AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY UNDER SECTION 3 OF UGC ACT, 1956

SCHOOL OF COMPUTING CHENNAI

Computer Science Engineering - Cyber Security

Topic: ZERO TRUST SECURITY FRAMEWORK FOR MICROSERVICE ARCHITECTURE DRIVEN WEB APPLICATIONS

Domain: Cybersecurity - DevOps

Members: Dinesh Kumar.N (ch.en.u4cys21014)
V. Jayaraj (ch.en.u4cys21026)

Supervisor: Dr. S. Udhayakumar
CSE- Cybersecurity

Problem Identification

- Traditional authentication and authorization mechanisms often rely on perimeter-based security, which fails to protect against internal threats and lateral movement within containerized environments.
- Increasing use of cloud services and remote work has expanded the attack surface, making it difficult to secure data and applications effectively.
- The complexity and scale of managing permissions and access controls in containerized and microservices architectures expose vulnerabilities.

Problem Statement

- The current reliance on perimeter-based security fails to provide the necessary protection for data and applications in the context of expanding cloud services and remote work, necessitating a shift to a Zero Trust Architecture.
- Zero Trust Architecture addresses these gaps by enforcing strict identity verification, continuous monitoring, and least-privilege access, significantly enhancing security in dynamic and distributed container deployments.

Project Significance

- Implementing Zero Trust Architecture (ZTA) in containerized environments is crucial for enhancing security.
- By implementing least-privilege access and continuous monitoring, organizations can better protect their sensitive data and applications from evolving threats.
- Containers are inherently ephemeral and dynamic, making traditional perimeter-based security ineffective.
- ZTA benefits organizations, DevOps teams, and IT security professionals by improving security posture, ensuring regulatory compliance, and enabling efficient, secure deployments, thus promoting safer and more resilient infrastructure.

Tech Stack Used

Frontend: React js -- TailwindCSS -- Redux

Big Data: Apache Spark -- PySpark -- mlib

Backend: Python -- flask -- Node js

DevOps:

- Docker
- Kubernetes - AWS EKS
- Helm Chart
- Git

Database: MongoDB /AWS DynamoDB

ZTA Feautures: JWT -- mTLS -- Falco -- R8

Android App: React Native -- Java

Objective

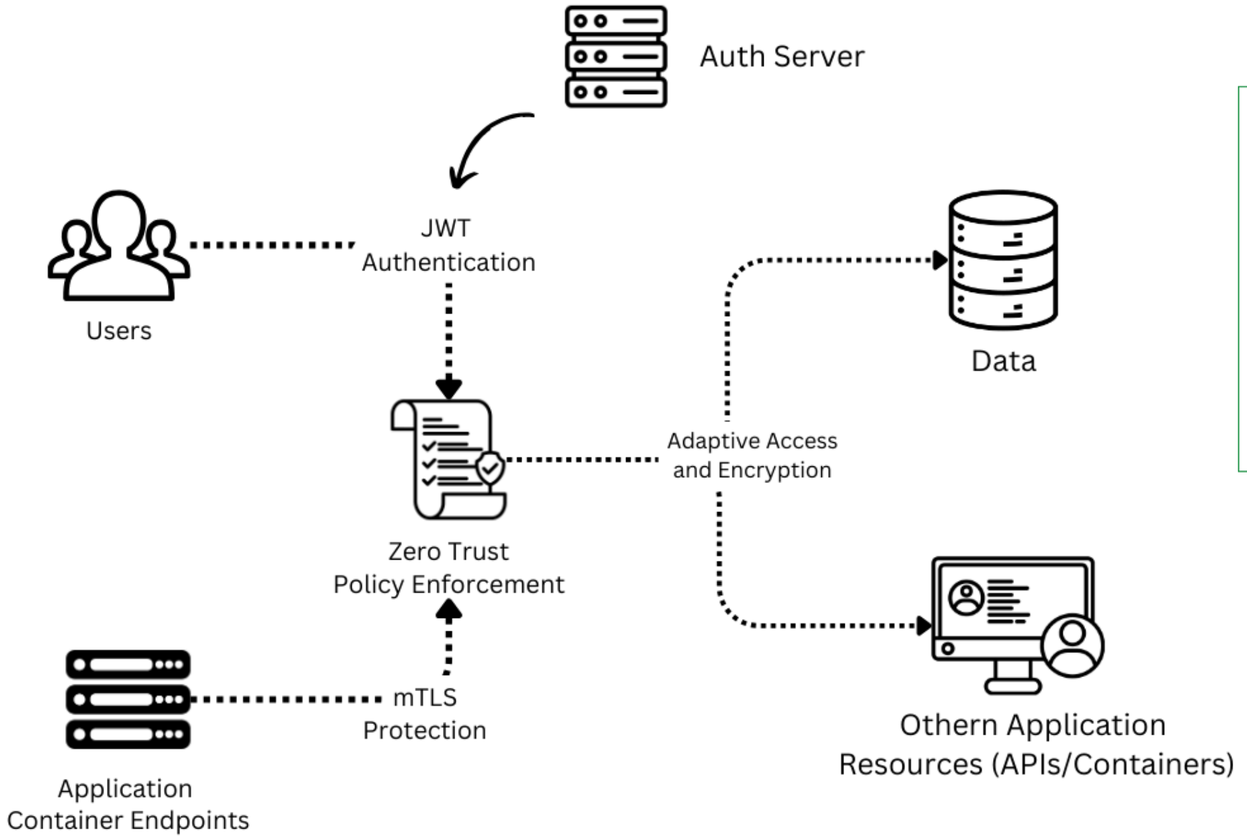
- Design and implement a Zero Trust Architecture that enforces strict identity verification and continuous monitoring, ensuring that every access request within containerized environments is authenticated and authorized.
- Develop mechanisms for least-privilege access, ensuring that users and applications have only the permissions necessary for their tasks, minimizing the risk of unauthorized actions.
- Integrate advanced security measures that protect against internal and external threats, ensuring a robust and resilient security posture in dynamic containerized environments.

