

# **ADVANCED TECHNIQUES IN RULE CREATION FOR THREAT DETECTION**

**Industrial Internship Project report submitted in partial fulfillment of the  
Requirements for the Award of the Degree of**

**BACHELOR OF TECHNOLOGY**

**in**

**Department of CSE- (Artificial Intelligence & Machine Learning)**

**By**

**Tholuchuri Sasi Sekhar**

**208X1A4260**

**Under the Guidance of**

**Dr. K. Siva Rama Prasad, M.Tech., Ph.D**

**Associate Professor**



**KALLAM HARANADHAREDDY INSTITUTE OF TECHNOLOGY  
(AUTONOMOUS)**

**Approved by AICTE- New Delhi, Accredited by NAAC with 'A'**

**Grade Permanently Affiliated to JNTUK, Kakinada**

**NH-5, Chowdavaram, Guntur**

**April-2024**

## DECLARATION

We hereby declare that the Industrial Internship project dissertation entitled **Advanced Techniques In Rule Creation For Threat Detection** submitted for the B.Tech. Degree is my original work and the dissertation has not formed the basis for the award of any degree, associateship, fellowship or any other similar titles.

Place: Guntur

Tholuchuri Sasi Sekhar

Date:

208X1A4260

## Certificate from Intern Organization

This is to certify that **THOLUCHURI SASI SEKHAR** Reg. No. **208X1A4260** of Kallam Haranadhareddy Institute of Technology underwent industry internship in **SMARTINTERNZ** from **FEB-2024** to **APR- 2024** The overall performance of the intern during his/her internship is found to be \_\_\_\_\_(Satisfactory/Not Satisfactory).

# Certificate



**Link:** <https://apsche.smartinternz.com/certificate/student/bfe6c14c945256de12a6add92c83b4d9>

## ACKNOWLEDGMENT

We profoundly express our gratitude and respect towards our honorable chairman **SRI KALLAM MOHAN REDDY, Chairman of KHIT** for his precious support in the college.

We sincerely express our deepest gratitude to dynamic director of our institute **Dr. M. UMA SHANKARA REDDY M.Sc., Ph.D., Director of KHIT** for his valuable guidance.

We would like to thank **Dr. B.S.B. REDDY MTech, Ph.D., Principal of KHIT, Chowdavaram, Guntur** for holds a senior position in the organization and has provided overall support and guidance to us during our work.

We are greatly indebted to **Dr.G.J. SUNNY DEOL MTech, Ph.D., Professor, & Head of the department, CSE(AI&ML) KHIT, GUNTUR** for providing the laboratory facilities fully as and when required and for giving us the opportunity to carry the project work in the college during Industry Internship.

We extend our deep sense of gratitude to our Internal Guide **Dr.K. SIVA RAMA PRASAD MTech, Ph.D., Associate Professor, Faculty Members & Support staff** for their valuable suggestions, guidance and constructive ideas in each and every step, which was indeed of great help towards the successful completion of our project.

We express our thanks to **Mr. M. CHENNAKESAVARAO, M.E, Ph.D, Project Co-Ordinator of CSE-Artificial Intelligence And Machine Learning Department, KHIT** for pretending as faculty supervisor from the institution has providing academic support and guidance throughout my internship and support during my work, helping to navigate the internship process

Tholuchuri Sasi Sekhar

Place: Guntur

208X1A4260

Date:

## ABSTRACT

Brute force and dictionary attacks on password-only remote login services are now widespread and ever increasing. Enabling convenient login for legitimate users while preventing such attacks is a difficult problem, Automated Turing Tests (ATTs) continue to be an effective, easy-to-deploy approach to identify automated malicious login attempts with reasonable cost of inconvenience to users. In this paper we discuss the inadequacy of existing and proposed login protocols designed to address large scale online dictionary attacks (e.g., from a botnet of hundreds of thousands of nodes). We propose a new Password Guessing Resistant Protocol (PGRP), derived upon revisiting prior proposals designed to restrict such attacks. While PGRP limits the total number of login attempts from unknown remote hosts to as low as a single attempt per username, legitimate users in most cases (e.g., when attempts are made from known, frequently-used machines) can make several failed login attempts before being challenged with an ATT. We analyze the performance of PGRP with two real-world datasets and find it more promising than existing proposals.

# TABLE OF CONTENTS

Abstract .....	i
Table of Contents .....	ii
List of Figures .....	iii
List of Tables .....	iv
Chapter 1 Introduction .....	1
1.1 Problem Statement.....	3
1.2 Existing System .....	4
1.3 Proposed System .....	5
Chapter 2 Literature Survey .....	7
2.1 Basic Concepts .....	8
2.2 Project related work .....	9
Chapter 3 Software Requirements Specifications .....	11
3.1 Functional Requirements .....	11
3.2 Software Requirements .....	12
3.3 Non Functional Requirements.....	13
3.4 Performance Requirements .....	14
3.5 Hardware Requirements.....	15
Chapter 4 System Design .....	16
4.1 Requirements Modeling .....	16
4.1.1 Class Diagram .....	16
4.1.2 Use Case Diagram.....	17
4.1.3 Sequence Diagram .....	18
4.1.4 Collaboration Diagram.....	19
4.1.5 Activity Diagram.....	20
Chapter 5 Methodologies .....	22
5.1 Modules Description .....	22
5.2 System Architecture .....	26
Chapter 6 System Implementation .....	28
6.1 Deployment Environment and Tools .....	28

6.2 Coding.....	29
6.2.1 Source Code.....	30
6.3 Data Flow Diagram.....	34
Chapter 7 Testing .....	37
7.1 Introduction to Testing.....	37
7.1.1 Testing Strategies.....	37
Chapter 8 Results .....	45
8.1 Screens and Reports.....	45
8.2 User Manual.....	50
Chapter 9 Limitations .....	52
Chapter 10 Conclusion.....	53
Chapter 11 Future Work .....	54
Chapter 12 References .....	56



## LIST OF FIGURES

Fig 4.1.1. Class Diagram .....	16
Fig 4.1.2. Use Case Diagram .....	17
Fig 4.1.3. Sequence Diagram.....	18
Fig 4.1.4 Collaboration Diagram .....	19
Fig 4.1.5. Activity Diagram.....	21
Fig 5.1.1. Creating New Accounts.....	22
Fig 5.1.2. Tracking Hacker .....	23
Fig 5.1.3. Send Password.....	24
Fig 5.1.4. Block Source IP.....	25
Fig 5.2 System Architecture .....	26
Fig 6.2.1. Data Flow Diagram .....	34
Fig 6.2.2. Represents the Overall Data Flow Graph .....	35
Fig 6.2.3. Data Distribution .....	36
Fig 6.2.4 Accuracy & Loss .....	36
Fig 7.1. Home Page .....	39
Fig 7.2. Admin Page .....	40
Fig 7.3. Account Creation Page.....	41
Fig 7.4. Login Page.....	42
Fig 7.5. Transaction Page .....	43
Fig 7.6. Bank Statement Page.....	44
Fig 8.1.6. Accuracy & Loss .....	46
Fig 8.1.7. Confusion Matrix .....	46
Fig 8.1.8. Metrics.....	47

## LIST OF TABLES

Table – 7.1: Home Page .....	39
Table – 7.2: Admin Page .....	40
Table – 7.3: Account Creation Page.....	41
Table – 7.4: Login Page.....	42
Table – 7.5: Transaction Page .....	43
Table – 7.6: Bank Statement .....	44
Table – 8.1: Reports of Test Cases .....	48

# 1. INTRODUCTION

Online guessing attacks on password-based systems are inevitable and commonly observed against web applications identified password guessing attacks on websites as a top cyber security risk. SSH servers that disallow standard password authentication may also suffer guessing attacks, e.g., through the exploitation of a lesser known/used SSH server configuration called keyboard interactive authentication. However, online attacks have some inherent disadvantages compared to offline attacks: attacking machines must engage in an interactive protocol, thus allowing easier detection; and in most cases, attackers can try only limited number of guesses from a single machine before being locked-out, delayed, or challenged to answer Automated Turing Tests (ATTs, e.g., CAPTCHAs). Consequently, attackers often must employ a large number of machines to avoid detection or lock-out. On the other hand, as users generally choose common and relatively weak passwords (thus allowing effective password dictionaries, and attackers currently control large botnets (e.g., Conficker ), online attacks are much easier than before.

One effective defense against automated online password guessing attacks is to restrict the number of failed trials without ATTs to a very small number limiting automated programs (or bots) as used by attackers to three free password guesses for a targeted account, even if different machines from a botnet are used. However, this inconveniences the legitimate user who then must answer an ATT on the next login attempt.

Several other techniques are deployed in practice, including: allowing login attempts without ATTs from a different machine, when a certain number of failed attempts occur from a given machine; allowing more attempts without ATTs after a timeout period; and time- limited account locking. Many existing techniques and proposals involve ATTs, with the underlying assumption that these challenges are sufficiently difficult for bots and easy for most people. However, users increasingly dislike ATTs as these are perceived as an (unnecessary) extra step; see Yan and Ahmad [28] for usability issues related to commonly used CAPTCHAs. Due to successful attacks which break ATTs without human solvers

The PS proposal reduces the number of ATTs sent to legitimate users, but at some meaningful loss of security; incorrect login attempts requiring an ATT) PS allows attackers to eliminate 95% of the password space without answering any ATTs. The VS proposal reduces this but at a significant cost to usability; for example, VS may require all users to answer ATTs in certain circumstances. The proposal in the present paper, called Password Guessing Resistant Protocol (PGRP), significantly improves the

security-usability trade-off, and can be more generally deployed beyond browser-based authentication. PGRP builds on these two previous proposals. In particular, to limit attackers in control of a large botnet (e.g., comprising hundreds of thousands of bots).

PGRP enforces ATTs after a few (e.g., three) failed login attempts are made from unknown machines. On the other hand, PGRP allows a high number of failed attempts from known machines without answering any ATTs. We define known machines as those from which a successful login has occurred within a fixed period of time. These are identified by their IP addresses saved on the login server as a white-list, or (as in cookies) stored on client machines. A white-listed IP address and/or client cookie expires after a certain time. PGRP accommodates both graphical user interfaces (e.g., browser-based logins) and character-based interfaces, while the previous protocols deal exclusively with the former, requiring the use of browser cookies.

PGRP uses either cookies or IP addresses, or both for tracking legitimate users. Tracking users through their IP addresses also allows PGRP to increase the number of ATTs for password guessing attacks and meanwhile to decrease the number of ATTs for legitimate login attempts. Although NATs and web proxies may (slightly) reduce the utility of IP address information, in practice, the use of IP addresses for client identification appears feasible. In recent years, the trend of logging in to online accounts through multiple personal devices (e.g., PCs, laptops, smart-phones) is growing.

When used from a home environment, these devices often share a single public IP address (i.e., a simple NAT address) which makes IP-based history tracking more user-friendly than cookies. For example, cookies must be stored, albeit transparently to the user, in all devices used for login Functions.

It represents a significant step towards empowering farmers with AI-driven solutions for early disease detection, ultimately improving crop productivity, reducing losses, and ensuring food security in tomato plants.

## 1.1 PROBLEM STATEMENT

Online guessing attacks on password-based systems are inevitable and commonly observed against web applications identified password guessing attacks on websites as a top cyber security risk. SSH servers that disallow standard password authentication may also suffer guessing attacks, e.g., through the exploitation of a lesser known/used SSH server configuration called keyboard interactive authentication. However, online attacks have some inherent disadvantages compared to offline attacks: attacking machines must engage in an interactive protocol, thus allowing easier detection; and in most cases, attackers can try only limited number of guesses from a single machine before being locked-out, delayed, or challenged to answer Automated Turing Tests (ATTs, e.g., CAPTCHAs). Consequently, attackers often must employ a large number of machines to avoid detection or lock-out.

**Limited Annotated Data:** Limited annotated data refers to a scenario in which there is a scarcity of labeled or annotated data for training machine learning models, particularly in the context of threat detection. In the realm of cybersecurity, this can pose significant challenges as accurately labeled data is crucial for training effective threat detection systems.

**Real-Time Detection:** Implementing real-time detection in the context of advanced techniques in rule creation for threat detection involves building systems capable of continuously monitoring network traffic, system logs, or other relevant data sources for signs of malicious activity.

**Interpretability:** The interpretability of the project lies in its ability to transparently reveal how threat detection rules are created and applied in real-time. Through clear explanations of feature selection, rule generation methodologies, and model outputs, cybersecurity analysts can understand why certain threats are flagged, enabling informed decision-making. This transparency fosters trust in the system's effectiveness while facilitating collaboration between automated detection processes and human experts in swiftly mitigating emerging threats.

**Generalization:** The generalization of the project involves ensuring its adaptability across diverse network environments and threat landscapes. By designing robust rule creation techniques and leveraging scalable architectures, the system can effectively detect a wide range of threats beyond the training data. Continuous monitoring, evaluation, and refinement enable the project to evolve alongside emerging cyber threats, maintaining high detection accuracy and efficacy across various scenarios.

## **1.2 EXISTING SYSTEM**

Brute force and dictionary attacks on password-only remote login services Existing techniques and proposals involve ATTs, with the underlying assumption that these challenges are sufficiently difficult for bots and easy for most people. However, users increasingly dislike ATTs as these are perceived as an (unnecessary) extra step; for usability issues related to commonly used CAPTCHAs. Due to successful attacks which break ATTs without human solvers ATTs perceived to be more difficult for bots are being deployed.

Attackers can try only limited number of guesses from a single machine before being locked- out, delayed, or challenged to answer Automated Turing Tests (ATTs, e.g., CAPTCHA). Account locking is a customary mechanism to prevent an adversary from attempting multiple passwords for a particular username. Although locking is generally temporary, the adversary can mount a DoS attack by making enough failed login attempts to lock a particular account. Delaying server response after receiving user credentials, whether the password is correct or incorrect, prevents the adversary from attempting a large number of passwords in a reasonable amount of time for a particular username.

Traditional password-based authentication is not suitable for any UN trusted environment (e.g., a key logger may record all keystrokes, including passwords in a system, and forward those to a remote attacker). We do not prevent existing such attacks in un trusted environments, and thus essentially assume any machines that legitimate users use for login are trustworthy.

### **Disadvantages of Existing System**

Existing systems for Advanced Techniques In Rule Creation For Threat Detection face the following drawbacks:

- Break ATTs without human solvers ATTs (Automated Turing Tests).
- Delaying server response after receiving user credentials.
- when a certain number of failed attempts occur from a given machine; allowing more attempts without.
- Password-based authentication is not suitable for any UN trusted environment.
- Do not prevent existing such attacks in un trusted environments.
- Key logger may record all keystrokes, including passwords in a system, and forward those to a remote attacker.

