National College of Ireland

Configuration Manual

MSc Research Project
MSc Cybersecurity

RAMANDEEP-
Student ID: X23327260

School of Computing
National College of Ireland

Supervisor: Dr. Rohit Verma

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Ramandeep-

Student ID: X23327260

Programme: MSc in Cybersecurity **Year:** 2024-25

Module: MSc Research Project

Lecturer: Dr. Rohit Verma

Submission Due Date: 11 August 2025

Project Title: IoT Security Framework Configuration Manual

Word Count: ..... **Page Count: 6**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Ramandeep

Date: 11 August 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

IoT Security Framework Configuration Manual

RAMANDEEP-
Student ID: X23327260

1 Project Setup Overview

This manual guide you through setting up a secure IoT network for supply chain operations, incorporating authentication, encryption, and blockchain logging using JavaScript, Next.js, Solidity, Auth0, and Ethereum (via Ganache and MetaMask).
.

2 Tools & Technologies

- Frontend: Next.js, JavaScript
- Authentication: Auth0 (M2M OAuth2)
- Encryption: AES (PyCryptodome or Node.js crypto module)
- Blockchain: Ethereum, Ganache, Solidity, MetaMask
- Testing: Postman, Wireshark, OWASP ZAP

3 Initial Environment Setup

3.1 Install Dependencies

```
npm install
npm install web3 dotenv
npm install next-auth
```

3.2 Install Ganache & Truffle

```
npm install -g ganache truffle
```

3.3 Install MetaMask Extension

- Browser: Chrome or Firefox
- Import test account using Ganache's private key

4 Auth0 Authentication Setup (M2M)

4.1 Create M2M Application

Go to https://auth0.com/ → Applications → M2M
Add:
Name: IoT Auth M2M
Callback URL: http://localhost:3000/api/auth/callback
Save Client ID and Client Secret

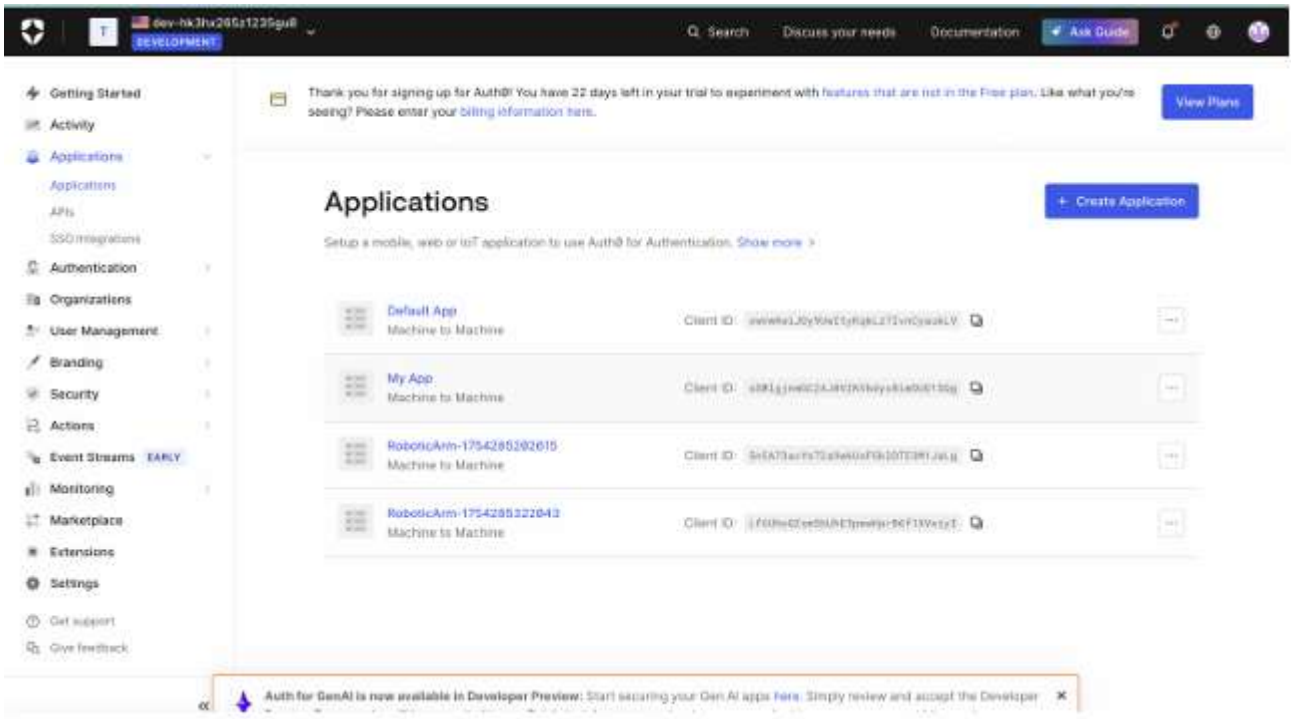

Figure 1: Auth0 Applications Dashboard (M2M Setup)

Figure 1 illustrates the Auth0 Applications Dashboard where the Machine-to-Machine (M2M) application is set up. It displays the client ID, domain, and token endpoint configuration, which serves as the foundation for secure OAuth2 token transfer between IoT devices and backend APIs within the zero-trust architecture.

