations by providing evidence of security controls and proactive threat monitoring.

Enhanced Incident Response: In the event of a security incident, IDS logs and alerts can provide valuable information for investigation and response teams. This information can help pinpoint the source of the attack, understand the scope of the damage, and accelerate the incident response process.

Insider Threat Detection: IDSs can help detect and prevent insider threats, such as malicious activity from employees or privileged users. By analyzing user behavior and system activity, IDSs can identify unusual patterns that may indicate insider threats.

Network Performance Monitoring: Some IDSs can also be used to monitor network performance and identify potential bottlenecks or performance issues. This information can be used to optimize network performance and ensure smooth operations.

CHAPTER-2: LITERATURE SURVEY

LITERATURE SURVEY

The field of Intrusion Detection Systems (IDS) has been extensively researched, leading to a diverse range of approaches and technologies. This literature survey explores key research and resources related to IDS, covering various aspects from fundamental concepts to advanced techniques.

Foundational Texts and Overviews:

Anderson, J. P. (1980) - Computer Security Technology Planning Study. This seminal work laid the groundwork for intrusion detection by exploring the concept of anomaly detection and the need for automated security monitoring.

Denning, D. E., & Neumann, P. G. (1985) - Requirements and Implementation for a Real-Time Intrusion Detection System. This paper formalized the concept of IDS and outlined key requirements for effective intrusion detection systems.

Lunt, T. F., et al. (1990) - A Real-Time Intrusion-Detection System. This research described the development of an early real-time IDS, focusing on rule-based detection and event correlation.

Detection Techniques

Signature-Based Detection: Numerous publications have explored signature- based detection, focusing on efficient pattern matching algorithms (e.g., Aho- Corasick) and signature generation techniques. Research continues on optimizing signature databases and minimizing false positives.

Anomaly-Based Detection: This area has seen significant research, with various machine learning algorithms being applied to model normal behavior and detect deviations. Research focuses on improving accuracy, handling evolving network behavior, and reducing false alarms.

Specification-Based Detection: This approach, though less common, defines expected system behavior and flags deviations as intrusions. Research in this area focuses on formal specification languages and efficient runtime monitoring.

Network-Based IDS (NIDS):

Roesch, M. (1999). Snort: Lightweight Intrusion Detection for Networks. This paper introduced Snort, a widely used open-source NIDS, which has spurred significant research on NIDS development and deployment.

Venter, H. S., & Eloff, J. H. P. (2010). A taxonomy for intrusion detection systems. This paper proposes a comprehensive taxonomy for classifying NIDS, providing a framework for understanding the various approaches and their strengths and weaknesses.

Host-Based IDS (HIDS):

Kim, D., & Solomon, M. G. (2016). Fundamentals of Information Systems Security. This book discusses HIDS in detail, covering their architecture, functionalities, and deployment considerations.

Various research papers have explored the use of system call analysis, file integrity monitoring, and registry monitoring for HIDS.

Advanced Topics:

Data Mining and Machine Learning for IDS: Numerous studies have explored the application of data mining and machine learning techniques for improving intrusion detection accuracy, including feature selection, anomaly detection, and threat prediction.

Cloud-Based IDS: Research in this area focuses on adapting IDS techniques to the unique challenges of cloud environments, such as scalability, elasticity, and multi-tenancy.

Intrusion Prevention Systems (IPS): Research on IPS focuses on extending IDS capabilities to include automated response actions, such as blocking malicious traffic or terminating suspicious processes.

Security Information and Event Management (SIEM): SIEM systems integrate logs and alerts from various security devices, including IDS, to provide a centralized view of security events and enable correlation and analysis.

Open Web Application Security Project (OWASP): OWASP provides valuable resources on web application vulnerabilities and security best practices, which are relevant to developing and deploying web application firewalls (WAFs), a type of specialized IDS.

NIST Cybersecurity Framework: NIST publications offer guidance on cybersecurity best practices, including recommendations for implementing and managing intrusion detection systems.

CHAPTER-3: SYSTEM ANALYSIS

SYSTEM ANALYSIS

3.1 Existing system:

Most traditional IDS implementations fall into the following categories:

Signature-Based IDS: Detects known attack patterns by comparing network traffic with a database of signatures.

Anomaly-Based IDS: Uses machine learning or statistical methods to detect deviations from normal behaviour.

Host-Based IDS (HIDS): Monitors activities on individual devices, analysing system logs and user behaviour.

Network-Based IDS (NIDS): Inspects network traffic for suspicious activity across multiple systems.

Limitations of the Existing IDS Despite their effectiveness, traditional IDS have several shortcomings:

High False Positives: Anomaly-based IDS often flag legitimate activity as threats.

High False Negatives: Signature-based IDS fail to detect new, unknown attack types.

Scalability Issues: Traditional IDS may struggle to handle large-scale network traffic in cloud or enterprise environments.

Slow Response Time: Delayed detection and response can allow attackers to exploit vulnerabilities before mitigation.

Lack of Context Awareness: IDS may lack deep visibility into user intent, making it difficult to differentiate between benign and malicious activities.

System Analysis Considerations: When analysing an existing IDS, the following factors must be evaluated:

- **Performance and Scalability:** Measuring system efficiency under different loads.
- **Integration with Other Security Tools:** Ensuring compatibility with SIEM (Security Information and Event Management) and firewalls.
- **Response Time:** Evaluating how quickly the system detects and responds to threats.
- **Adaptability to New Threats:** Checking how well the IDS updates to handle new vulnerabilities.

Given these limitations, organizations often enhance IDS with: Machine Learning & AI for better anomaly detection. Threat Intelligence Integration to detect zero-day attacks. Automated Response Mechanisms to reduce human intervention.

3.2 Proposed system:

Proposed System for Intrusion Detection System

This document outlines a proposed system for an Intrusion Detection System (IDS). The system will leverage a hybrid approach, combining signature-based and anomaly-based detection techniques for enhanced effectiveness.

System Architecture

The proposed system will consist of the following components:

1. **Data Acquisition Module:**
   - Responsible for collecting network traffic data from various sources, including network interfaces, firewalls, and security information and event management (SIEM) systems.
   - Supports various data formats, such as NetFlow, PCAP, and syslog.
2. **Preprocessing Module:**
   - Cleans and transforms the raw data into a suitable format for analysis.
   - This includes tasks such as data normalization, feature extraction, and dimensionality reduction.
3. **Signature-Based Detection Engine:**
   - Utilizes a database of known attack signatures (e.g., Snort rules, YARA rules) to match incoming traffic patterns against known threats.
   - Provides real-time alerts for identified threats.
4. **Anomaly-Based Detection Engine:**
   - Employs machine learning algorithms (e.g., Support Vector Machines, Isolation Forest, Autoencoders) to establish a baseline of normal network behavior.
   - Detects deviations from this baseline, indicating potential anomalies and potential threats.
5. **Correlation Engine:**
   - Correlates alerts from both signature-based and anomaly-based detection engines.
   - Reduces false positives and provides a more comprehensive view of potential threats.

Key Features

Hybrid Approach: Combines the strengths of signature-based and anomaly- based detection for enhanced accuracy and reduced false positives.

Scalability: Designed to handle high volumes of network traffic and adapt to evolving threat landscapes.

Flexibility: Configurable to meet specific security requirements and threat profiles.

Real-time Analysis: Provides real-time threat detection and alerting capabilities.

Automated Response: Supports automated response actions to mitigate threats quickly.

Integration Capabilities: Integrates with existing security infrastructure, such as firewalls, SIEM systems, and other security tools.

Benefits

Improved Threat Detection: Enhanced accuracy and reduced false positives compared to traditional IDS approaches.

Proactive Response: Enables proactive response to threats, minimizing potential damage.

Reduced Operational Costs: Automates threat detection and response processes, reducing the need for manual intervention.

Enhanced Security Posture: Provides a robust defence against a wide range of cyber threats.

This proposed system offers a comprehensive and effective solution for intrusion detection, providing organizations with the tools and capabilities to proactively defend against cyber threats and maintain a strong security posture.

Future Enhancements:

Future Enhancements of Intrusion Detection Systems Intrusion Detection Systems (IDS) are constantly evolving to keep pace with the ever- changing threat landscape. Here are some key areas of future enhancement:

1. Advanced Machine Learning and Artificial Intelligence:

- **Deep Learning:** Implementing deep learning models like neural networks and recurrent neural networks for more sophisticated anomaly detection. These models can learn complex patterns and identify subtle deviations from normal behaviour that traditional methods might miss.
- **AI-Powered Threat Intelligence:** Integrating AI to analyse threat intelligence feeds, identify emerging threats, and automatically update IDS signatures and detection rules.
- **Behavioural Analytics:** Leveraging AI to analyse user behaviour patterns and identify anomalies that may indicate insider threats or compromised accounts.

2. Enhanced Threat Intelligence Sharing:

- **Real-time Threat Feeds:** Integrating with real-time threat intelligence feeds from various sources, such as security research organizations and government agencies, to provide up-to-date threat information.
- **Automated Threat Sharing:** Automating the sharing of threat intelligence data among organizations to collectively improve threat detection and response capabilities.

3. Improved Contextual Awareness:

- **Contextual Analysis:** Incorporating contextual information, such as user location, device type, and time of day, to improve the accuracy of threat detection and reduce false positives.
- **Integration with Other Security Tools:** Seamless integration with other security tools, such as firewalls, SIEM systems, and endpoint security solutions, to provide a holistic view of the security landscape.

4. Automation and Orchestration:

- **Automated Response:** Automating response actions, such as blocking traffic, quarantining systems, and triggering incident response procedures, to minimize the impact of attacks.
- **Security Orchestration and Automation (SOAR):** Integrating IDS with SOAR platforms to automate security workflows, improve incident response times, and reduce the burden on security teams.

5. Focus on Emerging Threats:

- **IoT Security:** Addressing the unique challenges of securing IoT devices, including constrained resources and limited processing power.
- **Cloud Security:** Enhancing cloud-native IDS capabilities to protect cloud environments and address threats specific to cloud computing.
- **Zero-Trust Security:** Implementing zero-trust security principles, such as least privilege and continuous verification, to enhance the effectiveness of IDS in modern, dynamic environments.

CHAPTER-4: SYSTEM STUDY

SYSTEM STUDY

A system study for an Intrusion Detection System (IDS) project involves a thorough analysis of the current state of IDS technology, including their features, capabilities, limitations, and user requirements. Here's a proposed system study outline.

Eight Key Considerations for Feasibility Analysis:

1. **Analysis of Existing Intrusion Detection Systems:**
   - Review and analyse existing IDS solutions (both commercial and open-source), including NIDS, HIDS, and cloud-based IDS.
   - Identify their strengths and weaknesses: scalability, performance, accuracy (low false positives/negatives), deployment complexity, and cost.
   - Evaluate how existing systems address challenges like zero-day attacks, advanced persistent threats (APTs), and evolving attack vectors.
   - Analyse the effectiveness of different detection methods (signature- based, anomaly-based, behaviour-based).
2. **User Requirements Analysis:**
   - Conduct interviews, surveys, and focus groups with potential users (security analysts, network administrators, IT managers) to gather requirements.
   - Identify key use cases: real-time threat detection, incident response, compliance reporting, forensic analysis.
   - Define specific needs: types of attacks to detect, level of detail in alerts, integration with other security tools (SIEM, firewalls), reporting requirements.
   - Prioritize user needs based on importance and feasibility.
3. **Evaluation of Emerging Technologies:**
   - Explore emerging technologies relevant to IDS: machine learning (deep learning, reinforcement learning), artificial intelligence, big data analytics, cloud computing, and software-defined networking (SDN).
   - Assess their potential to enhance IDS capabilities: improved accuracy, automated threat detection, scalability, and adaptability.
   - Identify opportunities for innovation and differentiation.
4. **Assessment of Regulatory and Compliance Requirements:**
   - Review relevant regulations and compliance standards: PCI DSS, HIPAA, GDPR, NIST Cybersecurity Framework, ISO 27001.
   - Identify specific security controls and audit requirements related to intrusion detection and prevention.
   - Ensure the proposed IDS aligns with these requirements.
5. **Study of Security Threats and Attack Vectors:**
   - Investigate current and emerging security threats: malware, ransomware, phishing, DDoS attacks, insider threats, APTs, and zero- day exploits.
   - Analyze attack patterns and techniques to understand the vulnerabilities that need to be addressed.
   - Study the evolving threat landscape and its implications for IDS design.

6. **Evaluation of Integration Points:**
   - Identify potential integration points with other security tools: SIEM systems, firewalls, endpoint security solutions, threat intelligence platforms.
   - Assess how integration can enhance threat detection, incident response, and security management.
   - Define requirements for data exchange and interoperability.
7. **Assessment of Usability and User Experience:**
   - Evaluate the usability of existing IDS solutions: user interface, alert management, reporting capabilities, and configuration complexity.
   - Design a user-friendly interface that provides clear and actionable information.
   - Focus on efficient workflows for security analysts and administrators.
8. **Consideration of Resource Constraints:**
   - Analyse resource constraints: budget, time, manpower, and technical expertise.
   - Develop a realistic project plan that takes these constraints into account.
   - Prioritize features and functionalities based on feasibility and cost- benefit analysis.

CHAPTER-5: MODULES

.

MODULES

Intrusion Detection Systems (IDS) are crucial for network security, acting as a vigilant guardian against malicious activities. Let's explore the key modules that make up an IDS:

1. Data Collection:

- **Sensors:** These are the ears and eyes of the IDS, strategically placed to capture network traffic or system events. They can be network-based (monitoring traffic) or host-based (monitoring a specific system).
- **Data Sources:** Depending on the type of IDS, data sources can include network packets, log files, system calls, and application data.

2. Analysis Engine:

- **Detection Methods:** This is the brain of the IDS, where the collected data is analysed to identify potential intrusions. Common methods include:
  - **Signature-based:** Comparing data patterns to known attack signatures.
  - **Anomaly-based:** Identifying deviations from established "normal" behaviour.
  - **Statistical**
  - **-based:** Analyzing statistical properties of data for suspicious patterns.
- **Correlation:** This module links related events to provide a comprehensive view of an attack, even if individual events seem benign.

3. Alerting and Reporting:

- **Alert Generation:** When suspicious activity is detected, the IDS generates alerts to notify security personnel.
- **Alert Management:** This module helps prioritize and manage alerts, reducing "alert fatigue" and ensuring timely responses.
- **Reporting:** The IDS provides reports on security events, trends, and overall system health, aiding in security audits and improvements.

4. Response (in some cases):

- **Intrusion Prevention Systems (IPS):** Some IDSs have the capability to not just detect but also prevent intrusions by taking actions like blocking traffic or terminating connections.
- **Automated Response:** Depending on the configuration, the IDS can trigger automated responses to contain threats.

5. Preprocessing Module:

- The preprocessing module refines and formats the data collected to make it easier to analyze.
- This can involve filtering out irrelevant information, normalizing the data, or aggregating data for more efficient analysis.

6. Detection Module:

- This is the core component of the IDS that performs the actual analysis of the data to detect potential threats.
- It typically uses one of two primary detection methods:

- **Signature-based detection**: Relies on known attack patterns or signatures.
- **Anomaly-based detection**: Identifies deviations from normal behavior, indicating potential attacks.
- Some IDS systems use hybrid approaches that combine both techniques.

7. Database/Storage Module:

- This component stores all collected data, logs, alerts, and any other relevant information that needs to be retained for further analysis or compliance.
- The storage can be local, or it can involve centralized log management systems (e.g. SIEM systems).

8. Configuration/Management Module:

- This is used to configure and manage the IDS, including setting detection rules, adjusting thresholds for alerts, and updating signatures for signature- based detection systems.
- This module is important for fine-tuning the IDS to balance false positives/negatives and ensure optimal performance.

9. Integration Module:

- In some systems, the IDS may integrate with other security tools, such as firewalls, antivirus programs, or SIEM platforms, to provide a more holistic defence system.
- This module helps the IDS exchange data and coordinate responses with other security infrastructure.

Additional Modules:

- **User Interface:** A console for security personnel to interact with the IDS, configure settings, and view alerts.
- **Database:** For storing event data, attack signatures, and system configurations.
- **Integration:** Modules for integrating with other security tools like firewalls and SIEM systems.

An Intrusion Detection System (IDS) is designed to monitor network or system activities for malicious actions or policy violations. Typically, an IDS consists of several modules that work together to detect and respond to threats. These modules can vary depending on the specific implementation, but generally, the core modules include:

CHAPTER-6: SYSTEM DEVELOPMENT

SYSTEM DEVELOPMENT

The system development for an Intrusion Detection System involves several key steps to create a robust security monitoring platform. Initially, requirements are gathered to understand the organization's security needs and threat landscape. With this information, the system architecture is designed, considering server-side components (e.g., data processing engine, database), client-side components (management console), communication protocols, and data storage requirements.

Once the environment is set up, development begins with the creation of the core analysis engine, data collection modules, and the user interface. Integration and testing follow to ensure proper functionality and accuracy. Security measures are implemented to protect the IDS itself. The system is deployed to a production environment. Ongoing maintenance, performance optimization, rule updates, documentation, and continuous improvement complete the system development process, ensuring a robust, secure, and scalable intrusion detection platform.