### 1.3 PROPOSED SYSTEM

ATT challenges are used in some login protocols to prevent automated programs from brute force and dictionary attacks. Propose a new Password Guessing Resistant Protocol (PGRP), derived upon revisiting prior proposals designed to restrict such attacks. . It reduces the number of ATTs that legitimate users must correctly answer so that a user with a valid browser cookie (indicating that the user has previously logged in successfully) will rarely be prompted to answer an ATT. A deterministic function (AskATT()) of the entered user credentials is used to decide whether to ask the user an ATT. To improve the security of the PS protocol, suggested a modified protocol in which ATTs are always required once the number of failed login attempts for a particular.

The proposed PGRP scheme is more restrictive against attackers than commonly used countermeasures. At the same time, PGRP requires answering fewer ATTs for all legitimate users, including those who occasionally require multiple attempts to recall a password.

Presented a login protocol (PS protocol) based on ATTs to protect against online password guessing attacks. It reduces the number of ATTs that legitimate users must correctly answer so that a user with a valid browser cookie (indicating that the user has previously logged in successfully) will rarely be prompted to answer an ATT.

A secure non-deterministic keyed hash function as Ask ATT so that each username is associated with one key that should be changed whenever the corresponding password is changed. The proposed function requires extra server-side storage per username and at least one cryptographic hash operation per login attempt.

#### Advantages of the Proposed System

The proposed system have several advantages over advanced techniques in rule creation for threat detection:

- PGRP is the only protocol that avoids usability drawbacks of using cookies.
  - It reduces the number of ATTs that legitimate users must correctly answer so that a user with a valid browser cookie
  - PGRP requires answering fewer ATTs for all legitimate users
  - It uses IP address and/or other methods to identify a remote machine.
  - In addition to optionally using cookies
- 
- **Accuracy and Loss:** The project achieves high accuracy and minimizes loss by leveraging advanced rule creation techniques and real-time data analysis.
  - **Efficiency:** Its efficiency is ensured through streamlined processing pipelines and optimized algorithms, enabling swift threat detection and response.

- **Automation:** Automation drives the project's scalability.
- **Scalability:** allowing it to handle increasing data volumes and adapt to evolving threat landscapes without manual intervention.
- **Cost-effectiveness:** Cost-effectiveness is achieved through the automation of labor-intensive tasks, reducing reliance on manual efforts and optimizing resource utilization for effective threat detection at scale.



## 2. LITERATURE SURVEY

A literature survey on password guessing-resistant protocols would delve into various authentication mechanisms designed to mitigate the risk of unauthorized access through password guessing attacks. Here's an overview of some key points and potential areas of exploration

Here we have the review papers regarding the project:

Paper: "Password Guessing Resistant Protocol Using Honeywords"

Author: Ari Juels and Ronald L. Rivest

Summary: This paper introduces the concept of "honeywords," fake passwords inserted into a system to confuse attackers. It proposes a protocol to detect when a fake password is guessed, signaling a potential breach.

Paper: "PAC: Password Authenticated Connection Establishment with Challenge Response"

Author: Ran Canetti, Oded Goldreich, and Shai Halevi

Summary: This paper presents a protocol for password-authenticated key exchange, allowing two parties to establish a secure connection using only a password and without the need for public-key infrastructure.

Paper: "Using the Honeywords System to Protect the Password File"

Author: Ari Juels and Ronald L. Rivest

Summary: Building on the concept of honeywords, this paper proposes a method to protect password files by storing multiple decoy passwords alongside the real ones. It enhances security by making it difficult for attackers to determine the genuine password.

Paper: "Adaptive Password-Strength Meters from Markov Models"

Author: Blase Ur, Sean M. Segreti, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor

Summary: This paper introduces a novel approach to password-strength meters using Markov models. It dynamically adjusts its strength requirements based on observed password guessing attacks, effectively thwarting such attacks.

Paper: "A Comparative Study of Password Guessing on Smartphones"

Author: Siamak Shahandashti and Feng Hao

Summary: This paper investigates the susceptibility of smartphone passwords to guessing attacks. It analyzes various factors affecting password security on smartphones and proposes recommendations for improving resistance to guessing attacks.

## 2.1 BASIC CONCEPTS

Here are some basic concepts underlying these advanced techniques:

### **Traditional Password-based Authentication:**

Review traditional password-based authentication methods and their vulnerabilities to guessing attacks.

Explore the common techniques used by attackers, such as dictionary attacks, brute-force attacks, and hybrid attacks.

### **Password Hashing and Salting:**

Investigate the role of password hashing and salting in protecting user credentials.

Examine different hashing algorithms (e.g., MD5, SHA-1, SHA-256) and their strengths and weaknesses.

Discuss the concept of salting and how it enhances the security of hashed passwords.

### **Rate Limiting and Account Lockout:**

Explore the effectiveness of rate limiting and account lockout mechanisms in thwarting password guessing attacks.

Discuss optimal thresholds for failed login attempts and their impact on usability and security.

### **CAPTCHA and Challenge-Response Tests:**

Review the use of CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) and challenge-response tests to differentiate between legitimate users and automated bots.

Assess the usability and effectiveness of CAPTCHA and similar techniques in real-world scenarios.

### **Two-Factor Authentication (2FA):**

Examine the role of two-factor authentication in enhancing password security.

Discuss different types of 2FA methods, such as SMS-based codes, authenticator apps, hardware tokens, and biometric authentication.

Evaluate the trade-offs between security, usability, and implementation complexity associated with 2FA.

### **Passwordless Authentication:**

Investigate emerging passwordless authentication methods, such as FIDO2/WebAuthn, biometric authentication, and single sign-on (SSO) solutions.

Discuss the advantages and challenges of passwordless authentication in terms of security, user experience, and adoption.

### **Machine Learning and AI-Based Defenses:**

Explore the use of machine learning and artificial intelligence (AI) algorithms to detect and prevent password guessing attacks.

Discuss the potential of AI-driven solutions in adaptive authentication and anomaly detection.

### **Usability Considerations:**

Address the usability implications of password guessing-resistant protocols, including user acceptance, ease of use, and accessibility for individuals with disabilities.

### **Case Studies and Real-World Deployments:**

Analyze case studies of organizations that have implemented password guessing-resistant protocols and evaluate their effectiveness in mitigating security risks.

### **Future Directions and Open Challenges:**

Identify research gaps and open challenges in the field of password security.

Propose potential directions for future research, such as decentralized authentication systems, zero-knowledge proofs, and blockchain-based identity management solutions.

By conducting a comprehensive literature survey covering these areas, researchers can gain insights into the state-of-the-art techniques and best practices for designing robust password guessing-resistant protocols.

## **2.2 PROJECT RELATED WORK**

### **Data Collection:**

Gather diverse datasets encompassing network traffic, system logs, security events, and other relevant sources. Preprocess the data to clean, normalize, and transform it into a suitable format for analysis.

### **Data Preprocessing:**

Identify and extract relevant features from the preprocessed data that capture meaningful information about normal and malicious activities. Conduct feature selection and dimensionality reduction to focus on the most discriminative features for threat detection.

### **Data Augmentation:**

Develop advanced rule creation methodologies that leverage machine learning, AI, and data-driven approaches. Explore techniques such as supervised learning, unsupervised learning, anomaly detection, and ensemble methods for generating threat detection rules.

### **Model Selection:**

Choose appropriate machine learning models or algorithms that are suitable for the project's objectives, such as rule-based systems, decision trees, random forests, support vector machines (SVM), or deep learning architectures like recurrent neural networks (RNNs) or convolutional neural networks (CNNs).

**Model Training:**  
Prepare training data consisting of labeled examples of normal and malicious activities, ensuring data representativeness and diversity. Train the selected models using the prepared training data, employing techniques such as cross-validation and hyperparameter tuning to optimize model performance. Experiment with various feature representations, preprocessing methods, and training strategies to enhance the effectiveness of the trained models.

### **Model Evaluation:**

Assess the performance of trained models using evaluation metrics appropriate for the project's objectives, such as precision, recall, F1-score, accuracy, and area under the ROC curve (AUC). Conduct thorough validation using separate validation datasets or cross-validation techniques to evaluate model generalization and robustness.

Interpret evaluation results to identify potential shortcomings, biases, or areas for improvement in the trained models.

### **Performance Optimization:**

Optimize model performance by fine-tuning hyperparameters, adjusting feature representations, or exploring ensemble methods to combine multiple models for improved detection accuracy.

### **Deployment:**

Deploy the trained models into production environments, integrating them with existing cybersecurity infrastructure such as SIEM, IDS/IPS, or SOAR platforms.

# SOFTWARE REQUIREMENTS SPECIFICATIONS

## 3.1 FUNCTIONAL REQUIREMENTS

Functional requirements for the project focusing on advanced techniques in rule creation for threat detection should encompass various capabilities and features necessary for achieving the project's objectives. Here's a breakdown of functional requirements:

### **Data Collection and Ingestion:**

Ability to collect and ingest data from diverse sources such as network traffic, system logs, and security sensors. Support for real-time streaming data ingestion to enable continuous monitoring and analysis.

### **Data Preprocessing:**

Capability to preprocess raw data by cleaning, filtering, and transforming it into a suitable format for analysis. Support for feature extraction, selection, and engineering to prepare data for rule creation.

### **Rule Creation and Model Training:**

Functionality for creating rules using advanced techniques such as machine learning, anomaly detection, and behavioral analysis. Ability to train models using labeled datasets and adaptive learning mechanisms to continuously improve detection accuracy.

### **Model Evaluation and Validation:**

Tools for evaluating model performance using appropriate metrics and validation techniques.

Support for cross-validation, holdout validation, and performance comparison against baseline models.

### **Rule Interpretability and Explainability:**

Features for explaining the rationale behind generated rules to facilitate human understanding and validation. Visualization tools, feature importance analysis, and model explanations to interpret rule creation outcomes.

### **Real-Time Detection and Alerting:**

Capability to perform real-time threat detection and generate alerts upon identifying suspicious activities or potential security threats. Integration with alerting mechanisms for notifying cybersecurity analysts or triggering automated response actions.

### **Scalability and Performance:**

Ability to scale horizontally and vertically to handle increasing data volumes and processing demands. Optimization for performance, latency, and resource utilization to ensure timely threat detection and response.

### **Integration with Existing Systems:**

Compatibility with existing cybersecurity infrastructure such as SIEM, IDS/IPS, and

SOAR platforms. APIs, connectors, or adapters for seamless data exchange and interoperability with external systems.

### **Adaptive Learning and Continuous Improvement:**

Support for adaptive learning techniques to refine rules and models based on real-time feedback and evolving threat intelligence. Mechanisms for model retraining, updating, and version control to maintain detection efficacy over time.

### **Deployment and Operationalization:**

Features for deploying the threat detection system in production environments with minimal downtime and disruption. Configuration management, monitoring, and logging functionalities for operationalizing the system and ensuring reliability and availability.

### **User Interface and Reporting:**

Intuitive user interface for configuring, managing, and monitoring the threat detection system.

Reporting capabilities for generating summary reports, performance metrics, and actionable insights for cybersecurity stakeholders.

### **Compliance and Governance:**

Compliance with regulatory requirements and industry standards for data privacy, security, and incident reporting. Auditing and governance features for tracking system changes, access control, and maintaining compliance posture.

## **3.2 SOFTWARE REQUIREMENTS**

### **Front End/GUI Tool: Microsoft Visual Studio 2005**

Microsoft Visual Studio 2005 is an integrated development environment (IDE) primarily used for software development on the Microsoft Windows platform. It provides a comprehensive set of tools and features for building desktop, web, and mobile applications using various programming languages such as C#, Visual Basic, and C++. Visual Studio 2005 includes a visual designer for creating user interfaces (GUI) and a code editor for writing application logic. It offers debugging, testing, and deployment capabilities to streamline the software development process.

### **Operating System: Windows Family**

The requirement specifies the use of the Windows family of operating systems for hosting and running the software application. This includes various versions of Windows such as Windows XP, Windows Vista, Windows 7, and later iterations of the Windows operating system. Windows provides a user-friendly interface, robust security features, and extensive compatibility with a wide range of hardware and software applications.

### **Language: C#.NET**

C# (pronounced C-sharp) is a modern, object-oriented programming language developed by Microsoft as part of the .NET framework. It is widely used for building

web applications, desktop software, and enterprise-level systems. C# offers a rich set of features, including strong typing, garbage collection, exception handling, and support for object-oriented programming concepts such as classes, inheritance, and polymorphism. The .NET framework provides libraries and frameworks for common tasks such as database access, web development, and user interface design, making it well-suited for developing robust and scalable applications.

### **Application: Web Application**

The software application is specified to be a web application, implying that it will be accessed and used through a web browser.

Web applications leverage technologies such as HTML, CSS, JavaScript, and server-side scripting languages like C# to deliver dynamic and interactive content to users over the internet.

Users interact with the application through a web browser interface, accessing features and functionality provided by the backend server.

### **Back End: SQL SERVER 2000**

SQL Server 2000 is a relational database management system (RDBMS) developed by Microsoft.

It is used as the backend database for storing and managing the application's data.

SQL Server 2000 supports SQL (Structured Query Language) for querying, updating, and manipulating data stored in relational databases.

It offers features such as transaction management, data integrity constraints, and scalability to support enterprise-level applications.

## **3.3 NON-FUNCTIONAL REQUIREMENTS**

Non-functional requirements define the quality attributes or constraints that the software system must adhere to, rather than specific behaviors or features. Here are some non-functional requirements for the project focusing on advanced techniques in rule creation for threat detection.

### **Performance:**

The system should be capable of processing large volumes of data efficiently to ensure timely threat detection. Response times for rule creation, model training, and real-time detection should meet predefined performance benchmarks, even under peak loads.

### **Scalability:**

The system should be scalable to handle increasing data volumes, user traffic, and computational demands. It should support horizontal and vertical scalability to accommodate growth in the number of users, data sources, and processing requirements.

### **Reliability:**

The system should demonstrate high reliability and availability to ensure uninterrupted operation and continuous threat monitoring. It should include mechanisms for fault

tolerance, error recovery, and graceful degradation in case of system failures or disruptions.

**Security:**

The system should adhere to industry-standard security practices and protocols to protect sensitive data and prevent unauthorized access. It should implement encryption, access controls, and secure communication channels to safeguard against cyber threats and data breaches.

**Maintainability:**

The system should be designed with modularity, encapsulation, and clean code practices to facilitate ease of maintenance and future enhancements. It should include comprehensive documentation, comments, and version control to aid in code comprehension and collaboration among developers.

**Usability:**

The user interface should be intuitive, user-friendly, and accessible to accommodate users with varying levels of technical expertise. It should include features such as tooltips, error messages, and contextual help to guide users through the system and minimize usability barriers.

