IRONCORE LABS

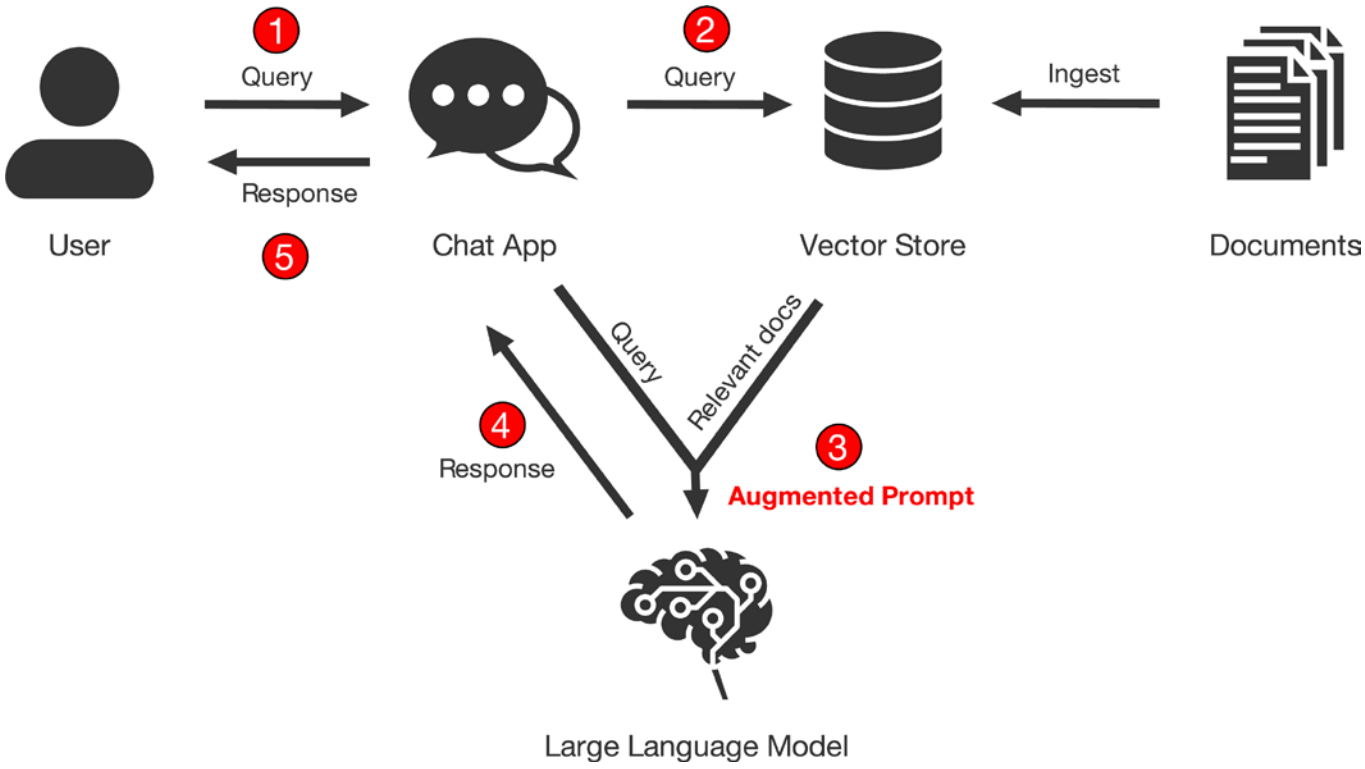
RAG SECURITY RISKS WHITE PAPER

INTRODUCTION

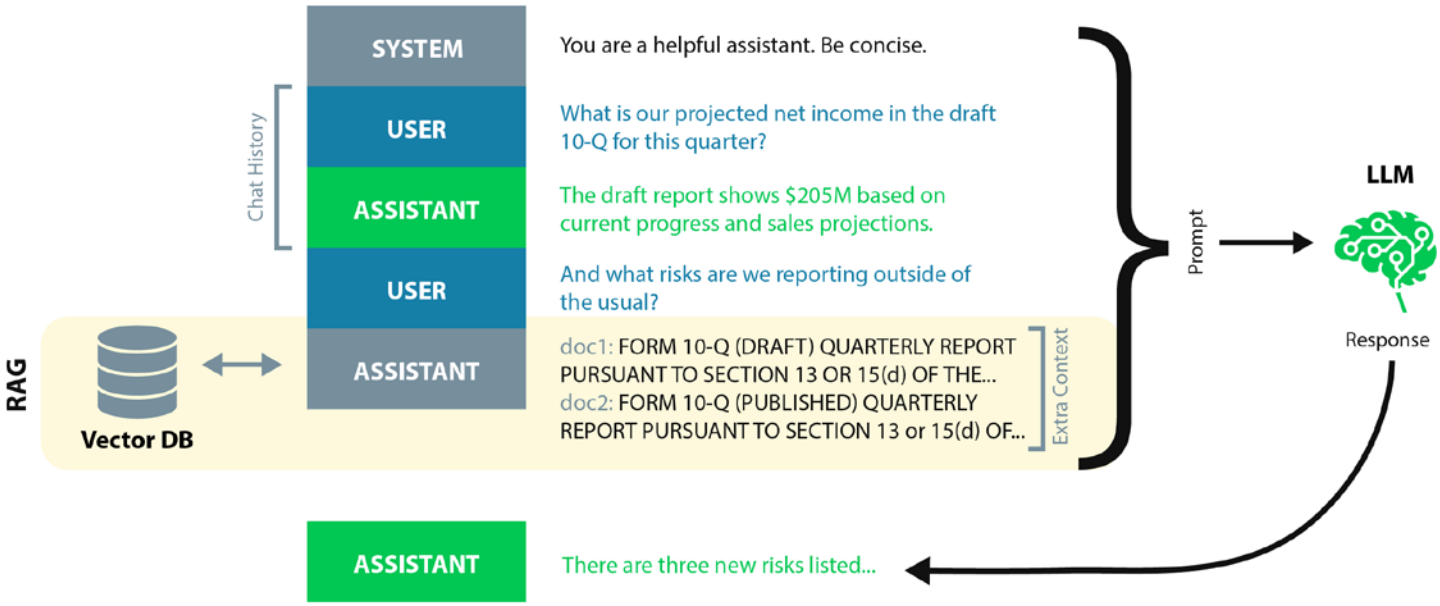
Retrieval-Augmented Generation (RAG) workflows are extremely popular in AI chat bots that use Large Language Models (LLMs). But as adoption has soared, few have stopped to study the security and privacy risks, of which there are many. This white paper will help the reader understand why RAG is so popular and how it should be approached by security professionals.

WHAT IS RAG?

RAG is a technique for enhancing the accuracy, reliability, and timeliness of Large Language Models (LLMs) that **allows them to answer questions about data they weren't trained on**, including private data, by fetching relevant documents and adding those documents as context to the prompts submitted to a LLM.



RAG EXAMPLE

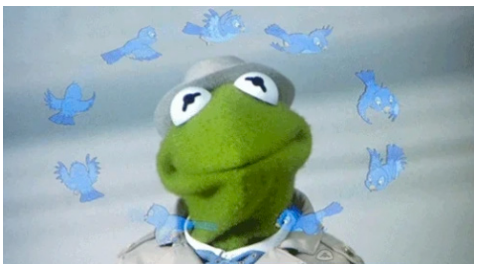


WHY USE RAG?

Retrieval-Augmented generation (RAG) is an architecture that solves several problems when using Large Language Models (LLMs):

1. LLMs "hallucinate" answers

LLMs are amazing at answering questions with clear and human-sounding responses that are authoritative and confident in tone. But in many cases, these answers are plausible sounding, but wholly or partially untrue.



