OWASP LLM Security Top 10

Die größten KI-Risiken und wichtigsten Schutzmaßnahmen

Stephan Schulz, F5, Senior Principal Solutions Engineer
Daniel Wolf, Controlware, Lead Technical Consultant

16.09.2025, Congress Park Hanau

Agenda

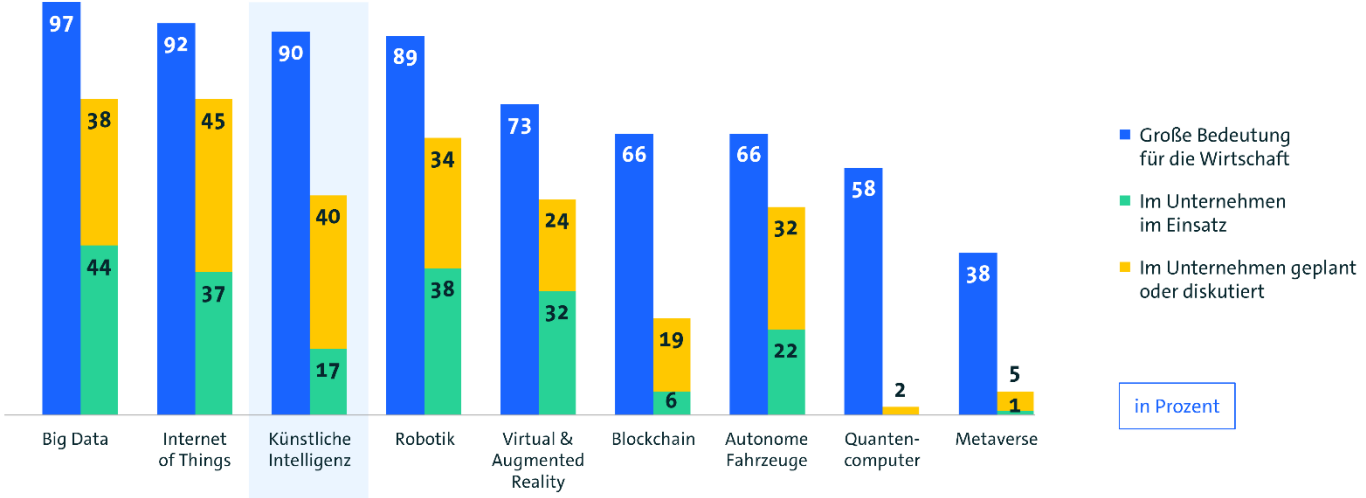
- Warum LLM Security jetzt?
- Architektur-Basics von LLM-Anwendungen
- OWASP LLM Security Top 10 – Überblick & Mapping

Warum LLM Security jetzt?

Warum jetzt LLM Security?

Digitale Technologien kommen in der Breite an

Welche Bedeutung haben die Technologien für die Wettbewerbsfähigkeit deutscher Unternehmen in der Zukunft und welche werden in Ihrem Unternehmen genutzt?



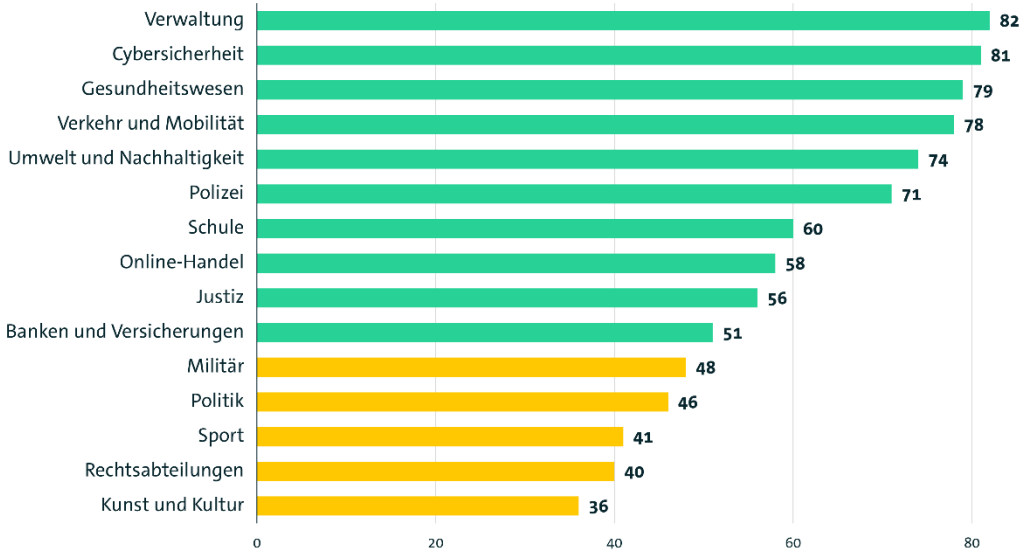
Basis: Alle Unternehmen (n=603) | Prozentwerte für »Sehr große Bedeutung« und »Eher große Bedeutung« | Quelle: Bitkom Research 2025