4.2 Set API Permissions

- Enable read:device, write:device, read:logs scopes
- Add to Postman for token testing

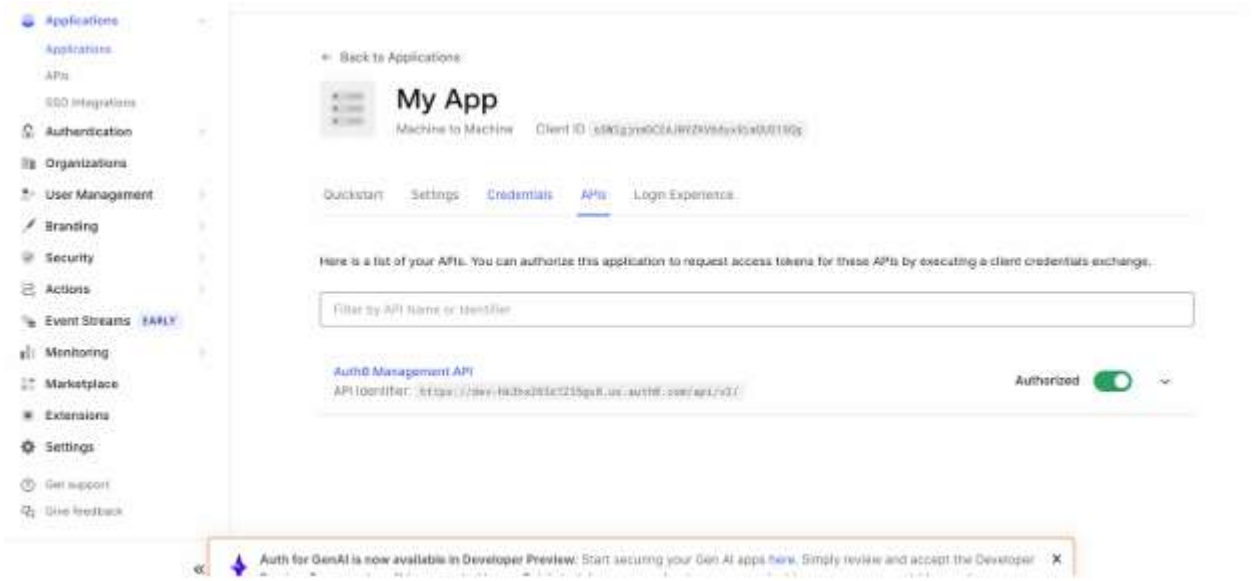

Figure 2: Auth0 App – API Credential Authorization Page

Figure 2 shows the API credential configuration within the Auth0 UI. In this, scopes and permissions are given to the registered M2M app. This is an important step for implementing

role-based API access control and specifying the operations an authenticated device can undertake on the backend.

4.3 Configure .env.local

```
AUTH0_CLIENT_ID=xxxxx
AUTH0_CLIENT_SECRET=xxxxx
AUTH0_DOMAIN=yourdomain.auth0.com
```

5 Blockchain Deployment

5.1 Ganache Setup

- Launch Ganache GUI or CLI
- Note the RPC Server URL (e.g., http://127.0.0.1:7545)

5.2 Compile & Migrate Contracts

```
truffle compile
truffle migrate --network development
```

5.3 Connect MetaMask

Use RPC: http://127.0.0.1:7545
Import Ganache account

6 Data Encryption Pipeline

6.1 Install AES Support

```
npm install crypto-js
```

6.2 AES Usage (example)

```
const CryptoJS = require("crypto-js")
const key = "iotkey2025"
const encrypted = CryptoJS.AES.encrypt("data", key).toString()
const decrypted = CryptoJS.AES.decrypt(encrypted, key).toString(CryptoJS.enc.Utf8)
```

7 Backend API and MetaMask Integration

7.1 Smart Contract Interaction

Use Web3.js to interact with Ethereum

```
import Web3 from "web3"
const web3 = new Web3(window.ethereum)
const contract = new web3.eth.Contract(ABI, contractAddress)
```

