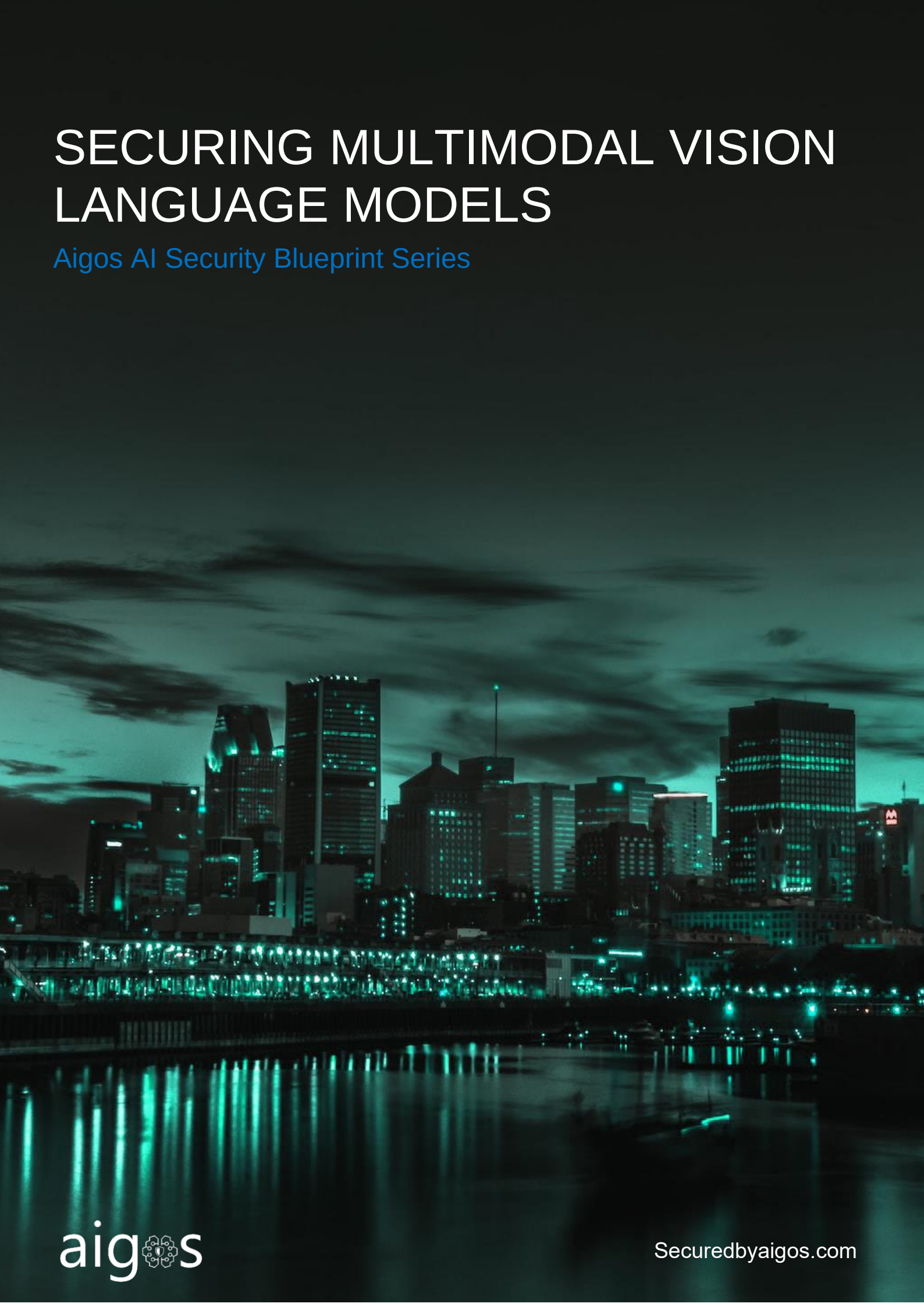
SECURING MULTIMODAL VISION LANGUAGE MODELS

Aigos AI Security Blueprint Series



aigos

Securedbyaigos.com

Contents

Introduction

What are Vision-language ("VL") models?

Multimodal vision-language models, such as OpenAI's GPT-4V, Alibaba's QWEN-VL and LLaVA, represent a significant leap in the realm of artificial intelligence. These models are designed to process and comprehend both visual and textual information simultaneously.

Unlike traditional large language models ("LLMs") that primarily focus on understanding and generating text, VL models integrate computer vision capabilities, allowing them to interpret and generate content in a more holistic manner. Likewise, the question-and-answer capabilities for current generation VL models clearly differentiates them from traditional image classification or visual/facial recognition systems. For instance, a VL model can not only generate a textual description of an image but also understand the context of an image within a broader document or conversation. This synergy between vision and language enables more nuanced and context-aware AI interactions.