1. **Requirements Gathering and Analysis:**
   - Identify stakeholders (security team, IT operations, management).
   - Gather requirements:
   - Specific threats to detect (e.g., malware, DDoS, SQL injection).
   - Network segments or systems to monitor.
   - Data sources to use (network traffic, logs, etc.).
   - Performance expectations (detection rate, false positive rate).
   - Integration requirements with other security tools (SIEM, firewalls).
   - Compliance requirements (e.g., PCI DSS, HIPAA).
2. **Architecture Design:**
   - Design the system architecture:
   - Sensor placement and data collection methods.
   - Data processing and analysis engine components.
   - Database design for storing events, rules, and configurations.
   - Communication protocols (e.g., for data transfer, alerts).
   - Scalability considerations.
3. **Environment Setup:**
   - Set up the development environment:
   - Install necessary software (e.g., analysis tools, database).
   - Configure network connectivity and data sources.
   - Set up a testing environment that mirrors the production network.
4. **Server-Side Development (Analysis Engine):**
   - Develop the core analysis engine:
   - Implement detection algorithms (signature-based, anomaly-based).
   - Develop data processing modules for normalizing and filtering data.
   - Implement alerting mechanisms.
   - Develop APIs for integration with other systems.
5. **Client-Side Development (Management Console):**

- Develop the user interface:
- Design dashboards for visualizing security events and trends.
- Create interfaces for configuring the IDS and managing rules.
- Implement alert management and reporting features.

6. Integration and Testing:

- Integrate the server-side and client-side components.
- Conduct thorough testing:
- Unit testing of individual modules.
- Integration testing of combined components.
- System testing to evaluate overall performance and accuracy.
- Performance testing to assess handling of large data volumes.
- Security testing to identify vulnerabilities in the IDS itself.

7. Security Implementation:

- Implement security measures to protect the IDS:
- Secure communication channels.
- Access control and authentication.
- Regular security audits and vulnerability scanning.
- Protection against tampering and data breaches.

8. Deployment:

- Deploy the IDS to the production environment:
- Configure sensors and data sources.
- Tune detection rules and thresholds.
- Integrate with other security tools.
- Monitor performance and resource usage.

9. Rule Management and Updates:

- Implement a process for managing and updating detection rules:
- Regularly update signature databases and anomaly detection models.
- Develop procedures for creating and modifying custom rules.
- Automate rule updates where possible.

10. Maintenance and Support:

- Provide ongoing maintenance and support:
- Monitor system health and performance.
- Address bug fixes and security vulnerabilities.
- Provide technical support to users.

11. Scaling and Optimization:

- Monitor performance and scalability:
- Implement load balancing and horizontal scaling as needed.
- Optimize data processing and analysis algorithms.
- Tune database performance.

12. Documentation:

- Document the system architecture, deployment process, and user manuals.
- Maintain up-to-date documentation for rules and configurations.

13. Continuous Improvement:

- Gather feedback from users and stakeholders.

- Continuously improve detection accuracy and performance.
- Research and incorporate new detection techniques and threat intelligence.

What is Python?

- Python is a high level, structured, open-source programming language that can be used for a wide variety of programming tasks.
- Python within itself is an interpreted programming language that is automatically compiled into byte code before execution.
- It is also a dynamically typed language that includes (but does not require one to use) object-oriented features.
- NASA has used Python for its software systems and has adopted it as the standard scripting language for its Integrated Planning System.
- Python is also extensively used by Google to implement many components of its Web Crawler and Search Engine & Yahoo! for managing its discussion groups.

History of Python

Python was created by **Guido Van Rossum**.
The design began in the late 1980s and was first released in February1991.

Why the name Python?

No. It wasn't named after a dangerous snake. Rossum was fan of a comedy series from late 70s. The name "Python" was adopted from the same series "Monty Python's Flying Circus".

<u>Features of Python Programming</u>

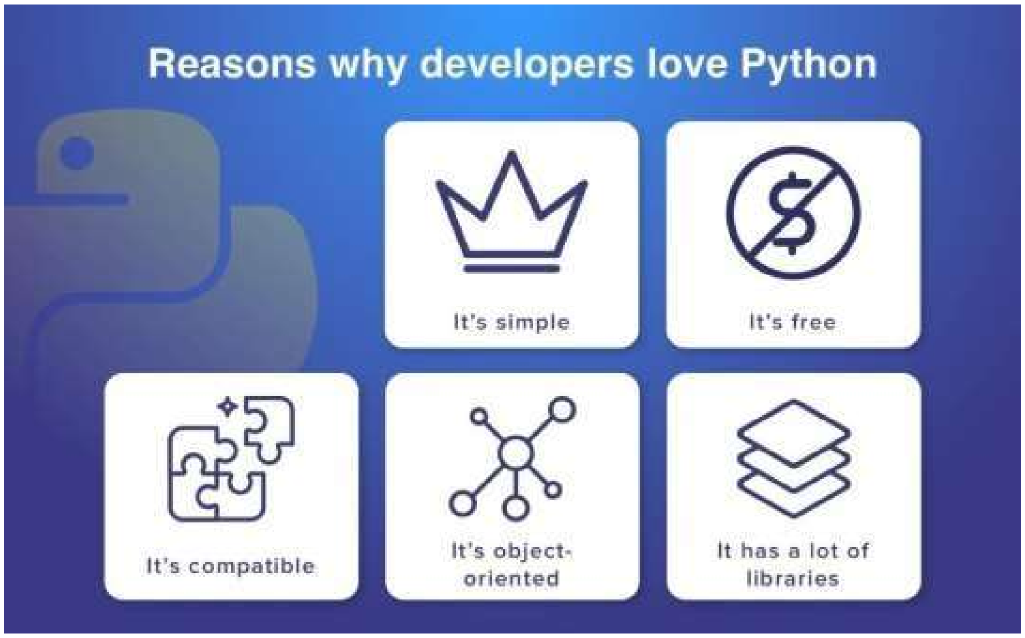


Fig.1: Python Features

1. **A simple language which is easier to learn**
    - Python has a very simple and elegant syntax.
    - It's much easier to read and write Python programs compared to other languages like: C++, Java ,C#.
    - Python makes programming fun and allows you to focus on the solution rather than syntax.
    - If you are a newbie, it's a great choice to start your journey with Python.
2. **Free and open-source**
    - You can freely use and distribute Python, even for commercial use.
    - Not only you can use and distribute software's written in it, you can even make changes to the Python's source code.
3. **Portability**
    - You can move Python programs from one platform to another and run it without any changes.
    - It runs seamlessly on almost all platforms including windows, mac OS and Linux
4. **Extensible and Embeddable**
    - Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code.
    - This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.
5. **A high-level, interpreted language**
    - Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and soon.
    - Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations.
6. **Large standard libraries to solve common tasks**
    - Python has several standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself.
    - For example: Need to connect My SQL database on a Web server? You can use My SQL dB library using import My SQL database.
    - Standard libraries in Python are well tested and used by hundreds of people. So, you can be sure that it won't break your application.
7. **Object-oriented**
    - Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively.
    - With OOP, you can divide these complex problems into smaller sets by creating objects.

Python Version History
Implementation started - December 1989 Released in 1990

PYTHON VERSIONS	DATE OF RELEASED
0.9	20-Feb-91
1	25-Oct-94
2	16-Oct-00
3	3-Dec-08
3.1	27-Jun-09
3.2	20-Feb-11
3.3	29-Sep-12
3.4	16-Mar-14
3.5	13-Sep-15
3.6	23-Dec-16
3.7	27-Jun-18
3.8	14-Oct-19
3.9	5-Oct-20
3.10.0	4-Oct-21
3.10.4	24-Mar-22

Table.1: Python Versions

4 Reasons to Choose Python as First Language

1. Simple Elegant Syntax
Programming in Python is fun. It's easier to understand and write Python code.
Why? The syntax feels natural. Take this source code for an example:

```
a = 2
b = 3
sum= a+b
print (sum)
```

Even if you have never programmed before, you can easily guess that this program adds two numbers and prints it.

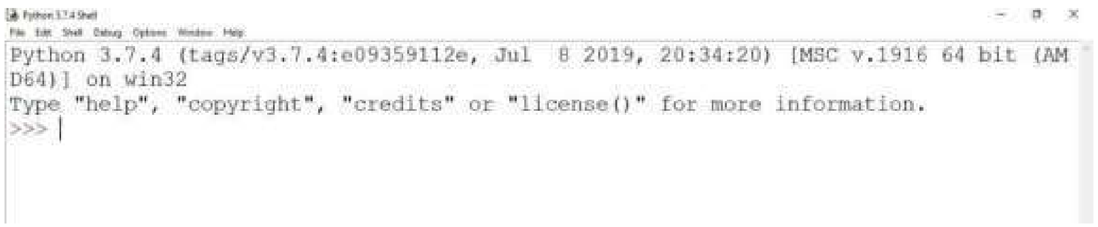
2. Not overly strict
You don't need to define the type of a variable in Python. Also, it's not necessary to add semicolon at the end of the statement. Python enforces you to follow good practices (like proper indentation). These small things can make learning much easier for beginners.

3. Expressiveness of the language
Python allows you to write programs having greater functionality with fewer lines of code. Here's a link to the source code of Tic-tac-toe game with a graphical interface and a smart computer opponent in less than 500 lines of code. This is just an example. You will be amazed how much you can do with Python once you learn the basics.

4. Great Community and Support
Python has a large supporting community. There are numerous active forums online which can be handy if you are stuck. Some of them are:

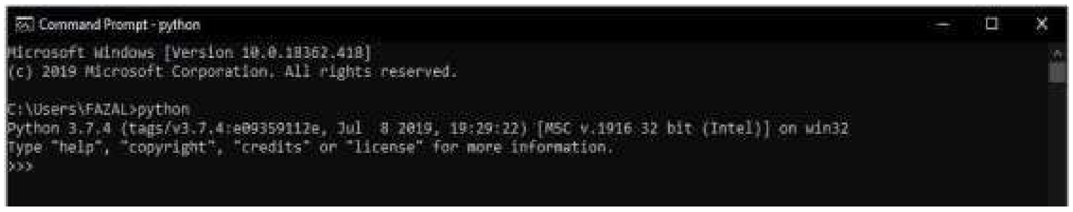
- Google Forum for Python
- Python Questions – Stack Overflow

PYTHON HAS TWO BASIC MODES:
Interactive mode: is a command line shell which gives immediate output for each statement, while running previously statements in **active** memory. This mode is also referred as REPL (Read Evaluate Print Loop)



```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```

We can start an interactive session from Command Prompt Directly.



```
Microsoft Windows [Version 10.0.18362.418]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\FAZAL>python
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Fig.3: Command prompt

Normal mode: is where the scripted python file (.py) run in the Python interprete

CHAPTER-7: SYSTEM DESIGN

SYSTEM DESIGN

7.1 Architecture

Elements and Their Descriptions:

1. **Server (Leftmost):** Represented by a blue tower icon with multiple horizontal layers, symbolizing a typical server computer. This is likely the source or destination of network traffic being monitored.
2. **Cloud (Top Center):** A stylized cloud with the label "Packets from Network". This represents the network environment (e.g., the internet or a local network) from which data packets are traveling.
3. **Router (Center):** A rounded rectangle with antenna-like symbols and the label "Router". This signifies a network device responsible for forwarding data packets between different networks.[1]
4. **Firewall (Right Center):** A vertical rectangle with a brick-like pattern, labeled "Firewall". This represents a security system that controls network traffic based on predetermined rules, acting as a barrier against unauthorized access.
5. **Intrusion Detection System (Rightmost):** Represented by a computer monitor icon displaying three server icons and the label "Intrusion Detection System". This is the core component of the image, indicating the system responsible for monitoring network traffic for malicious activity.
6. **User (Bottom Left):** A stylized human figure, labeled "User", representing an individual or entity interacting with the network or monitoring the IDS.
7. **Laptop with "Check for Packets" (Bottom Center):** A laptop icon with the label "Check for Packets". This symbolizes the action of monitoring or inspecting network traffic data.
8. **Arrows and Flow:** The arrows connecting the elements indicate the direction of data flow. Data packets move from the network (cloud) through the router, firewall, and finally to the server or user. The connection between the laptop and the router suggests the user's monitoring activity.
9. **Labels:** Each element is accompanied by a text label, clearly identifying its function within the IDS framework.

Interpretation and Context:

The image illustrates a common scenario where network traffic passes through a router and firewall before reaching a server or user. The IDS is positioned to monitor this traffic, likely by inspecting packets and logs, to identify suspicious patterns or malicious activities. The user's laptop suggests an interface for monitoring the IDS alerts or configuring its settings.

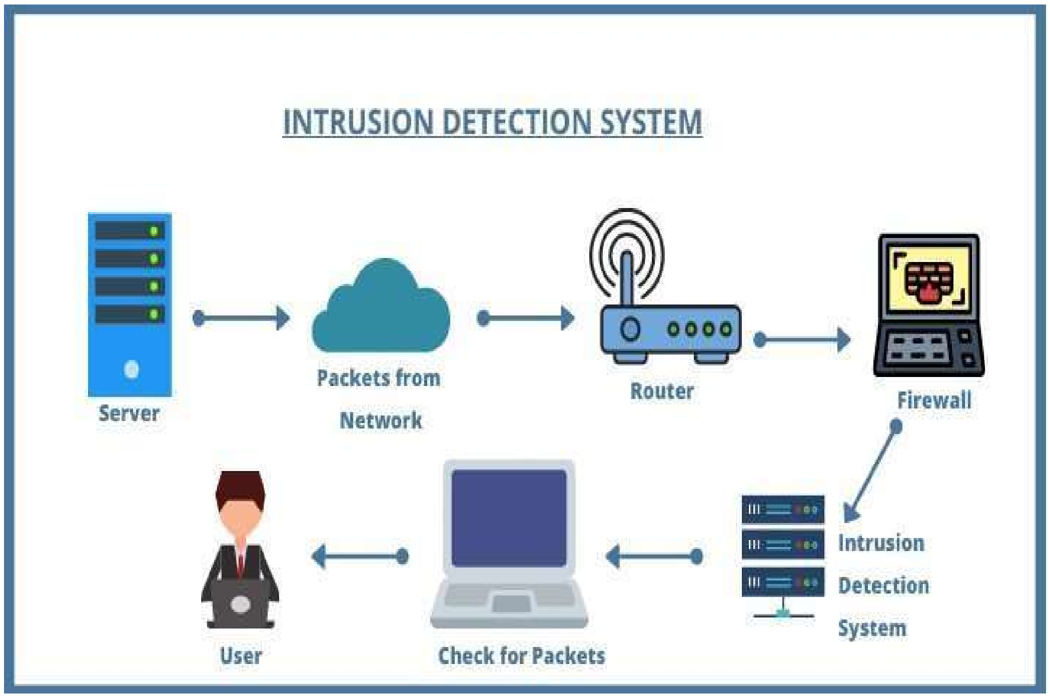
Educational Value:

This image serves as a valuable educational tool for understanding the basic components and data flow within an Intrusion Detection System. It simplifies the complex process of network security monitoring into a visually digestible format.

Potential Improvements:

While effective in its simplicity, the image could be enhanced by:

- **More Specific Icons**: Using more distinct icons for elements like the firewall and IDS could further clarify their functions.
- **Color Coding**: Employing color coding to differentiate between normal traffic flow and potential threats could enhance understanding.
- **Detailed Packet Inspection Visualization**: Adding a visual representation of packet inspection or anomaly detection could provide a deeper insight into the IDS process.



7.2 Introduction to UML:

UML, or Unified Modeling Language, is a standardized modeling language used in software engineering for visualizing, specifying, constructing, and documenting the artifacts of software systems. It's a blueprint language that helps developers and stakeholders understand the system's structure, behavior, and interactions. Think of it as the architectural drawings for a software project.

7.2.1 Use Case Diagram

UML (Unified Modelling Language) Use Case Diagram for an Intrusion Detection System (IDS). It visually represents the interactions between different actors (users or external systems) and the functionalities provided by the IDS. Let's break down the diagram's elements and their meanings:

Use Case Diagram

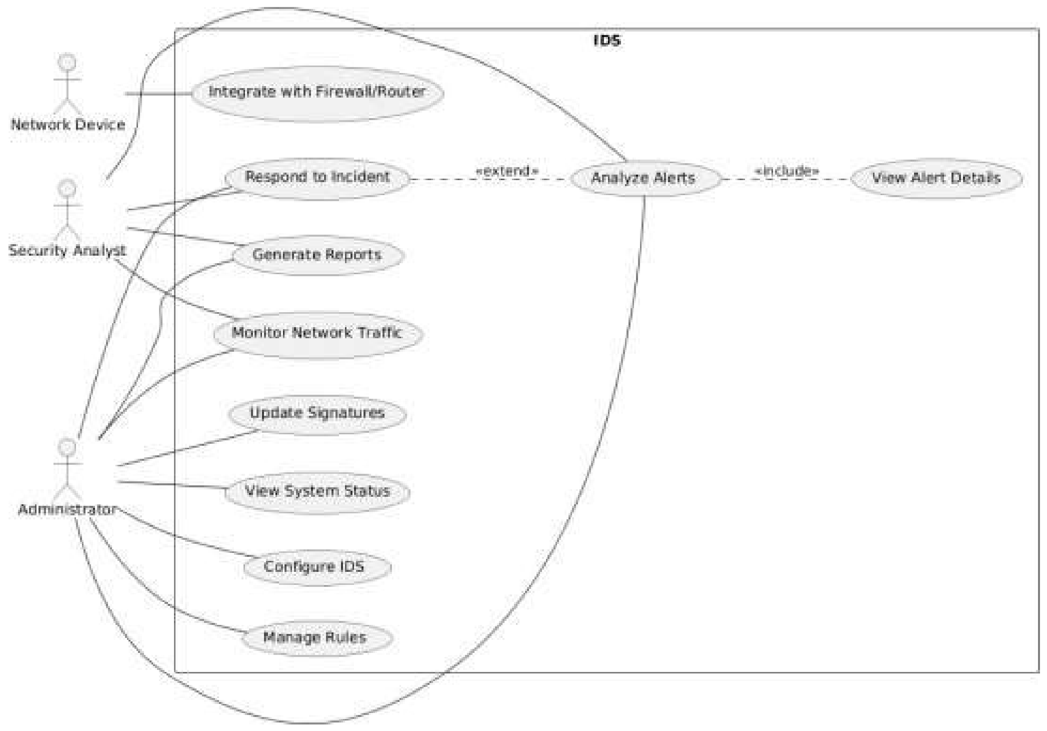


Fig.5: Use Case Diagram

Elements and their Descriptions:

1. **System Boundary (Rectangle):** The rectangle labeled "IDS" encloses all the use cases, representing the scope of the Intrusion Detection System. Everything inside the rectangle is a function or service provided by the IDS.