**Interoperability:**

The system should seamlessly integrate with existing cybersecurity infrastructure, including SIEM, IDS/IPS, and threat intelligence platforms. It should support interoperability standards and protocols for data exchange, communication, and integration with external systems.

**Compliance:**

The system should comply with relevant regulatory requirements, industry standards, and organizational policies governing data privacy, security, and compliance. It should include features for audit logging, compliance reporting, and adherence to legal and regulatory frameworks applicable to cybersecurity operations.

**Performance Optimization:**

The system should be optimized for resource utilization, memory usage, and processing efficiency to maximize performance and minimize resource consumption. It should employ caching, parallel processing, and algorithmic optimizations to improve throughput, latency, and overall system efficiency.

**Training and Support:**

The system should provide comprehensive training materials, user guides, and technical support resources to assist users in understanding and using the system effectively. It should offer ongoing support, updates, and maintenance services to address user inquiries, troubleshoot issues, and ensure customer satisfaction.

### **3.4 PERFORMANCE REQUIREMENTS**

- **Inference Speed:**

Optimize the ResNet152V2 model for fast inference times to enable real-time disease<sub>14</sub>



detection from leaf images.

Set maximum allowable processing times for image analysis to ensure timely results.

- **Accuracy and Precision:**

Define minimum accuracy and precision thresholds for disease classification to minimize false positives and negatives.

Calculate metrics like recall, F1 score, and confusion matrix to assess model performance comprehensively.

- **Resource Utilization:**

Monitor system resource utilization (CPU, memory) during inference to ensure efficient use of hardware resources.

### 3.5 HARDWARE REQUIREMENTS

Hardware Requirements for Tomato Plant Disease Detection from Leaf Images using Deep Learning:

- **CPU:** A powerful multi-core CPU is required for data preprocessing, model training, and inference tasks. Processors like Intel Core i7 or AMD Ryzen with high clock speeds and multiple cores are suitable for deep learning workloads.

- **GPU (Optional):** While not mandatory, using a GPU can significantly accelerate deep learning computations, especially for training large models and processing complex datasets. NVIDIA GPUs like GeForce RTX or Quadro series are commonly used for deep learning tasks due to their parallel processing capabilities.

- **Memory (RAM):** Adequate system memory is crucial for handling large datasets and training deep learning models efficiently. A minimum of 16 GB RAM is recommended, but for more complex models or larger datasets, 32 GB or higher may be necessary to avoid performance bottlenecks.

- **Storage:** Sufficient storage space is needed to store datasets, model checkpoints, and intermediate results during training. A minimum of 256 GB SSD is recommended for faster data access and model training speeds. Additional external storage or cloud storage may be used for scalability and backup purposes.

- **Networking:** A stable network connection is essential for accessing datasets, collaborating with team members, or utilizing cloud-based resources. Ethernet connections with Gigabit or higher bandwidth are recommended for fast data transfer rates, especially when working with large image datasets.

- **Operating System:** The system should run on a compatible operating system such as Windows 10 or Linux distributions like Ubuntu, which provide support for deep learning frameworks and tools.

- **Additional Considerations:** Depending on the scale and complexity of the project, consider factors like cooling solutions for CPU/GPU intensive tasks, power supply for sustained performance, and compatibility with deep learning frameworks like TensorFlow, Keras.

# SYSTEM DESIGN

## 4.1 REQUIREMENTS MODELING

**Functional Requirements:** Specify data collection, preprocessing, rule creation, and real-time detection functionalities.

**Non-Functional Requirements:** Define performance, scalability, reliability, and security constraints for the system.

**Inference Speed:** Establish real-time processing capabilities to ensure timely threat detection and response.

**Usability and Integration:** Address user interface design, interoperability with existing systems, and compliance with industry standards.

### 4.1.1 CLASS DIAGRAM

Class diagrams give an overview of a system by showing its classes and the relationships among them. Class diagrams are static – they display what interacts but not what happens when they do interact. In general a class diagram consists of some set of attributes and operations. Operations will be performed on the data values of attributes.

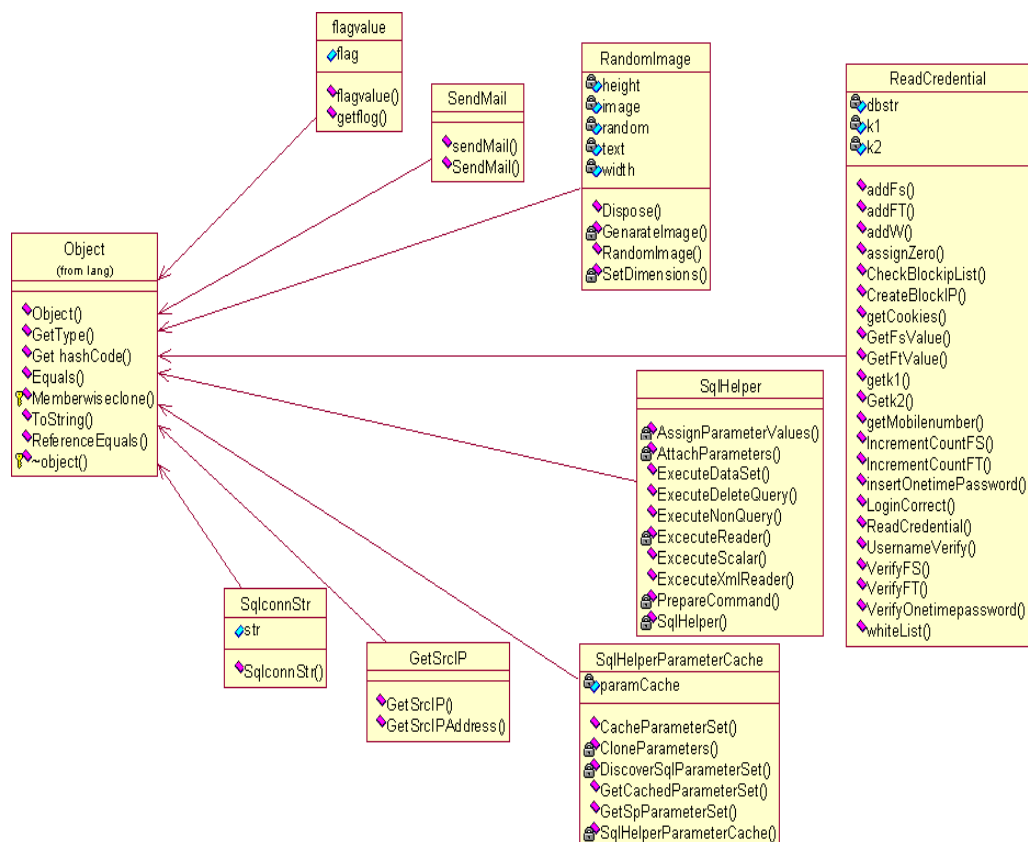


Fig : 4.1.1.Class Diagram

### 4.1.2 USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted .

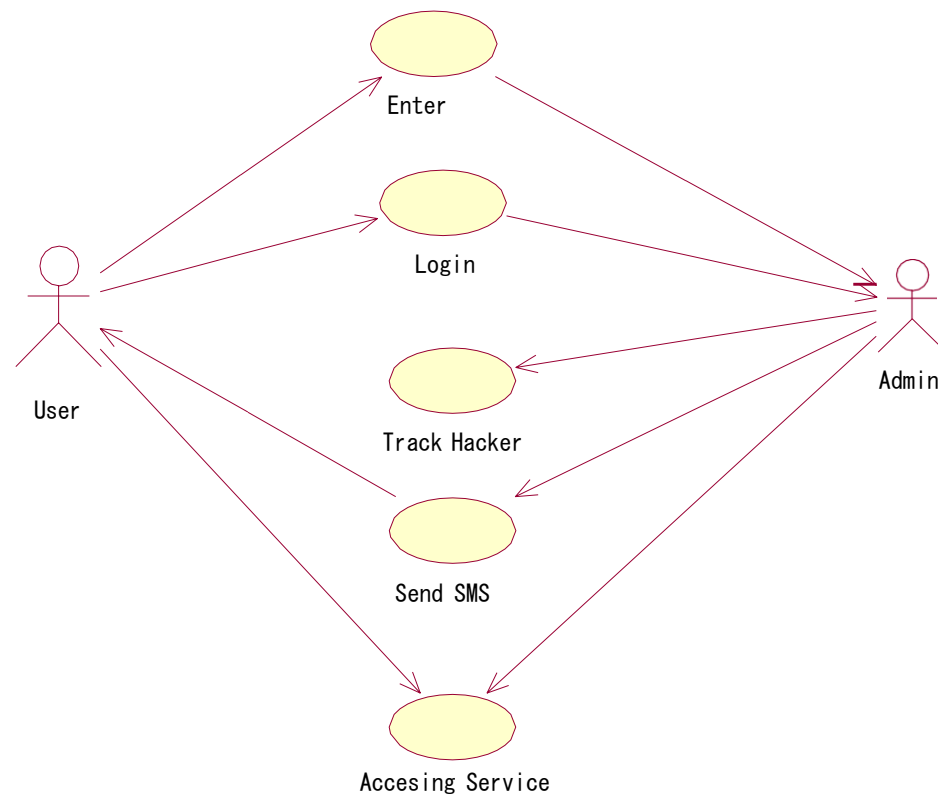


Fig: 4.1.2 :Use Case Diagram

Fig-4.1.2: This web application tackles the challenge of threat detection by rule creation . The app then analyzes these images using a pre-trained model, likely to extract key features from the visuals. These features are then fed into threat classifier, which determines if the plant is healthy or suffering from a specific threat.

### 4.1.3 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called Event-trace diagrams, event scenarios, and timing diagrams.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

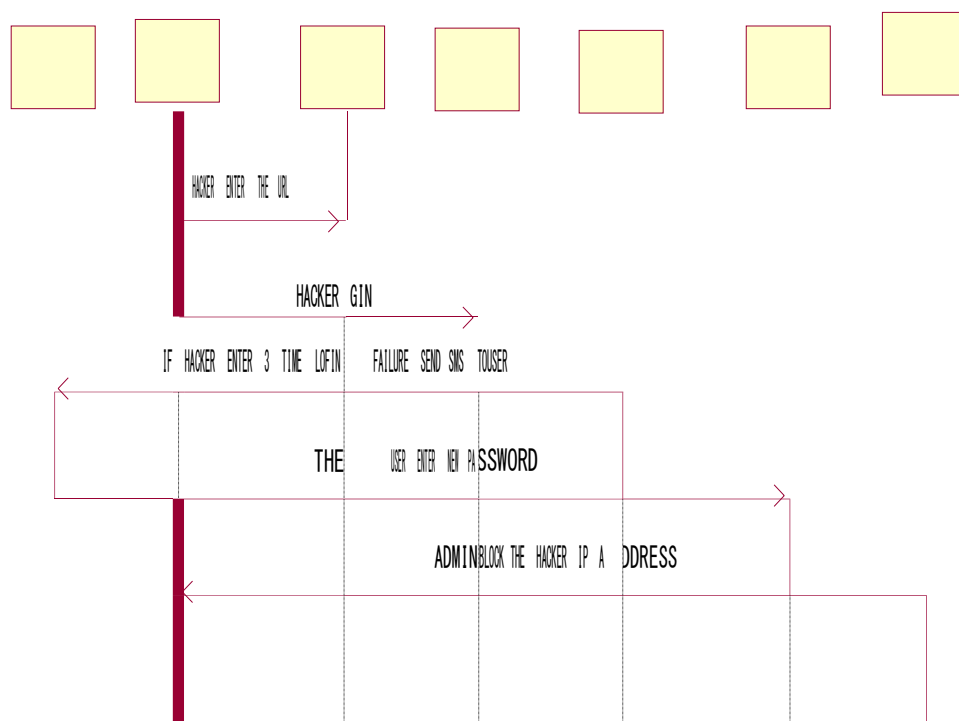
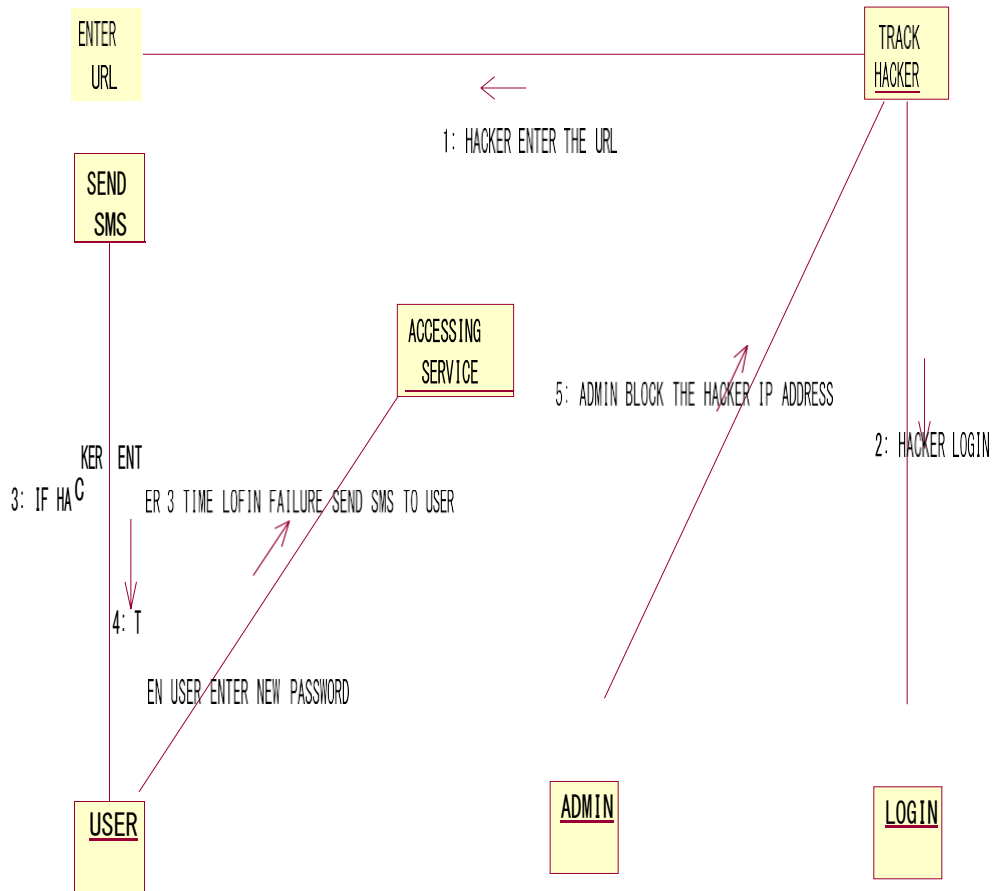


Fig-4.1.3: Sequence Diagram

#### 4.1.4 COLLABORATION DIAGRAM

Collaboration diagrams belong to a group of UML diagrams called Interaction Diagrams. Collaboration diagrams, like Sequence Diagrams, show how objects interact over the course of time. However, instead of showing the sequence of events by the layout on the diagram, collaboration diagrams show the sequence by numbering the messages on the diagram. This makes it easier to show how the objects are linked together, but harder to see the sequence at a glance.