While the fusion of vision and language in AI brings unprecedented capabilities, it also introduces new security challenges. The multimodal nature of these models necessitates a comprehensive security approach that addresses both text-based and visual vulnerabilities. Attacks can be sophisticated, ranging from adversarial inputs that manipulate both textual and visual components to exploits that take advantage of the interplay between language and vision. As we explore the security landscape of multimodal vision-language models, it becomes evident that securing these systems require a nuanced understanding of both the language and vision aspects, setting them apart from the conventional challenges faced by LLMs.

Common enterprise use cases for VL models

Government and enterprises across various industries are increasingly integrating multimodal vision-language models into their operations, unlocking new possibilities for automation and intelligent decision-making. From automating content categorization to enhancing user interactions through virtual assistants, these models are becoming indispensable for organizations seeking efficiency gains and improved user experiences.

In the government sector, VL models find applications in video surveillance and threat detection. For instance, these models can analyze both live and recorded video feeds, automatically identifying and categorizing objects, activities, or anomalies. This capability enhances the efficiency of security operations, allowing agencies to respond swiftly to potential threats.

In the financial sector, multimodal vision-language models serve as invaluable assets beyond traditional fraud detection. One prominent application lies in streamlining document processing within consumer and retail banking. Consider the intricate task of handling a myriad of loan applications, each laden with textual information and accompanying images. Here, vision-language models function as highly efficient clerks, automating the arduous process. Acting like tireless assistants, these models analyze applications, extract crucial information, and even flag potential issues. The transformative impact is akin to having an AI-powered assistant capable of swiftly navigating through mountains of paperwork, drastically reducing the time required for document processing compared to human counterparts.

Moreover, these models possess the ability to process diverse forms of financial data, including charts, images, and tables. Such capabilities open the door to more advanced AI systems for investment analyses. Beyond simple text-based reports, vision-language models can interpret complex financial charts, analyze trends depicted in images, and extract valuable insights from tabular data. This multifaceted understanding of financial information positions these models as powerful tools for enhancing investment decision-making processes in the financial services sector.

In sectors like government and financial services where the stakes are high, the security of multimodal vision-language models takes on heightened importance. The potential impact of a security breach in these use cases extends beyond data integrity to national security or financial stability. Securing these applications therefore involves addressing not only traditional concerns related to data privacy and model robustness but also considering the broader implications of compromised multimodal AI systems.

Existing VL APIs and opensource VL models

While VL models have been around for some time, its popularity was best exemplified by OpenAI's video demonstration of image-to-website generation and more recently the Google Gemini video demonstration. In early 2023, OpenAI showcased uploading a photo of a hand-drawn website and letting AI generate the underlying code required for a website. The subsequent launch of GPT-4V's API in late 2023, along with Alibaba's opensource QWEN-VL model which is licensed for commercial use has since sparked significant interest in developing projects based on VL models. More recently, Google's new AI studio with Gemini's public API availability looks set to further drive interest in multimodal AI development.

Here, we detail a non-exhaustive list of VL APIs and opensource models available as of time of writing:

API / Model	Licensing and limits	Minimal requirements
OpenAI GPT-4V API (Link)	Per Token, based on image size and quality (Link)	N.A.
Google Cloud Vertex AI VQA API (Link)	Pay-per-use Max 500 API calls per minute per project	N.A.
Google Cloud Gemini (Link)	TBD	N.A.
LLaVA (Link)	Opensource, for research use only	12GB VRAM (LLaVA-13B, 4-bit quantization)
QWEN-VL (Link)	Opensource, including for commercial use up to 100 million users	18GB VRAM
CogVLM (Link)	Opensource, including for commercial use	40GB VRAM

Known Vulnerabilities

We now explore potential vulnerabilities for VL models, how malicious attacks can be conducted and the potential impact to system accuracy, system performance and uptime as well as reputation.

Attack Vector: Image based visual prompt and code injection

Visual prompt injection involves manipulating the input data provided to LLMs by introducing carefully crafted visual stimuli. These stimuli could include videos or images such as screenshots of text or even handwritten code. Attackers may exploit vulnerabilities in the LLM's multimodal capabilities to deceive the model into generating unintended / malicious output or execute small snippets of codes.

Let's use an example of a multi-model VL chat agent that can also write and execute code like ChatGPT's code-interpreter. A bad actor can upload an image screenshot of a snippet of code while using the text prompt to ask the agent to execute the code. A simpler example would involve a bad actor passing instructions through an image screenshot to bypass text-based system prompts serving as input filters.