2. **Actors (Stick Figures):** These represent external entities that interact with the IDS
   - **Network Device**: Likely represents a router, firewall, or other network infrastructure component that the IDS might integrate with or receive data from.

- **Security Analyst**: A human analyst responsible for monitoring, analyzing, and responding to security alerts and incidents.
- **Administrator:** A user with elevated privileges responsible for configuring, managing, and maintaining the IDS.

3. **Use Cases (Ovals):** These represent specific functionalities or actions that the actors can perform with the IDS:
   - **Integrate with Firewall/Router**: Indicates the IDS's ability to work in conjunction with existing network security devices.
   - **Respond to Incident**: Represents the actions taken to address a
   - detected security breach or attack.
   - **Analyze Alerts**: The process of examining and interpreting security alerts generated by the IDS.
   - **View Alert Details:** Accessing detailed information about specific security alerts.
   - **Generate Reports:** Creating reports on security events, trends, and system status.
   - **Monitor Network Traffic:** The core function of observing and
   - analyzing network data for suspicious activity.
   - **Update Signatures:** Keeping the IDS's detection rules and patterns up- to-date.
   - **View System Status:** Checking the operational status and health of the IDS.
   - **Configure IDS:** Setting up and customizing the IDS's parameters and behavior.
   - **Manage Rules:** Defining, modifying, and managing the rules used by the IDS to detect intrusions.
4. **Relationships (Lines with Arrows):** These illustrate the interactions between actors and use cases:
   - **Solid Lines**: Connect actors to the use cases they participate in.
   - **Dashed Lines with Stereotypes:** Add more specific meaning to the

relationships:

- <<extend>>: Indicates that the "Respond to Incident" use case can be extended by the "Analyze Alerts" use case. This means that analyzing alerts is an optional step in responding to an incident.
- <<include>>: Indicates that the "Analyze Alerts" use case includes the "View Alert Details" use case.[9] This means that viewing alert details is a mandatory part of analyzing alerts.

Interpretation and Context:

The diagram shows the various ways users and external systems interact with the IDS. It highlights the key functions of the IDS, from basic network traffic monitoring to more advanced features like incident response and reporting. The use case diagram effectively communicates the scope and functionality of the IDS from a user-centric perspective.

7.2.2 Class Diagram

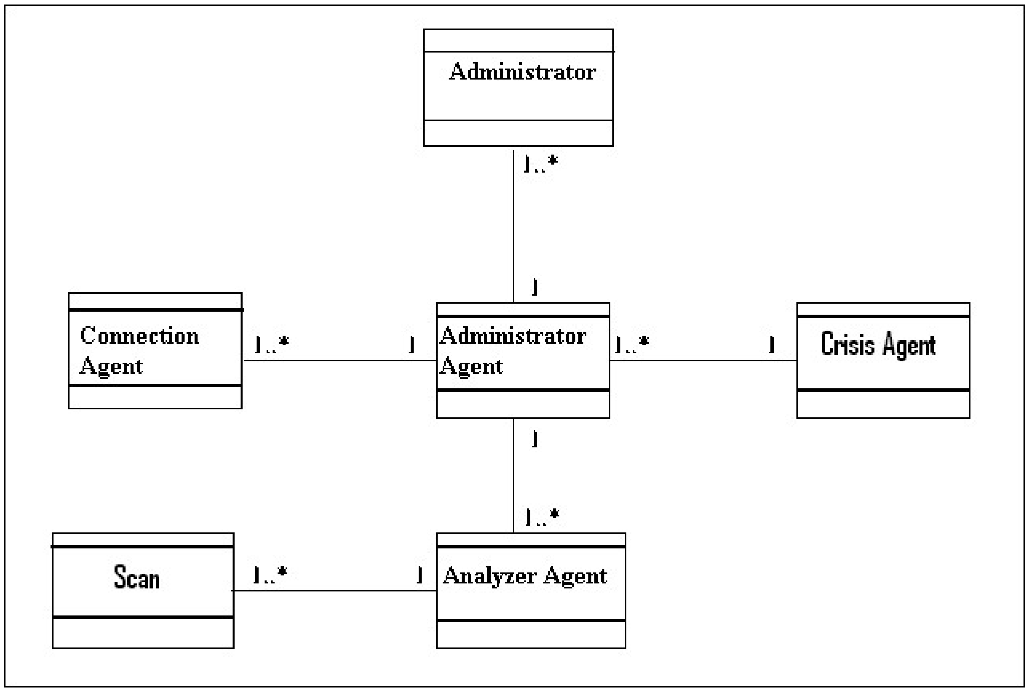


Fig.6: Class Diagram

1. Classes (Rectangles):

- **Administrator**: This class represents an administrative user or entity with high-level privileges.
- **Administrator Agent**: This acts as an intermediary or agent for the Administrator, potentially handling tasks or interactions on their behalf.
- **Connection Agent**: This class is responsible for managing connections, possibly network connections or connections to other systems/resources.
- **Crisis Agent**: This class handles crisis situations or critical events, likely responsible for taking appropriate actions during emergencies.
- **Scan**: This class represents a scanning process or component, possibly for security vulnerabilities or other relevant data.
- **Analyzer Agent:** This class is responsible for analyzing data, potentially the results of scans or other system information.

2. Relationships (Lines connecting the classes):

- **Association (Solid Lines):** These lines indicate a relationship or interaction between the classes. The labels at the ends of the lines specify the nature of the relationship.
- **Multiplicity (Numbers at the ends of the lines):** These numbers indicate the quantity of instances of one class that can be associated with an instance of another class.
- ***1.. (One to many)**:** Indicates that one instance of the class at this end can be associated with one or more instances of the class at the other end.

- **Administrator -- Administrator Agent (1..*):** One Administrator can be associated with one or more Administrator Agents.
- **Administrator Agent -- Connection Agent (1..*):** One Administrator Agent can be associated with one or more Connection Agents.
- **Administrator Agent -- Crisis Agent (1..*):** One Administrator Agent can be associated with one or more Crisis Agents.
- **Administrator Agent -- Analyzer Agent (1..*):** One Administrator Agent can be associated with one or more Analyzer Agents.
- **Analyzer Agent -- Scan (1..*):** One Analyzer Agent can be associated with one or more Scans.

7.2.3 Sequence Diagram

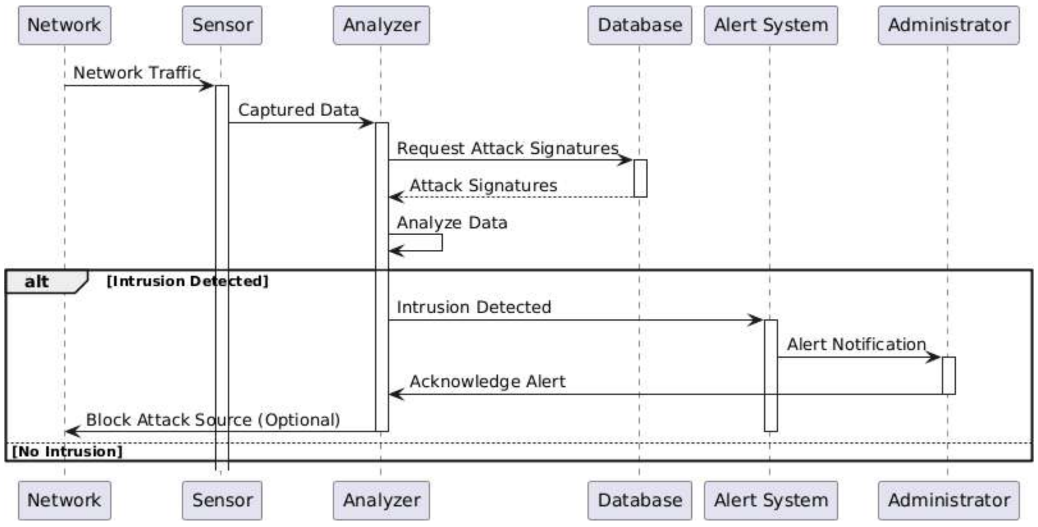


Fig.7: Sequence Diagram

This image is a Sequence Diagram illustrating the flow of events and interactions within an Intrusion Detection System (IDS). It visually depicts the sequence of actions taken by different components of the IDS in response to network traffic, including the detection of an intrusion.

Let's break down the elements and their meanings:

1. Lifelines (Vertical Dashed Lines):

These represent the participants or components in the interaction, depicted as vertical dashed lines with a box containing the component's name at the top. The participants shown are

- **Network:** Represents the network being monitored.
- **Sensor:** A component that captures network traffic.
- **Analyzer**: A component that analyzes the captured data.
- **Database**: Stores attack signatures or other relevant data.
- **Alert System**: A component responsible for generating alerts.
- **Administrator**: A user or entity managing the IDS.

1. **Activation Bars (Rectangles on Lifelines):**

   - These represent the period during which a participant is active in the interaction. The longer the bar, the longer the participant is involved in the process.

2. **Messages (Arrows):**

   - These represent communication or actions between participants. The direction of the arrow indicates the direction of the message. Different arrow styles represent different types of messages.
   - Solid arrow with a filled arrowhead: Represents a synchronous call (the sender waits for a response).
   - Solid arrow with an open arrowhead: Represents an asynchronous call (the sender doesn't wait for a response).

- Dashed arrow with an open arrowhead: Represents a return message.

Sequence of Events:

1. **Network Traffic:** The process begins with "Network Traffic" flowing through the network.
2. **Captured Data:** The Sensor captures this network traffic and generates "Captured Data".
3. **Request Attack Signatures:** The Analyzer requests "Attack Signatures" from the Database.
4. **Attack Signatures:** The Database responds with "Attack Signatures".
5. **Analyze Data:** The Analyzer analyzes the "Captured Data" using the "Attack Signatures".
6. **alt [Intrusion Detected]:** This indicates an alternative flow or conditional logic. If an intrusion is detected:
   - **Intrusion Detected:** The Analyzer detects an intrusion.[9]
   - **Alert Notification**: The Analyzer sends an "Alert Notification" to the Alert System.
   - **Alert Notification**: The Alert System notifies the Administrator.[10]
   - **Acknowledge Alert**: The Administrator acknowledges the alert.
   - **Block Attack Source (Optional):** The Administrator can optionally choose to block the attack source.
7. **[No Intrusion]:** If no intrusion is detected, the process ends without further action.

Interpretation and Context:

The diagram clearly shows the sequence of steps involved in intrusion detection:

- **Traffic Capture:** The Sensor captures network traffic.
- **Analysis:** The Analyzer retrieves attack signatures and analyzes the captured data.
- **Detection:** The Analyzer determines if an intrusion has occurred.
- **Alerting:** If an intrusion is detected, the Alert System notifies the Administrator.
- **Response:** The Administrator can then take action, such as blocking the attack source.

7.2.4 Collaboration Diagram:

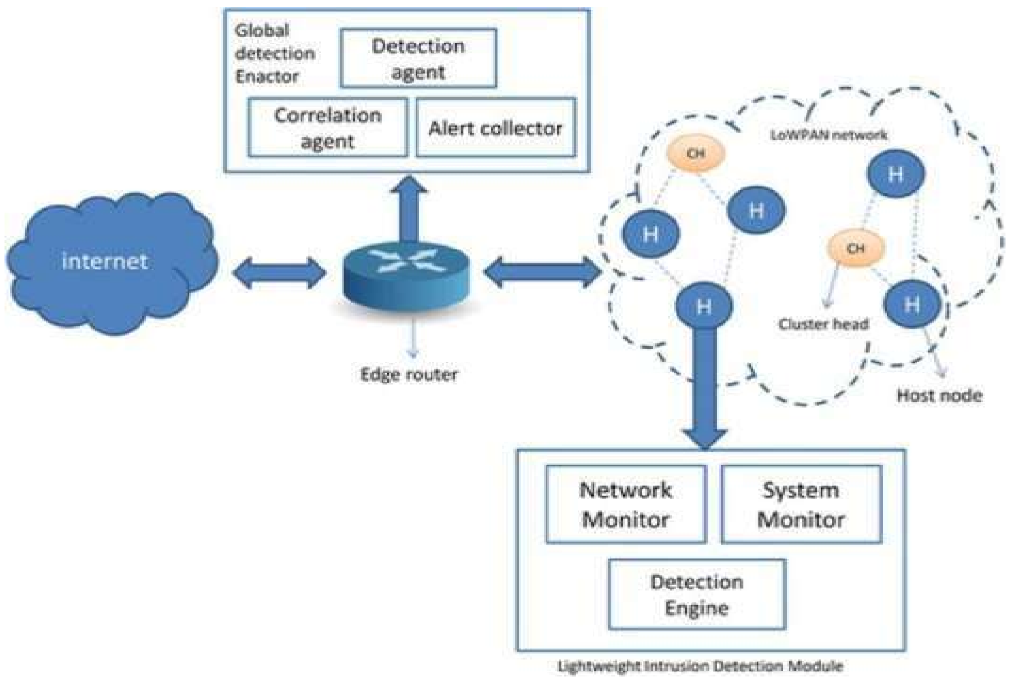


Fig.8: Collaboration Diagram

This diagram illustrates COLIDE, a collaborative intrusion detection framework for IoT. It features a hierarchical structure with

- **LoWPAN:** A network of resource-constrained IoT devices (H - hosts, CH - cluster heads).
- **Edge Router**: Connects LoWPAN to the internet, hosting network/system monitors and a detection engine.
- **Global Detection Enactor**: A centralized system that correlates alerts from the edge and makes decisions.

The system uses layered detection, from lightweight modules in the LoWPAN to comprehensive analysis at the edge and global levels, to secure IoT networks.

7.3 INPUT AND OUTPUT DESIGN

7.3.1 INPUT DESIGN

Network Traffic:

- **Packet Headers:** Source/destination IPs, ports, protocols (TCP, UDP, ICMP), flags, timestamps. Essential for network anomaly detection and signature-based attacks.
- **Packet Payloads**: The actual data within the packets. Needed for deep packet inspection (DPI) to detect malicious code, exploits, or sensitive data exfiltration. (Consider privacy implications and legal restrictions for payload inspection).
- **Flow Data:** Summarized network traffic over a period (e.g., total bytes, packets, duration). Useful for bandwidth analysis, DDoS detection, and identifying unusual traffic patterns. Tools like NetFlow or IPFIX can provide this.

System Logs:

- **Firewall Logs:** Blocked connections, dropped packets, intrusion attempts.
- **Operating System Logs:** User logins/logouts, file access, process creation, system events. Crucial for detecting insider threats, privilege escalation, and malware activity.
- **Application Logs:** Web server logs (access attempts, errors), database logs (queries, modifications), authentication logs. Provide insights into application-specific attacks.

Security Events:

- **Honeypot Activity:** Logs from honeypots that attract attackers, providing valuable information on attack techniques.
- **Vulnerability Scanner Reports:** Output from vulnerability scans highlighting potential weaknesses in the system. IDS can use this to prioritize monitoring.
- **Threat Intelligence Feeds:** Data from external sources about known threats, attack patterns, and malicious IPs. Used to update IDS rules and improve detection accuracy.

User/Entity Behavior Data (UEBA):

- **User Login Patterns:** Time of day, location, devices used.
- **File Access Patterns:** Files accessed, frequency, permissions.
- **Application Usage:** Applications used, duration, data accessed.
- **Device Activity (IoT Specific):** Sensor readings, actuator commands, communication patterns. Essential for detecting anomalies in IoT environments.

Input Design Considerations:

- **Data Volume:** Network traffic and logs can be voluminous. Efficient data collection, filtering, and storage mechanisms are essential.
- **Data Fidelity:** Ensure the data is accurate and reliable. Compromised sensors or log manipulation can undermine the IDS.
- **Data Correlation:** The IDS should be able to correlate data from different sources to get a comprehensive view of potential attacks.