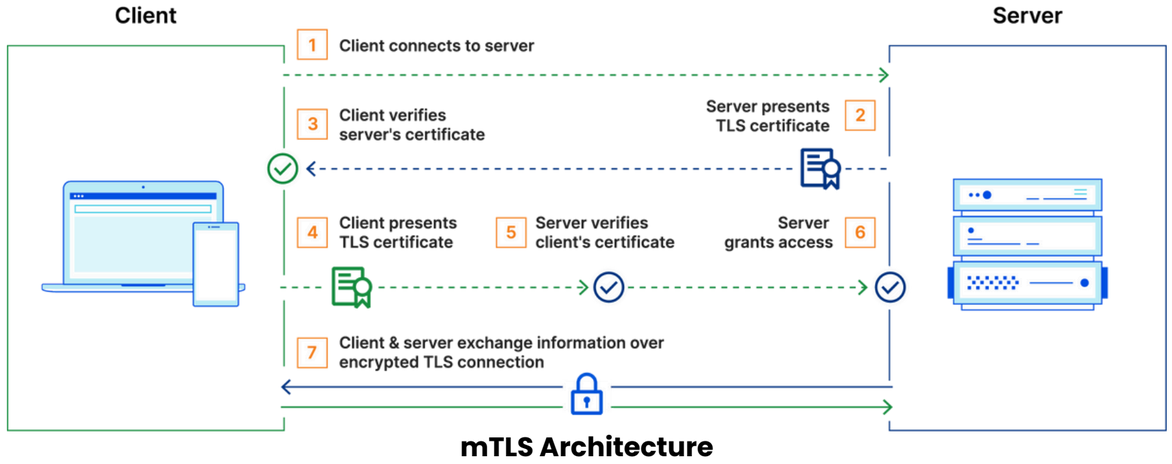
Scope

To Evaluate current security.
Develop a Zero Trust framework for containers, integrate continuous identity verification and access controls.
Develop a POC for ZTA in web and android applications, conduct testing, provide documentation.

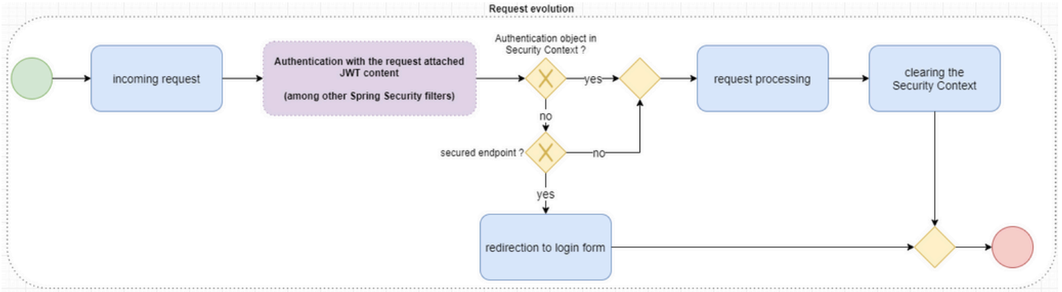
Github Repo: https://github.com/Dinesh-4320/Zero-Trust-Project

