Technical Specification Document for TIE 202 - Velonix
Table of Contents

Introduction

As organizations increasingly adopt Large Language Models (LLMs) and therefore RAG systems to boost productivity and gain insights, concerns around data privacy and information leakage are becoming critical. Technical specifications play a key role in aligning stakeholders, defining requirements, and enabling secure implementation.

This document outlines the technical direction and objectives for Velonix, a cybersecurity solution built to help organizations securely adopt LLM technologies without compromising sensitive data.

The 4 W's Approach:

Who: Velonix is built for major industries in Saudi Arabia—including finance, healthcare, energy, and government—where protecting sensitive data is vital.

What: Velonix is a cybersecurity tool that integrates, in the background, with Retrieval-Augmented Generation (RAG) systems to control access to sensitive information. It ensures that only authorized users can retrieve specific data, while others receive limited or alternate access.

Where: Designed for seamless integration into existing IT infrastructures, Velonix adapts to different organizational environments without requiring major infrastructure changes.

Why: While LLMs offer immense value, their reliance on data retrieval systems increases the risk of sensitive information exposure. Velonix mitigates this risk through preemptive, real-time cybersecurity tools tailored to each organization’s needs.

Project Objectives and Scope

Objectives	Description
Minimum Viable Product (MVP)	Build and demonstrate the core access-control features of Velonix for RAG system security tools.
Feasibility & Functionality	Validate and test Velonix’s seamless integration with existing systems and its ability to prevent unauthorized data access.
Stakeholder Feedback	Gather feedback from industry partners to improve performance, usability, and compliance features.

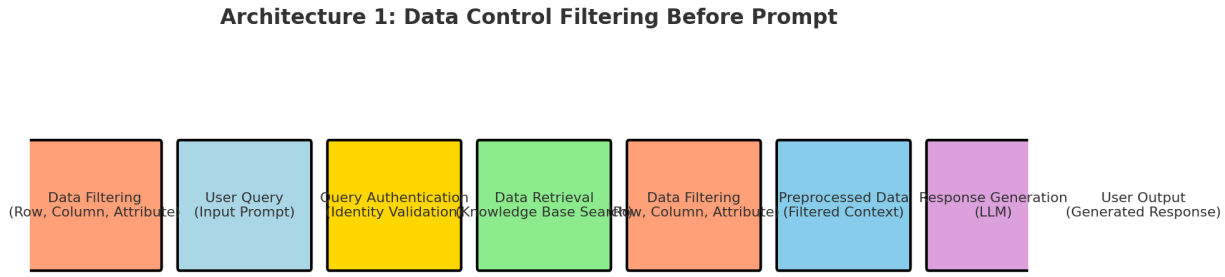
2. System Architecture Overview

2.1 Generic Architecture

Here are the three architecture diagrams showing different placements of the Data Control Filtering in the RAG pipeline:

Architecture 1: Data Control Filtering Before Prompt

Data is filtered based on user permissions even before a query is made. This approach ensures users only interact with datasets they are allowed to query. In this architecture, queries run on a smaller dataset, reducing computation time and improving response speed. This is also the most secure and effective in reducing data leakages however it is not flexible and if the filter is too strict, the LLM may not retrieve enough context, leading to lower-quality responses. In addition, this architecture may need a custom retrieval pipeline for filtering data before query execution.

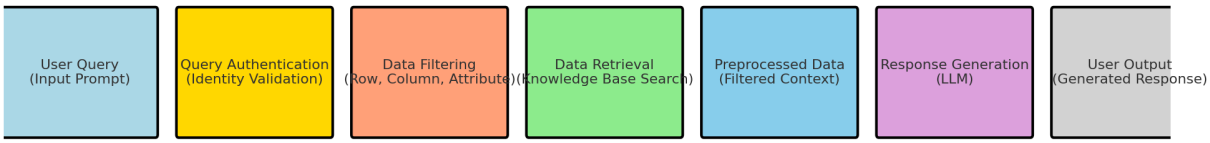


Architecture 2: Data Control Filtering After Prompt

In this architecture, data control filtering occurs after the user prompt is made but before data retrieval. The system dynamically modifies the query to exclude unauthorized data, ensuring only permitted information is retrieved and passed to the LLM. After the user prompt is made, before retrieval, the system modifies the query to exclude unauthorized data before passing it to the LLM. This approach offers several advantages. It allows for more flexible permissions, as access policies can be enforced at query time, adapting to individual user roles. It also enables better handling of ambiguous queries, allowing the system to reframe vague or broad requests instead of rejecting them outright. Furthermore, the LLM can retrieve richer context than pre-retrieval filtering, since the query operates on the full dataset with tailored access rules.