7.3.2 Output Design:

The IDS needs to provide meaningful information to security personnel or trigger automated responses. Here are key outputs:

Alerts:

- **Detailed Information:** Timestamp, source/destination IPs, type of attack, severity, affected systems.
- **Prioritization:** Rank alerts based on severity and potential impact.
- **Correlation:** Group related alerts to provide context and identify complex attacks.

Reports:

- **Summary Reports:** Overview of security events over a period.
- **Detailed Reports:** In-depth analysis of specific incidents.
- **Trend Analysis**: Identify patterns and trends in attack activity.

Visualizations:

- **Dashboards:** Real-time view of network security status.
- **Graphs and Charts:** Visualize attack trends, traffic anomalies, and system.

Automated Responses (IPS Functionality):

- **Blocking Traffic:** Drop malicious packets or block IP addresses.
- **Quarantining Devices:** Isolate compromised devices from the network.
- **Triggering Other Security Tools:** Notify firewalls, SIEM systems, or other security components.

Logs:

- **Detailed Logs of Detected Events:** For forensic analysis and incident response.
- **Logs of IDS Activity:** For auditing and troubleshooting the IDS itself.

Output Design Considerations:

- **Alert Fatigue:** Avoid generating too many false positives. Proper tuning and correlation are crucial.
- **Actionable Information:** Provide clear and concise information that security personnel can use to take action.
- **Integration with Other Systems:** Integrate with SIEM systems, firewalls, and other security tools for a coordinated response.
- **IoT Specifics:** Consider the limited display capabilities of some IoT devices. Simple alerts or status indicators might be necessary.

COLIDE Framework Considerations:

- **Distributed Alerts:** The COLIDE framework emphasizes collaboration. Alerts need to be efficiently communicated between the edge devices and the global enactor.
- **Lightweight Alerts**: IoT devices might need to generate simplified alerts to conserve bandwidth and processing power.
- **Correlation at the Enactor:** The global enactor needs to correlate alerts from multiple sources to get a comprehensive view of potential attacks.

CHAPTER-8: SOURCE CODE

Source code

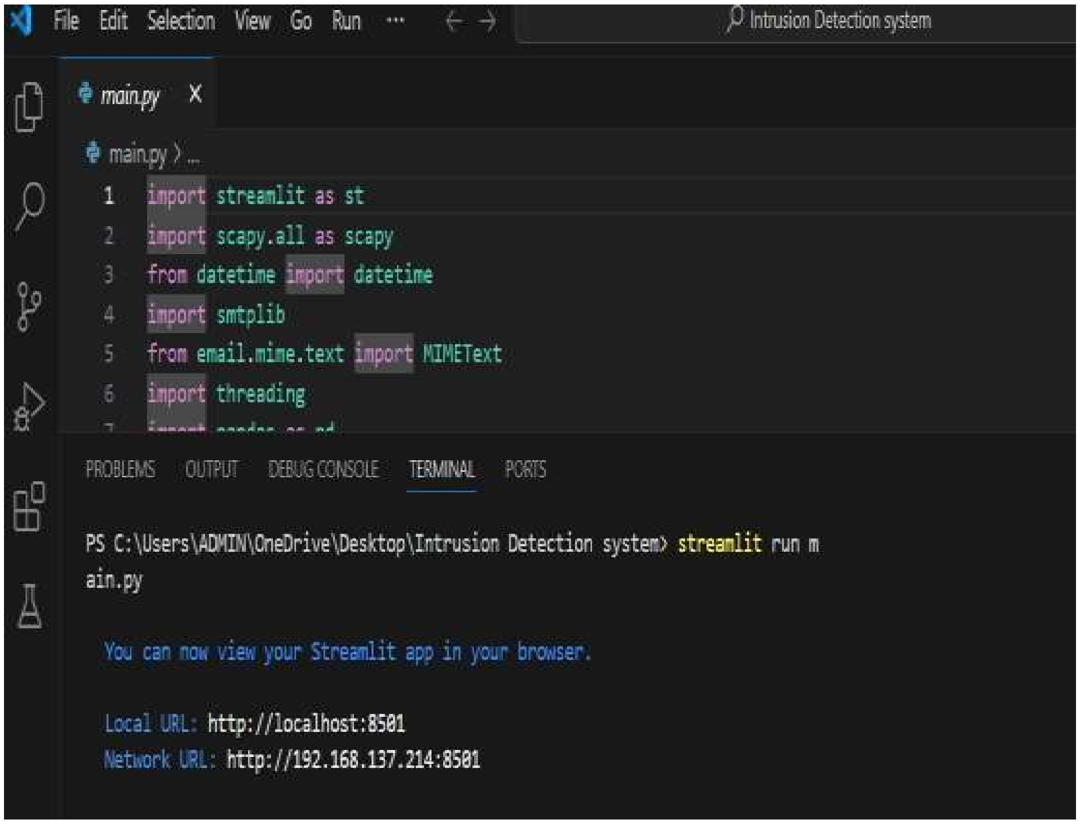
```
import streamlit as st
import scapy.all as scapy
from datetime import datetime
import smtplib
from email.mime.text import MIMEText
import threading
import pandas as pd
import time
# Global Variables
packet_logs = []
 alerts = []
custom_rules = []
sent_alerts = set()  # To limit duplicate email alerts per session
alert_threshold = 10
baseline_traffic = {}
signature_database = [
{'pattern': 'SYN  flood', 'protocol': 6},
{'pattern': 'ARP spoof', 'protocol': 2054}
]

# Control Flags
stop_flag = threading.Event()

# Packet Capture and Analysis
def capture_packets():
        def process_packet(packet):
                if stop_flag.is_set():
                        return
                        if packet.haslayer(scapy.IP):
                                src_ip = packet[scapy.IP].src
                                dst_ip = packet[scapy.IP].dst
        protocol = packet[scapy.IP].proto\ timestamp = datetime.now()
        # Log the packet
        packet_logs.append((timestamp, src_ip, dst_ip, protocol))
```

```
# Perform anomaly and signature detection

check_anomalies(src_ip, dst_ip)

check_signatures(src_ip, dst_ip, protocol)

scapy.sniff(prn=process_packet, store=False, stop_filter=lambda x: stop_flag.is_set())


# Signature-Based Detection

def check_signatures(src_ip, dst_ip, protocol):
        for signature in signature_database:
           if protocol == signature['protocol']:
              alert_message = f"Detected: {signature['pattern']} from {src_ip} to
              {dst_ip}"
        if alert_message not in sent_alerts:
               alerts.append((datetime.now(), src_ip, dst_ip, signature['pattern']))
               send_alert(alert_message) sent_alerts.add(alert_message)


# Anomaly Detection
def check_anomalies(src_ip, dst_ip):

        global baseline_traffic

        key = f"{src_ip}-{dst_ip}"
        baseline_traffic[key] = baseline_traffic.get(key, 0) + 1

        if baseline_traffic[key] > alert_threshold:
                alert_message = f"Anomalous traffic from {src_ip} to {dst_ip}"
        if alert_message not in sent_alerts:
                alerts.append((datetime.now(), src_ip, dst_ip, 'Anomaly detected'))
                send_alert(alert_message)
                sent_alerts.add(alert_message)

Send Alerts via Email with Rate Limiting def send_alert(message):

        try:
        sender = "malleswarigottipati404@gmail.com"
        receiver = "pasamkoteswararao07@gmail.com"
        #Corrected domain to gmail.com
       password = "asrv keul kmcu zdzj"
       # Replace with the generated App  Password

msg = MIMEText(message)
msg['Subject'] = "Intrusion Detection Alert"
msg['From'] = sender
msg['To'] = receiver
```

```
with smtplib.SMTP_SSL('smtp.gmail.com', 465) as server:
          server.login(sender, password) server.sendmail(sender, receiver,

          msg.as_string())

          print(f"Alert sent: {message}")

# Add a small delay to prevent overwhelming the server time.sleep(1)

          except smtplib.SMTPException as e: print(f"SMTP error: {e}")


           except Exception as e:
                    print(f"Error sending alert: {e}")

                    # Streamlit Dashboard with Dynamic Updates

                    def main():

                              st.title("Intrusion Detection System")
                              # Dynamic placeholders for packet logs, alerts, and rules

                              logs_placeholder = st.empty()

                              alerts_placeholder = st.empty()

                              rules_placeholder = st.empty()


                              # Start/Stop Buttons
                              if st.button("Start Detection"):

                              stop_flag.clear()

                              threading.Thread(target=capture_packets,daemon=True).start()


if st.button("Stop Detection"):

          stop_flag.set()

          # Custom Rule Creation

           st.subheader("Custom Rule Creation")

          src_ip = st.text_input("Source IP")

          dst_ip = st.text_input("Destination IP")


          if st.button("Add Rule"):
                    rule = {'Source IP': src_ip, 'Destination IP': dst_ip}

                    custom_rules.append(rule)

                    alert_message = f"Custom rule triggered: {src_ip} -> {dst_ip}"
                    alerts.append((datetime.now(), src_ip, dst_ip, "Custom Rule
                    Triggered"))
            if alert_message not in sent_alerts:
                    send_alert(alert_message)
                    sent_alerts.add(alert_message)
                    # Continuously update the interface while detection is running
                    while not stop_flag.is_set():
                    # Update packet logs dynamically
                    logs_placeholder.subheader("Packet Logs")
                    packet_df = pd.DataFrame(packet_logs, columns=['Timestamp',
                    'Source IP', 'Destination IP', 'Protocol'])
                    logs_placeholder.dataframe(packet_df)

# Update alerts dynamically
alerts_placeholder.subheader("Alerts"\jnjmnmmmmmmmmmmmmmmmmmmmmmm
mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
mmmmmmmmmmmmmmmmmmmmmm'        '0 '

alerts_df = pd.DataFrame(alerts, columns=['Timestamp', 'Source IP', 'Destination IP',
'Alert Type'])
alerts_placeholder.dataframe(alerts_df)

# Update custom rules dynamically rules_
placeholder.subheader("Custom Rules")
rules_df = pd.DataFrame(custom_rules)
rules_placeholder.dataframe(rules_df)

# Refresh the UI every second
time.sleep(1)
# Final update when the detection stops st.success("Detection stopped.")
logs_placeholder.dataframe(pd.DataFrame(packet_logs,
columns=['Timestamp', 'Source IP', 'Destination IP',
'Protocol']))
    alerts_placeholder.dataframe(pd.DataFrame(alerts, columns=['Timestamp', 'Source
IP', 'Destination IP', 'Alert Type']))
rules_placeholder.dataframe(pd.DataFrame(custom_rules
))


# Main Function
      if __name__== "__main__":
main()
```

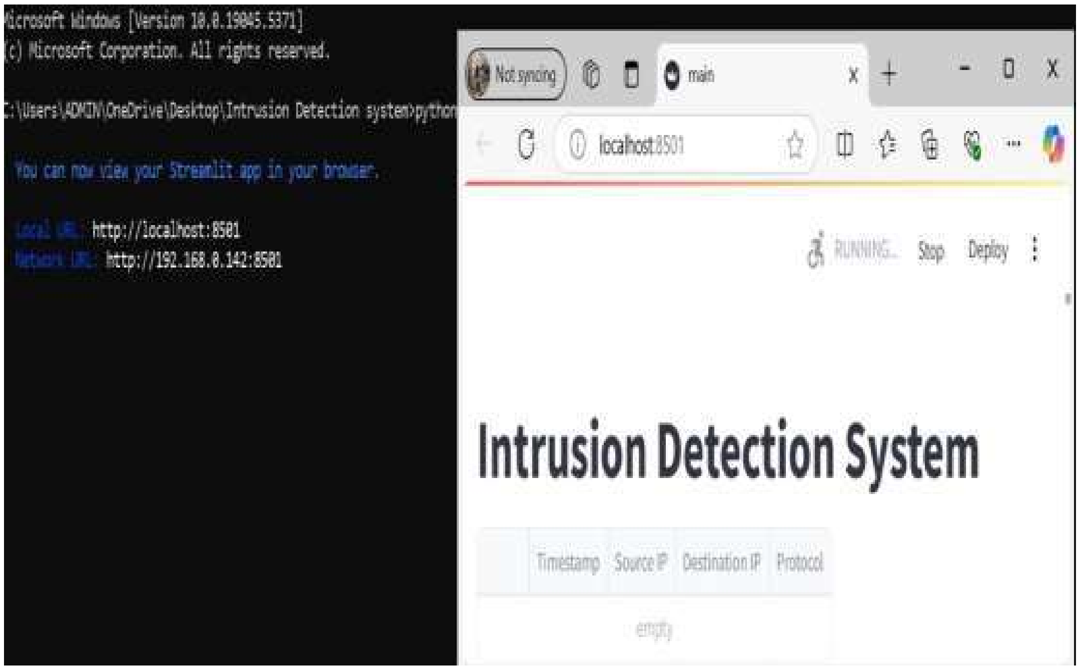
CHAPTER -9: SCREENSHOTS

OUTPUT:


File Edit Selection View Go Run ···
Intrusion Detection system
main.py
main.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
import streamlit as st
import scapy.all as scapy
from datetime import datetime
import smtplib
from email.mime.text import MIMEText
import threading
```

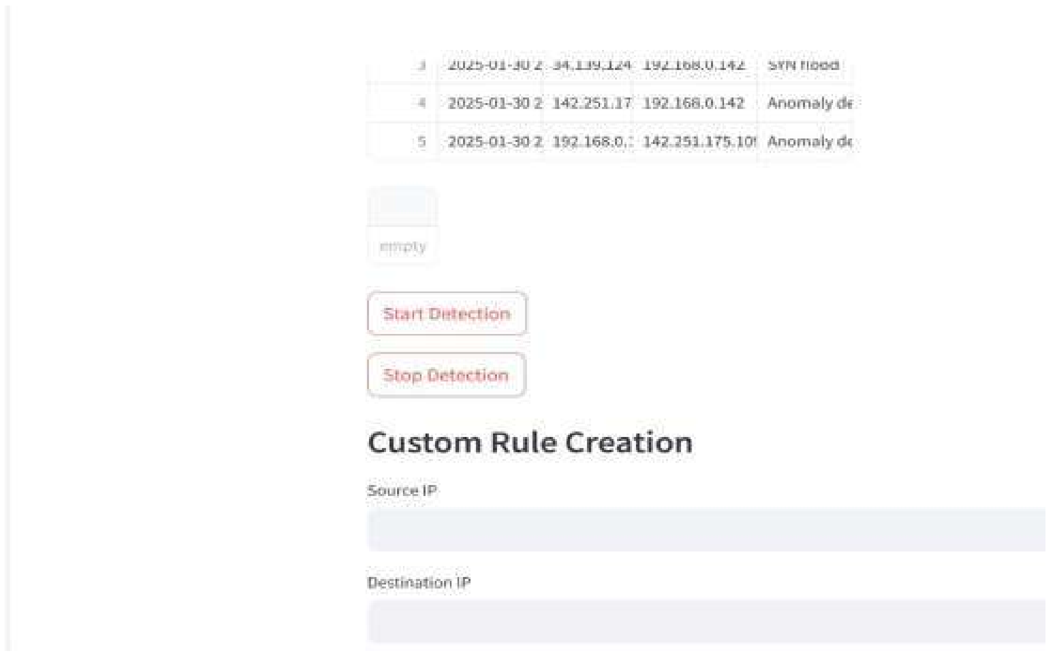
```
PS C:\Users\ADMIN\OneDrive\Desktop\Intrusion Detection system> streamlit run m
ain.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.137.214:8501
```



```
Microsoft Windows [Version 10.0.19045.5371]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ADMIN\OneDrive\Desktop\Intrusion Detection system>python