General ZTA Diagram



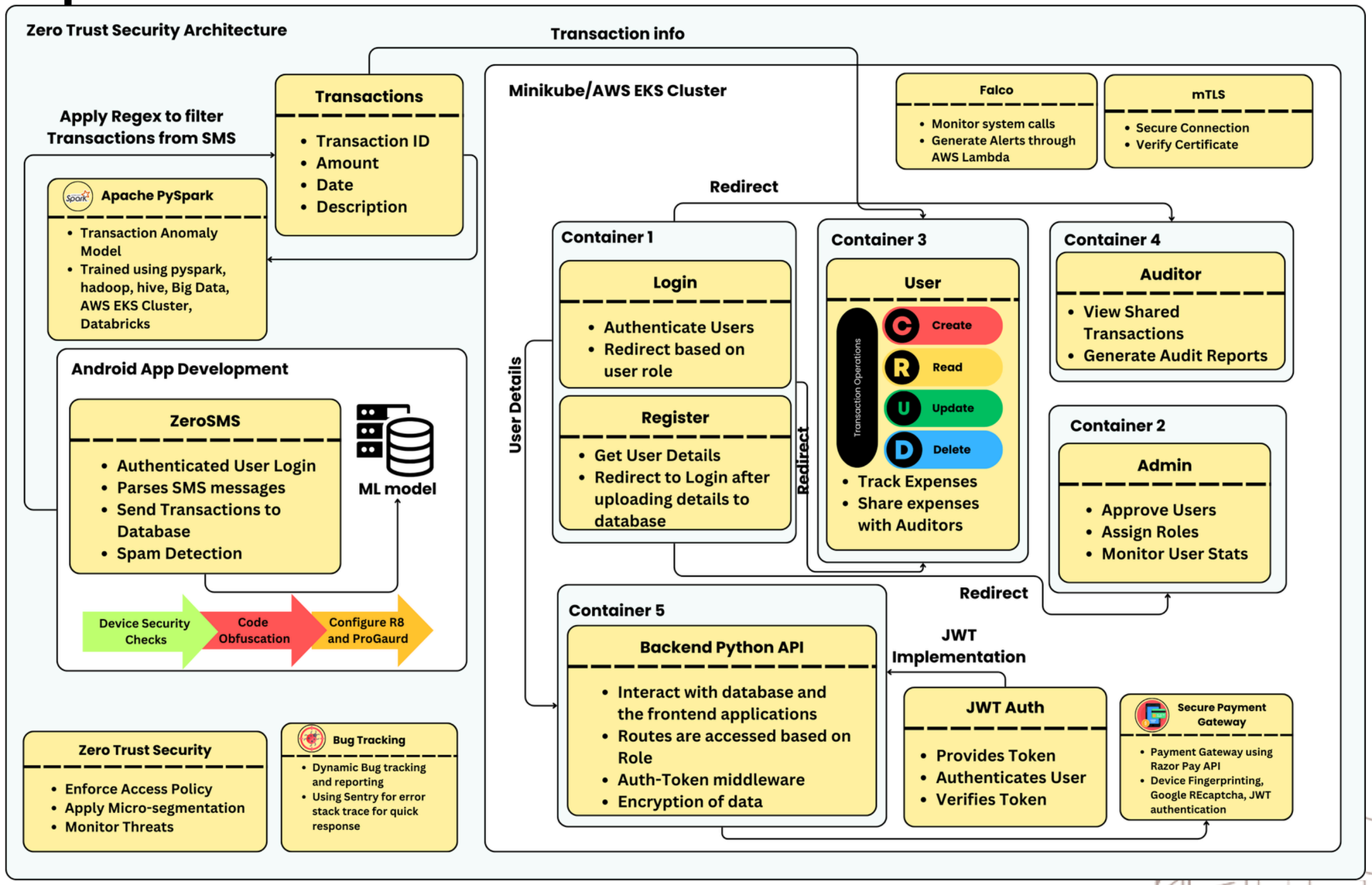


mTLS Architecture

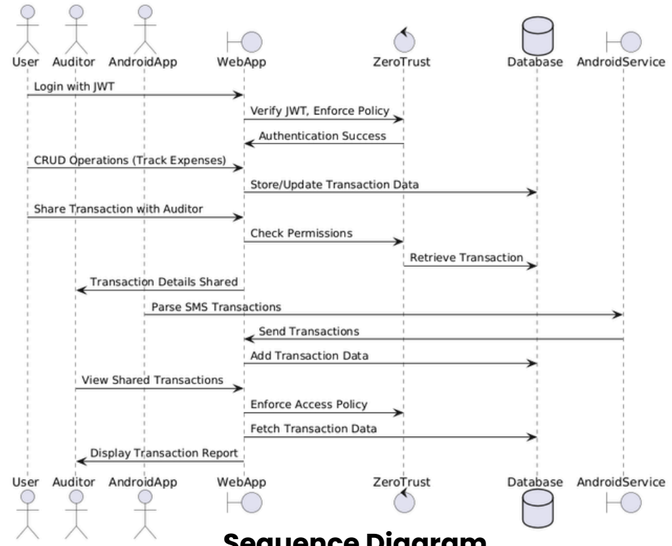


JWT life cycle

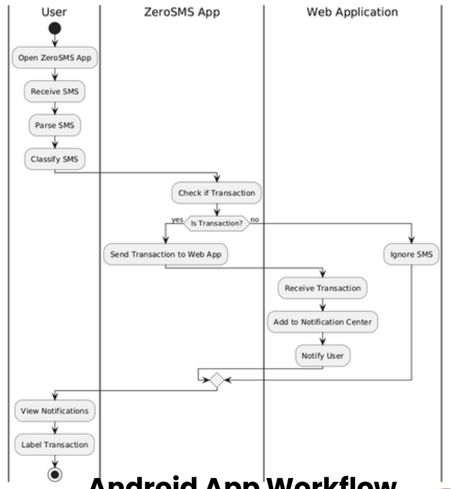
Implementation



Web and Android Application Architecture Diagram

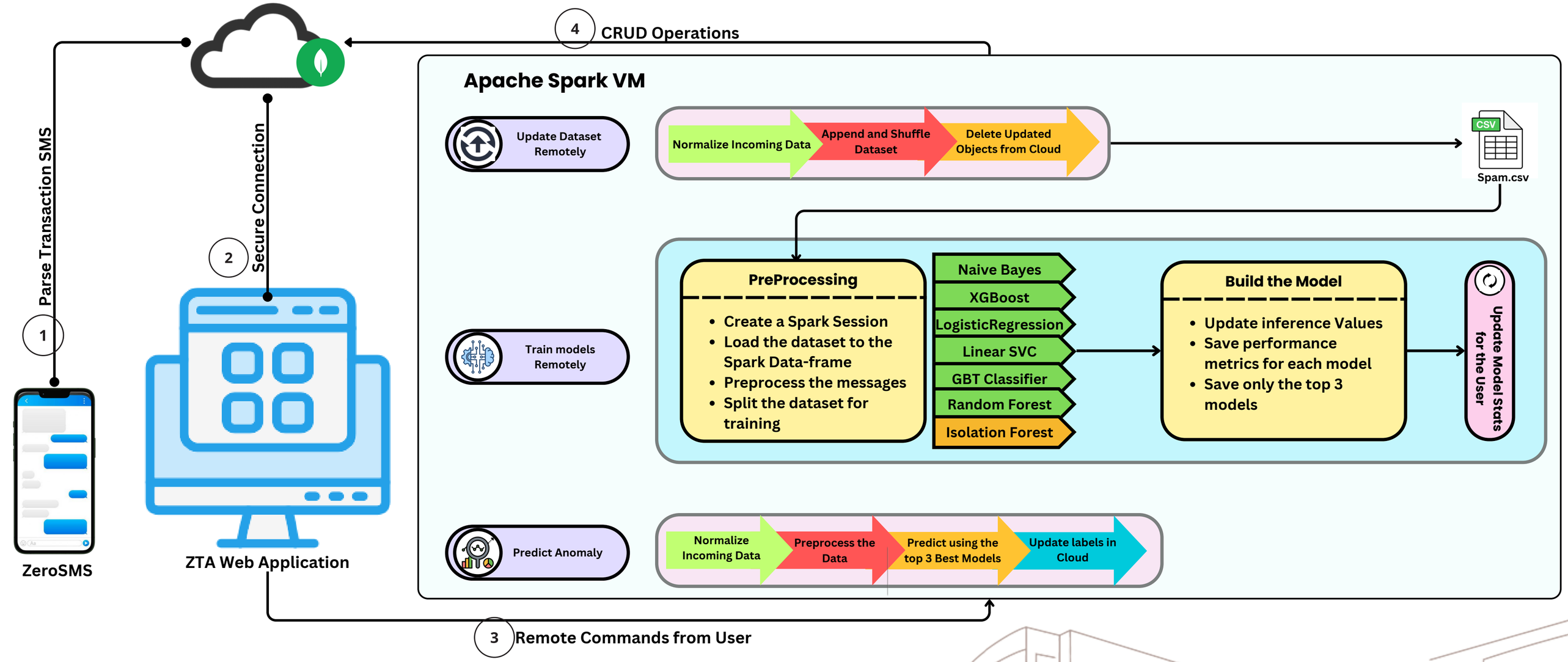


Sequence Diagram



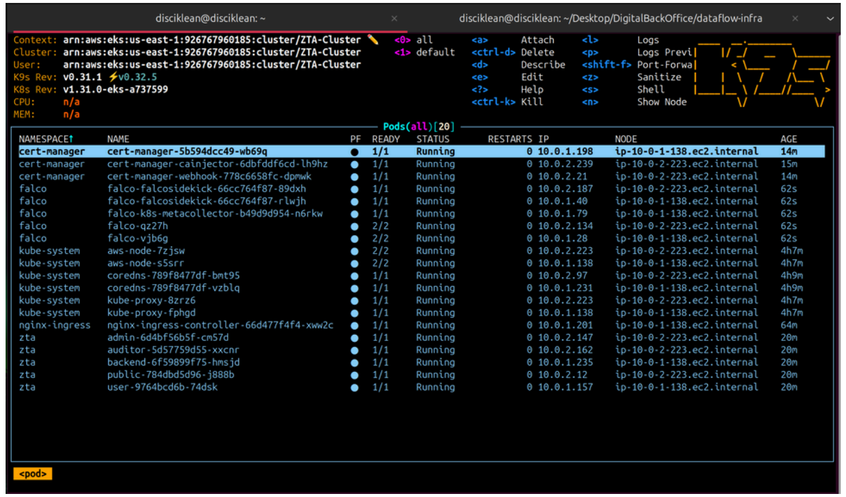
Android App Workflow

Zero Trust Security Framework for MIicroservice Architecture Driven Web Applications