However, this design has its drawbacks. There is a higher risk of data leakage if filtering is not applied strictly enough, making it susceptible to prompt injection attacks. Additionally, this method incurs increased processing overhead, as the retrieval system still searches the entire dataset before filtering, which can lead to slower query response times.

Corrected Architecture 2: Data Control Filtering After Prompt but Before Retrieval

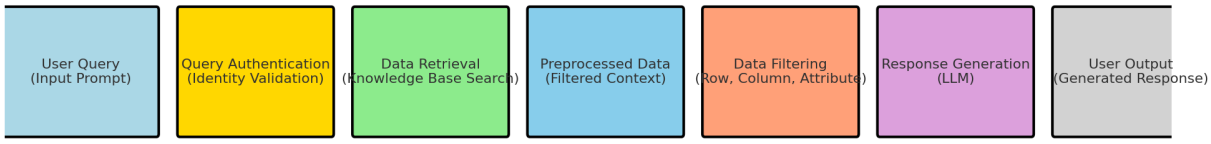


Architecture 3: Data Control Filtering After Retrieval

In this architecture, the system retrieves data from the knowledge base first, then applies filtering based on user permissions. This design enables the retrieval of the most complete and relevant information before access restrictions are enforced. It can be beneficial in scenarios where data relationships are complex, and the filtering process needs full context to make accurate decisions. Additionally, it may reduce the chances of incomplete results, as the LLM has access to a broader dataset before filtering is applied.

However, this approach comes with significant trade-offs. Since the full dataset is accessed before any filtering takes place, there is a higher risk of unauthorized data exposure, especially if the filtering logic is flawed or incomplete. It also increases vulnerability to prompt injection attacks, where malicious input attempts to bypass access controls. From a technical standpoint, this architecture demands robust internal safeguards and strict post-retrieval sanitization mechanisms to prevent leakage. It also introduces greater processing overhead, as the system spends resources retrieving and then filtering potentially large volumes of data. While it provides flexibility and completeness in data context, this model is the least secure among the three if not implemented with strong enforcement protocols.

Architecture 3: Data Control Filtering After Retrieval



Combined Solution

An alternative architecture could integrate two or more data control filtering stages to balance security, flexibility, and system performance. This layered approach may include:

- Pre-Prompt Filtering: restricting the dataset available to the user before a query is made.
- In-Prompt Filtering: modifying the query after it is submitted but before data retrieval.
- Post-Retrieval Filtering: sanitizing the output just before it reaches the LLM.

By combining multiple stages, the system can reduce the risk of sensitive data exposure at various points in the RAG pipeline. For instance, pre-prompt filtering may limit broad access from the start, while in-prompt filtering adds contextual adaptation based on user roles. Post-retrieval filtering can then serve as a final safety net before data is processed by the LLM. This multi-layered design aligns with modern security principles and regulatory requirements, but it introduces added complexity. It requires a deeper understanding of system behavior under different access policies, as well as further testing to ensure that performance, maintainability, and compliance standards are consistently met.

2.2 Software Specifications

The software stack for the system includes components that support secure data processing, real-time response, and seamless integration into enterprise environments. Each category is selected to satisfy specific operational, functional, and security needs.

Category	Specifications
Firmware	Supports automated updates, remote diagnostics, and rollback mechanisms. Firmware also enforces security patches and manages system logs for predictive maintenance.

Data Protocols	TLS-secured HTTP and OAuth-based APIs are used for data exchange. These protocols ensure confidentiality and integrity during communication between clients and back-end systems.
Cloud Services	Scalable cloud platforms such as AWS and Azure are used for data storage, analytics, and visualization. Vector databases like Weaviate or Pinecone support high-speed context retrieval.
Notifications	Real-time alerts are delivered through email, SMS, and mobile/web dashboard notifications to inform users of suspicious activities or system anomalies.

This software configuration is designed to meet enterprise-grade requirements. It ensures system resilience through local caching during cloud outages, enables secure transmission with TLS 1.3, and supports compliance through modular integration and privacy-focused designs.