bitkom

Warum jetzt LLM Security? – cont.

Die Menschen wünschen sich einen breiten KI-Einsatz

Würden Sie sich wünschen, dass in diesen Bereichen künftig KI genutzt wird?



in Prozent

Basis: Personen ab 16 Jahren (n=1.005) | Prozentwerte für »Ja, auf jeden Fall« oder »Eher ja« | Quelle: Bitkom Research 2025

bitkom

Warum jetzt LLM Security? – cont.

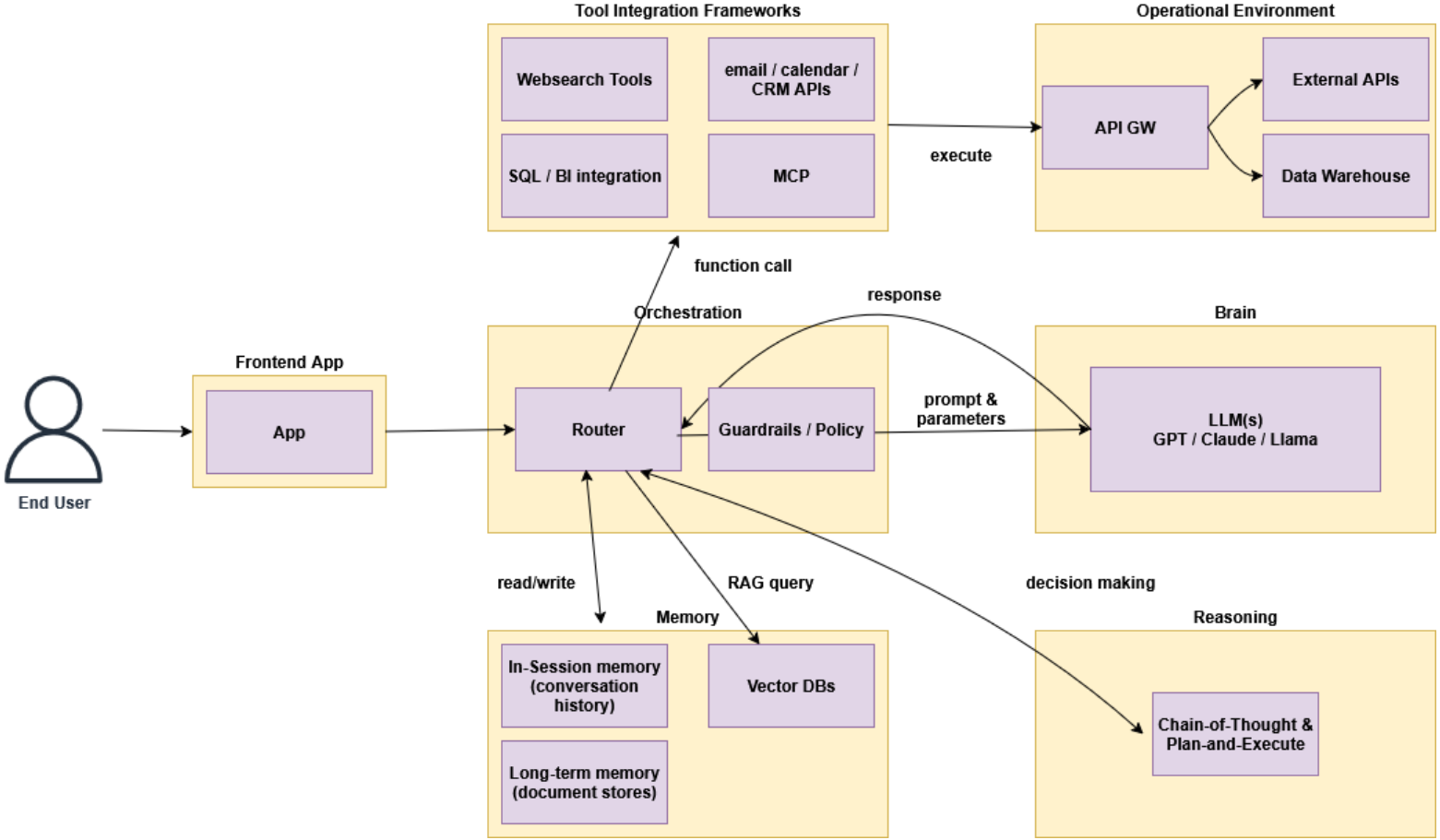
Liste aktueller KI Security Fails, kurz vorm Security Day aktuell einfügen