N. Dinesh Kumar
V. Jayaraj

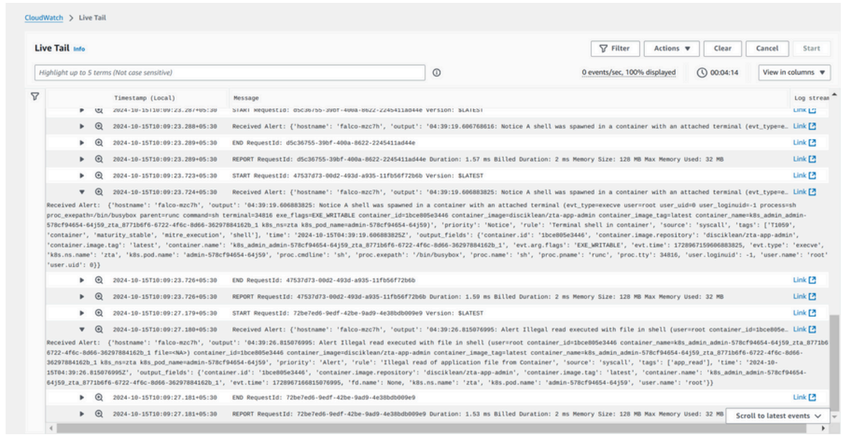

4
CRUD Operations
Apache Spark VM
Update Dataset Remotely
Normalize Incoming Data
Append and Shuffle Dataset
Delete Updated Objects from Cloud
CSV
Spam.csv
Parse Transaction SMS
1
2
Secure Connection
Train models Remotely
PreProcessing
Create a Spark Session
Load the dataset to the Spark Data-frame
Preprocess the messages
Split the dataset for training
Naive Bayes
XGBoost
LogisticRegression
Linear SVC
GBT Classifier
Random Forest
Isolation Forest
Build the Model
Update inference Values
Save performance metrics for each model
Save only the top 3 models
Update Model Stats for the User
Predict Anomaly
Normalize Incoming Data
Preprocess the Data
Predict using the top 3 Best Models
Update labels in Cloud
ZeroSMS
ZTA Web Application
3
Remote Commands from User