RAG architectures allow a prompt to tell an LLM to use provided source material as the basis for answering a question, which means the LLM can cite its sources and is less likely to imagine answers without any factual basis.

2. LLMs are trained on stale training sets

LLMs are generally trained on large repositories of text data that were processed at a specific point in time and are often sourced from the Internet. In practice, these training sets are often two or more years old.

More recent news, events, scientific discoveries, public projects, etc., will not be included in that training data, which means the LLM can't answer questions on recent information.

RAG architectures allow for more recent data to be fed to an LLM, when relevant, so that it can answer questions based on the most up-to-date facts and events.

3. LLMs lack knowledge of private data

LLMs are trained on public data from across social media, news sites, Wikipedia, and so forth, which helps them to understand language and to answer questions (sometimes accurately) based on public domain knowledge and opinions. But this limits their knowledge and utility. For an LLM to give personalized answers to individuals or businesses, it needs knowledge that is often private.

RAG architectures allow non-public data to be leveraged in LLM workflows so organizations and individuals can benefit from AI that is specific to them.

TOP 5 SECURITY RISKS WITH RAG

1. Proliferation of private data

One of the primary concerns with RAG is the introduction of a new data store called a vector database. This involves new infrastructure and new types of data, which are often copies of private data that's well protected elsewhere. Vector databases store embeddings, which are derived from the private data, but which can be reversed back to near perfect approximations of the original data via inversion attacks.

Being relatively new, the security offered by vector databases is immature. These systems are changing fast, and bugs and vulnerabilities are near certainties (which is true of all software, but more true with less mature and more quickly evolving projects). Many vector database companies don't even have controls in place to stop their employees and engineering teams from browsing customer data. And they've made the case that vectors aren't important since they aren't the same as the source data, but of course,

inversion attacks show clearly how wrong that thinking is.

Vector embeddings and vector databases are an underprotected gold mine for attackers.

2. Oversharing and access mismatches

RAG workflows are unintentionally exposing overshared documents in repositories like SharePoint. Worse still, data from domain-specific applications, like CRMs, ERPs, HR systems, and more, are all finding their way into vector databases. These databases don't have the domain-specific business logic required to control who can see what, which leads to massive oversharing.

3. Simple data discovery for hackers

AI systems are great for surfacing information to the people who need it, but they're also great at surfacing that information to attackers. Previously, an attacker might have had to reverse engineer SQL tables and joins, then spend a lot of time crafting queries to find information of interest, but now they can ask a helpful chat bot for the information they want. And it will be nicely summarized as well. This essentially decreases the time required to effectively respond to an incident and will make incidents more severe, even when the perpetrator is unsophisticated.

LLM prompts are the new APIs and prompt injections are the latest avenue of attack on your infrastructure, which makes these AI systems weak points that need to be better protected by your security program.

4. Leaky LLM prompt/response logs

Prompts from users, and especially those with augmented data included, can be incredibly revealing. This sensitive data flows through systems that can be compromised or that may have bugs. These systems may by default keep logs of prompts and responses. Looking at OpenAI specifically, we've seen account takeovers, leaked credentials, clever hidden injections, and OpenAI bugs that leaked chat histories across users. Some customers now have control over retention times for these records, but using private data in RAG workflows that utilize third-party LLMs still presents risks. Even if you are running LLMs on systems under your direct control, there is still an increased threat surface.

5. RAG poisoning

Many people today are aware of model poisoning, where intentionally crafted, malicious data used to train an LLM results in the LLM not performing correctly. Few realize that similar attacks can focus on data added to the query process via RAG. Any sources that might get pushed into a prompt as part of a RAG flow can contain poisoned data, prompt injections, and more. A devious employee might add or update documents crafted to give executives who use chat bots bad information. And when RAG workflows pull from the Internet at large, such as when an LLM is being asked to summarize a web page, the prompt injection problem grows worse.