Architektur-Basics

LLM Buzzwords 101

- LLM
- Inference
- Agent
- RAG
- MCP
- Reasoning
- Chain-of-Thought

Key Components of Agentic Architectures



OWASP LLM Top 10

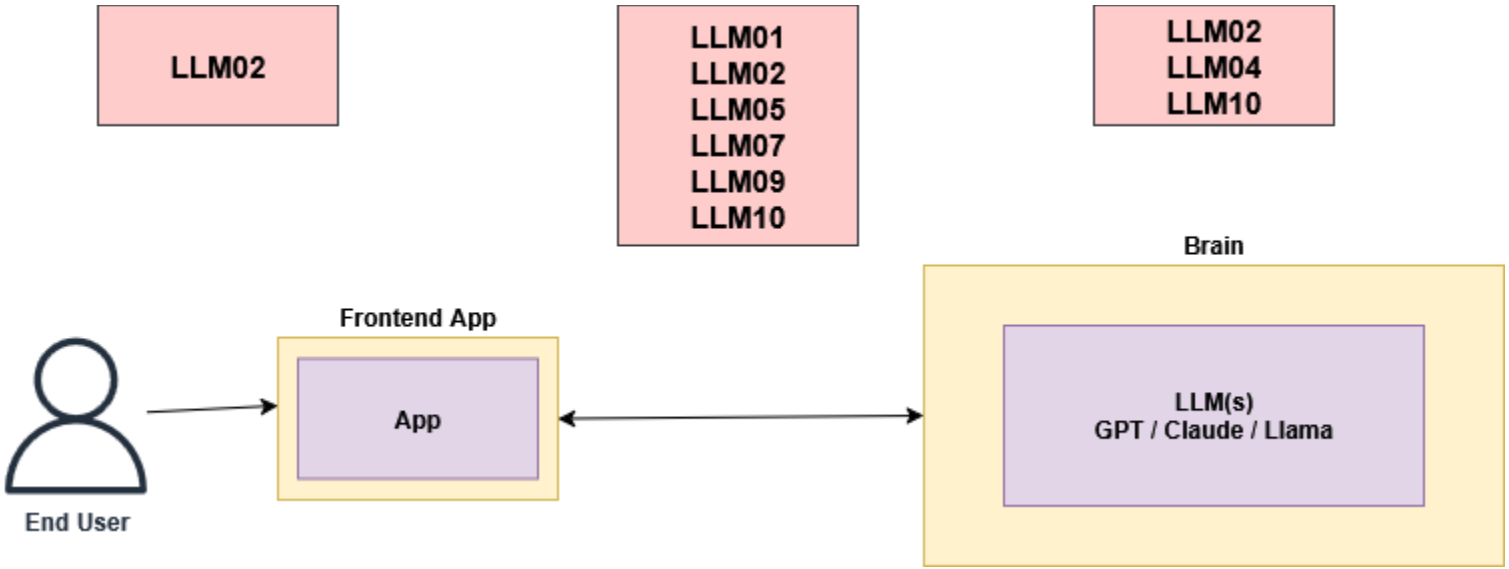
OWASP Top 10 for LLM Applications 2025

- LLM01:2025 Prompt Injection
- LLM02:2025 Sensitive Information Disclosure
- LLM03:2025 Supply Chain
- LLM04:2025 Data and Model Poisoning
- LLM05:2025 Improper Output Handling