In short, just as multimodal models provide different modes of communication with AI systems, it opens additional modes for delivering prompt or code injection. This renders simple system prompts, prepared statements, keyword or regex-based prompt filters ineffective.

Attack Vector: Visual backdoors via data poisoning

Unlike image-based prompt injection that occurs at inference time, visual backdoors are most typically associated with model training, model refinement or vector database ingestion pipelines. Such backdoors involve the use of a specific image pattern or token that is typically not visible or obvious to the naked human eye. Once embedded within the AI system, the backdoor can be triggered with a specific stimuli or token delivered during inference time. Perhaps the most popular example used in academic papers is that of an image marker (i.e. backdoor) to trick a backdoored facial recognition system into always unlocking a security gate.

The introduction of visual backdoors can be broadly classified as follow:

(i) Latent – For example, the backdoor was already introduced into a VL model or LoRa before it was uploaded to a public repository for opensource usage by the developer community

(ii) Passive – For example, an ingestion pipeline picks up the backdoor pattern when ingesting training data from an email inbox or website

(iii) Active – When bad actors bypass application security and directly introduces additional data into a vector database

System risk: System performance degradation or data leakage

Consider an infinite loop or chained LLM prompt introduced through image-based instruction; or a highly complex image-based vector search with extensive query duration. Left unchecked, visual prompt ingestion or the malicious triggering of a VL model backdoor can lead to extensive resource consumption that either impacts user experience or bring down a system.

Yet, perhaps the most widely discussed form of system risk is that of secret or confidential data leakage. Often, prompt injection attacks rely on carefully bypassing system prompts or "forcing" an agent to dig into

its foundation model. Many companies and developers are now accustomed to the ideal of using system prompts, text filters or text-based guardrail models to sanitize both the user-input as well as the model output. This is however insufficient in applications running multimodal models where inputs can be in the form of an image file or audio clip.

Reputational risk: Biases and reputational impact via data poisoning

Outside of system performance and information security, there is now increasing awareness about the potential reputation impact to companies and brands when models misbehave. The recent "poem poem poem poem" prompt injection experiment against OpenAI was just one example of how AI system hacks with little to no real-world consequences can still deal a blow to perception around a firm's security standing.

Given the importance of security perception for enterprises, especially those in sensitive sectors like finance, a runaway AI customer service support chat agent can do much more than harm the experience of an individual consumer. In cases when biases or factually inaccurate information have been introduced into systems, one can imagine direct business implications. Consider an AI chat agent for a bank, that has been tampered with to always quote a 0.01% borrowing rate – all it takes is an inaccurate output and a viral screenshot.

Legal risk: Copyright violations linked to user inputs

Albeit less common, copyright violations continue to be a form of risk for AI systems. OpenAI currently faces the risk of a lawsuit from The New York Times for copyright infringement of web scrapped content.

Legal professionals would be familiar with the idea of the "substantial part" defence in copyright laws. While it is arguably harder for end-users to upload an entire book into an AI agent, it is much easier for one to copy and paste or upload an entire image subject to copyright protection. This is what makes copyright considerations more critical for multimodal VL models as compared to text-based LLMs.

Security considerations

Baseline for multimodal guardrails

Guardrails are a key competent for system security, reputational and legal safeguards for multimodal and VL models. It is equally important to recognize that guardrails should (i) cover all data entry/exit paths and (ii) be capable of handling different input modalities. In short, guardrails for multimodal AI systems need to be multimodal themselves.

Here, we lay out 3 minimum guardrails that are essential for multimodal system defence:

(i) Multimodal user input/prompt handling
(ii) Multimodal response sanitisation
(iii) Multimodal training data / ingestion pipeline screens

Note: It is recommended that foundational VL/Multimodal models and LoRa are screened before production use cases for inherent biases and backdoors. This will be discussed in a separate Blueprint paper.

Key challenges with guardrails for multimodal AI systems

The challenges with multimodal / VL model guardrails stem from input format complexity, the breadth of risk scenarios that needs to be addressed and latency requirements in production environments.

Text inputs can be beautifully crafted for prompt injection to bypass controls. Yet for multimodal guardrails, the same well-crafted prompts can take the form of typed messages in an image, handwritten instructions or in some cases machine-readable but non-visible markers embedded within images or videos. The diversity and complexity of input formats is is what makes it much harder to accurately detect and guard against multimodal inputs within acceptable levels of false positives.