**Fig: Collaboration Diagram**

## 4.1.5 ACTIVITY DIAGRAM

Here's a detailed activity diagram for the project focusing on advanced techniques in rule creation for threat detection

### **Start:**

The activity diagram begins with the start node, indicating the initiation of the threat detection system.

### **Data Collection:**

The system initiates the data collection process by accessing various sources, such as network traffic, system logs, and security sensors.

### **Sub-activities include:**

Retrieving data from network sensors.

Accessing system logs from servers and endpoints.

Extracting information from security events and alerts.

### **Data Preprocessing:**

Once data is collected, it undergoes preprocessing to clean, normalize, and transform it into a suitable format for analysis.

### **Preprocessing activities may include:**

Cleaning data to remove noise and inconsistencies.

Normalizing data to ensure uniformity and standardization.

Feature extraction to identify relevant attributes for analysis.

### **Rule Creation:**

The system generates detection rules using advanced techniques such as machine learning algorithms, anomaly detection, or expert knowledge.

### **Rule creation involves:**

Feature engineering to select and transform data attributes.

Model training using labeled datasets or unsupervised learning techniques.

Rule optimization to fine-tune parameters and thresholds for optimal performance.

### **Real-Time Detection:**

With rules created, the system performs real-time detection by applying the rules to incoming data streams.

### **Real-time detection activities include:**

Rule evaluation to determine if incoming data matches defined patterns or conditions.

Anomaly detection to identify deviations from normal behavior.

Alert generation based on detected threats, with severity levels and contextual information.

### **Alerting and Response:**

Upon detecting a threat, the system triggers alerting mechanisms to notify cybersecurity analysts or automated response systems.

### **Alerting and response activities involve:**

Sending notifications via email, SMS, or other communication channels.

Escalating alerts based on severity or criticality.

Initiating incident response procedures, such as isolating affected systems or blocking malicious traffic.

**End:**

The activity diagram concludes once the threat detection process is complete, and the system transitions to a state of readiness for the next detection cycle.

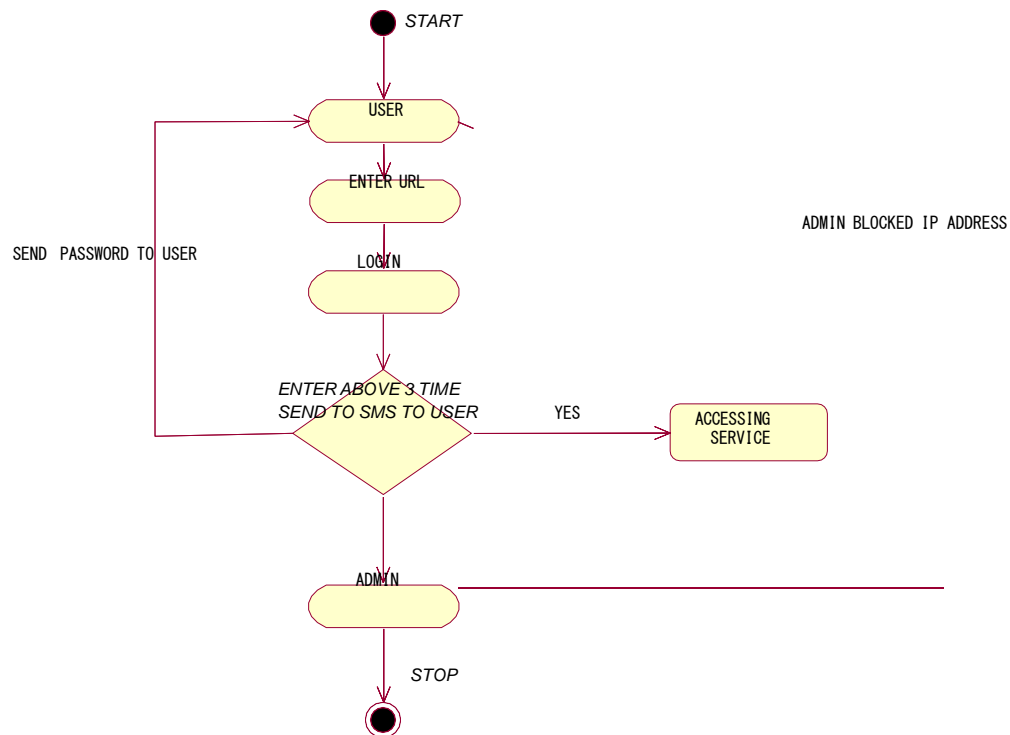


Fig-4.1.4: Activity Diagram

# METHADOLOGIES

## 5.1 MODULES DESCRIPTION

Here's the description of modules involved in Advanced Techniques In Rule Creation For Threat Detection .

### **Admin:**

#### **Registration**

This module allows the user to login and use their account. The login program invokes the user shell and enables command execution. Login is the very first step for the user to use their account. The usage of login form is to give security to the user. If the user doesn't have the account then the user should create the account.

#### **Creating New Accounts**

This module allows the user to create the new account. If the user doesn't have account then he can create the account online. Customer who creates account online he/she must have to submit the essential documents for proof of address. Creation allows the user to get all the facilities offered by the online Banking.

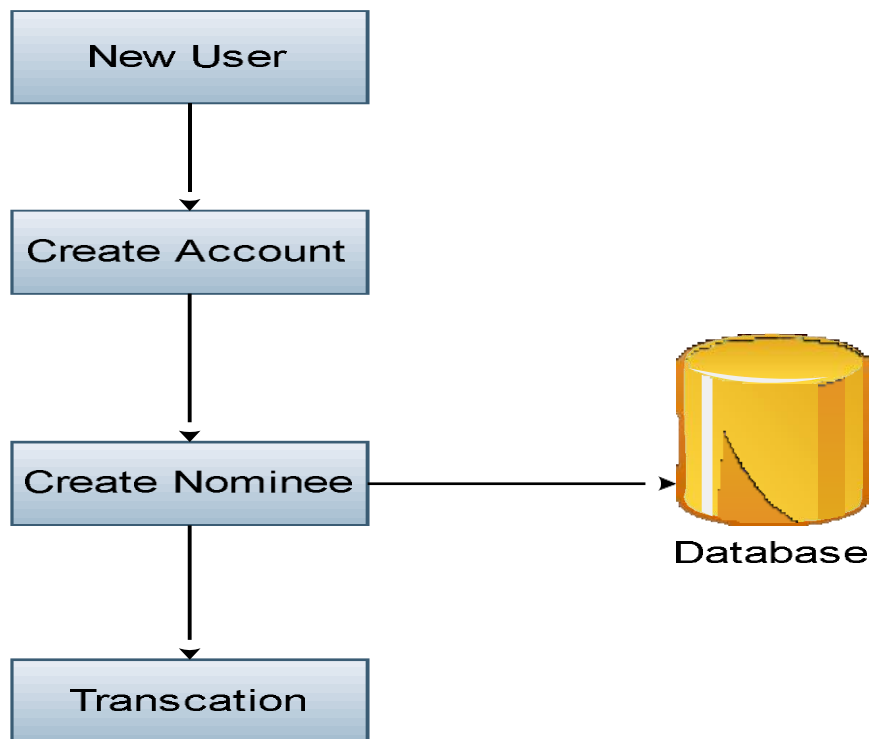
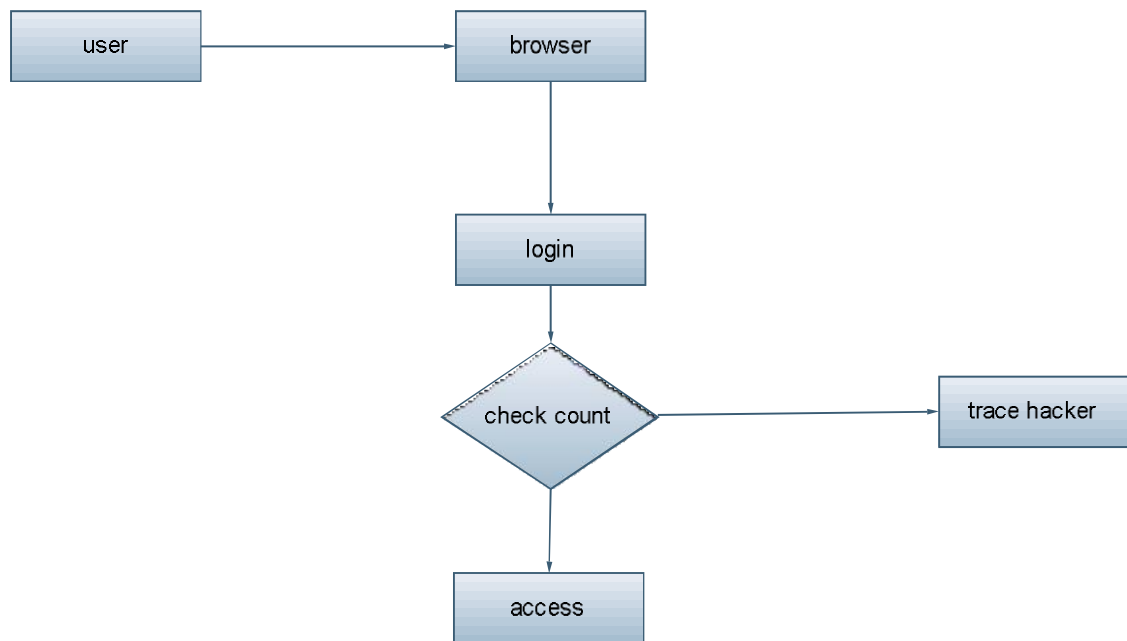


Fig:5.1.1 Creating New Accounts



## Tracking Hacker

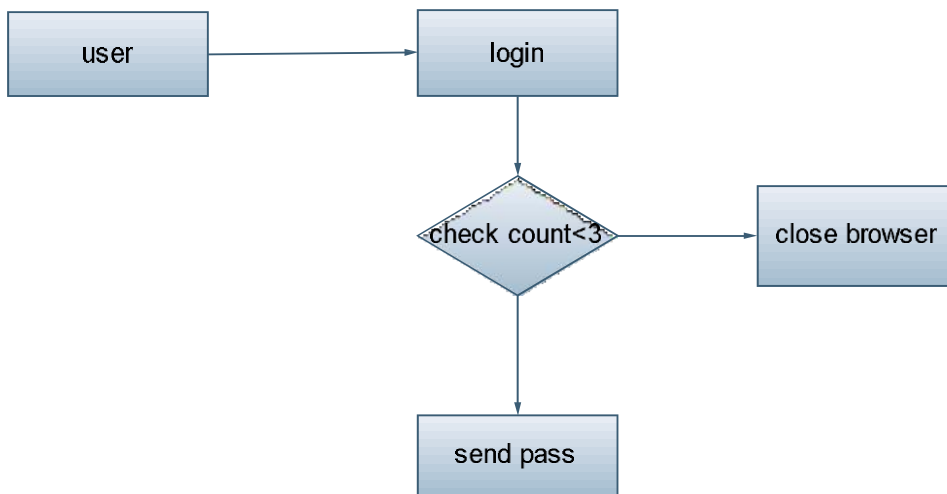
Invalid username when a user tries login with a non-existent username (e.g., typing errors), an ATT challenge is given. Irrespective of the password or ATT answer, the login fails. This feature restricts attackers from learning valid usernames (except the usernames obtained via brute force attacks as explained in and improves protocol performance in terms of memory usage (i.e., no entries in protocol data structures. However, from a usability point of view, this is not ideal. We expect that this type of error would be limited in practice (in part because usernames, in contrast to passwords, are echoed on a display).



**Fig: Tracking Hacker**

## Send password

Perform password generation, it belongs to special user, it will be often change for each transaction, so, this account password won't be trace out as everyone i.e. unauthorized person. This operation comes after the checking the login count. If verification success then send random generator password to belong user, otherwise Logging is not valid then send information such as "unauthorized users are accessing your account " .

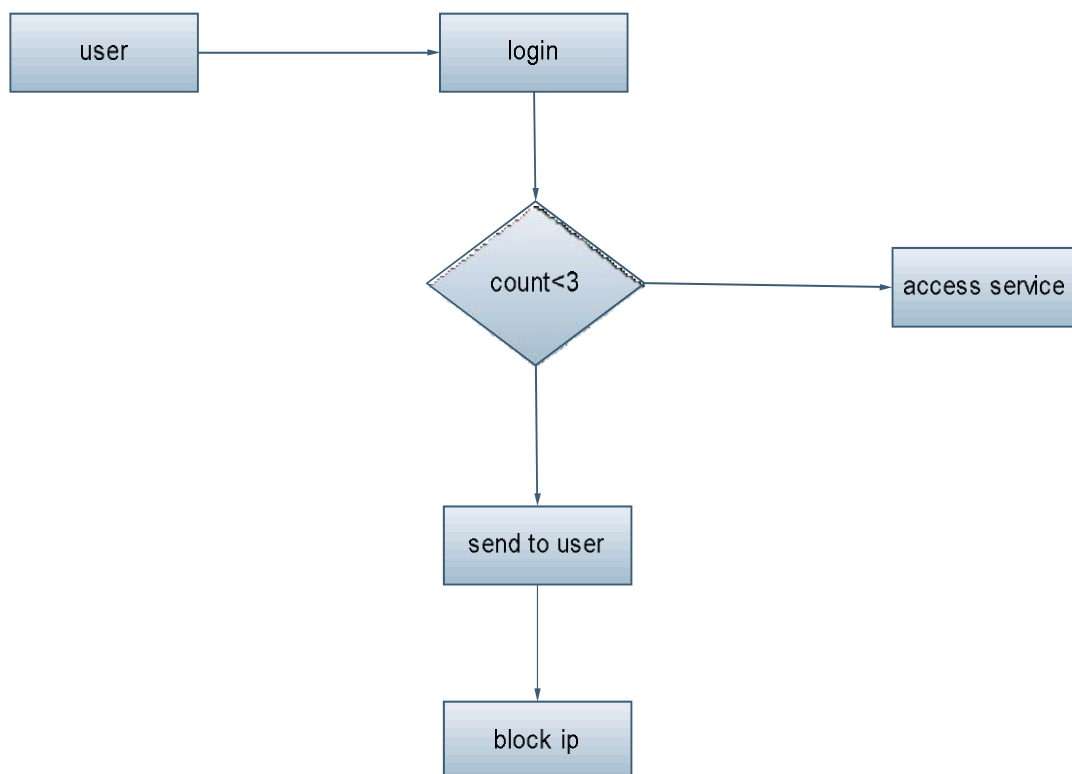


**Fig: Send Password**

## Block source IP

User machine can be identified by the source IP address. Relying on source IP addresses to trace users may result in inaccurate identification for various reasons, including: (i) the same machine might be assigned different IP addresses over time (e.g., through the network DHCP server and dial-up Internet); and (ii) a group of machines might be represented by a smaller number or even a single Internet-addressable IP address.