Results and Analysis

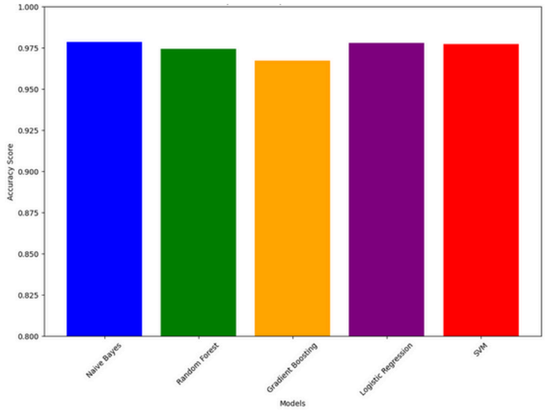
The web application, deployed on an EKS cluster with node groups, ensures scalability, fault tolerance, and seamless service communication for a smooth user experience. The user dashboard supports secure CRUD operations, while data sharing with auditors provides selective access. Auditors have read-only access to transactions, and the admin dashboard allows dynamic role assignment and access control.



Security is enhanced using AWS Lambda and Falco, which monitor the cluster for anomalies and trigger real-time alerts for threat detection. Session logs offer a comprehensive audit trail, tracking user activities and ensuring compliance with Zero Trust principles by continuously monitoring user actions.



Naive Bayes was the best model with an F1 score of 0.918 and accuracy of 0.978, showing a strong balance of precision and recall. SVM had the highest precision (1.0) and ROC AUC score (0.984), while Logistic Regression also performed well with an F1 score of 0.909. Random Forest and Gradient Boosting had slightly lower F1 scores but maintained high accuracy across all models.

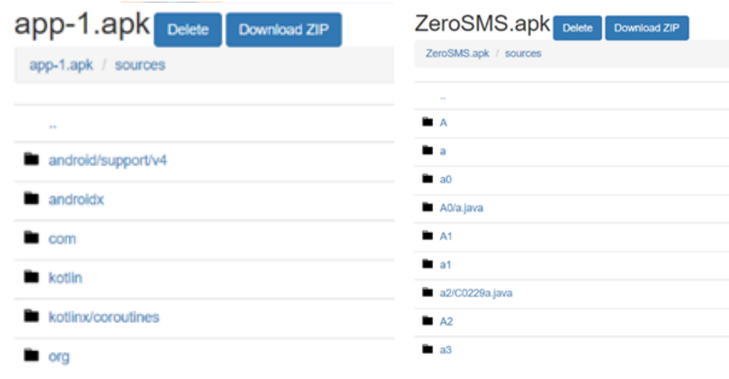


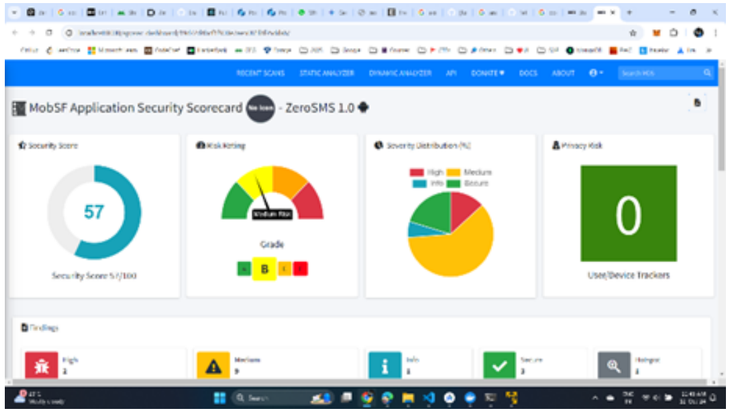
Results and Analysis

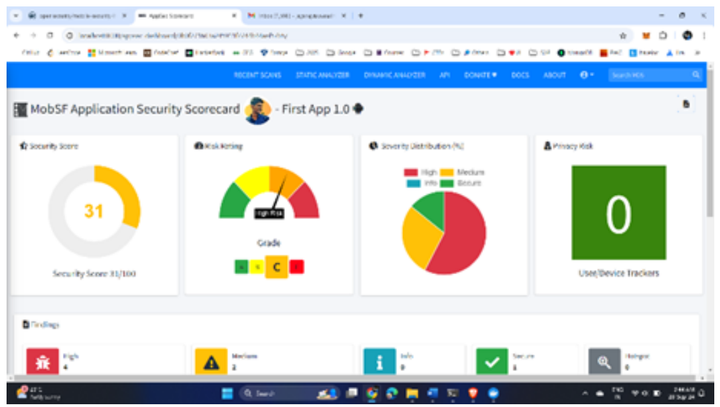
To prepare for production release, the app leverages the JailMonkey library to detect rooting and unsafe environments, while utilizing R8 and ProGuard tools for code optimization and obfuscation to enhance performance and security. The APK is securely signed with a private key from a trusted keystore, ensuring authorized distribution.