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.0.142:8501
```

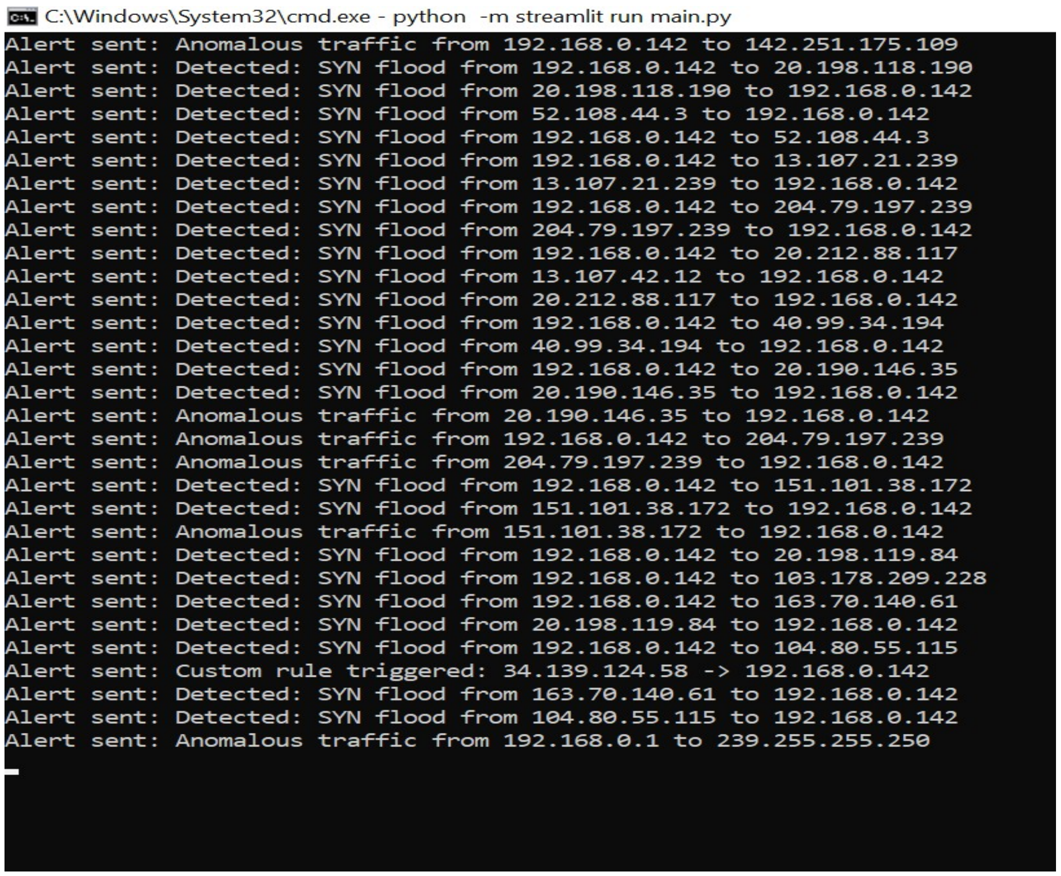
Intrusion Detection System

	Timestamp	Source IP	Destination IP	Protocol



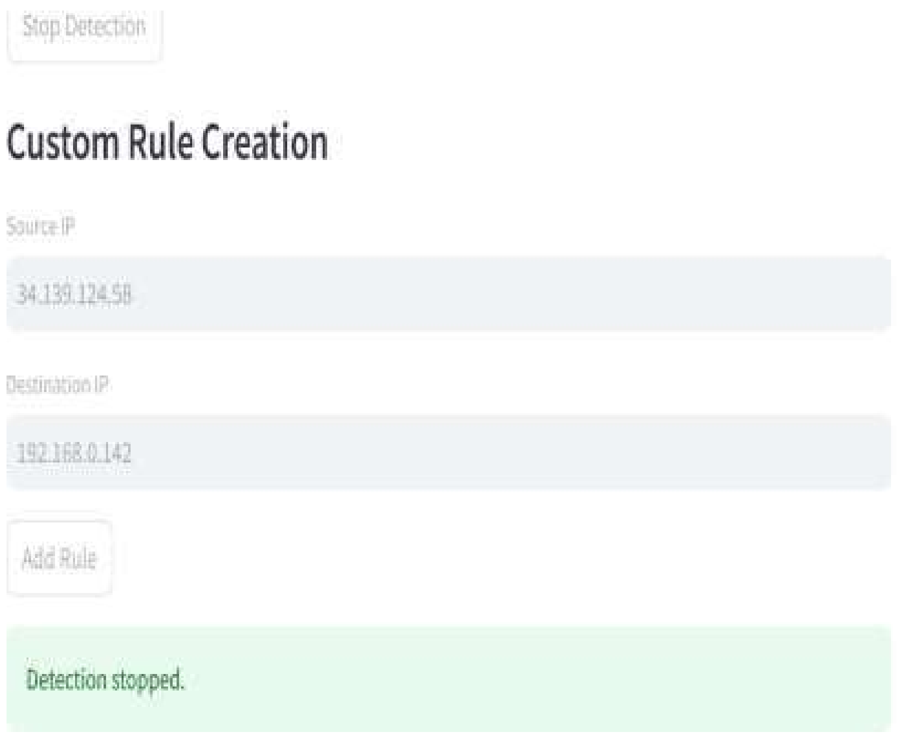
3	2025-01-30 2	34.139.124	192.168.0.142	SYN flood
4	2025-01-30 2	142.251.17	192.168.0.142	Anomaly de
5	2025-01-30 2	192.168.0.1	142.251.175.10	Anomaly de

Custom Rule Creation



C:\Windows\System32\cmd.exe - python -m streamlit run main.py

```
Alert sent: Anomalous traffic from 192.168.0.142 to 142.251.175.109
Alert sent: Detected: SYN flood from 192.168.0.142 to 20.198.118.190
Alert sent: Detected: SYN flood from 20.198.118.190 to 192.168.0.142
Alert sent: Detected: SYN flood from 52.108.44.3 to 192.168.0.142
Alert sent: Detected: SYN flood from 192.168.0.142 to 52.108.44.3
Alert sent: Detected: SYN flood from 192.168.0.142 to 13.107.21.239
Alert sent: Detected: SYN flood from 13.107.21.239 to 192.168.0.142
Alert sent: Detected: SYN flood from 192.168.0.142 to 204.79.197.239
Alert sent: Detected: SYN flood from 204.79.197.239 to 192.168.0.142
Alert sent: Detected: SYN flood from 192.168.0.142 to 20.212.88.117
Alert sent: Detected: SYN flood from 13.107.42.12 to 192.168.0.142
Alert sent: Detected: SYN flood from 20.212.88.117 to 192.168.0.142
Alert sent: Detected: SYN flood from 192.168.0.142 to 40.99.34.194
Alert sent: Detected: SYN flood from 40.99.34.194 to 192.168.0.142
Alert sent: Detected: SYN flood from 192.168.0.142 to 20.190.146.35
Alert sent: Detected: SYN flood from 20.190.146.35 to 192.168.0.142
Alert sent: Anomalous traffic from 20.190.146.35 to 192.168.0.142
Alert sent: Anomalous traffic from 192.168.0.142 to 204.79.197.239
Alert sent: Anomalous traffic from 204.79.197.239 to 192.168.0.142
Alert sent: Detected: SYN flood from 192.168.0.142 to 151.101.38.172
Alert sent: Detected: SYN flood from 151.101.38.172 to 192.168.0.142
Alert sent: Anomalous traffic from 151.101.38.172 to 192.168.0.142
Alert sent: Detected: SYN flood from 192.168.0.142 to 20.198.119.84
Alert sent: Detected: SYN flood from 192.168.0.142 to 103.178.209.228
Alert sent: Detected: SYN flood from 192.168.0.142 to 163.70.140.61
Alert sent: Detected: SYN flood from 20.198.119.84 to 192.168.0.142
Alert sent: Detected: SYN flood from 192.168.0.142 to 104.80.55.115
Alert sent: Custom rule triggered: 34.139.124.58 -> 192.168.0.142
Alert sent: Detected: SYN flood from 163.70.140.61 to 192.168.0.142
Alert sent: Detected: SYN flood from 104.80.55.115 to 192.168.0.142
Alert sent: Anomalous traffic from 192.168.0.1 to 239.255.255.250
```


Stop Detection

Custom Rule Creation

Source IP

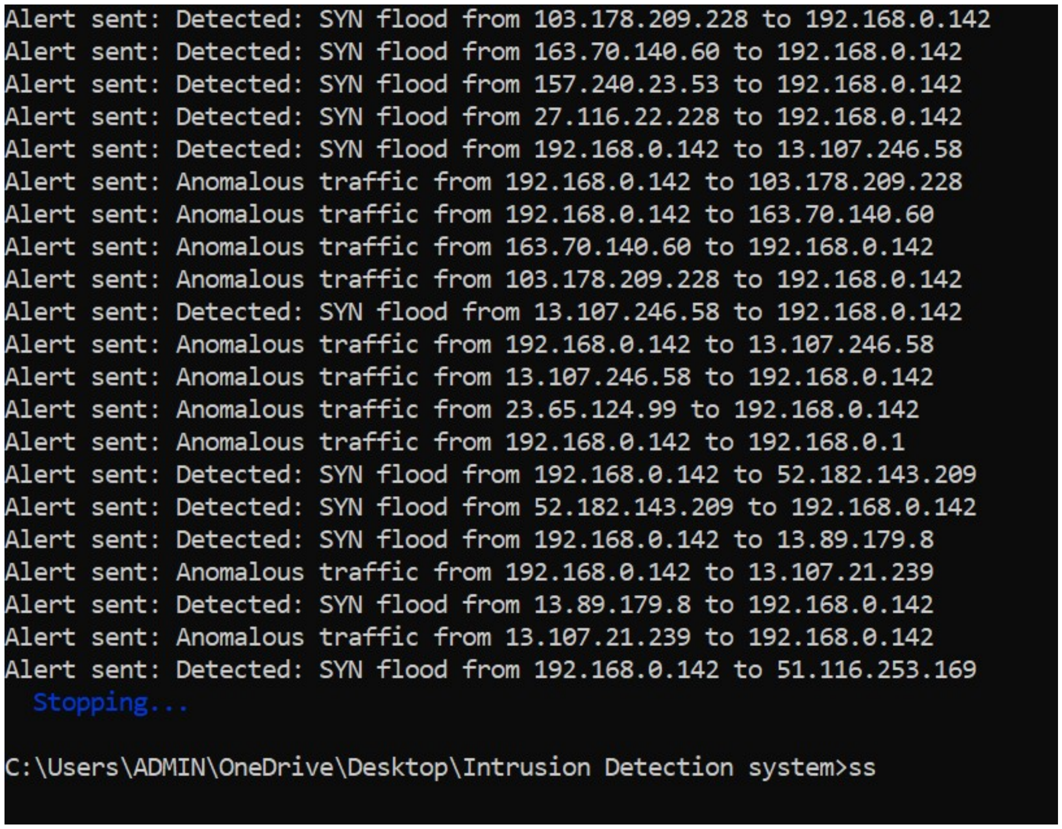
34.139.124.58

Destination IP

192.168.0.142

Add Rule

Detection stopped.


```
Alert sent: Detected: SYN flood from 103.178.209.228 to 192.168.0.142
Alert sent: Detected: SYN flood from 163.70.140.60 to 192.168.0.142
Alert sent: Detected: SYN flood from 157.240.23.53 to 192.168.0.142
Alert sent: Detected: SYN flood from 27.116.22.228 to 192.168.0.142
Alert sent: Detected: SYN flood from 192.168.0.142 to 13.107.246.58
Alert sent: Anomalous traffic from 192.168.0.142 to 103.178.209.228
Alert sent: Anomalous traffic from 192.168.0.142 to 163.70.140.60
Alert sent: Anomalous traffic from 163.70.140.60 to 192.168.0.142
Alert sent: Anomalous traffic from 103.178.209.228 to 192.168.0.142
Alert sent: Detected: SYN flood from 13.107.246.58 to 192.168.0.142
Alert sent: Anomalous traffic from 192.168.0.142 to 13.107.246.58
Alert sent: Anomalous traffic from 13.107.246.58 to 192.168.0.142
Alert sent: Anomalous traffic from 23.65.124.99 to 192.168.0.142
Alert sent: Anomalous traffic from 192.168.0.142 to 192.168.0.1
Alert sent: Detected: SYN flood from 192.168.0.142 to 52.182.143.209
Alert sent: Detected: SYN flood from 52.182.143.209 to 192.168.0.142
Alert sent: Detected: SYN flood from 192.168.0.142 to 13.89.179.8
Alert sent: Anomalous traffic from 192.168.0.142 to 13.107.21.239
Alert sent: Detected: SYN flood from 13.89.179.8 to 192.168.0.142
Alert sent: Anomalous traffic from 13.107.21.239 to 192.168.0.142
Alert sent: Detected: SYN flood from 192.168.0.142 to 51.116.253.169
  Stopping...