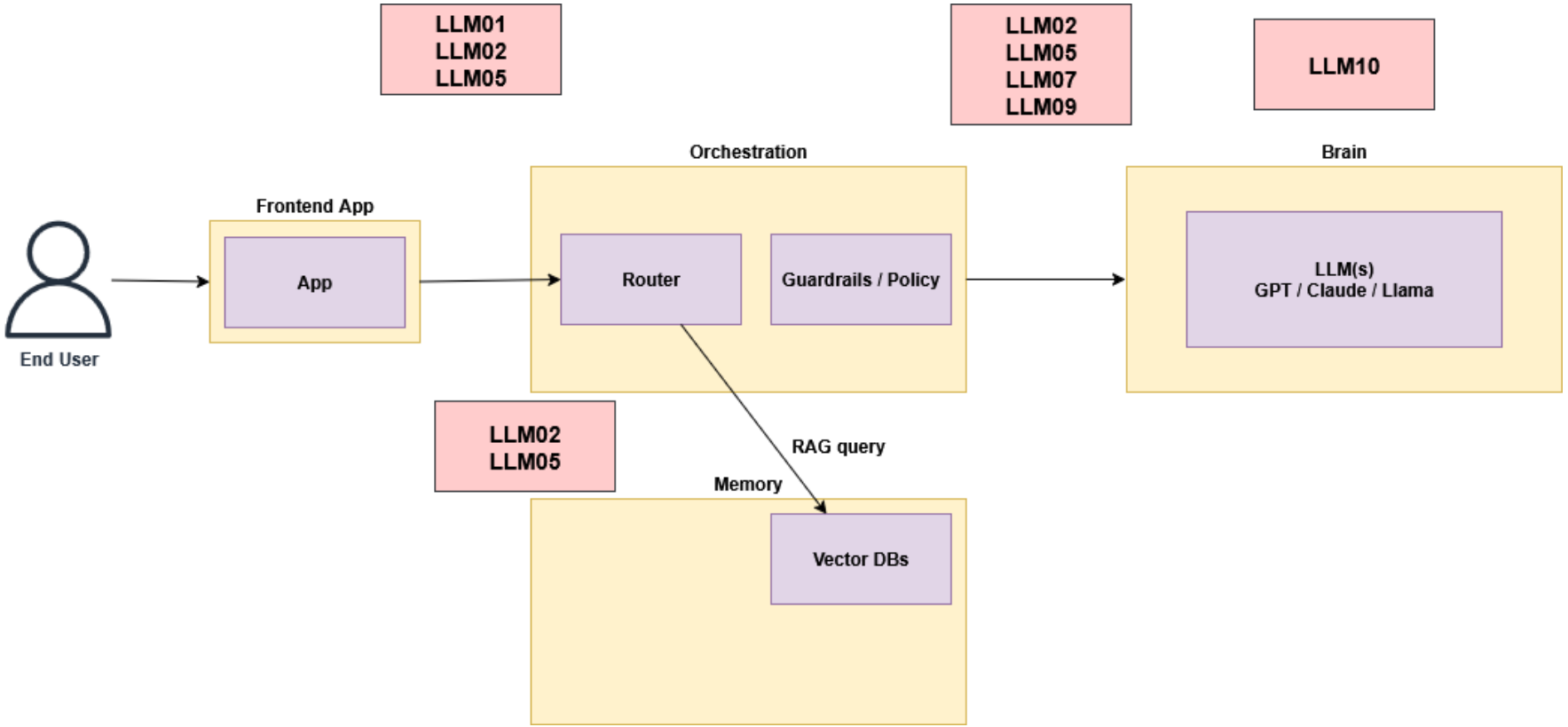
OWASP Top 10 for LLM Applications 2025 - cont.

- LLM06:2025 Excessive Agency
- LLM07:2025 System Prompt Leakage
- LLM08:2025 Vector and Embedding Weakness
- LLM09:2025 Misinformation
- LLM10:2025 Unbounded Consumption

