Automated Misconfiguration & Threat Detection in Public Cloud Storage

Background

The adoption of public cloud services such as AWS-S3 and Google Cloud has been steadily on the rise throughout the last years.
However, this trend poses a very serious security threat, namely user misconfigurations (public buckets, weak permissions …), which often lead to data breaches.
Notable incidents, such as the Capital One breach in 2019, exposed sensitive data of over 100 million customers due to misconfigured cloud storage settings.
Manually checking these settings is slow and unreliable, especially when systems are growing in complexity.
A small automated tool, which scans for misconfigurations and raises flags for risks, can prevent such issues.

Project Goals

1) **Develop an automated tool** that detects common misconfigurations in public cloud storage services.
2) **Implement an ML-based anomaly detection component** using unsupervised learning to identify suspicious access patterns in synthetic cloud logs.
3) **Provide clear and easy-to-understand output** that explains each issue and its severity, suitable even for non-experts.
4) **Apply proper software engineering practices**, including thorough requirements analysis and detailed documentation.
5) **Design and implement a modular system architecture**, making the tool easy to maintain and extend to additional cloud providers in the future.

Key Engineering and Technological Aspects

1) **API Integration** - for public cloud providers for retrieval of configuration data (logging, encryption settings, access policies..).
2) **Rule-Based Misconfiguration Checks -** for detecting common issues such as public access and missing encryption.
3) **Lightweight ML Anomaly Detection -** using a simple unsupervised ML model on synthetic logs for identifying suspicious behaviours.
4) **Risk Classification -** assigning severity levels (Low / Medium / High) based on impact and exposure.
5) **Modular Architecture and Design -** designing the system as separate modules for data retrieval, analysis, ML, and reporting to make it easy to maintain and extend.
6) **Simple UI -** providing a basic CLI or minimal web interface that displays findings and recommended actions.
7) **Testing and Validation -** using synthetic configurations and logs to test both the rule-based and ML components.

Scope and Complexity

The scope of the project is estimated as one quarter, and includes test, QA and deployment.
The project presents meaningful engineering complexity through several challenges: working with cloud APIs and authentication, designing a modular architecture with loosely coupled components, combining rule-based and ML-based detection methods, and generating realistic synthetic datasets for validation. The system requires coordinated requirements analysis, design, implementation, thorough testing, and comprehensive documentation.