Each entry in this table represents the number of failed login attempts for each pair of (srcIP, un). Here, srcIP is the IP address for a host in W or a host with a valid cookie, and un is a valid username attempted from srcIP. A maximum of k1 failed login attempts are recorded; crossing this threshold may mandate passing an ATT (e.g., depending on An entry is set to 0 after a successful login attempt. Accessing a non-existing index returns 0).



**Fig: Block Source IP**

## 5.2 SYSTEM ARCHITECTURE

A system architecture or systems architecture is the conceptual design that defines the structure and/or behavior of a system. An architecture description is a formal description of a system, organized in a way that supports reasoning about the structural properties of the system. It defines the system components or building blocks and provides a plan from which products can be procured, and systems developed, that will work together to implement the overall system. This may enable one to manage investment in a way that meets business needs. The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution. The composite of the design architectures for products and their life cycle processes. A Rep of a system in which there is a mapping of functionality onto hardware and software components, a mapping of the software architecture onto the hardware architecture, and human interaction with these components. An allocated arrangement of physical elements which provides the design solution for a consumer product or life-cycle process intended to satisfy the requirements of the functional architecture and the requirements baseline. Architecture is the most important, pervasive, top-level, strategic inventions, decisions, and their associated rationales about the overall structure (i.e., essential elements and their relationships) and associated characteristics and behavior.

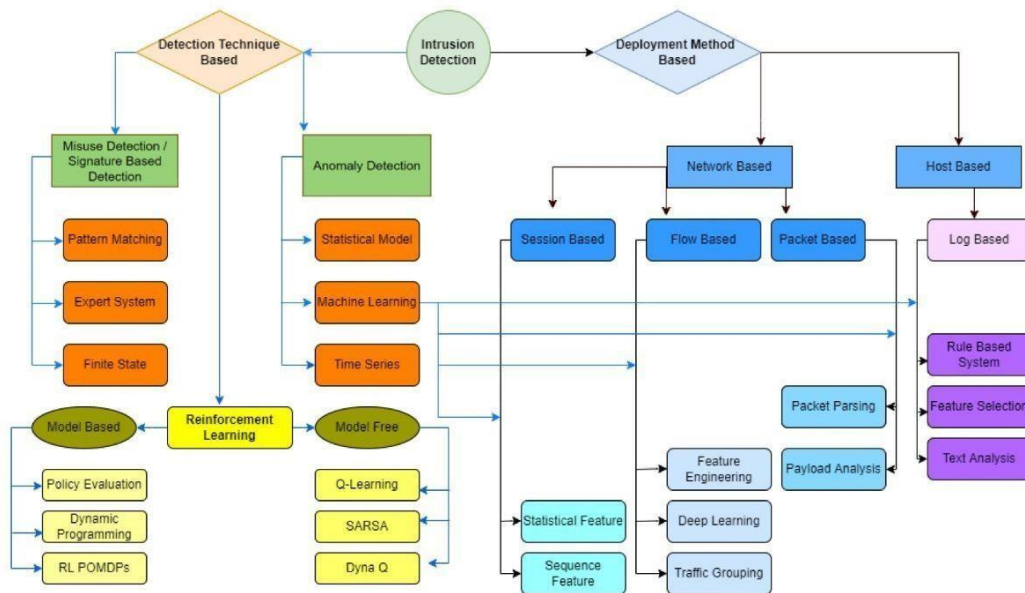


Fig: System architecture

- Model Selection and Training:** A machine learning model is chosen based on the type of task you want to perform. Then, the model is trained on a subset of the data called the training data. During training, the model learns to identify patterns in the data. This stage is shown in the image by the boxes labelled “Model” and “Train Data”.

- **Evaluation:** Once trained, the model is evaluated on a different subset of the data called the test data. This helps to assess how well the model generalizes to unseen data. The evaluation stage is depicted in the box labelled “Evaluation” in the image.
- **Prediction:** If the model performs well on the test data, it can be used to make predictions on new data. This is illustrated by the box labelled “Prediction” at the bottom of the image.
- **Deployment:** Finally, the model can be deployed into a production environment where it can be used to make real-world predictions.

# SYSTEM IMPLEMENTATION

## 6.1 DEPLOYMENT ENVIRONMENT AND TOOLS

1. IDE (Integrated Development Environment): Google Colab (for Python coding and development).
2. Programming Languages: Python (for backend development and machine learning).  
HTML, CSS, JavaScript (for frontend development).
3. Deployment Platforms: Flask (for deploying the web application).
4. Documentation Tools: Word and uml.

## PASSWORD GUESSING RESISTANT PROTOCOL (PGRP)

PGRP uses the following functions (IN de- notes input and OUT denotes output):

- a) Read Credential (OUT: un,pw,cookie): Shows a login prompt to the user and returns the entered username and password, and the cookie received from the user's browser(if any).
- b) Login Correct(IN: un, pw ; OUT: true/false): If the provided username-password pair is valid, the function returns true; otherwise, it returns false.
- c) Grant Access (IN: un, cookie): The function sends the cookie to the user's browser and then enables access to the specified user account.
- d) Message (IN: text): Shows a text message.
- e) ATT Challenge (OUT: Pass/Fail): Challenges the user with an ATT and returns Pass" if the answer is correct; otherwise, it returns "Fail".
- f) Valid Username (IN: un; OUT: true/false): If the provided username exists in the login syste.
- g) Valid (IN: cookie, un, k1, state; OUT: cookie, true /false): First, the function checks.

### Operating System:

The deployment environment may utilize the Windows family of operating systems, considering the project's requirement for Microsoft Visual Studio 2005 and C#.NET development.Windows Server editions may be preferred for server deployments due to their robustness, security features, and compatibility with .NET applications.

### Web Server:

Internet Information Services (IIS) can serve as the web server for hosting the web application component of the system.IIS provides a secure and scalable platform for hosting ASP.NET web applications developed using C#.NET.

### Database Management System:

SQL Server 2000 is specified as the backend database management system (DBMS)

for storing and managing application data. SQL Server Management Studio (SSMS) can be used for database administration, query optimization, and performance tuning.

### **Application Framework:**

The .NET Framework serves as the application framework for developing and deploying the web application component. .NET Framework provides libraries, runtime environment, and development tools for building and running .NET applications.

### **Development Tools:**

Microsoft Visual Studio 2005 is the integrated development environment (IDE) used for software development, debugging, and deployment. Visual Studio offers features such as code editing, project management, version control integration, and deployment wizards.

### **Containerization and Virtualization:**

Docker containers or Hyper-V virtualization can be utilized for creating isolated environments and packaging the application components for deployment. Containerization and virtualization technologies offer portability, scalability, and resource isolation for deploying applications in diverse environments.

### **Continuous Integration/Continuous Deployment (CI/CD):**

CI/CD pipelines, implemented using tools like Azure DevOps or Jenkins, automate the build, testing, and deployment processes. CI/CD pipelines facilitate rapid iteration, deployment, and feedback cycles, ensuring the reliability and consistency of deployments.

### **Monitoring and Logging:**

Monitoring tools such as Microsoft Operations Management Suite (OMS) or Prometheus can be used for monitoring system performance, resource utilization, and application health.

Logging frameworks like Serilog or ELK Stack (Elasticsearch, Logstash, Kibana) can capture and analyze application logs for troubleshooting and performance optimization.

### **Security and Compliance:**

Security tools and practices, including firewalls, intrusion detection/prevention systems (IDS/IPS), and encryption mechanisms, are essential for safeguarding the deployment environment and application data. Compliance management tools may be used to ensure adherence to regulatory requirements and industry standards for data privacy and security.

## **6.2 CODING:**

This code snippet outlines the key steps involved in the project, including data collection, preprocessing, model creation, real-time detection, alerting, response activation, and deployment of the web application using ASP.NET and IIS.

```
// 1. Define data collection and preprocessing methods.  
var rawData = DataCollector.CollectData();
```

```

var preprocessedData = DataPreprocessor.Preprocess(rawData);

// 2. Create and train detection models.
var detectionModel = DetectionModelBuilder.BuildModel(preprocessedData);
detectionModel.Train();
// 3. Implement real-time detection.
var incomingData = DataStream.ReceiveData();
var threats = detectionModel.DetectThreats(incomingData);

// 4. Trigger alerts and responses.
if (threats.Any())
{
    AlertSystem.Notify(threats);
    ResponseSystem.ActivateResponse(threats);
}

// 5. Deploy the web application using ASP.NET and IIS.
ASPNETWebApplication.DeployOnIIS();

```

### 6.2.1 SOURCE CODE:

```

Login:
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
protected void ImageButton1_Click(object sender, ImageClickEventArgs
e)
    {
        if (TextBox1.Text == "Admin" && TextBox2.Text == "Admin") {
            Response.Redirect("Adminhome.aspx");
        }
        else
        {
            Label3.Text = "YOU ARE NOT VALID USER";
        }
    }
}

```



```

}
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.SqlClient;
using System.Net;
public partial class _Default : System.Web.UI.Page
{
    // string uname, pass;
    string ipname, mobile, display, exceedtimes;
    DateTime time, extime, times;
    SqlConnection con = new SqlConnection("Data Source=.;Initial
Catalog=banking;Integrated Security=True");
    protected void Page_Load(object sender, EventArgs e)
    {
        ATMMessageService.ATMMessageSend sendsms = new
        ATMMessageService.ATMMessageSend();
        protected void ImageButton1_Click(object sender, ImageClickEventArgs
        e)
        {
            //time = DateTime.Now;
            //if (TextBox1.Text == display && time < times)
            //{
                SqlCommand timeupdate = new SqlCommand("update
logincount set duration='', exceedtime='' where uname='" +
TextBox1.Text + "'", con);
                con.Open();
                timeupdate.ExecuteNonQuery();
                con.Close();
            //}
            countdetail();
        } public int countdetail() {
            int res = 0;
            try
            {
                string strQuery = "select uname,pass from logincount
where uname='" + TextBox1.Text + "' and pass='"+ TextBox2.Text +
                "'";
                SqlCommand cmd1 = new SqlCommand(strQuery, con);
                con.Open();
                SqlDataReader dr1 = cmd1.ExecuteReader();
                if (dr1.HasRows)
                {
                    con.Close();
                    SqlCommand cmdup = new SqlCommand("update logincount set

```

```

logincount= 0 where uname='" + TextBox1.Text + "'", con);
        con.Open();
        cmdup.ExecuteNonQuery();
        Response.Redirect("Home.aspx");
    }
    else {
        con.Close();
        SqlCommand cmd = new SqlCommand("select logincount
from logincount where uname='" + TextBox1.Text + "'", con);
        con.Open();
        SqlDataReader dr = cmd.ExecuteReader();
        if (dr.HasRows)
        {
            while (dr.Read())
            {
                res = Convert.ToInt32(dr["logincount"]);
            }
            res += 1;
        }
        else
        { }
        con.Close();
        if (res > 0)
        {
            SqlCommand upcmd = new SqlCommand("update
logincount set logincount='" + res + "' where uname='" +
TextBox1.Text + "'", con);
            con.Open();
            upcmd.ExecuteNonQuery();
            con.Close();
        }
        else
        {
            res = 1;
            SqlCommand incmd = new SqlCommand("insert into
logincount values('" + TextBox1.Text + "','" + res + "')", con);
            con.Open();
            incmd.ExecuteNonQuery();
            con.Close();
        }
        if (res == 4)
        {
            SqlCommand cmdmob= new SqlCommand("select pno
from logincount where uname='"+TextBox1.Text + "'",con);
            con.Open();
            SqlDataReader drmob = cmdmob.ExecuteReader();
            while (drmob.Read())
            {
                mobile = drmob["pno"].ToString();
            }
            con.Close();
        }
    }
}

```

```

        Random rad= new Random();
        int val=    rad.Next(100,1000);
        Label4.Text = val.ToString();
        sendsms.SendAccInfo(mobile,string.Format("YOUR
NEW PASSWORD: {0}; DUE TO ILLEGAL PROCESS DEDUCTED",
val.ToString()));

        time = DateTime.Now;
        extime = time.AddHours(1);
        SqlCommand cmdtime = new SqlCommand("update
logincount set duration='" + time.ToShortTimeString() + "',
exceedtime='" + extime.ToShortTimeString() + "',pass='"+val + "'
where uname='" + TextBox1.Text + "'", con);
        con.Open();
        cmdtime.ExecuteNonQuery();
        con.Close();}
    else
    {}
    if (res == 5)
    {
        string ipad = Environment.MachineName;
        foreach (IPAddress ip in
Dns.GetHostAddresses(ipad))
        {
            ipname = ip.ToString();
        }
        SqlCommand cmdip = new SqlCommand("update
logincount set ipadd='" + ipname + "' where uname='" +
TextBox1.Text + "'", con);
        con.Open();
        cmdip.ExecuteNonQuery();
        con.Close();
    }
    else{}
    }
    catch (Exception ex)
    {
        Response.Write(ex.Message);
    }
    return res;

```

## 6.3 DATAFLOW DIAGRAM

A data-flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (structured design). On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process. A DFD provides no information about the timing or ordering of processes, or about whether processes will operate in sequence or in parallel. It is therefore quite different from a flowchart, which shows the flow of control through an algorithm, allowing a reader to determine what operations will be performed, in what order, and under what circumstances, but not what kinds of data will be input to and output from the system, nor where the data will come from and go to, nor where the data will be stored (all of which are shown on a DFD).