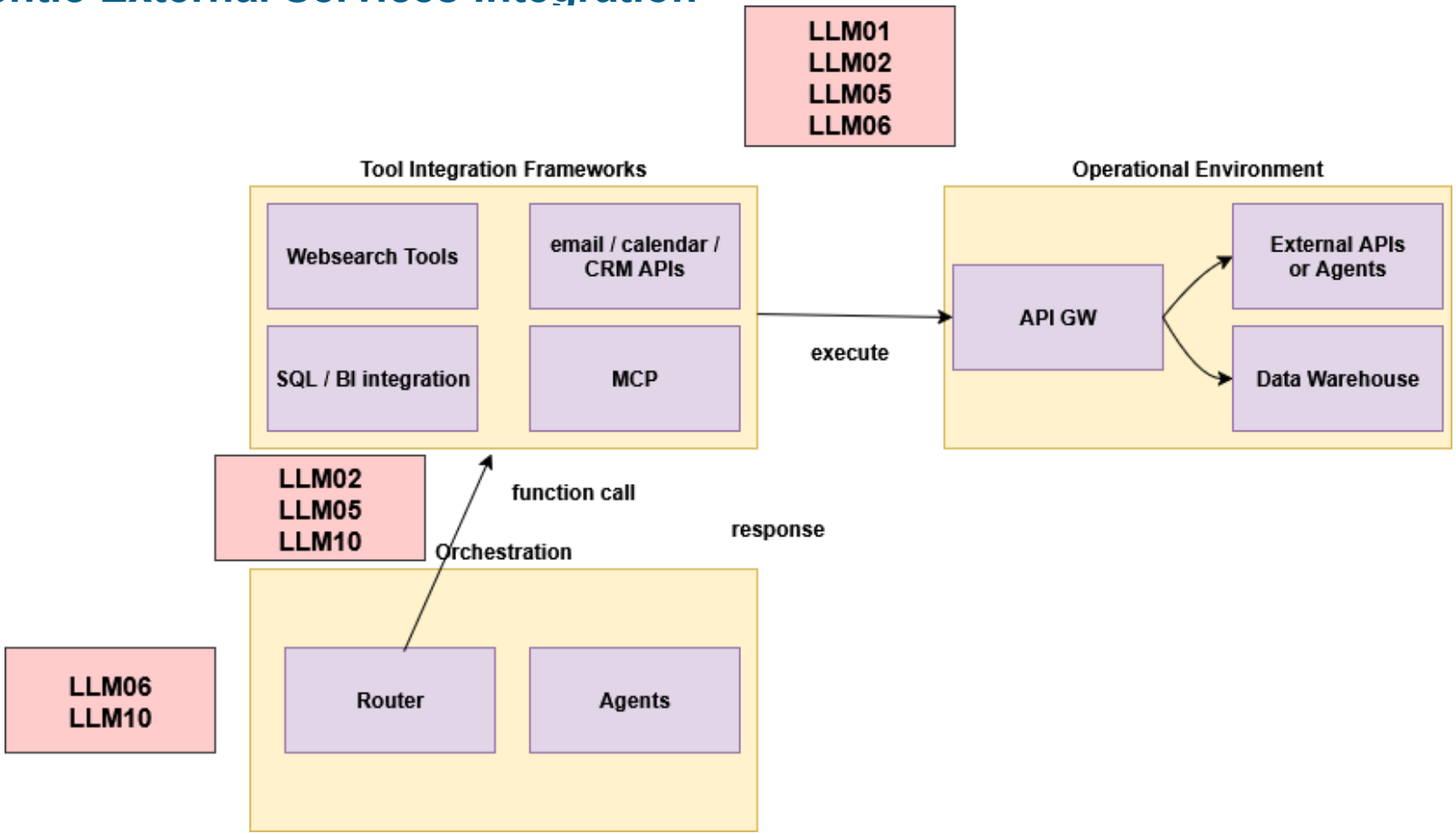
Inference



Inference with RAG

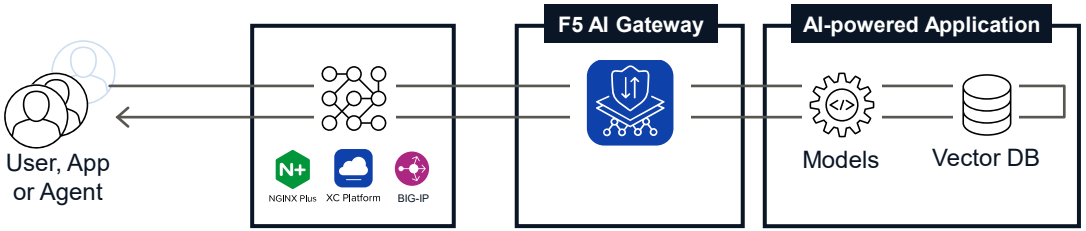


Agentic External Services Integration



F5 AI Gateway (AIGW)

F5 AI Gateway - Overview



Inspect inbound prompts and outbound responses to prevent bad outcomes or data leakage and route prompts to the proper models for cost efficiency.