SIX STEPS TO PROTECT SENSITIVE DATA IN RAG ARCHITECTURES

RAG brings significant enhancements to response relevance, reduces the occurrence of hallucinations, and allows LLMs to provide customized responses based on private data. However, it is crucial to acknowledge that the integration of RAG into applications introduces new risks for organizations, particularly in relation to sensitive data.

AI systems in general operate better with access to more data – both in model training and as sources for RAG. **These systems have strong gravity for data, but poor protections for that data, which make them both high value and high risk.**

To effectively combat these security risks and ensure the responsible implementation of RAG, organizations should adopt the following measures:

1. Robust data protection with encryption

Data protection is critical, and vector databases lack many of the basics. Advanced application-layer encryption that keeps vectors usable while preventing their use to anyone without the key, is critical to protecting these massive repositories of private data.

IronCore Labs' Cloaked AI is inexpensive and dead simple to integrate, with a growing number of integration examples showing how to use it with various vector databases.

2. Zero-retention LLM chat histories

Minimize the number of places where sensitive data gets stored by turning off any retention of logs for prompts and responses in third-party LLM systems. When using a dedicated and secure system, make sure logs are protected as well at least as well as your most sensitive data since they may contain copies of it.

3. Use confidential compute housed models

A number of startups are running LLMs – generally open source ones – in confidential computing environments where those running the computers -- if done right, both the infrastructure provider (eg, AWS) and the startup -- are unable to see the data flowing through their systems. These confidential compute approaches can further minimize the risks from putting private data inside prompts. Running your own models is also an option if you have the expertise and security attention to truly secure those systems and if the infrastructure provider and data sovereignty-type rules aren't a concern.

There is no single vendor solution or magic bullet to protect AI systems. Step one is to understand the problems and step two is to mitigate them.

4. Minimize private data used in AI systems

Many systems have custom logic for access controls. For example, a manager should only be able to see the salaries of people in her organization, but not peers or higher-level managers. But access controls in AI systems can't mirror this logic, which means extra care must be taken with what data goes into which systems and how the exposure of that data – through the chat workflow or presuming any bypasses – would impact an organization. Broad access controls, such as specifying who can view employee information or financial information, can be better managed in these systems.

5. Reduce agency

Many startups and big companies that are quickly adding AI are aggressively giving more agency to these systems. For example, they are using LLMs to produce code or SQL queries or REST API calls and then immediately executing them using the responses. These are stochastic systems, meaning there's an element of randomness to their results, and they're also subject to all kinds of clever manipulations that can corrupt these processes. Consider allow lists and other mechanisms to control what actions, specifically, can be taken, and to add layers of security to any AI agents and consider any agent-based AI system to be high risk if it touches systems with private data.

6. Security best practices

These are still software systems and all of the best practices for mitigating risks in software systems, from security by design to defense-in-depth and all of the usual processes and controls for dealing with complex systems still apply and are more important than ever.

IronCore Labs is at the forefront of protecting AI systems, workflows, and data.
Want more info? Sign up for a tech Q&A below.

Let's talk!

IRONCORE LABS

ABOUT IRONCORE

IronCore Labs, the leading provider of application data security for modern cloud applications, offers a platform to help software developers and businesses rapidly add application-layer encryption into the heart of their software. The platform protects data wherever it lives and allows it to still be used with encrypted search solutions for keyword indices and vector databases. By protecting the "memory of AI," IronCore Labs is at the forefront of ground breaking new technology to handle quickly emerging security issues while empowering cloud software companies to move fast in adopting new technologies. IronCore's solutions help customers meet data privacy regulations, address international transfer limitations, and fulfilling hold-your-own key (HYOK/BYOK) requests to build trust and profitable relationships with their customers.

IronCore Labs
1750 30th Street #500
Boulder, CO 80301, USA

Inquiries
Email: info@ironcorelabs.com
Phone: +1.415.968.9607

CONNECT WITH US

- ironcorelabs.com/blog
- linkedin.com/company/ironcore-labs
- twitter.com/ironcorelabs
- ironcorelabs.com