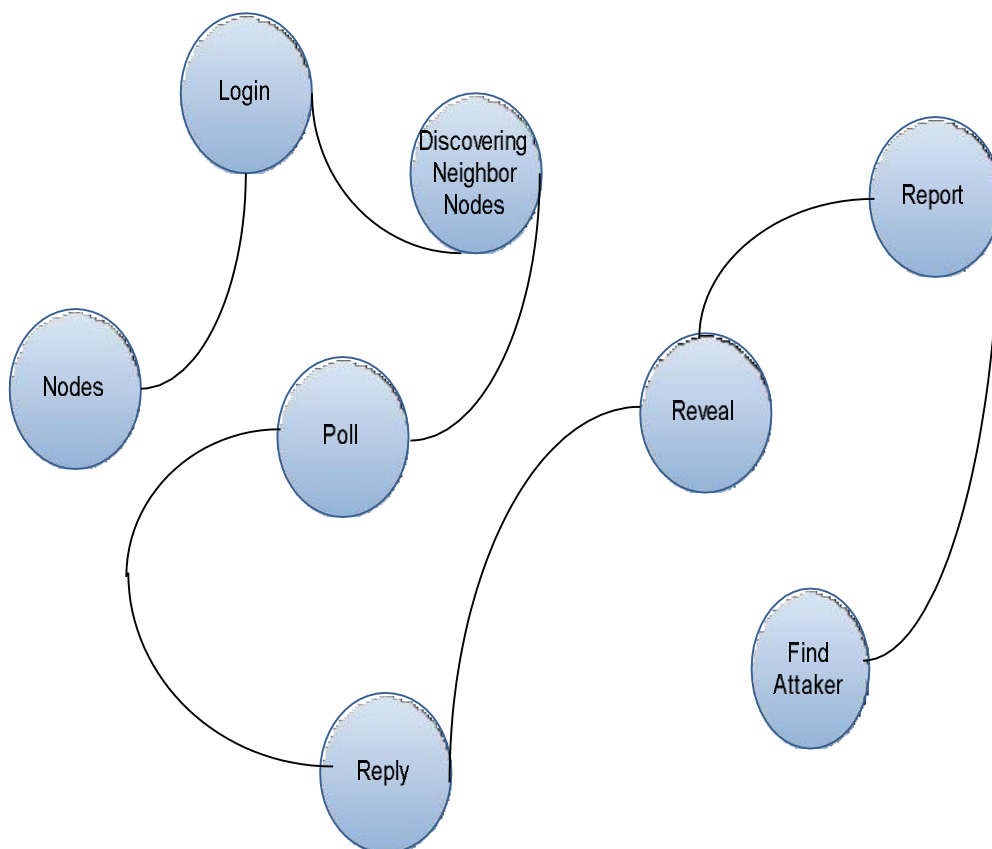


Fig : Data Flow Diagram

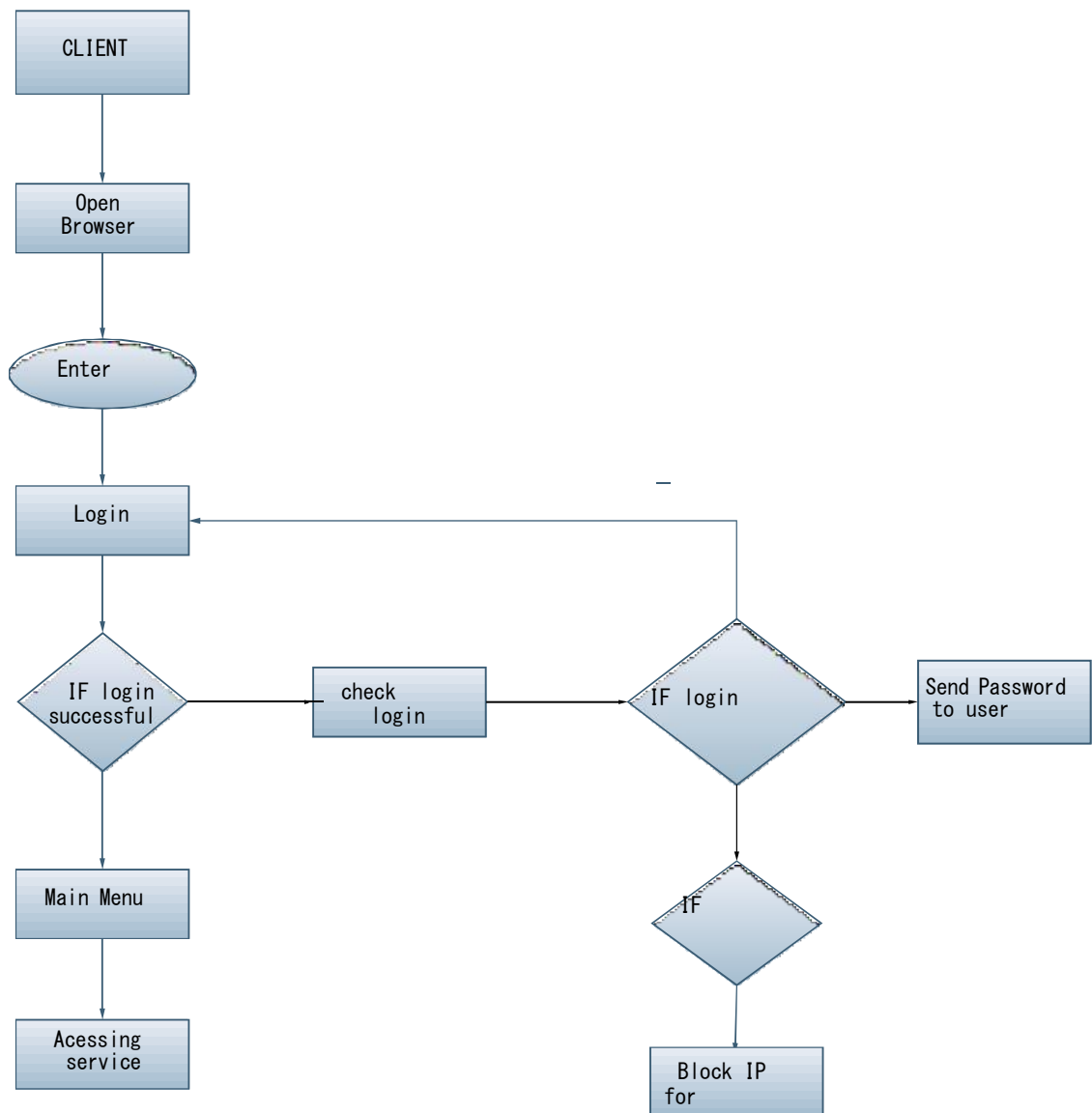
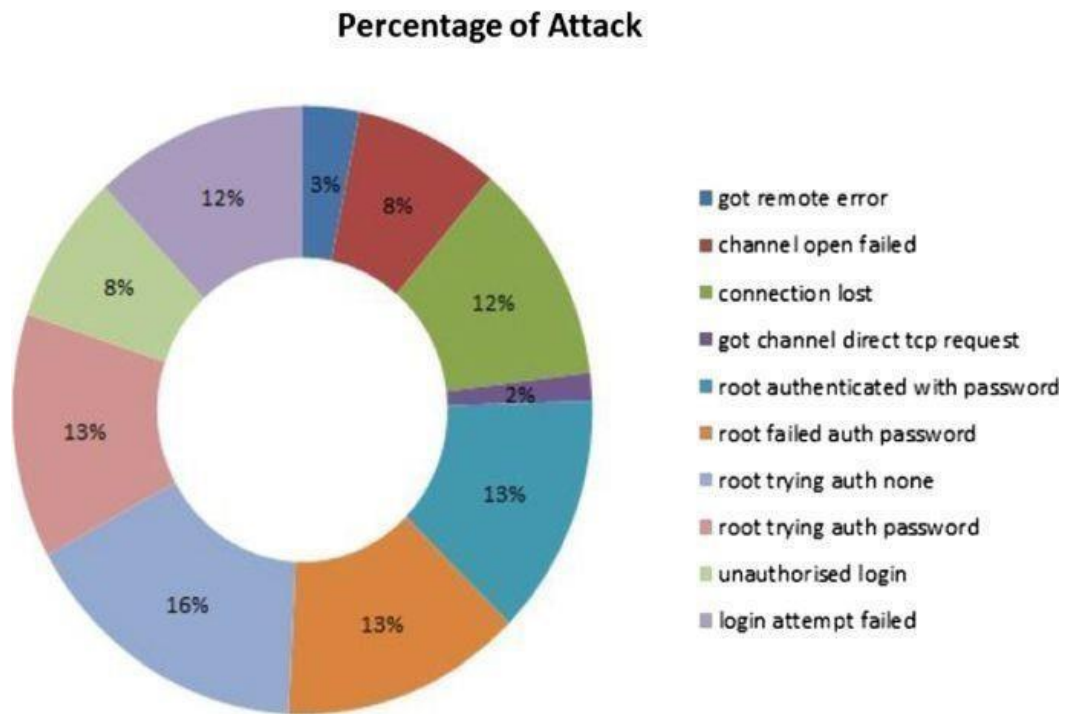
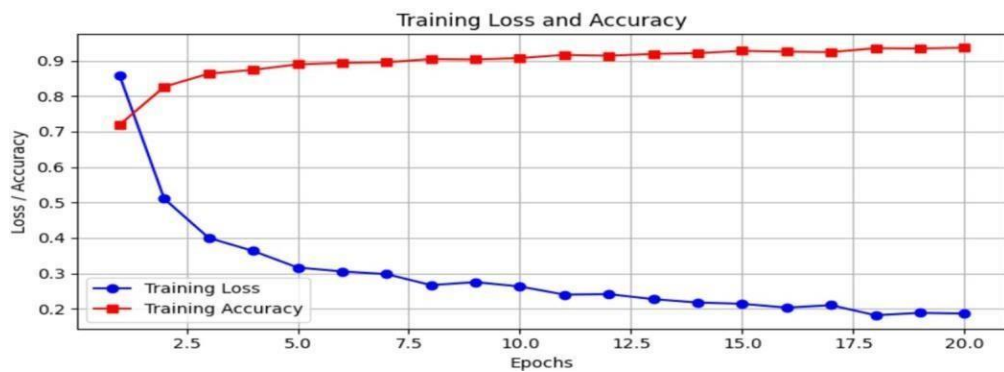


Fig : Represents the over all dat flow graph



**Fig:** Data Distribution for Train, Validate and Test Purposes after Augmentation (prevents Overfit).



**Fig:** Accuracy and Loss during Training Phase.

# **TESTING**

## **7.1 INTRODUCTION TO TESTING**

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

### **7.1.1 TESTING STRATEGIES**

In the context of advanced techniques in rule creation for threat detection project, here are some testing strategies that can be employed to ensure the accuracy, reliability, and effectiveness of the system.

## **UNIT TESTING**

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

## **INTEGRATION TESTING**

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

## **FUNCTIONAL TESTING**

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

Functional testing is centered on the following items:

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

Invalid Input : identified classes of invalid input must be rejected.  
Functions : identified functions must be exercised.  
Output : identified classes of application outputs must be exercised.  
Systems/Procedures : interfacing systems or procedures must be invoked.

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

## **SYSTEM TESTING**

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

## **WHITE BOX TESTING**

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

## **BLACK BOX TESTING**

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



## 7.1.1 TEST CASES:

### Testing Unit: Home Page

Result is shown below



Fig n- 7.1 Home Page

**Test case T01:** Writing test cases for home Page for getting access to all the specified pages.

<b>TID:</b> 01	<b>Priority:</b> High
<b>Test Unit:</b> Home Page	
<b>Test Description:</b> For getting access to all the specified Pages as well getting know the description of web application	
<b>Test Data:</b> No Data	
<b>Actions:</b> Click on Login Provide	
<b>Expected Result:</b> Showcase the information	<b>Output:</b> Showcase the information
<b>Note:</b> Executed Successfully	

Table-7.1: Writing test cases for homepage

## Testing Unit: About Admin Page

Result is shown below

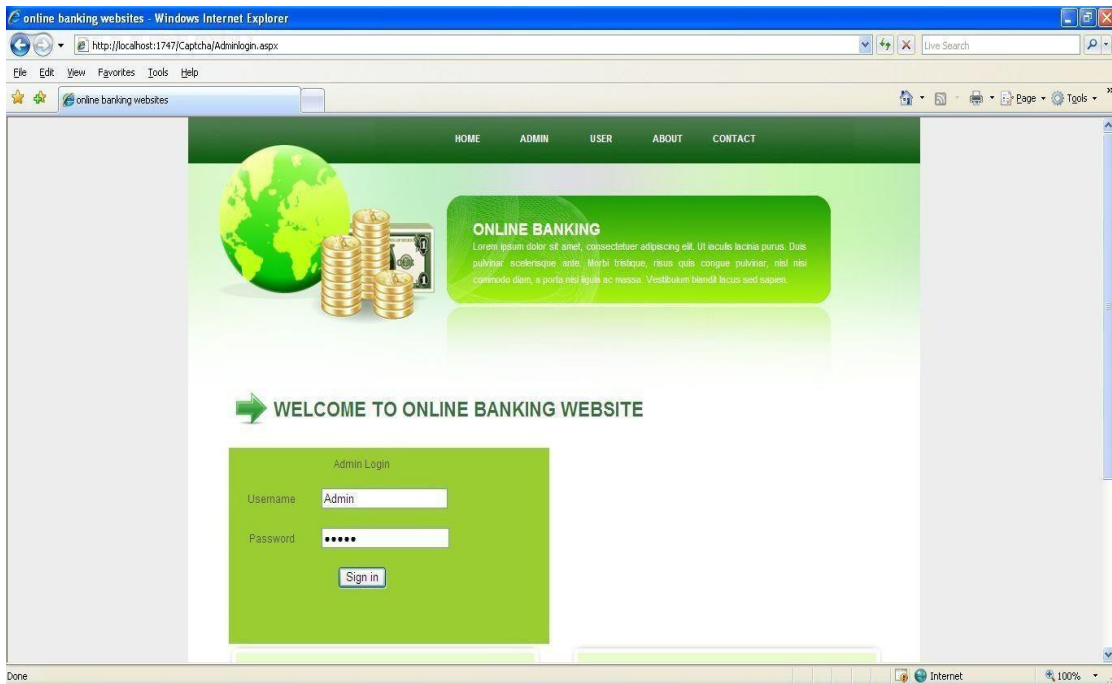


Fig -7.2: Result of testcase -1

### Test case T02: Writing test cases for About Admin Page

<b>TID:</b> 02	<b>Priority:</b> High
<b>Test Unit:</b> About Page	
<b>Test Description:</b> To know the Algorithm and Techniques	
<b>Test Data:</b> No Data	
<b>Actions:</b> Click on Sign in bar	
<b>Expected Result:</b> Know and used algorithm and techniques	<b>Output:</b> Know and used algorithm and techniques
<b>Note:</b> Executed Successfully	

