SIEM Project Developer Guide

Table of Contents

1. Introduction
2. System Architecture
3. Development Environment Setup
4. Project Structure
5. Coding Standards
6. Git Workflow
7. Testing
8. Deployment
9. Troubleshooting

Introduction

Welcome to the SIEM Project Developer Guide. This document provides comprehensive information for developers working on our Security Information and Event Management (SIEM) system.

System Architecture

Our SIEM system follows a microservices architecture: - Frontend: React-based single-page application - Backend: Django REST Framework API - Database: MariaDB - Log Ingestion: Custom service using NXLog for Windows event logs

Key components: - Log Processing Pipeline - Alert Generation Engine - Reporting Service - User Authentication and Authorization

Development Environment Setup

1. Clone the repository:

   ```
   git clone https://github.com/kylerobertson84/i14-ssiem.git
   cd siem-project
   ```

2. Run the setup script:

   ```
   ./scripts/setup_dev_environment.sh
   ```

3. Access the applications:

   - Frontend: http://localhost:3000
   - Backend API: http://localhost:8000
   - API Documentation: http://localhost:8000/api/docs/

Project Structure

i14-ssiem/ - backend/ - siem_project/ - accounts/ - logs/ - alerts/ - reports/ - frontend/ - src/ - components/ - tests/ - pages/ - services/ - public/ - database/

- docs/ - scripts/

Coding Standards

- Python: Follow PEP 8 style guide
- JavaScript: Follow Airbnb JavaScript Style Guide
- Use meaningful variable and function names
- Write clear comments and docstrings
- Maintain test coverage for new code

Git Workflow

1. Create a new branch for each feature or bug fix
2. Make small, focused commits
3. Write clear commit messages
4. Create a pull request to merge changes into the develop branch
5. Require code review before merging

Testing

Automate testing:

- The project has been set up to include automate testing you can run this by the following code:
- Backend: `./scripts/run_backend_unit_tests.sh`
- Frontend: `./scripts/run_frontend_tests.sh`
- Write unit tests for all new features
- Maintain integration tests for critical user flows

Deployment

1. Ensure all tests pass locally
2. Merge your feature branch into develop
3. Once develop is stable, merge into main
4. Run the deployment script:

   `./scripts/setup_dev_environment.sh`

Troubleshooting

- Check Docker logs: `docker-compose logs`
- Verify database connections in Django settings
- Ensure all required environment variables are set
- Check browser console for frontend errors

For more detailed information on specific components, refer to their respective README files in each directory.