C:\Users\ADMIN\OneDrive\Desktop\Intrusion Detection system>ss
```

CHAPTER-10: SYSTEM TESTING

System Testing of Intrusion Detection Systems (IDS)

Intrusion Detection Systems (IDS) are crucial for network security, acting as sentinels that monitor network traffic and system activity for malicious behaviour. To ensure their effectiveness, rigorous system testing is essential. This testing process evaluates the IDS's ability to accurately detect intrusions, minimize false alarms, and maintain performance under various conditions.

Types of IDS and Testing Considerations

Before diving into testing methodologies, it's important to understand the different types of IDS and their unique characteristics:

- **Network-based IDS (NIDS):** Monitors network traffic for suspicious patterns. Testing focuses on its ability to analyse high volumes of data, identify various attack signatures, and handle network protocols.
- **Host-based IDS (HIDS)**: Monitors activity on individual systems. Testing emphasizes its capacity to detect malicious file changes, registry modifications, and unauthorized process execution.
- **Signature-based IDS**: Detects intrusions by comparing network traffic or system activity to known attack patterns (signatures). Testing involves exposing the IDS to a wide range of known attacks to verify its detection capabilities.
- **Anomaly-based IDS**: Identifies deviations from normal behaviour. Testing focuses on its ability to establish a baseline of normal activity, adapt to changes in network traffic, and minimize false positives.

10.1 Key Aspects of IDS System Testing

1. **Functional Testing:**
   - **Attack Simulation:** Introduce various types of attacks (e.g., port scans, denial-of-service attacks, malware) to assess the IDS's ability to detect and alert on them.
   - **False Positive Testing**: Subject the IDS to normal network traffic and system activity to ensure it doesn't trigger alerts unnecessarily.
   - **Alert Accuracy**: Verify that the IDS provides accurate information about detected intrusions, including the type of attack, source and destination IP addresses, and timestamps.
   - **Evasion Testing:** Attempt to bypass the IDS using techniques like traffic fragmentation, encryption, or obfuscation to evaluate its resilience. Performance Testing:
   - **Traffic Load**: Evaluate the IDS's performance under different network traffic loads to ensure it can handle peak activity without performance degradation.
   - **Resource Utilization**: Monitor the IDS's CPU, memory, and disk usage to ensure it doesn't consume excessive resources.
   - **Scalability**: Assess the IDS's ability to scale to accommodate growing network traffic and the addition of new systems.

2. **Security Testing:**
   - **Vulnerability Scanning**: Identify potential vulnerabilities in the IDS itself that could be exploited by attackers.

- **Penetration Testing**: Attempt to compromise the IDS to gain unauthorized access or disable its functionality.
- **Security Hardening:** Ensure the IDS is configured securely, with strong passwords, access controls, and up-to-date patches.

Best Practices for IDS System Testing

- Define clear testing objectives and criteria.
- Develop a comprehensive test plan that covers all aspects of the IDS.
- Use a variety of testing techniques and tools.
- Document all test results and findings.

TYPES OF TESTS:

1. Unit testing

Unit testing focuses on verifying the functionality of individual components or modules of the IDS in isolation. It's a fundamental step in the software development lifecycle, ensuring that each unit performs as expected before being integrated into the larger system. Why Unit Test an IDS?

- **Early Bug Detection**: Identify and fix defects in individual components early in the development process, reducing the cost and effort of debugging later.
- **Code Quality:** Encourage developers to write modular, testable code, leading to better design and maintainability.
- **Regression Prevention**: Ensure that changes to one part of the IDS don't unintentionally break other parts.
- **Improved Confidence**: Increase confidence in the reliability and correctness of the IDS.

What to Unit Test in an IDS?

The specific units to be tested will vary depending on the IDS architecture, but some common examples include:

- **Packet Parsing Module**: Verify that the module correctly extracts relevant information from network packets (e.g., protocol, source/destination IP addresses, ports).
- **Signature Matching Engine**: Test the engine's ability to accurately compare network traffic against known attack signatures.
- **Anomaly Detection Algorithm**: Evaluate the algorithm's ability to establish a baseline of normal behavior and identify deviations.
- **Alert Generation Module**: Ensure that the module generates alerts with the correct information and severity levels.

How to Unit Test an IDS?

1. **Isolate the Unit**: Focus on testing one unit at a time, mocking or stubbing any dependencies on other modules.
2. **Define Test Cases**: Create a set of test cases that cover various scenarios, including valid inputs, invalid inputs, edge cases, and error conditions.
3. **Use Testing Frameworks**: Utilize unit testing frameworks (e.g., JUnit, pytest) to automate test execution, assertion checking, and report generation.
4. **Write Assertions**: Define clear assertions to verify the expected behavior of the unit under test.

10.2 Integration testing

Integration testing focuses on verifying the interaction and communication between different modules or components of the IDS after they have been unit tested. It ensures that these units work together correctly as a cohesive system. This is crucial because even if individual components function perfectly in isolation, issues can arise when they are combined.

- **Monitor System Behavior**: Observe the behavior of the integrated system during testing, paying attention to network traffic, system logs, and generated alerts.
- **Verify Results:** Compare the actual results with the expected results to determine if the integration is successful.

Challenges in Integration Testing an IDS:

- **Complexity:** IDS can be complex systems with multiple interacting components, making integration testing challenging.
- **Environment Setup**: Setting up a realistic test environment can be time-consuming and resource-intensive.
- **Dependency Management**: Managing dependencies between different modules can be complex.

Best Practices for Integration Testing an IDS:

- **Plan Integration Testing Early:** Start planning integration testing during the design phase of the IDS.
- **Use a Phased Approach**: Integrate modules incrementally, starting with the most critical ones.
- **Automate Testing**: Automate integration tests to ensure frequent and consistent testing.
- **Document Test Results:** Maintain detailed records of integration test results.
- **Continuous Integration**: Integrate integration testing into the continuous integration pipeline.

10.3 Functional test

Functional testing of an Intrusion Detection System (IDS) is all about verifying that it performs its intended functions correctly. This means checking if it can accurately detect intrusions, generate alerts, and take any necessary actions, all while minimizing false alarms.

Here's a breakdown of what functional testing of an IDS involves:

1. **Defining Test Objectives:**
   - What types of attacks should the IDS detect? (e.g., port scans, denial-of- service attacks, malware, SQL injection)
   - What actions should the IDS take upon detection? (e.g., generate alerts, block traffic, log events)
   - What level of accuracy is required? (e.g., minimize false positives and false negatives)

2. **Creating Test Cases:**
   - **Attack Simulation**: Use tools or scripts to simulate various types of attacks. This could involve sending malicious network traffic, attempting unauthorized access to systems, or executing malicious code.
   - **Normal Traffic Generation**: Generate normal network traffic and system activity to ensure the IDS doesn't trigger false alarms.
   - **Edge Cases:** Test the IDS with unusual or borderline scenarios to see how it handles them.

3. **Setting up the Test Environment:**
   - **Realistic Network**: Create a test network that mimics a real-world environment, including servers, clients, and network devices.
   - **IDS Configuration**: Configure the IDS to monitor the network traffic and system activity in the test environment.
   - **Attack Tools**: Use tools like Nmap, Metasploit, or Scapy to simulate attacks.

4. **Executing the Tests:**
   - **Run Attack Simulations**: Launch the simulated attacks against the test network.
   - **Monitor IDS Response**: Observe the IDS's behavior, including whether it detects the attacks, generates alerts, and takes appropriate actions.
   - **Collect Logs**: Gather logs from the IDS and other relevant systems for analysis.

10.4 System Test

System testing of IDs focuses on verifying that the ID system functions correctly within the larger system it's integrated with. It's not just about testing the ID generation in isolation, but how those IDs behave and interact with other components. Here's a breakdown of what system testing of IDs entails:

- **ID Retrieval and Search**: Test the system's ability to retrieve data based on IDs. This includes various search methods (exact match, partial match, etc.) and large datasets. Performance testing is crucial here – how quickly can the system find data given an ID?
- **Integration with Other Systems**: If the IDs are used to interact with other systems (e.g., via APIs), test these integrations thoroughly. Ensure IDs are correctly passed and interpreted by the other systems, and that data exchange is seamless. Consider different data formats and communication protocols.
- **Error Handling**: Test how the system handles invalid IDs or ID conflicts. Appropriate error messages should be returned, and the system should not crash or become unstable. Test boundary conditions (e.g., extremely long IDs, special characters) to ensure robustness.
- **Security**: Verify that IDs are not easily guessable or predictable, especially if they have security implications (e.g., access control). Test for vulnerabilities related to ID manipulation or unauthorized access.
- **Performance and Scalability**: Test the performance of ID generation and retrieval under realistic load conditions. Simulate peak usage scenarios to ensure the system can handle the volume of ID requests without performance degradation. Scalability testing checks how well the system can handle increasing data volumes and user traffic over time.
- **Usability:** If users interact with IDs (e.g., entering them into forms), test the usability of the ID system. Are the IDs easy to read and remember? Are there any common errors users make when working with IDs?
- **Compliance:** Ensure the ID system complies with any relevant regulations or standards, such as data privacy laws (e.g., GDPR)

10.5 White Box Testing

Black box testing of IDs (Identifiers) involves testing the functionality of a system or application without any knowledge of its internal structure or code. This method focuses solely on the inputs and outputs, treating the system as a "black box." In the context of IDs, it means verifying that the system correctly handles various types of IDs, ensuring they are generated, validated, stored, and used as expected.

Key aspects of black box testing of IDs:

1. **ID Generation:**
   - **Uniqueness:** Verify that each ID generated is unique within its scope (e.g., within a specific system or database).
   - **Format:** Ensure IDs adhere to the defined format (e.g., length, character types, special characters).
   - **Predictability:** Check if IDs are generated randomly or sequentially. Randomness is usually preferred for security reasons.
   - **Collision:** Test for potential collisions (duplicate IDs) when generating a large number of IDs.

2. **ID Validation:**
   - **Valid IDs:** Verify that the system correctly accepts and processes valid IDs.
   - **Invalid IDs**: Test how the system handles invalid IDs (e.g., incorrect format, non-existent IDs). Error messages should be clear and informative.
   - **Boundary Conditions**: Test IDs at the boundaries of allowed values (e.g., minimum and maximum length).
   - **Special Cases**: Check how the system handles special characters or reserved words in IDs.

3. **ID Storage:**
   - **Data Integrity**: Ensure that IDs are stored correctly and consistently in the database or storage system.
   - **Security**: Verify that IDs are stored securely, especially if they contain sensitive information.
   - **Retrieval:** Test the retrieval of IDs based on various criteria (e.g., searching by ID, retrieving IDs associated with a user).

4. **ID Usage:**
   - **Functionality**: Verify that IDs are used correctly in different parts of the system (e.g., authentication, authorization, data retrieval).
   - **Performance**: Test the performance of the system when handling a large number of IDs.
   - **Security**: Ensure that IDs cannot be easily guessed or manipulated to gain unauthorized access.

10.6 Black Box Testing

Black box testing of IDs focuses on verifying the functionality of a system or application related to how it handles identifiers (IDs), without any knowledge of the internal code or structure. It treats the system as a "black box," concentrating solely on inputs and outputs.

Key aspects of black box testing of IDs:

1. **ID Generation:**
   - **Uniqueness:** Ensure each ID generated is unique within its scope.
   - **Format:** Verify IDs adhere to the defined format (length, character types, special characters).
   - **Predictability:** Check if IDs are random or sequential. Randomness is usually preferred for security.
   - **Collision:** Test for potential collisions (duplicate IDs) when generating many IDs.
2. **ID Validation:**
   - **Valid IDs**: Confirm the system correctly accepts and processes valid IDs.
   - **Invalid IDs**: Test how the system handles invalid IDs (incorrect format, non-existent IDs). Error messages should be clear.
   - **Boundary Conditions**: Test IDs at the limits of allowed values (minimum and maximum length).
   - **Special Cases**: Check how the system handles special characters or reserved words in IDs.
3. **ID Storage:**
   - **Data Integrity**: Ensure IDs are stored correctly and consistently.
   - **Security:** Verify IDs are stored securely, especially if they contain sensitive information.
   - **Retrieval**: Test ID retrieval based on various criteria (searching by ID, retrieving IDs associated with a user).
4. **ID Usage:**
   - **Functionality**: Verify IDs are used correctly in different parts of the system (authentication, authorization, data retrieval).
   - **Performance:** Test system performance when handling numerous IDs.
   - **Security**: Ensure IDs cannot be easily guessed or manipulated to gain unauthorized access.

CHAPTER-11: CONCLUSION

CONCLUSION

Intrusion detection currently attracts considerable interest from both the research community and commercial companies. Research prototypes continue to appear, and commercial products based on early research are now available. In this paper, I have given an overview of the current state of the art of intrusion detection, based on a proposed taxonomy illustrated with examples of past and current projects. The taxonomy clearly 14 highlights the properties of these intrusion-detection systems, covering both past and current developments adequately.

Information sources for these tools are either a C2 audit trail, syslog, or network packets. Whereas system sources were widely used in the early stages of research, the current focus of research prototypes as well as products is on protecting the infrastructure rather than the end-user station, and this paradigm has led to the use of network sniffers that analyse packets. As shown, quite a number of research issues concerning the efficiency of both network and host audit sources, the formatting and existence of a common audit trail format, and even the contents of the audit trail itself, still await an answer.

There are also a number of unsolved issues concerning the analysis of the audit trail. Signature analysis is clearly in the commercial domain now, but has been shown to be insufficient for detecting all attacks. Therefore, work is still in progress to experiment with new approaches to both knowledge-based and behaviour-based intrusion detection. The detection of abuse-of-privilege attacks (primarily insider attacks) is also the subject of ongoing work.

CHAPTER-12: FUTURE SCOPE

FUTURE SCOPE