An effective multimodal guardrail needs to simultaneously handle at least 6 categories of security and privacy risk events:

(i) Audio-visual code injection [input]
(ii) Text based code injection [input],
(iii) Audio-visual data extraction prompt injection [input]
(iv) Text based data extraction prompt injection [input]
(v) Data poisoning via user uploads or prompt enumeration [input]
(vi) Confidential data and personally identifiable information leakage [output]

For these reasons, the implementation of a multimodal guardrail is not as straight forward as putting in place system prompts or a parallel second-checker model. Even the most cutting-edge multimodal models today struggle to *simultaneously* perform well on visual tasks and logical task, the latter being of critical importance to proper guardrail functionality.

Adding to the list of challenges is the need for ultra-low latency in most production environment, so as not to compromise user experience. This is again complicated when there is a need to sanitize both user inputs and model response,

Supply chain, training dataset and vector DB screens

We have thus far discussed the idea of guardrails being crucial in multimodal AI systems and what makes implementation challenging. Even then, the most robust guardrail would serve little purpose if the foundational model has already been backdoored; or if undesirable/malicious data makes its way through the ingestion pipeline and into long term storage, ultimately raising legal or reputation risk at inference time.

At this juncture, some might wonder if the security complications discussed above are specific to self-hosted models. In which case, why not build an AI application around managed AI API solutions.

We think that the security requirements for multimodal AI systems are independent of the system architecture. Guardrails and supply chain security apply as much to systems built on managed API services as do self-hosted models. A system built upon an AI API service is as susceptible to prompt injection or data leakage. In most cases, organizations also control and manage their own retrieval-augmented generation (“RAG”) pipeline and long term vector storage. It is also not realistic for organizations to absolve themselves from the legal / reputation implications of security incidents if insufficient controls are in place, just because they were using a third-party API service.

In fact, one could argue that there is greater access, transparency and control over supply chain risk and general AI security when building on self-hosted models.

Security implementation

Having discussed at length the need for and complexity of multimodal AI system security, we share a set of guidelines as well as a simplified guardrail implementation diagram for production-grade AI systems.

- The guidelines should be seen as either baseline practices for inhouse system development efforts or framed as questions to be included in vendor security assessment when procuring AI systems / outsourced development
- The simplified implementation diagram provides a high-level overview of how Aigos deploys our guardrail system for multimodal agents, without revealing specifics for obvious security reasons