Table-7.2: Writing test cases for About Admin Page

### Testing Unit: account creation page

Result is shown below

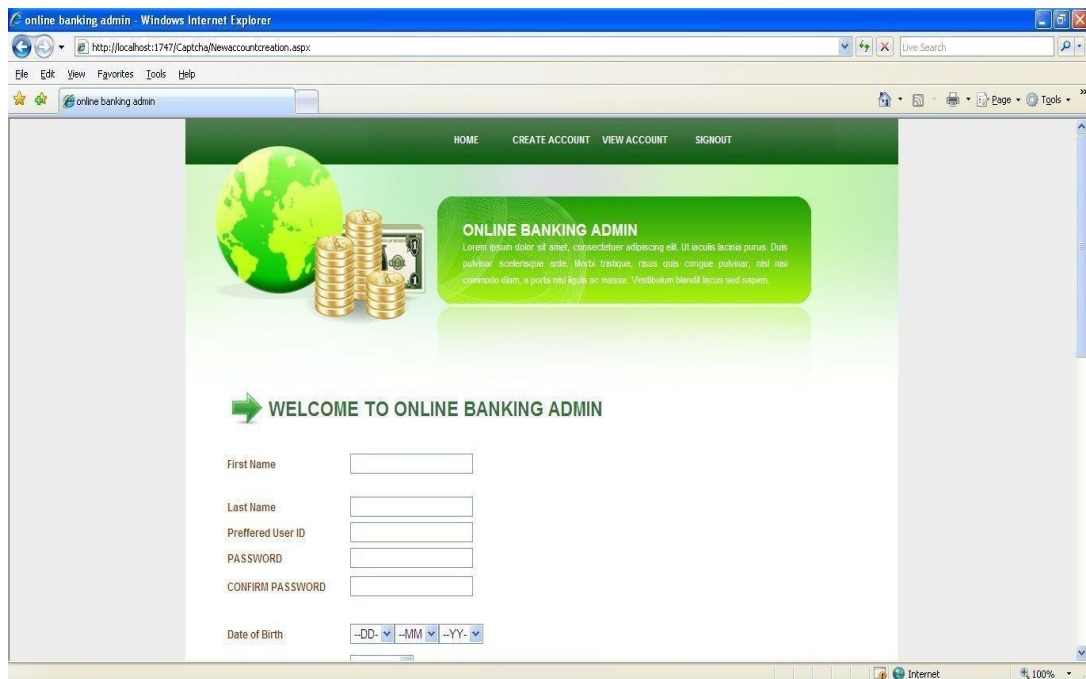


Fig no-7.3: Result of testcase -2

### Test case T03: Writing test cases for account creation Page

<b>TID:</b> 03	<b>Priority:</b> High
<b>Test Unit:</b> account creation Page	
<b>Test Description:</b> To get the contact details of developer	
<b>Test Data:</b> No Data	
<b>Actions:</b> Click on submitbar	
<b>Expected Result:</b> Contact Information(Mobile NO, Email, Linkeidn, Github)	<b>Output:</b> Contact Information (Mobile NO, Email, Linkeidn, Github)
<b>Note:</b> Executed Successfully	

Table-7.3: Writing test cases for contact page

## Testing Unit: User Login Page

Result is shown below

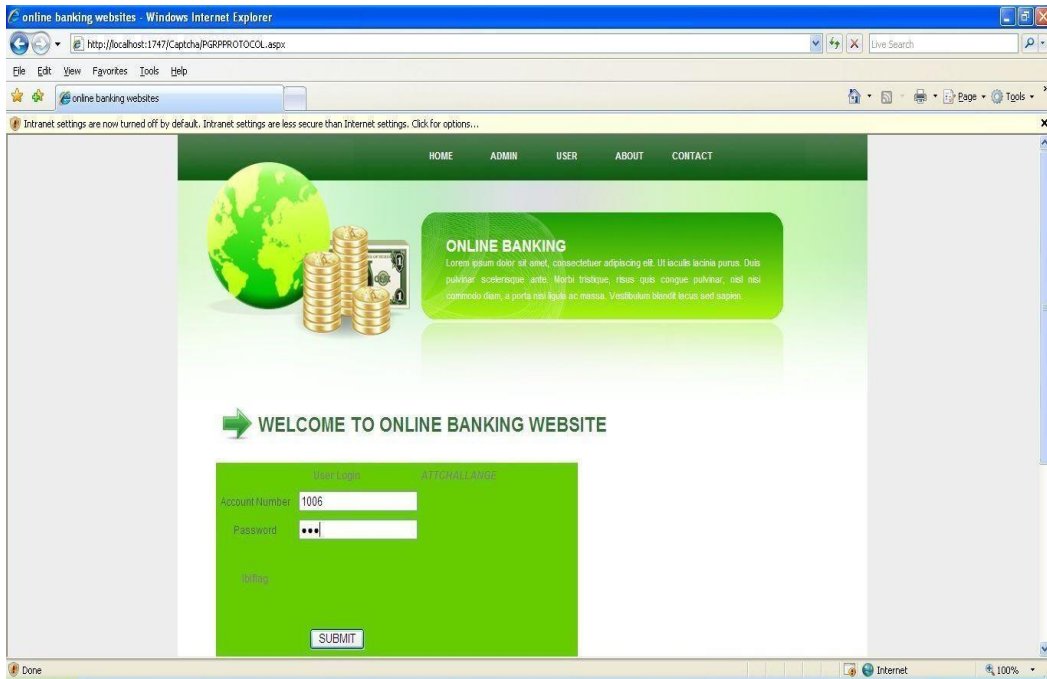


Fig-7.4: Result of testcase -3

### Test case T04: User Login Page

<b>TID:</b> 04	<b>Priority:</b> High
<b>Test Unit:</b> User Login Page	
<b>Test Description:</b> User Login Page	
<b>Test Data:</b> number and password	
<b>Actions:</b> Click on Submit	
<b>Expected Result:</b> Shows the application	<b>Output:</b> Shows the application
<b>Note:</b> Executed Successfully	

Table-7.4: Writing test cases for browsing an image

## Testing Unit: Transaction Page

Result is shown below

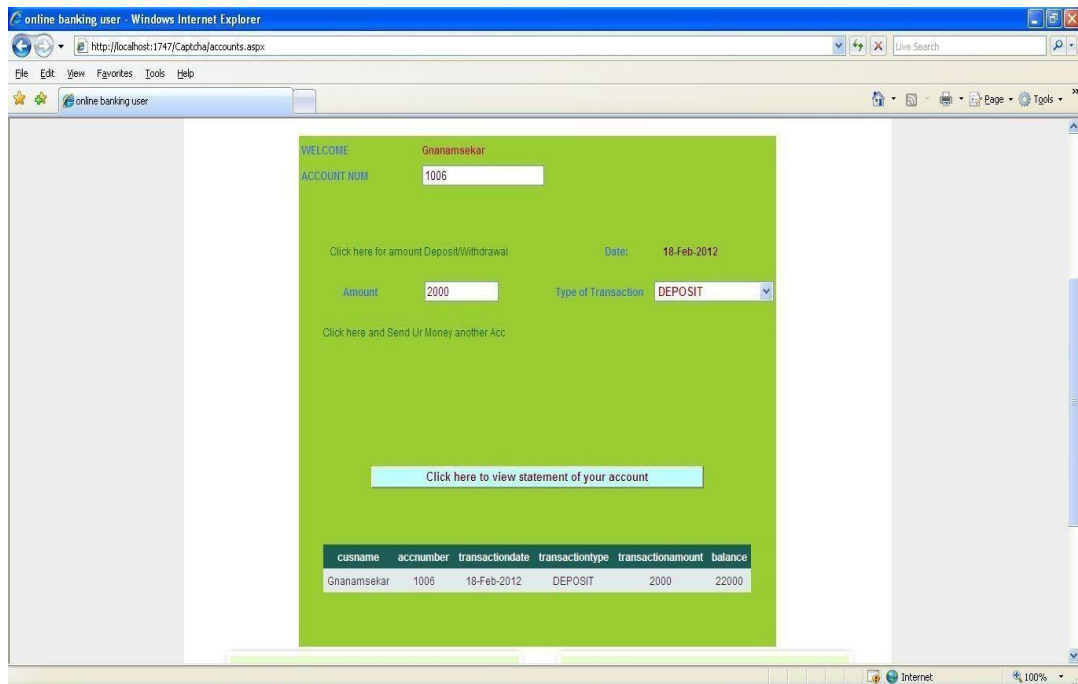


Fig -7.6: Result of testcase -4

### Test case T05: Transaction Page

<b>TID:</b> 05	<b>Priority:</b> High
<b>Test Unit:</b> Transaction Page	
<b>Test Description:</b> To check whether the amount transferred or not	
<b>Test Data:</b> Transaction amount	
<b>Actions:</b> Click on Deposit Button	
<b>Expected Result:</b> amount transferred	<b>Output:</b> amount transferred
<b>Note:</b> Executed Successfully	

Table-7.5: Writing test cases for transaction

## Testing Unit: Bank Statement

Result is shown below



Fig -7.7: Result of testcase -5

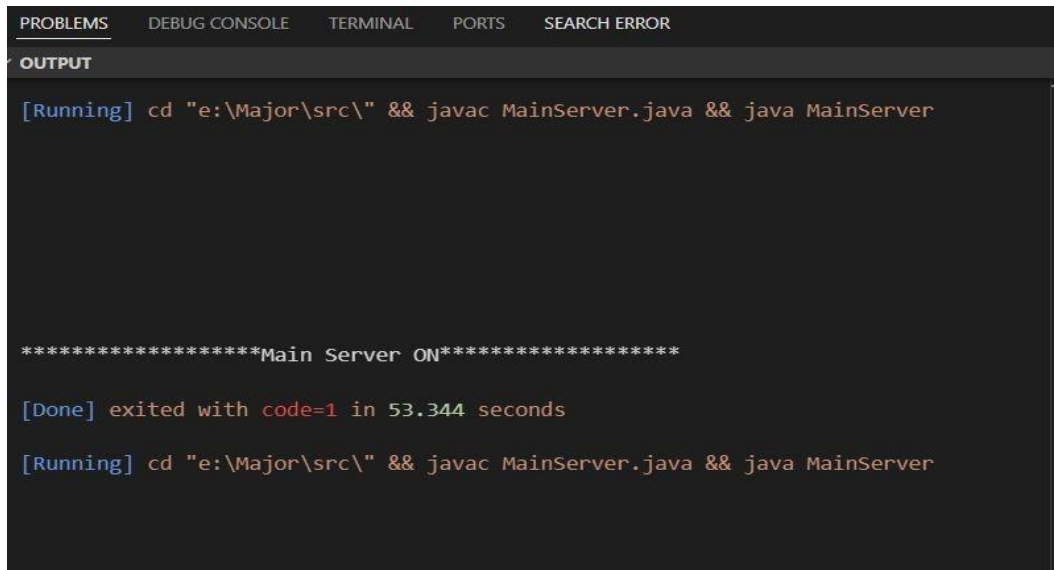
### Test case T06: Bank Statement

<b>TID: 06</b>	<b>Priority: High</b>
<b>Test Unit:</b> Bank Statement	
<b>Test Description:</b> To check whether the money deposited or not	
<b>Test Data:</b> amount	
<b>Actions:</b> Click on Get Button	
<b>Expected Result:</b> No file selected	<b>Output:</b> No file selected
<b>Note:</b> Executed Successfully	

Table-7.6: Writing test cases for no file selection.

# RESULTS

## 8.1 SCREENS AND REPORTS:



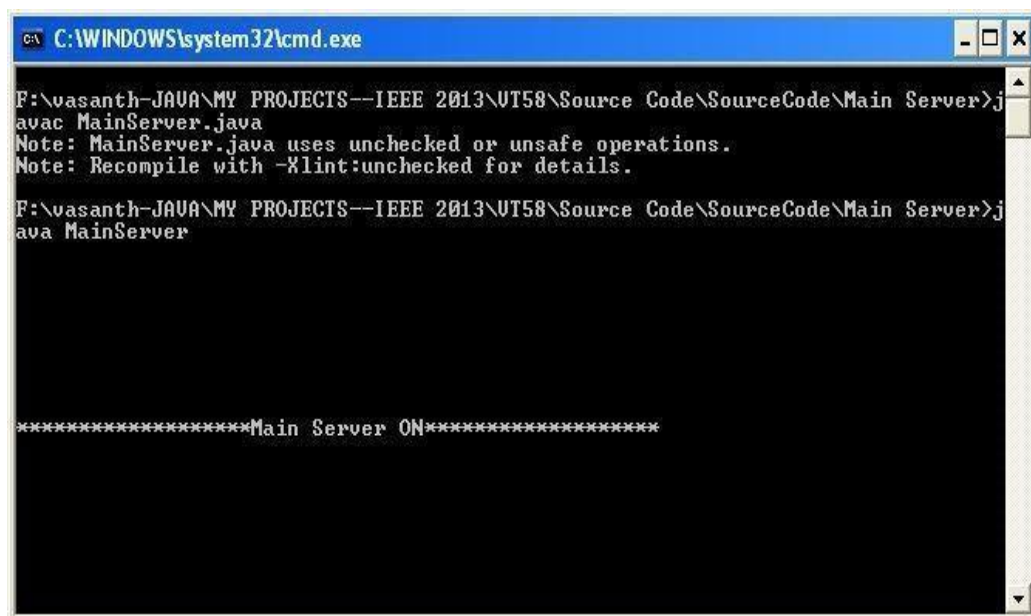
The screenshot shows an IDE's Output window with tabs for PROBLEMS, DEBUG CONSOLE, TERMINAL, PORTS, and SEARCH ERROR. The OUTPUT tab is active, displaying the following text:

```
[Running] cd "e:\Major\src\" && javac MainServer.java && java MainServer

*****Main Server ON*****

[Done] exited with code=1 in 53.344 seconds

[Running] cd "e:\Major\src\" && javac MainServer.java && java MainServer
```



The screenshot shows a Windows Command Prompt window titled "C:\WINDOWS\system32\cmd.exe". The command prompt shows the following text:

```
F:\vasanth-JAVA\MY PROJECTS--IEEE 2013\UT58\Source Code\SourceCode\Main Server>j
avac MainServer.java
Note: MainServer.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

F:\vasanth-JAVA\MY PROJECTS--IEEE 2013\UT58\Source Code\SourceCode\Main Server>j
ava MainServer

*****Main Server ON*****
```

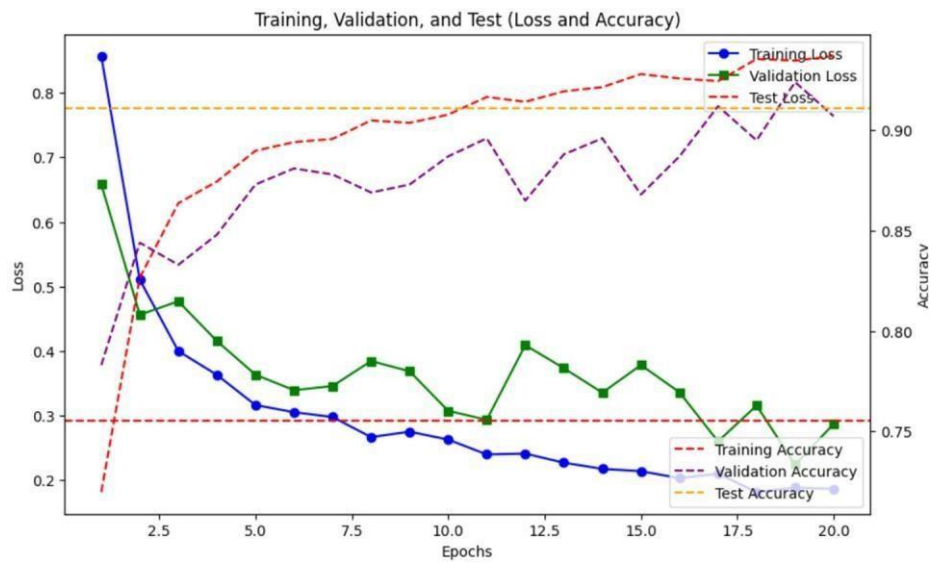


Fig-8.1.6: Accuracy and Loss Comparison for Training, Validation and Testing.