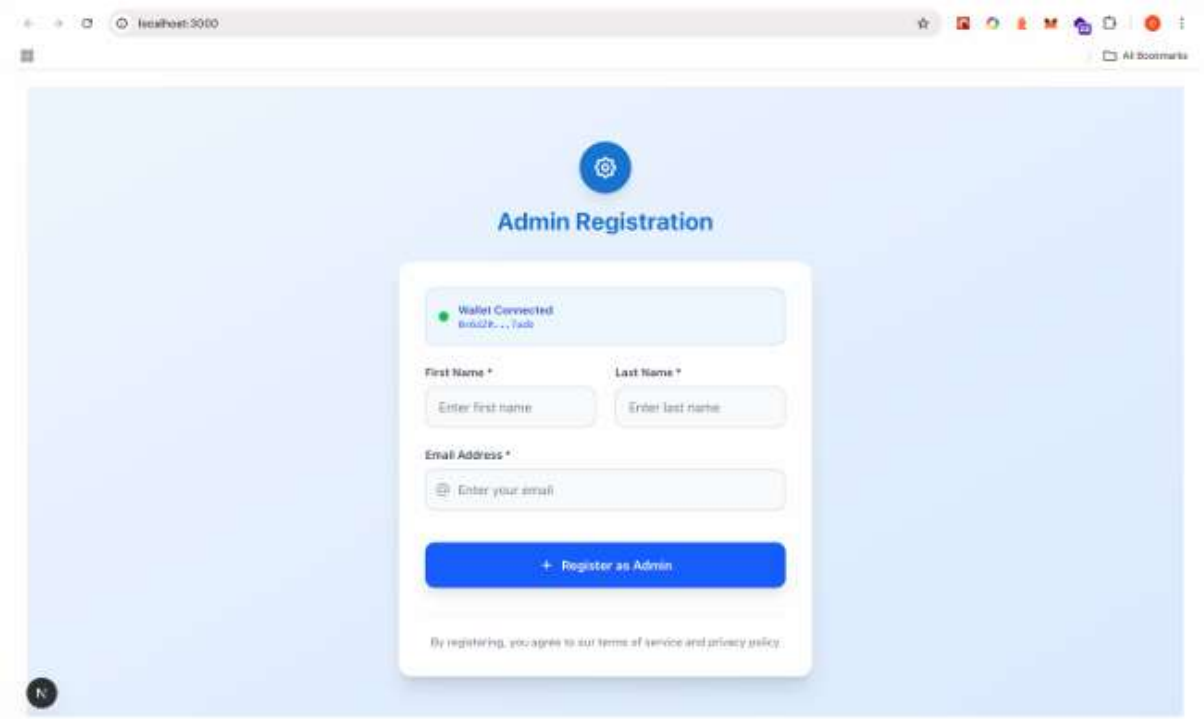

Figure 3: Admin Registration Page with Wallet Integration

Figure 3 illustrates the admin registration page with MetaMask wallet authentication integrated. With this configuration, administrative users are cryptographically associated with a verified Ethereum wallet, providing secure smart contract interaction and transaction signing, which is a must for role-based access and blockchain-based robot operation logging.

7.2 Form Submission & Logging

On robotic input, trigger MetaMask for transaction confirmation.
Log includes: device_id, operation_type, timestamp