The future of Intrusion Detection Systems (IDS) is bright and promising, driven by the ever- evolving cybersecurity landscape and technological advancements. Here are some key trends and future directions:

1. AI and Machine Learning Integration:

- **Enhanced threat detection**: AI and machine learning algorithms can analyze vast amounts of data to identify patterns and anomalies that traditional signature-based methods might miss. This enables the detection of zero-day exploits and sophisticated attacks.
- **Adaptive systems**: Future IDSs will be more adaptive, learning from network behavior and user activity to fine-tune their detection capabilities and minimize false positives.
- **Automated response**: AI can automate responses to detected threats, such as blocking malicious traffic or isolating infected devices, reducing the need for human intervention.

2. Cloud-Native Security:

- **Cloud-based IDS**: With the increasing adoption of cloud computing, cloud-native IDSs are becoming essential. These systems are designed to protect cloud environments and integrate seamlessly with cloud platforms.
- **Serverless security**: As serverless computing gains popularity, IDSs will need to adapt to monitor and secure these dynamic and ephemeral environments.

3. Focus on User Behavior:

- **User and Entity Behavior Analytics (UEBA):** IDSs will incorporate UEBA to detect insider threats and compromised accounts by analyzing user behavior and identifying deviations from normal patterns.

4. Integration with other security tools:

- **Holistic security posture:** IDSs will be integrated with other security tools, such as firewalls, SIEM systems, and threat intelligence platforms, to provide a more comprehensive and coordinated security approach.
- **XDR (Extended Detection and Response):** XDR platforms combine data from various security tools to provide a unified view of threats and enable faster and more effective responses.

5. Emphasis on Prevention:

- **Intrusion Prevention Systems (IPS):** The focus will shift towards proactive prevention rather than just detection. IDSs will evolve into IPSs with advanced capabilities to block or mitigate threats in real-time.

CHAPTER-13: REFERENCES

REFERENCES

1. Kalyankumar Dasari, Mohmad Ahmed Ali, NB Shankara, K Deepthi Reddy, M Bhavsingh, K Samunnisa, "**A Novel IoT-Driven Model for Real-Time Urban Wildlife Health and Safety Monitoring in Smart Cities**" 2024 8th International Conference on I-SMAC, Pages 122-129.

2. Kalyan Kumar Dasari & Dr, K Venkatesh Sharma, "**A Study on Network Security through a Mobile Agent Based Intrusion Detection Framework**", JASRAE, vol: 11, Pages: 209-214, 2016.

3. Dr.K.Sujatha, Dr.Kalyankumar Dasari , S. N. V. J. Devi Kosuru , Nagireddi Surya Kala , Dr. Maithili K , Dr.N.Krishnaveni, " **Anomaly Detection In Next-Gen Iot:Giant Trevally Optimized Lightweight Fortified Attentional Convolutional Network**" Journal of Theoretical and Applied Information Technology, 15th January 2025. Vol.103. No.1,pages: 22-39.

4. Kalyankumar Dasari, Dr. K. Venkatesh Sharma, "**Analyzing the Role of Mobile Agent in Intrusion Detection System**", JASRAE, vol : 15, Pages: 566-573,2018.

5. Kalyan Kumar Dasari&, M Prabhakar, "**Professionally Resolve the Password Security knowledge in the Contexts of Technology**", IJCCIT, Vol: 3, Issue:1, 2015.

6. S Deepajothi, Kalyankumar Dasari, N Krishnaveni, R Juliana, Neeraj Shrivastava, Kireet Muppavaram, "**Predicting Software Energy Consumption Using Time Series-Based Recurrent Neural Network with Natural Language Processing on Stack Overflow Data**", 2024 Asian Conference on Communication and Networks (ASIANComNet), Pages:1-6, Publisher: IEEE.

7. S Neelima, Kalyankumar Dasari, A Lakshmanarao, Peluru Janardhana Rao, Madhan Kumar Jetty, "**An Efficient Deep Learning framework with CNN and RBM for Native Speech to Text Translation**", 2024 3rd International Conference for Advancement in Technology (ICONAT), Pages: 1-6,Publisher :IEEE.

8. A Lakshmanarao, P Bhagya Madhuri, Kalyankumar Dasari, Kakumanu Ashok Babu, Shaik Ruhi Sulthana, "**An Efficient Android Malware Detection Model using Convnets and Resnet Models**",2024 International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS), Pages :1-6, Publisher : IEEE.

9. Dr.D.Kalyankumar, Saranam Kavyasri, Mandadi Mohan Manikanta, Pandrangi Veera Sekhara Rao, GanugapantaVenkata Pavan Reddy, "**Build a Tool for Digital Forensics to Analyze and Recover Information from Compromised Systems**", IJMTST, Vol: 10, Issue: 02, Pages:173-180, 2024.

10. Dr.D.Kalyankumar, Kota Nanisai Krishna, Gorantla Nagarjuna, PuvvadaVenkata Naga Sai Jagadesh Kumar, Modepalli Yeswanth Chowdary, "**Email Phishing Simulations Serve as a Valuable Tool in Fostering a Culture of Cyber security Awareness**", IJMTST, Vol: 10, Issue: 02, Pages:151- 157, 2024.

11. Dr.D.Kalyankumar, Muhammad Shaguftha, Putti Venkata Sujinth, Mudraboyina Naga Praveen Kumar, Namburi Karthikeya, "**Implementing a Chatbot with End-To-End Encryption for Secure and Private Conversations**", IJMTST, Vol: 10, Issue: 02, Pages:130-136, 2024.

12. Dr.D.Kalyankumar, Panyam Bhanu Latha, Y. Manikanta Kalyan, Kancheti Deepu Prabhunadh, Siddi Pavan Kumar, "**A Proactive Defense Mechanism against Cyber Threats Using Next-Generation Intrusion Detection System**", IJMTST, Vol: 10, Issue: 02, Pages:110-116, 2024.

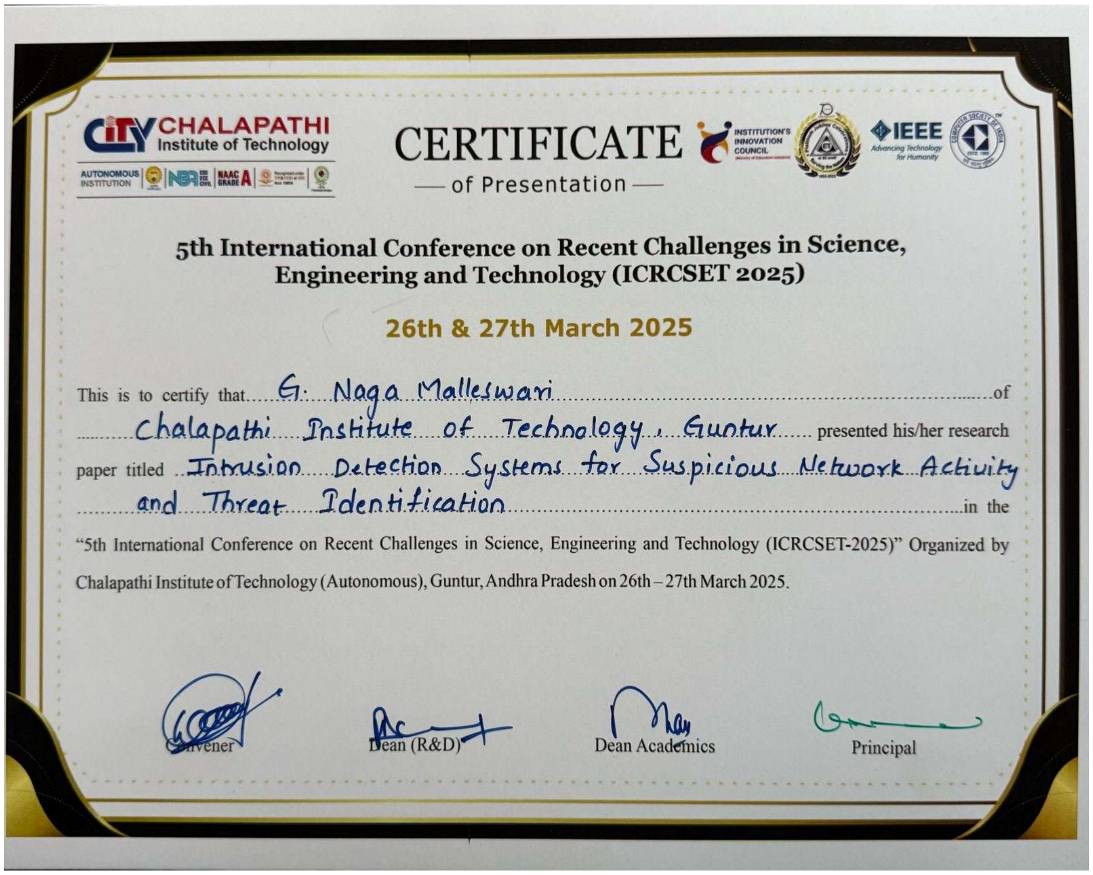
13. Kalyan Kumar Dasari, K Dr , "**Mobile Agent Applications in Intrusion Detection System (IDS)**'-JASC, Vol: 4, Issue : 5, Pages: 97-103, 2017.

14. V.Monica, D. Kalyan Kumar, "**BACKGROUND SUBTRACTION BY USING DECOLOR ALGORITHM**", IJATCSE, Vol. 3, No.1, Pages: 273 – 277 (2014). 285 JNAO Vol. 16, Issue. 1: 2025.

15. GanugapantaVenkata Pavan Reddy Dr.D.Kalyankumar, Saranam Kavyasri, Mandadi Mohan Manikanta, Pandrangi Veera Sekhara Rao "**Build a Tool for Digital Forensics to Analyze and Recover Information from Compromised Systems**", IJMTST, Vol: 10, Issue: 02, Pages:173-180, 2024.

CHAPTER-14: CERTIFICATES

CERTIFICATES



CHALAPATHI Institute of Technology

AUTONOMOUS INSTITUTION | NBA | NAAC GRADE A

CERTIFICATE

— of Presentation —

INSTITUTION'S INNOVATION COUNCIL

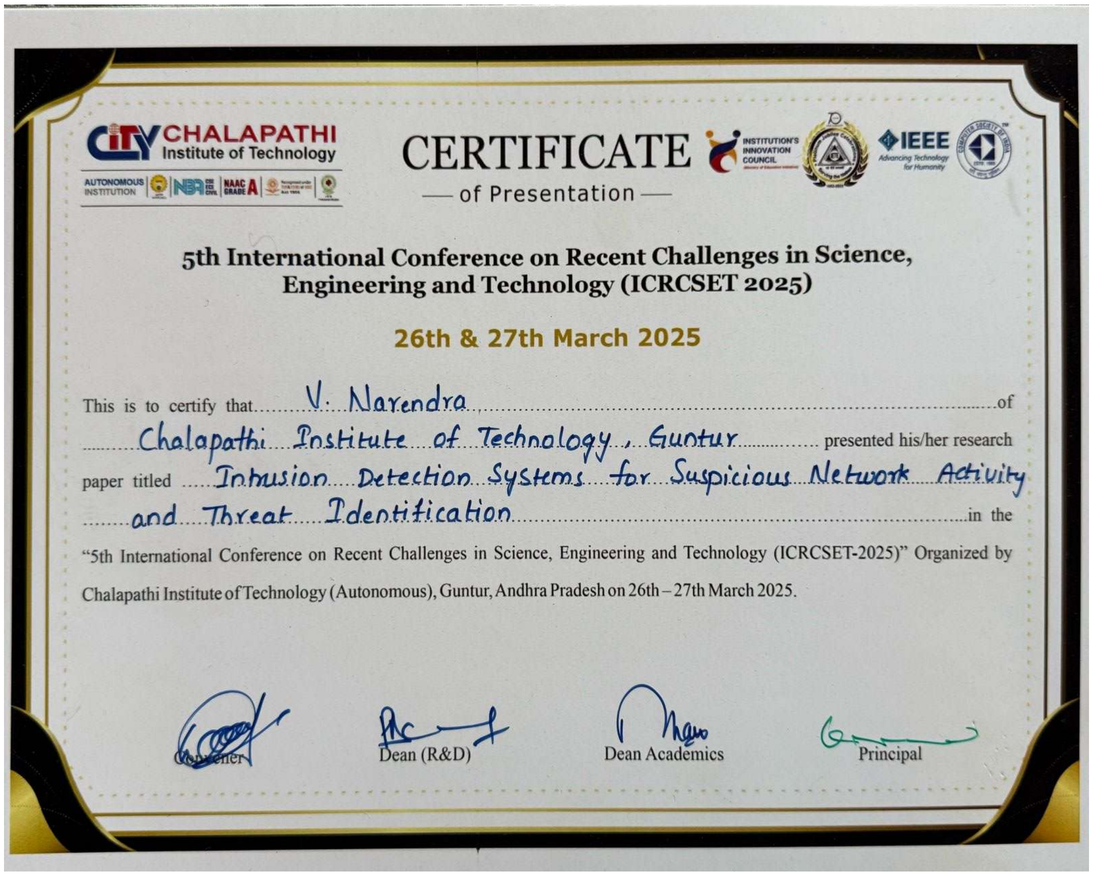
IEEE Advancing Technology for Humanity

5th International Conference on Recent Challenges in Science, Engineering and Technology (ICRCSET 2025)

26th & 27th March 2025

This is to certify that G. Naga Malleswari of Chalapathi Institute of Technology, Guntur presented his/her research paper titled Intrusion Detection Systems for Suspicious Network Activity and Threat Identification in the "5th International Conference on Recent Challenges in Science, Engineering and Technology (ICRCSET-2025)" Organized by Chalapathi Institute of Technology (Autonomous), Guntur, Andhra Pradesh on 26th – 27th March 2025.

Convener | Dean (R&D) | Dean Academics | Principal



CHALAPATHI Institute of Technology

AUTONOMOUS INSTITUTION | NBA | NAAC GRADE A

CERTIFICATE

of Presentation

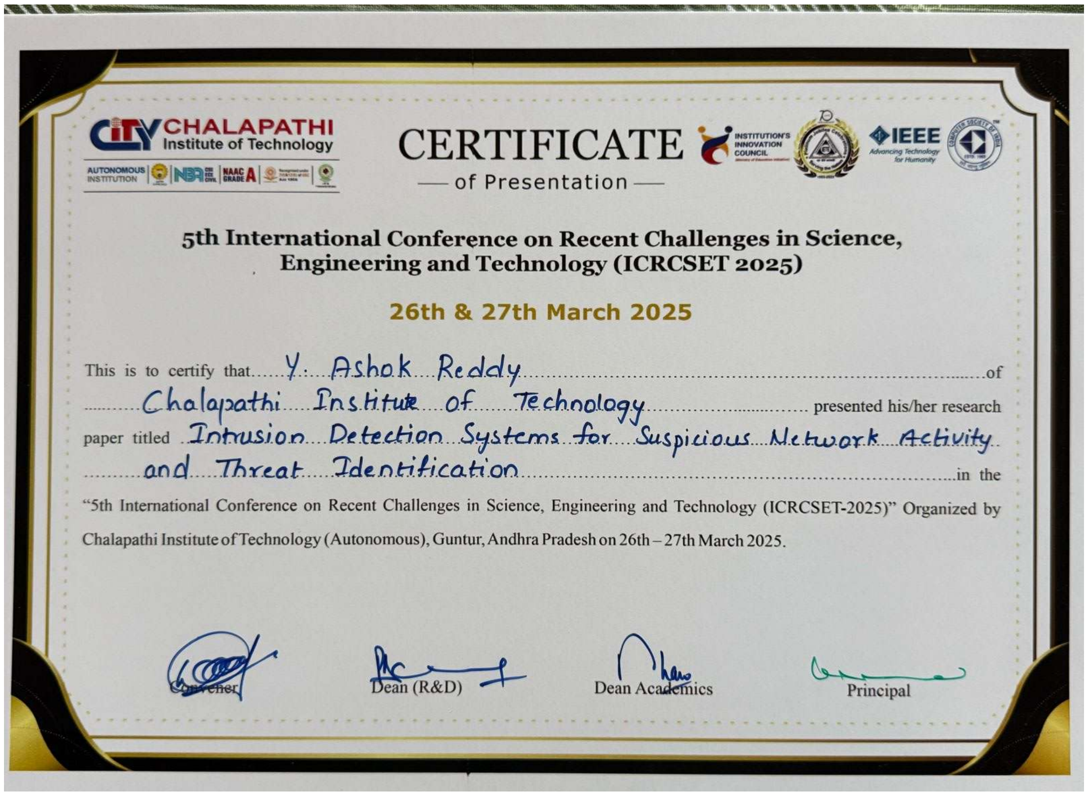
INSTITUTION'S INNOVATION COUNCIL | IEEE Advancing Technology for Humanity

5th International Conference on Recent Challenges in Science, Engineering and Technology (ICRCSET 2025)

26th & 27th March 2025

This is to certify that V. Narendra of Chalapathi Institute of Technology, Guntur presented his/her research paper titled Intrusion Detection Systems for Suspicious Network Activity and Threat Identification in the "5th International Conference on Recent Challenges in Science, Engineering and Technology (ICRCSET-2025)" Organized by Chalapathi Institute of Technology (Autonomous), Guntur, Andhra Pradesh on 26th – 27th March 2025.

Convener | Dean (R&D) | Dean Academics | Principal


CHALAPATHI Institute of Technology

AUTONOMOUS INSTITUTION | NBA | NAAC GRADE A

CERTIFICATE

— of Presentation —

INSTITUTION'S INNOVATION COUNCIL

IEEE Advancing Technology for Humanity

5th International Conference on Recent Challenges in Science, Engineering and Technology (ICRCSET 2025)

26th & 27th March 2025

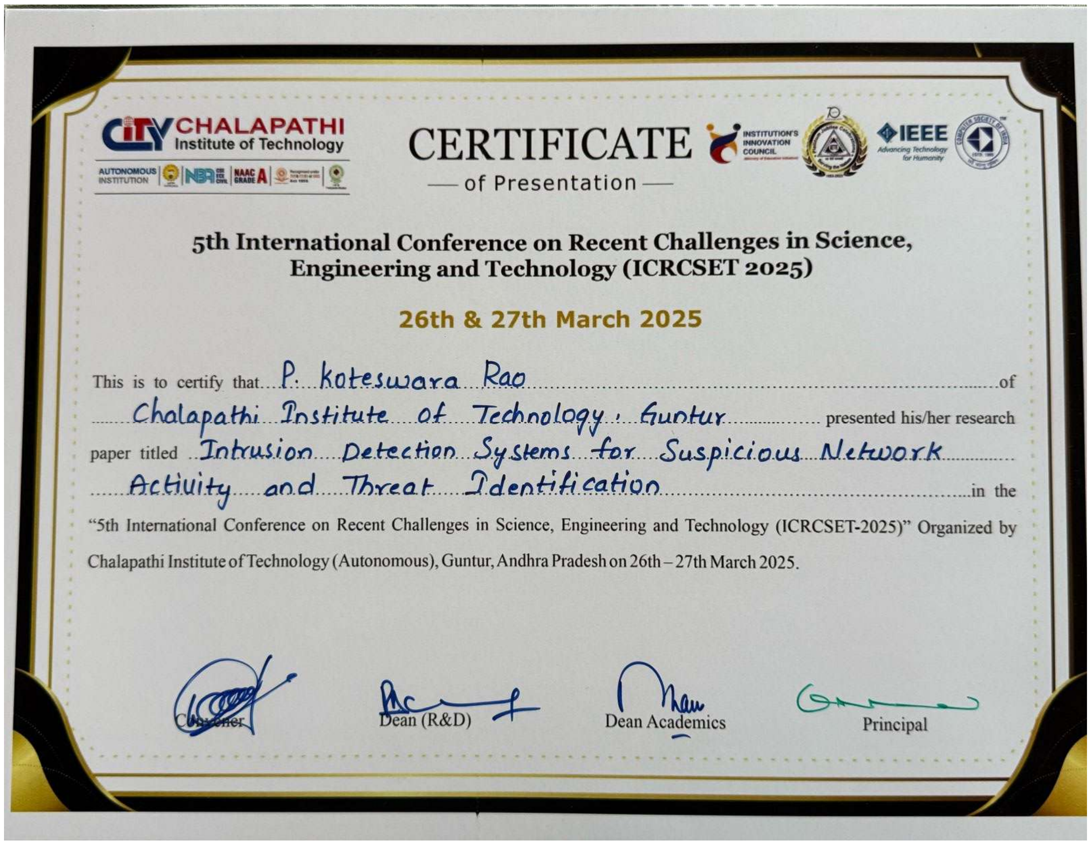
This is to certify that Y. Ashok Reddy of Chalapathi Institute of Technology presented his/her research paper titled Intrusion Detection Systems for Suspicious Network Activity and Threat Identification in the "5th International Conference on Recent Challenges in Science, Engineering and Technology (ICRCSET-2025)" Organized by Chalapathi Institute of Technology (Autonomous), Guntur, Andhra Pradesh on 26th – 27th March 2025.

[illegible] Convener

[illegible] Dean (R&D)

[illegible] Dean Academics

[illegible] Principal


CHALAPATHI Institute of Technology

AUTONOMOUS INSTITUTION | NBA | NAAC GRADE A

CERTIFICATE

— of Presentation —

INSTITUTION'S INNOVATION COUNCIL

IEEE Advancing Technology for Humanity

5th International Conference on Recent Challenges in Science, Engineering and Technology (ICRCSET 2025)

26th & 27th March 2025

This is to certify that P. koteswara Rao of Chalapathi Institute of Technology, Guntur presented his/her research paper titled Intrusion Detection Systems for Suspicious Network Activity and Threat Identification in the "5th International Conference on Recent Challenges in Science, Engineering and Technology (ICRCSET-2025)" Organized by Chalapathi Institute of Technology (Autonomous), Guntur, Andhra Pradesh on 26th – 27th March 2025.

[illegible] Convener | [illegible] Dean (R&D) | [illegible] Dean Academics | [illegible] Principal

CHAPTER-15: JOURNAL PAPER

Journal of Nonlinear Analysis and Optimization
Vol. 16, Issue. 1: 2025
ISSN : **1906-9685**

A Network Security Tool That Monitors Network Traffic for Suspicious Activity and Potential Threats

[1]T. Yedukondalu, [2] G. Naga Malleswari, [3] V. Narendra, [4] Y. Ashok Reddy, [5] P. Koteswara Rao

1Asst.Professor, Department of CSE-Cyber Security
2,3,4,5 UG Scholar, Department of CSE-Cyber Security
Chalapathi Institute of Technology, Guntur, Andhra Pradesh, India-522016.

ABSTRACT

The Intrusion Detection System (IDS) Implementation project addresses critical challenges in modern network security by developing a robust and adaptable tool for identifying and responding to malicious activities. An IDS acts as a crucial defence mechanism, monitoring network traffic and system behaviour for suspicious patterns that may indicate an intrusion attempt. While traditional IDS solutions offer basic threat detection, they often struggle with evolving attack vectors, high false positive rates, and difficulties in real-time analysis and response. This project offers an innovative solution by designing a Python-based platform that leverages machine learning algorithms and advanced anomaly detection techniques to enhance intrusion detection accuracy and efficiency. The primary objective of the IDS Implementation project is to provide a user-friendly and intelligent tool that overcomes the limitations of conventional IDS systems. This tool empowers security administrators with real-time threat identification, automated alert generation, and adaptive learning capabilities to stay ahead of emerging threats. It promotes a proactive approach to network security by integrating threat intelligence feeds and visualization tools, enhancing understanding of attack patterns and enabling faster incident response. Operating within a framework that prioritizes accessibility and ethical use, the IDS Implementation project ensures adherence to legal boundaries and respects privacy considerations. By focusing on innovation and usability, this project sets a benchmark in intrusion detection, addressing the growing need for advanced security measures in today's interconnected world. The IDS Implementation project tackles the evolving challenges of network security by creating a sophisticated, user-friendly tool that bridges the gap between traditional intrusion detection and modern network requirements. As cyber attacks become increasingly sophisticated and frequent, IDS solutions must evolve to provide real-time, adaptive, and intelligent protection against malicious intrusions and data breaches. This project delivers a solution designed with Python and machine learning, offering an intelligent platform for detecting, analysing, and responding to security threats tailored to individual network environments. solutions often suffer from signature-based detection limitations, high false positive rates, complexity in rule management, and a lack of real-time insights. These challenges hinder effective threat detection and response, leaving networks vulnerable to advanced persistent threats and zero-day exploits. This project aims to address these shortcomings by incorporating machine learning algorithms and anomaly detection techniques to improve detection accuracy and reduce false positives.

Keywords: Intrusion Detection System, Traditional IDS, Machine Learning Algorithm, and Secure Algorithm.

1. Introduction

In today's interconnected world, cyber security has become paramount. Organizations and individuals alike face constant threats from malicious actors seeking to exploit vulnerabilities and gain unauthorized access to sensitive data and systems. To combat these threats, Intrusion Detection Systems (IDS) have emerged as crucial tools in safeguarding digital assets. An IDS is a security system that monitors network traffic and system activity for malicious activity or policy violations. It acts as a vigilant guardian, constantly analyzing data flows and user behaviour to identify any deviations from normal patterns. By detecting and alerting on suspicious activity, IDS empowers organizations to proactively respond to threats, minimize damage, and enhance their overall security posture. This project focuses on the development and implementation of and the proposed IDS, including its architecture, algorithms, implementation, and evaluation. IDS can be broadly categorized into two main types: Network-Based Intrusion Detection Systems (NIDS): These systems monitor network traffic flowing in and out of a network segment. They analyze data packets for malicious patterns, such as unauthorized access attempts, port scans, and denial-of-service attacks. NIDS are typically deployed at strategic points within a network, such as at the perimeter or within critical segments.

Host-Based Intrusion Detection Systems (HIDS): These systems focus on monitoring the activity within individual hosts (computers or servers). They analyse system logs, audit trails, and other system-level data to detect suspicious activity, such as unauthorized file access, malware infections, and privilege escalations. HIDS provide a deeper level of visibility into the internal operations of a system. Signature-based Detection: This method relies on predefined patterns or signatures of known attacks. IDS systems compare incoming traffic or system events against these signatures. If a match is found, an alert is triggered. This approach is effective against known threats but may miss novel or zero-day attacks. Anomaly-based Detection: This method focuses on identifying deviations from normal system behavior. IDS systems establish a baseline of normal activity and then analyze incoming traffic or system events for any unusual patterns. Anomalies, such as sudden spikes in traffic, unusual user activity, or unexpected system changes, can indicate potential threats. This approach is more effective at detecting novel attacks but may generate a higher number of false positives. Behavior-based Detection: This method analyzes user behavior and system activity to identify suspicious patterns. IDS systems learn normal user behavior over time and then flag any deviations from this established baseline. This approach can effectively detect insider threats and other forms of malicious activity that may not be easily detectable using other methods.