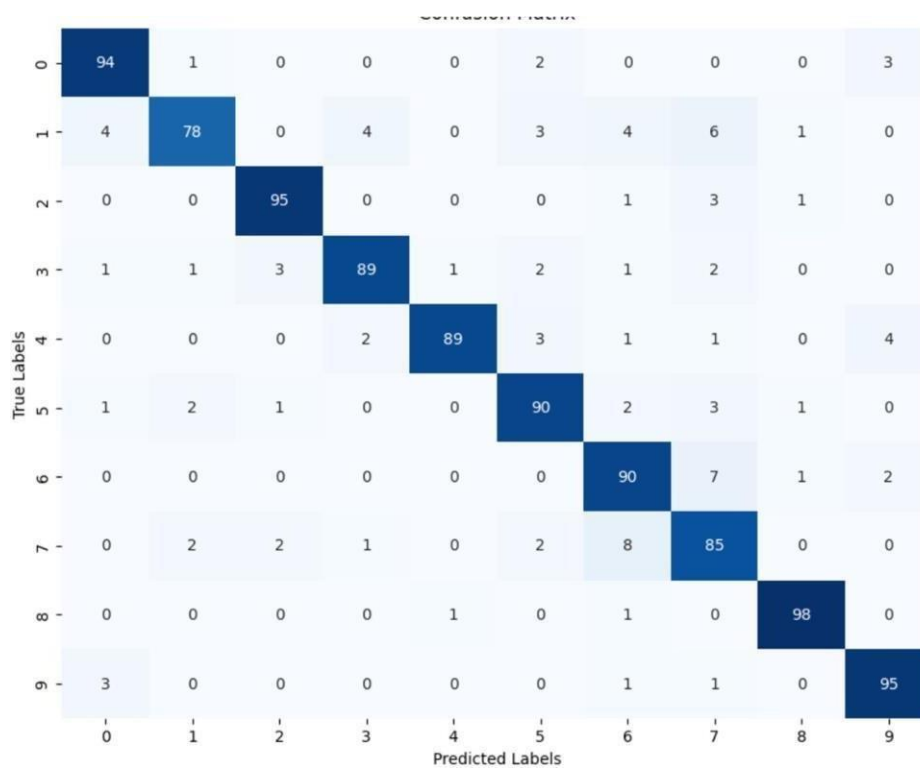


Fig-8.1.7: Confusion Matrix



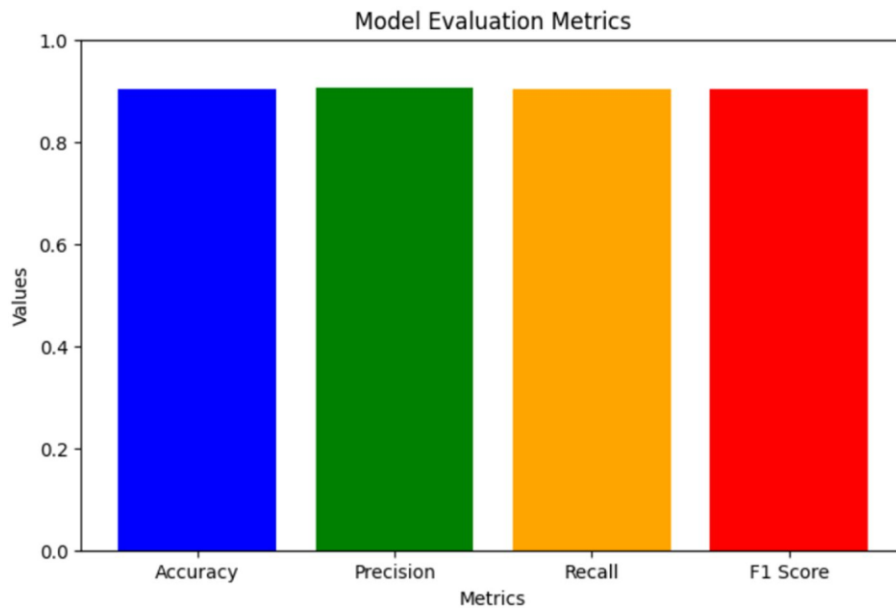


Fig-8.1.8: Model Evaluation Metrics During Testing Phase

## 8.2 USER MANUAL

### Advanced Techniques In Rule Creation For Threat Detection

This manual will guide you through the usage of the system, providing instructions on how to effectively leverage its capabilities for threat detection in your organization's cybersecurity operations.

#### 1. Introduction:

The Advanced Techniques in Rule Creation for Threat Detection system is designed to enhance your organization's cybersecurity posture by leveraging advanced techniques such as machine learning, anomaly detection, and behavioral analysis to create effective threat detection rules.

#### 2. System Overview:

The system operates by collecting and preprocessing data from various sources, creating detection rules using advanced techniques, and performing real-time detection to identify and mitigate security threats proactively.

#### 3. Getting Started:

To begin using the system, ensure that you have the necessary permissions and access credentials provided by your organization's IT department. You will also need a compatible web browser to access the system's user interface.

## **4. User Interface:**

The user interface provides intuitive controls and navigation options to interact with the system's features. Familiarize yourself with the layout and functionality of the user interface to effectively utilize the system.

## **5. Functionality:**

**Data Collection:** Collect data from diverse sources such as network traffic, system logs, and security sensors.

**Preprocessing:** Preprocess raw data by cleaning, filtering, and transforming it into a suitable format for analysis.

**Rule Creation:** Create detection rules using advanced techniques such as machine learning, anomaly detection, and behavioral analysis.

**Real-Time Detection:** Perform real-time threat detection by applying detection rules to incoming data streams.

**Alerting and Response:** Trigger alerts and response actions upon detecting security threats, notifying cybersecurity analysts or activating automated response mechanisms.

## **6. Best Practices:**

Regularly update detection rules and models to adapt to evolving threats.

Monitor system performance and adjust configurations as needed to optimize detection efficacy.

Collaborate with cybersecurity experts and share insights to enhance threat detection capabilities.

## **7. Troubleshooting:**

If you encounter any issues or errors while using the system, refer to the troubleshooting section of this manual for guidance on resolving common issues.

## **8. FAQs:**

Browse the frequently asked questions section for answers to common queries about the system's usage, features, and functionality.

## **9. Support and Feedback:**

For further assistance or feedback regarding the system, contact the IT support team or provide feedback through the designated channels.

## LIMITATIONS

Advanced techniques for rule creation in cybersecurity threat detection offer powerful capabilities but also come with several limitations:

**Complexity:** Advanced techniques often involve complex algorithms and models, making them difficult to understand and manage, especially for less experienced analysts. This complexity can lead to errors in rule creation or misinterpretation of results.

**High False Positive Rates:** Sophisticated rules may detect a wider range of behaviors, leading to an increase in false positives. This can overwhelm security teams with irrelevant alerts, leading to alert fatigue and potentially missing real threats.

**Resource Intensive:** Implementing and maintaining advanced rule creation techniques often requires significant computational resources and expertise. Small organizations or those with limited budgets may struggle to deploy and manage these techniques effectively.

**Dependency on Quality Data:** Advanced techniques often rely on high-quality, labeled training data to learn patterns and anomalies effectively. If the data used for training is incomplete, biased, or not representative of real-world threats, the effectiveness of the rules may be compromised.

**Dynamic Threat Landscape:** Advanced techniques may struggle to keep up with rapidly evolving threats and attack techniques. Rules based on historical data or static models may not effectively detect emerging threats or zero-day attacks.

**Overfitting:** Advanced models may become too specialized on the training data, resulting in overfitting. This means they perform well on the training data but fail to generalize to new, unseen data, leading to poor detection performance in real-world scenarios.

**Interpretability:** Some advanced techniques, such as machine learning models, lack interpretability. This makes it challenging for analysts to understand why a particular alert was triggered, hindering the investigation and response process.

## CONCLUSION

In conclusion, the Advanced Techniques in Rule Creation for Threat Detection project leverages cutting-edge methodologies to fortify cybersecurity defenses against evolving threats. Drawing parallels with the highlights of the PGRP protocol, which prioritizes proactive threat mitigation and robust risk management, this project similarly emphasizes preemptive detection measures and comprehensive response strategies. By amalgamating advanced rule creation techniques with real-time monitoring capabilities, the system fosters a dynamic defense ecosystem, akin to the adaptive nature of the PGRP protocol. As organizations navigate the intricate landscape of cyber threats, this project serves as a beacon of resilience, empowering stakeholders to confront challenges head-on and safeguard critical assets with agility and precision.

Online password guessing attacks on password-only systems have been observed for decades. Present-day attackers targeting such systems are empowered by having control of thousand to million node botnets. In previous ATT-based login protocols, there exists a security-usability trade-off with respect to the number of free failed login attempts (i.e., with no ATTs) versus user login convenience (e.g., less ATTs and other requirements). In contrast, PGRP is more restrictive against brute force and dictionary attacks while safely allowing a large number of free failed attempts for legitimate users. Our empirical experiments on two datasets (of one-year duration) gathered from operational network environments show that while PGRP is apparently more effective in preventing password guessing attacks (without answering ATT challenges), it also offers more convenient login experience, e.g., fewer ATT challenges for legitimate users even if no cookies are available. However, we reiterate that no user-testing of PGRP has been conducted so far. PGRP appears suitable for organizations of both small and large number of user accounts. The required system resources (e.g., memoryspace) are linearly proportional to the number of users in a system. PGRP can also be used with remote login services where cookies are not applicable.

## FUTURE WORK

Looking ahead, several avenues for future work can further enhance the impact and capabilities of our project:

**Enhanced Rule Creation Techniques:** Continuously explore and integrate emerging technologies such as deep learning, natural language processing, and graph analytics to enhance the sophistication and effectiveness of rule creation for threat detection.

**Integration with Threat Intelligence Platforms:** Develop mechanisms to integrate the system with external threat intelligence feeds and platforms, enabling automatic updates of detection rules based on real-time threat intelligence data.

**Behavioral Analysis and User Entity Behavior Analytics (UEBA):** Extend the capabilities of the system to include advanced behavioral analysis techniques and UEBA to detect anomalous user behavior and insider threats more effectively.

**Automated Response and Orchestration:** Implement automated response and orchestration capabilities to enable the system to automatically mitigate detected threats through predefined response actions, reducing manual intervention and response time.

**Scalability and Performance Optimization:** Continuously optimize the system's architecture and algorithms to ensure scalability and performance, allowing it to handle increasing data volumes and processing requirements efficiently.

**Compliance and Regulatory Requirements:** Enhance the system's capabilities to facilitate compliance with industry-specific regulations and data protection laws, such as GDPR, HIPAA, and PCI DSS, by incorporating compliance checks and reporting functionalities.

**User Interface and Experience (UI/UX) Enhancements:** Focus on improving the user interface and experience of the system to make it more intuitive, user-friendly, and accessible, enabling cybersecurity analysts to interact with the system more effectively.

**Collaborative Threat Intelligence Sharing:** Implement mechanisms for collaborative threat intelligence sharing among organizations, enabling the exchange of threat indicators, detection rules, and insights to enhance collective cybersecurity defenses.

**Adaptive Learning and Self-Learning Capabilities:** Develop adaptive learning algorithms and self-learning capabilities within the system to enable it to

continuously evolve and adapt to emerging threats and changing attack patterns autonomously.

**Real-Time Visualization and Reporting:** Enhance real-time visualization and reporting capabilities to provide cybersecurity stakeholders with actionable insights and situational awareness, facilitating informed decision-making and response planning.

By focusing on these areas of future development, the project can remain at the forefront of cybersecurity innovation, continually evolving to address emerging threats and challenges effectively.

## REFERENCES

1. Tziampazis, C. (2021). Open Security Intelligence, analysis and countermeasures (Master's thesis, Universitat Politècnica de Catalunya).
2. Sonawane, H. S., Deshmukh, S., Joy, V., & Hadsul, D. (2022, June). Torsion: Web Reconnaissance using Open Source Intelligence. In 2022 2nd International Conference on Intelligent Technologies (CONIT) (pp. 1-4). IEEE.
3. Sharmila, C., Gopalakrishnan, J., Shanmuga Prasath, P., & Daniel, Y. (2022, May). Multipurpose Linux Tool for Wi-Fi Based Attack, Information Gathering and Web Vulnerability Scanning Automations. In International Conference on Image Processing and Capsule Networks (pp. 587-598). Cham: Springer International Publishing.
4. Zoder, B. O. (2020). Automated Collection of Open Source Intelligence. Pro gradu-tutkielma, Masaryk University.
5. Bratulescu, R. A., Vatasoiu, R. I., Mitroi, S. A., Suci, G., Sachian, M. A., Dutu, D. M., & Calescu, S. E. (2022, August). Fraudulent Activities in the Cyber Realm: DEFRAUDify Project: Fraudulent Activities in the Cyber Realm: DEFRAUDify Project. In Proceedings of the 17th International Conference on Availability, Reliability and Security (pp. 1-5).
6. Wright, T., Whitfield, S., Cahill, S., & Duffy, J. (2020, December). Project umbra. In 2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM) (pp. 748-751). IEEE.
7. Rajamäki, J., & McMenamin, S. (2024, March). Utilization and Sharing of Cyber Threat Intelligence Produced by Open-Source Intelligence. In International Conference on CyberWarfare and Security (Vol. 19, No. 1, pp. 607-611).
8. Zouave, E., Bruce, M., Colde, K., Jaitner, M., Rodhe, I., & Gustafsson, T. (2020). Artificially intelligent cyberattacks. Stockholm: Totalförsvarets forskningsinstitut FOI [Online] Available: [https://www.statsvet.uu.se/digitalAssets/769/c\\_769530-1\\_3-k\\_rapport-foi-vt20.pdf](https://www.statsvet.uu.se/digitalAssets/769/c_769530-1_3-k_rapport-foi-vt20.pdf) [Accessed: Sep. 28, 2022].