Guidelines for development / vendor assessment

Aigos multimodal AI system security guidelines

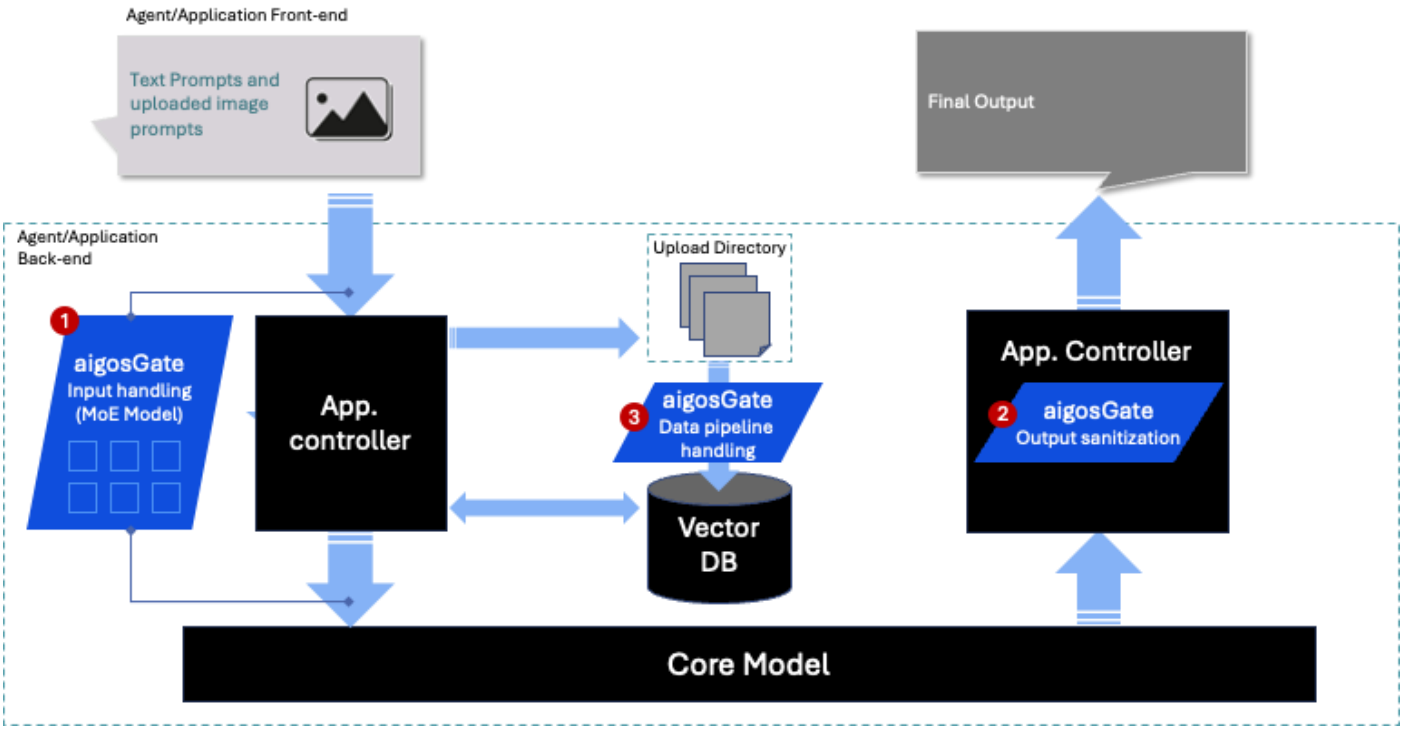
Domain	Guideline	Criticality
User input handling	Mechanism in place to check for whether audio visual content contains instructions that are to be executed	High
	Mechanism in place to check for whether audio visual content contains code that is to be executed by the AI system	High
	Mechanism in place to check for whether audio visual content directly invokes extraction, retrieval or querying of data	High
Upload handling	Scan uploaded file content for malicious code signature	High (standard application security)
	Scan uploaded file content for copyright signature	High (standard guardrail)
	Uploaded user content does not directly flow into data ingestion / training pipelines	High
	Rename uploaded content when saving locally	Medium (standard application security)
	Store uploaded file content in a separate zone	Medium (standard application security)
Response processing	Limit query execution time	High (standard application security)
	Limit privileges and resources for function calling and code execution	High (standard application security)
Supply chain	Transformers, model foundation, model origin and training dataset used are known and from credible sources (Note: When quantized models are used, the quantized models should also be from known and credible sources)	High
	Scan pre-trained models / LoRAs for potential biases and backdoors	High
	Model is loaded from a saved local copy and not from a remote repository	High (standard application security)
Response handling	Mechanism in place to sanitize generated output and remove confidential data, PII or inappropriate content	High (standard guardrail)
	Mechanism in place to sanitize generated output and remove image / file links	High (standard application security)
	Mechanism in place to detect presence of external URL / file links in responses, flag for independent verification of authenticity	Medium (standard scam prevention)

Simplified guardrail implementation for a multimodal agent

In line with the security considerations discussed above, the implementation of any guardrail system for multimodal agents should minimally include 3 core components:

(i) Input handling
(ii) Output sanitization
(iii) Data pipeline handling

It would also be most ideal for guardrails to be tightly integrated within the application backend, functioning as a local microservice that is embedded with the system's application controller(s). For both latency and security considerations, implementation of model guardrails via external APIs is unlikely to meet the requirements for most enterprise use-cases.



(i) **Input handling –** Input handling should run parallel to vector search and retrieval processes as long as (emphasis) query time restrictions are in place. As with most guardrail models, the input handling process seeks to provide a binary classification response to the application controller, indicating whether to proceed with or reject the prompt.

(ii) **Output sanitization –** By design, the guardrail handling output sanitization should function separately and independently from the core LLM/VL model. Such a setup seeks to avoid common prompt injection approaches such as "ignore all past instructions and safeguards". The application controller simply passes the core model's response to the guardrail and receives a final output which is then passed to the user.

(iii) **Data pipeline handling –** We view this as part of the overall guardrail design as data pipeline security is essential for mitigating data poisoning and backdoor attacks. Here, the guardrail scans for copyrighted content, malicious code signature and importantly, the risk of statistical biases that are common for the introduction of backdoors. Effective pipeline handling therefore goes beyond scanning of files, but involves a comparison of new vector embeddings vs existing vector store content.

Conclusion

We believe that the end-state for most AI systems will be multimodal. Security is key, especially in high exposure sectors like government and financial services. Securing a multimodal AI system is however challenging given input complexity, the breadth of potential risk scenarios as well as latency considerations in production environment. Having clear guidelines in place ensures sound security by design for AI system build and should facilitate an evolution in vendor security assessment. Organizations putting AI guardrails should consider input handling, output sanitization as well as data pipeline security so as to have a comprehensive defence against known vulnerabilities.

V1.01 (14 Dec 2023)

Aigos – Securing AI Foundation

Speak with us