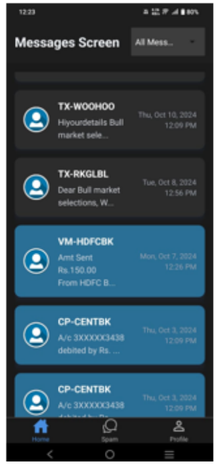
The below figures compares and tests ZeroSMS APK against standard APKs. Due to R8 and ProGuard obfuscation, reverse engineering ZeroSMS is significantly more challenging. A MobSF scan shows ZeroSMS achieving a security score of 57 (Grade B), 83.87% higher than a standard APK, which scored 31 (Grade C). ZeroSMS has SMS permissions and allows "http" traffic for backend connectivity, impacting its security score.

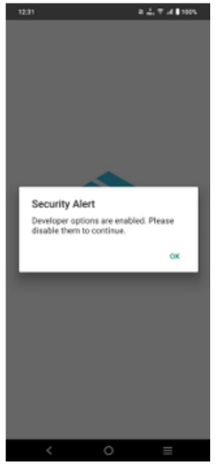
```
    signingConfigs {
        debug {
            storeFile file('debug.keystore')
            storePassword 'android'
            keyAlias 'androiddebugkey'
            keyPassword 'android'
        }
        release {
            if (project.hasProperty('MYAPP_UPLOAD_STORE_FILE')) {
                storeFile file(MYAPP_UPLOAD_STORE_FILE)
                storePassword MYAPP_UPLOAD_STORE_PASSWORD
                keyAlias MYAPP_UPLOAD_KEY_ALIAS
                keyPassword MYAPP_UPLOAD_KEY_PASSWORD
            }
        }
    }
    buildTypes {
        debug {
            signingConfig signingConfigs.debug
        }
        release {
            // Caution! In production, you need to generate your own keystore file.
            // see https://reactnative.dev/docs/signed-apk-android.
            signingConfig signingConfigs.release
            debuggable false
            shrinkResources true
            minifyEnabled true
            proguardFiles getDefaultProguardFile("proguard-android.txt"), "proguard-rules.pro"
        }
    }
}
```











ZERO TRUST SECURITY FRAMEWORK FOR MIICROSERVICE ARCHITECTURE DRIVEN WEB APPLICATIONS

JAYARAJ VISWANATHAN CH.EN.U4CYS21026

N. DINESH KUMAR CH.EN.U4CYS21014

MENTOR: Dr. S. Udhaya Kumar

AMRITA VISHWA VIDYAPEETHAM — SCHOOL OF COMPUTING CHENNAI

Tools and Technologies: AWS EKS, Kubernetes, Docker, Falco, React js, React Native, Android Studio, R8, ProGaurd, Apache Spark, Big Data

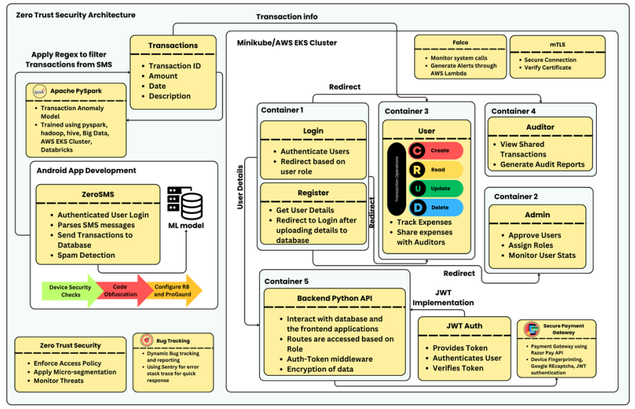
https://github.com/Dinesh-4320/Zero-Trust-Project

INTRODUCTION AND OBJECTIVE

In today's digital landscape, traditional security models fall short against modern cyber threats. As organizations move to cloud-native architectures, the attack surface grows, increasing vulnerability. The Zero Trust Security Model addresses these risks by enforcing continuous authentication, strict access control, and encryption. This project aims to implement a secure web application using Zero Trust principles, focusing on identity verification with JWT and mTLS, adaptive access control, encrypted communication, and real-time threat detection.