Intrusion Detection Systems (IDS) serve a crucial role in safeguarding computer systems and networks from malicious activity. Here are some key usages: Proactive Threat Detection: IDSs can identify and alert on suspicious activity in real-time, such as unauthorized access attempts, data breaches, and denial-of-service attacks. This allows for immediate response and mitigation of potential damage. Improved Security Posture: By continuously monitoring network traffic and system activity, IDSs provide valuable insights into potential vulnerabilities and threats. This information can be used to enhance security measures, strengthen defenses, and improve overall network security. Reduced Risk of Data Breaches: IDSs play a crucial role in preventing data breaches by detecting and blocking malicious activities that could compromise sensitive data. Compliance with Regulations: Many industries and organizations are subject to regulatory requirements that mandate specific security measures. IDSs can help organizations demonstrate compliance with these regulations by providing evidence of security controls and proactive threat monitoring. Enhanced Incident Response: In the event of a security incident, IDS logs and alerts can provide valuable information for investigation and response teams. This information can help pinpoint the source of the attack, understand the scope of the damage, and accelerate the incident response process. Insider Threat Detection: IDSs can help detect and prevent insider threats, such as malicious activity from employees or privileged users. By analyzing user behavior and system activity, IDSs can identify unusual patterns that may indicate insider threats. Network Performance Monitoring: Some IDSs can also be used to monitor network performance and identify potential bottlenecks or performance issues. This information can be used to optimize network performance and ensure smooth operations.

2. EXISTING SYSTEM

Most traditional IDS implementations fall into the following categories: Signature-Based IDS: Detects known attack patterns by comparing network traffic with a database of signatures. Anomaly-Based IDS: Uses machine learning or statistical methods to detect deviations from normal behaviour. Host-Based IDS Monitors activities on individual devices, analysing system logs and user behaviour. Network-Based IDS : Inspects network traffic for suspicious activity across multiple systems. Limitations of the Existing IDS despite their effectiveness, traditional IDS have several shortcomings: High False Positives: Anomaly-based IDS often flag legitimate activity as threats. High False Negatives: Signature-based IDS fail to detect new, unknown attack types. Scalability Issues: Traditional IDS may struggle to handle large-scale network traffic in cloud or enterprise environments. Slow Response Time: Delayed detection and response can allow attackers to exploit vulnerabilities before mitigation. Lack of Context Awareness: IDS may lack deep visibility into user intent, making it difficult to differentiate between benign and malicious activities. System Analysis Considerations: When analysing an existing IDS, the following factors must be evaluated: Detection Accuracy: Assessing false positives and negatives. Performance and Scalability: Measuring system efficiency under different loads. Integration with Other Security Tools: Ensuring compatibility with SIEM (Security Information and Event Management) and firewalls. Response Time: Evaluating how quickly the system detects and responds to threats. Adaptability to New Threats: Checking how well the IDS update to handle new vulnerabilities. Given these limitations, organizations often enhance IDS with: Machine Learning & AI for better anomaly detection. Threat Intelligence Integration to detect zero-day attacks. Automated Response Mechanisms to reduce human intervention.

3. PROPOSED SYSTEM

Proposed System for Intrusion Detection System: This document outlines a proposed system for an Intrusion Detection System (IDS). The system will leverage a hybrid approach, combining signature-based and anomaly- based detection techniques for enhanced effectiveness. System Architecture. The proposed system will consist of the following components: Data Acquisition Module: Responsible for collecting network traffic data from various sources, including network interfaces, firewalls, and security information and event management (SIEM) systems. Supports various data formats, such as NetFlow, PCAP, and syslog. Preprocessing Module: Cleans and transforms the raw data into a suitable format for analysis. This includes tasks such as data normalization, feature extraction, and dimensionality reduction. Signature-Based Detection Engine: Utilizes a database of known attack signatures (e.g., Snort rules, YARA rules) to match incoming traffic patterns against known threats. Provides real-time alerts for identified threats. Anomaly-Based Detection Engine: Employs machine learning algorithms (e.g., Support Vector Machines, Isolation Forest, Auto encoders) to establish a baseline of normal network behaviour. Detects deviations from this baseline, indicating potential anomalies and potential threats. Correlation Engine: Correlates alerts from both signature-based and anomaly-based detection engines. Reduces false positives and provides a more comprehensive view of potential threats.

Key Features: Hybrid Approach: Combines the strengths of signature-based and anomaly- based detection for enhanced accuracy and reduced false positives. Scalability: Designed to handle high volumes of network traffic and adapt to evolving threat landscapes. Flexibility: Configurable to meet specific security requirements and threat profiles. Real-time Analysis: Provides real-time threat detection and alerting capabilities. Automated Response: Supports automated response actions to mitigate threats quickly. Integration Capabilities: Integrates with existing security infrastructure, such as firewalls, SIEM systems, and other security tools. Benefits to improved Threat Detection: Enhanced accuracy and reduced false positives compared to traditional IDS approaches. Proactive Response: Enables proactive response to threats, minimizing potential damage. Reduced Operational Costs: Automates threat detection and response processes, reducing the need for manual intervention. Enhanced Security Posture: Provides a robust defence against a wide range of cyber threats. This proposed system offers a comprehensive and effective solution for intrusion detection, providing organizations with the tools and capabilities to proactively defend against cyber threats and maintain a strong security posture.

4. SYSTEMSTUDY

The system development for an Intrusion Detection System involves several key steps to create a robust security monitoring platform. Initially, requirements are gathered to understand the organization's security needs and threat landscape. With this information, the system architecture is designed, considering server-side components (e.g., data processing engine, database), client-side components (management console), communication protocols, and data storage requirements. Once the environment is set up, development begins with the creation of the core analysis engine, data collection modules, and the user interface. Integration and testing follow to ensure proper functionality and accuracy. Security measures are implemented to protect the IDS itself. The system is deployed to a production environment. Ongoing maintenance, performance optimization, rule updates, documentation, and continuous improvement complete the system development process, ensuring a robust, secure, and scalable intrusion detection platform.

Requirements Gathering and Analysis: Identify stakeholders (security team, IT operations, management). Gather requirements: Specific threats to detect (e.g., malware, DDoS, SQL injection).

Network segments or systems to monitor. Data sources to use (network traffic, logs, etc.). Performance expectations (detection rate, false positive rate). Integration requirements with other security tools (SIEM, firewalls). Compliance requirements (e.g., PCI DSS, HIPAA). Architecture Design: Design the system architecture is Sensor placement and data collection methods. Data processing and analysis engine components. Database design for storing events, rules, and configurations. Communication protocols (e.g., for data transfer, alerts). Scalability considerations.

Environment Setup Set up the development environment. Install necessary software (e.g., analysis tools, database). Configure network connectivity and data sources are set up a testing environment that mirrors the production network. Server-Side Development develops the core analysis engine: Implement detection algorithms. Develop data processing modules for normalizing and filtering data. Implement alerting mechanisms. Develop APIs for integration with other systems. Client-Side Development the user interface: Design dashboards for visualizing security events and trends. Create interfaces for configuring the IDS and managing rules. Implement alert management and reporting features. This adaptation of the chat application development process to an IDS context provides a structured approach to building a robust and effective intrusion detection system. Remember to tailor the specifics to your organization's individual needs and security requirements.

4.1 DESIGN AND ARCHITECTURE

This is the core part of the system study, detailing the overall design of the encryption algorithm. It should include algorithmic design: Step-by-step explanation of how the encryption and decryption processes work. Encryption steps outline of how plaintext is transformed into cipher text.

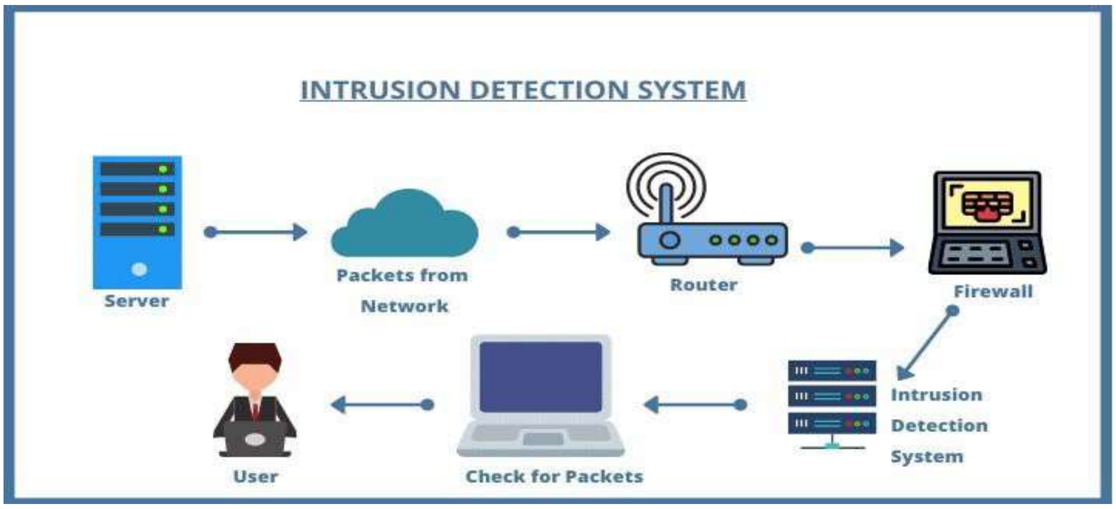


Fig: 1.System Design

4.2. Elements and Their Descriptions:

Server (Leftmost): Represented by a blue tower icon with multiple horizontal layers, symbolizing a typical server computer. This is likely the source or destination of network traffic being monitored. Cloud (Top Center): A stylized cloud with the label "Packets from Network". This represents the network environment (e.g., the internet or a local network) from which data packets are traveling.

Router (Center): A rounded rectangle with antenna-like symbols and the label "Router". This signifies a network device responsible for forwarding data packets between different networks.1

Firewall (Right Center): A vertical rectangle with a brick-like pattern, labeled "Firewall". This represents a security system that controls network traffic based on predetermined rules, acting as a barrier against unauthorized access.

Intrusion Detection System (Rightmost): Represented by a computer monitor icon displaying three server icons and the label "Intrusion Detection System". This is the core component of the image, indicating the system responsible for monitoring network traffic for malicious activity.

User (Bottom Left): A stylized human figure, labeled "User", representing an individual or entity interacting with the network or monitoring the IDS.

Laptop with "Check for Packets" (Bottom Center): A laptop icon with the label "Check for Packets". This symbolizes the action of monitoring or inspecting network traffic data.

Arrows and Flow: The arrows connecting the elements indicate the direction of data flow. Data packets move from the network (cloud) through the router, firewall, and finally to the server or user. The connection between the laptop and the router suggests the user's monitoring activity.

Labels: Each element is accompanied by a text label, clearly identifying its function within the IDS framework.

Interpretation and Context: The image illustrates a common scenario where network traffic passes through a router and firewall before reaching a server or user. The IDS is positioned to monitor this traffic, likely by inspecting packets and logs, to identify suspicious patterns or malicious activities. The user's laptop suggests an interface for monitoring the IDS alerts or configuring its settings.

Educational Value: This image serves as a valuable educational tool for understanding the basic components and data flow within an Intrusion Detection System. It simplifies the complex process of network security monitoring into a visually digestible format.

5. CONCLUSION

Intrusion detection currently attracts considerable interest from both the research community and commercial companies. Research prototypes continue to appear, and commercial products based on early research are now available. In this paper, I have given an overview of the current state of the art of intrusion detection, based on a proposed taxonomy illustrated with examples of past and current projects. The taxonomy clearly 14 highlights the properties of these intrusion-detection systems, covering both past and current developments adequately. Information sources for these tools are a C2 audit trail, syslog, or network packets. Whereas system sources were widely used in the early stages of research, the current focus of research prototypes as well as products is on protecting the infrastructure rather than the end-user station, and this paradigm has led to the use of network sniffers that analyse packets. As shown, quite a number of research issues concerning the efficiency of both network and host audit sources, the formatting and existence of a common audit trail format, and even the contents of the audit trail itself, still await an answer. There are also a number of unsolved issues concerning the analysis of the audit trail. Signature analysis is clearly in the commercial domain now, but has been shown to be insufficient for detecting all attacks. Therefore, work is still in progress to experiment with new approaches to both knowledge-based and behaviour-based intrusion detection. The detection of abuse-of-privilege attacks is also the subject of ongoing work.

REFERENCES

[1] Kalyankumar Dasari, Mohmad Ahmed Ali, NB Shankara, K Deepthi Reddy, M Bhavsingh, K Samunnisa, "A Novel IoT-Driven Model for Real-Time Urban Wildlife Health and Safety Monitoring in Smart Cities" 2024 8th International Conference on I-SMAC, Pages 122-129.

[2] Kalyan Kumar Dasari & Dr, K Venkatesh Sharma, "A Study on Network Security through a Mobile Agent Based Intrusion Detection Framework", JASRAE, vol: 11, Pages: 209-214, 2016.

[3] Dr.K.Sujatha, Dr.Kalyankumar Dasari , S. N. V. J. Devi Kosuru , Nagireddi Surya Kala , Dr. Maithili K , Dr.N.Krishnaveni, " Anomaly Detection In Next-Gen Iot:Giant Trevally Optimized Lightweight Fortified Attentional Convolutional Network," Journal of Theoretical and Applied Information Technology, 15th January 2025. Vol.103. No.1,pages: 22-39.

[4] Kalyankumar Dasari, Dr. K. Venkatesh Sharma, "Analyzing the Role of Mobile Agent in Intrusion Detection System", JASRAE, vol : 15, Pages: 566-573,2018.

[5] Kalyan Kumar Dasari&amp, M Prabhakar, "Professionally Resolve the Password Security knowledge in the Contexts of Technology", IJCCIT, Vol: 3, Issue:1, 2015.

[6] S Deepajothi, Kalyankumar Dasari, N Krishnaveni, R Juliana, Neeraj Shrivastava, Kireet Muppavaram, "Predicting Software Energy Consumption Using Time Series-Based Recurrent Neural Network with Natural Language Processing on Stack Overflow Data", 2024 Asian Conference on Communication and Networks (ASIANComNet), Pages:1-6, Publisher: IEEE.

[7] S Neelima, Kalyankumar Dasari, A Lakshmanarao, Peluru Janardhana Rao, Madhan Kumar Jetty, "An Efficient Deep Learning framework with CNN and RBM for Native Speech to Text Translation", 2024 3rd International Conference for Advancement in Technology (ICONAT), Pages: 1-6,Publisher :IEEE.

[8] A Lakshmanarao, P Bhagya Madhuri, Kalyankumar Dasari, Kakumanu Ashok Babu, Shaik Ruhi Sulthana, "An Efficient Android Malware Detection Model using Convnets and Resnet Models",2024 International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS), Pages :1-6, Publisher : IEEE

[9] Dr.D.Kalyankumar, Saranam Kavyasri, Mandadi Mohan Manikanta, Pandrangi Veera Sekhara Rao, GanugapantaVenkata Pavan Reddy, "Build a Tool for Digital Forensics to Analyze and Recover Information from Compromised Systems", IJMTST, Vol: 10, Issue: 02, Pages:173-180, 2024.

[10] Dr.D.Kalyankumar, Kota Nanisai Krishna, Gorantla Nagarjuna, PuvvadaVenkata Naga Sai Jagadesh Kumar, Modepalli Yeswanth Chowdary, "Email Phishing Simulations Serve as a Valuable Tool in Fostering a Culture of Cyber security Awareness", IJMTST, Vol: 10, Issue: 02, Pages:151-157, 2024.

[11] Dr.D.Kalyankumar, Muhammad Shaguftha, Putti Venkata Sujinth, Mudraboyina Naga Praveen Kumar, Namburi Karthikeya, "Implementing a Chatbot with End-To-End Encryption for Secure and Private Conversations", IJMTST, Vol: 10, Issue: 02, Pages:130-136, 2024.

[12] Dr.D.Kalyankumar, Panyam Bhanu Latha, Y. Manikanta Kalyan, Kancheti Deepu Prabhunadh, Siddi Pavan Kumar, "A Proactive Defense Mechanism against Cyber Threats Using Next-Generation Intrusion Detection System", IJMTST, Vol: 10, Issue: 02, Pages:110-116, 2024.

[13] Kalyan Kumar Dasari, K Dr , "Mobile Agent Applications in Intrusion Detection System (IDS)ʽ-JASC, Vol: 4, Issue : 5, Pages: 97-103, 2017.

[14] V.Monica, D. Kalyan Kumar, "BACKGROUND SUBTRACTION BY USING DECOLOR ALGORITHM", IJATCSE, Vol. 3, No.1, Pages: 273 – 277 (2014).

[15] GanugapantaVenkata Pavan Reddy Dr.D.Kalyankumar, Saranam Kavyasri, Mandadi Mohan Manikanta, Pandrangi Veera Sekhara Rao “Build a Tool for Digital Forensics to Analyze and Recover Information from Compromised Systems”, IJMTST, Vol: 10, Issue: 02, Pages:173-180, 2024.