8 Security Testing & Validation

8.1 API Token Test via Postman

- Obtain JWT via Auth0
- Test secure endpoints /api/device/register, /api/logs

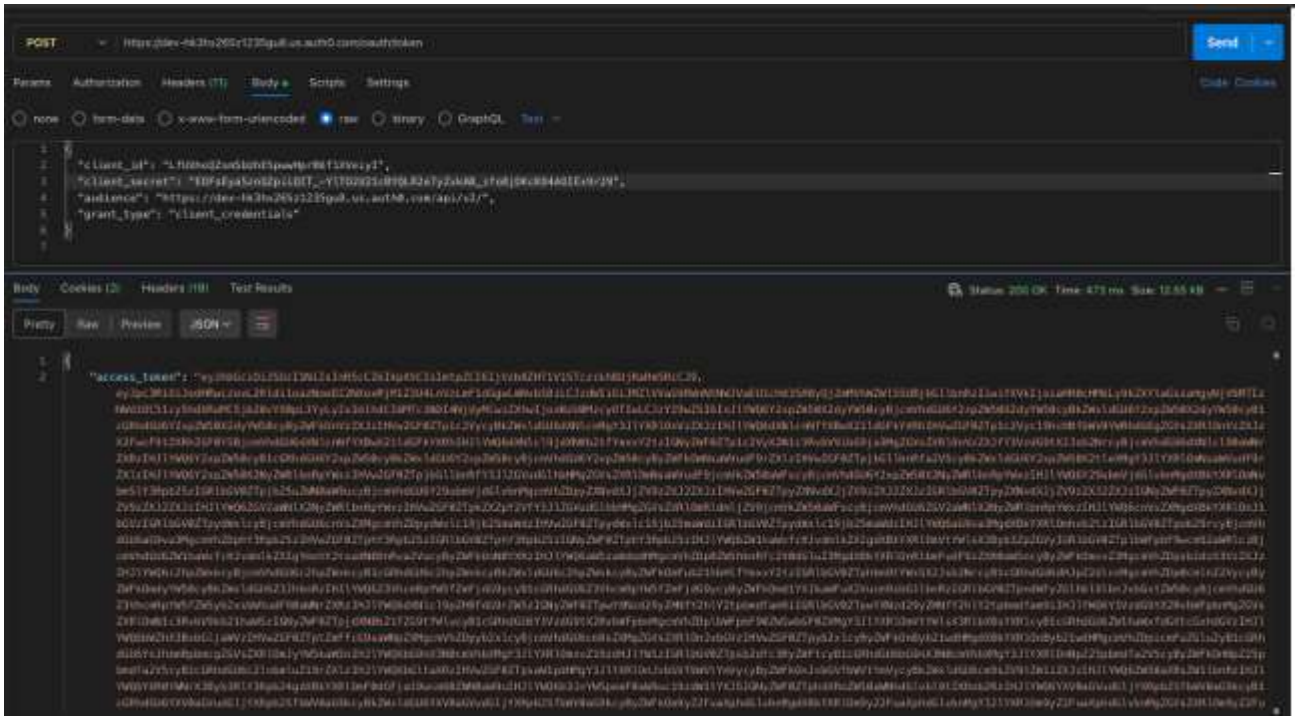

Figure 4: Postman – Token Request and JWT Access Token Response

Figure 4 shows a live Postman request that fetches a JWT access token from Auth0. This illustrates how devices or users authenticate securely and obtain scoped access tokens, which are then used to authorize API calls so that only authenticated entities interact with the system.

8.2 Wireshark Packet Validation

- Monitor real-time encrypted traffic
- Confirm no plaintext data leakage

8.3 OWASP ZAP Test

- Run ZAP scan on http://localhost:3000
- Export security report (optional)

Figure 5 : ZAP Vulnerability Scan Result Output

Figure 5 illustrates the result of a security vulnerability scan performed with OWASP ZAP. The report verifies that the system set up under the given configuration successfully passed all standard security tests, showing immunity to usual web-based attacks, thereby establishing the integrity of the implementation prior to production deployment.

9 Deployment Guide

9.1 Host Frontend (e.g., Vercel or Netlify)

Update .env with production Auth0 and Ethereum details

9.2 Secure Smart Contract on Public Testnet

- Use Sepolia or Goerli via Infura
- Fund test ETH for transactions

9.3 Troubleshooting

Issue	Fix
MetaMask "pending" forever	Ensure Ganache RPC matches MetaMask
Auth0 token invalid	Check client credentials and scopes
Data not logged on blockchain	Confirm transaction confirmation
Encrypted payload unreadable	Verify encryption key matches at both ends

9.4 Recommended Enhancements

- Migrate to public Ethereum testnet for scalability testing
- Integrate alert system for anomaly detection
- Add real-time analytics dashboard
- Apply rate-limiting and JWT expiration

10 References

Abu, N. *et al.* (2022) 'Internet of things applications in precision agriculture: A review', *Journal of Robotics and Control (JRC)*, 3(3), pp. 338–347.

Khan, A.A. *et al.* (2022) 'Internet of Things (IoT) security with blockchain technology: A state-of-the-art review', *IEEE Access*, 10, pp. 122679–122695.

Sallam, K., Mohamed, M. and Mohamed, A.W. (2023b) 'Internet of Things (IoT) in supply chain management: challenges, opportunities, and best practices', *Sustainable machine intelligence journal*, 2, pp. 3–1.

Varriale, V. *et al.* (2021) 'Sustainable supply chains with blockchain, IoT and RFID: A simulation on order management', *Sustainability*, 13(11), p. 6372.

Zhu, L. *et al.* (2023) 'Enhancing the security and privacy in the IoT supply chain using blockchain and federated learning with trusted execution environment', *Mathematics*, 11(17), p. 3759.