ALGORITHM TO IMPLEMENT ZTA

- Authenticate User
  - a. Validate Credentials
  - b. Assign Role
- Enforce Least Privilege
  - a. If authorized, grant minimal access
  - b. Else, deny access
- Request Access to Resource
  - a. Verify Token and Permissions
  - b. Log Access if valid, else deny
- Secure Containers
  - a. Initialize and Secure Kubernetes
  - b. Deploy Containers with mTLS
- Monitor and Detect Threats
  - a. Audit Activity
  - b. Detect Anomalies
- Encrypt Data at Rest and In Transit
- Implement Real-Time Threat Response
  - a. AI/ML for proactive threat detection
  - b. Automate incident response for detected anomalies



METHODOLOGY

The system integrates a mobile Android app, web application, and security module following Zero Trust principles. The Android app parses SMS data for transaction information, while the web application handles user management, transactions, and applies security policies.

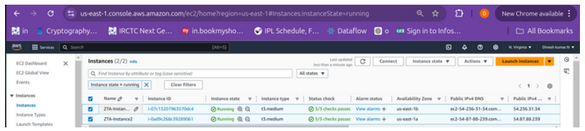
Security protocols include:

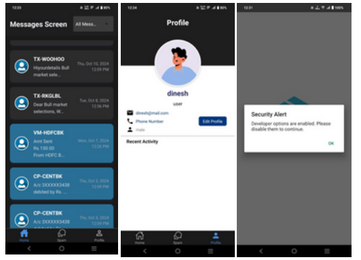
- JWT Authentication for secure user sessions.
- mTLS for encrypted communication between services.
- Policy Enforcement Point (PEP) for real-time access control.
- Falco for real-time threat monitoring via system calls.

Roles like Admin, Auditor, and User are defined to manage access and operations securely. Additional measures, such as code obfuscation and detection of rooted devices, further strengthen security.

RESULT ANALYSIS

The web application is designed to efficiently manage personal and business financial transactions by enabling users to manually input and track their expenses, savings, and investments in an organized manner. In addition to manual entry, the app offers seamless synchronization of transactions through SMS parsing using the integrated ZeroSMS app, ensuring that users can automatically log financial activities received via text messages. To help users better understand their financial status, the application provides detailed visual insights through interactive charts and graphs, allowing for easier monitoring of spending patterns, savings growth, and investment performance. The app also supports role-based access control (RBAC), granting specific permissions to administrators and auditors for oversight and management purposes, ensuring that users with different roles have appropriate access levels based on their responsibilities. Built on a robust Zero Trust security framework, the application ensures that all access points are secure, utilizing JWT (JSON Web Token) authentication to verify the identity of each user, reducing the risk of unauthorized access.

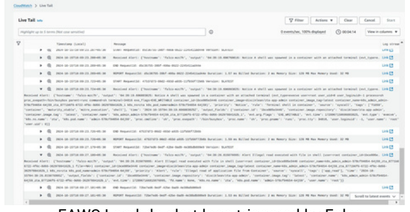




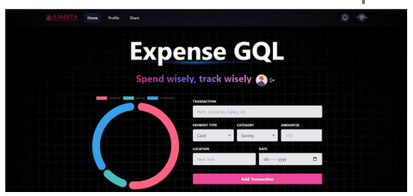
Root | Developer Options | Debugging detection using “JailMonkey” Library

CONCLUSION

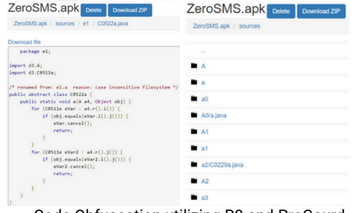
This project implemented the Zero Trust strategy to enhance the security of containerized environments through a proof of concept (PoC) that included a secure web app, Python API, and Android application. Key measures such as JWT authentication, continuous monitoring, mTLS, and RBAC were integrated to demonstrate effective Zero Trust principles. Future enhancements could focus on applying machine learning for anomaly detection, automating security audits, and integrating security into CI/CD pipelines. Additionally, training developers in Zero Trust practices and conducting real-world deployments will further improve security in containerized environments.



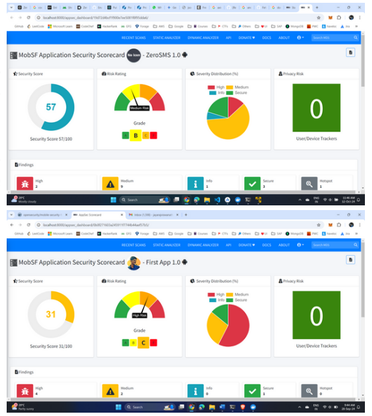
FAWS Lambda alert logs triggered by Falco



User Dashboard of the deployed application



Code Obfuscation utilizing R8 and ProGaurd



ZeroSMS - Security score that is 83.87% higher than that of the standard APK in MobSF Scan