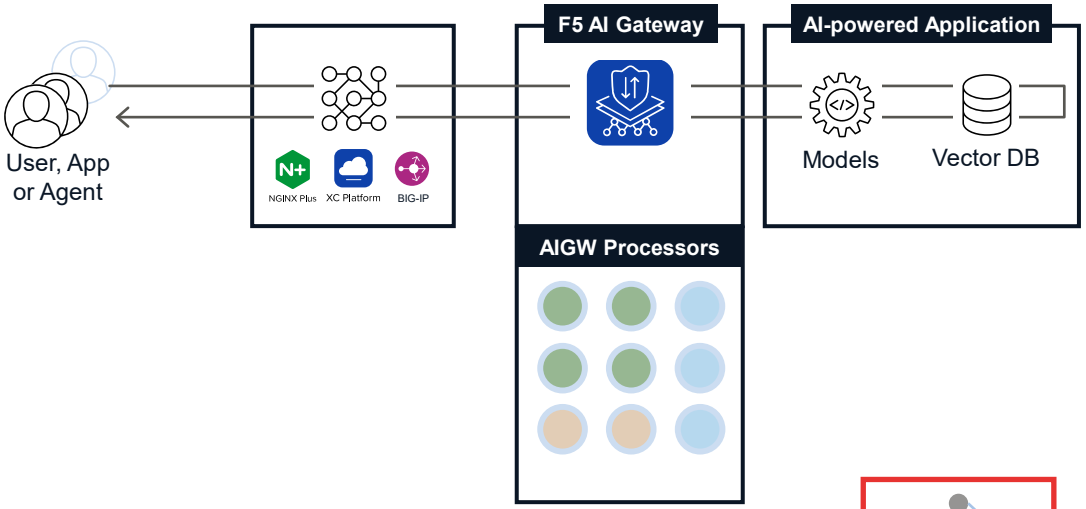
Problem Areas Addressed

- Mitigate unauthorized data outputs
- Prevent attacks on LLMs (OWASP Top Ten)
- Visibility on AI App transactions

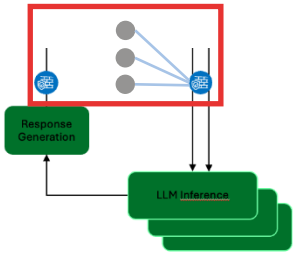
Key Value Proposition

- Fine-grained access control to sensitive data
- Maximized AI application responses
- SDK enables tightening of AI Gateway security and controls to business operations

F5 AI Gateway – Components



- Processor provided by F5
- Processor provided by an F5 Partner
- Processor developed by Customer



Integration:

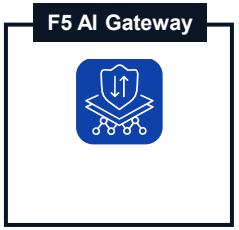
- Container / Kubernetes deployable
- Processor SDK (Python, Go, Rust)
  – typically build as separate container

Support for:

- OpenAI API, Azure OpenAI, Anthropic, Ollama

AIGW is placed „inline“ while processors are interacting with AIGW based on rule set

AIGW - more details…



- **Architecture and Deployment**
  - built on a portable microservices architecture, enabling deployment wherever AI applications are being built
  - deployable to public/private clouds or on-premises data centers.
- **Proxy Architecture for LLM Traffic**
  - proxy architecture for LLM traffic with support for leading AI platforms (like Anthropic, ChatGPT, Ollama, and Azure AI Studio)
- **Processor Framework**
  - each version comes with processors out-of-the-box (running within the core) for different tasks like: prompt-injection or language-id and more

AIGW Processors

- **SDK and Extensibility**
  - AIGW processors – option to adapt to custom and new requirements
  - SDK for Python, Rust, and Go

- **Concept**
  - custom processors like iRules - focus on “rule” logic (“what to do” or “how to do it…”)
  - optionally can leverage it’s own (S)LM to verify (i.e. identify language, etc.). The model then runs typically within the container.
  - depending on the processor type, some are dependent on language (semantic, meaning, ”slangs”, etc.)
  - there are “experimental“ versions with support for more languages (notGA yet)
  - in general: missing functionality can programmed with custom AI processors

Processor details	Supported
Deterministic	No
GPU acceleration support	Yes
Base Memory Requirement	655 MB
Input stage	Yes
Response stage	No
Recommended position in stage	Beginning
Supported language(s)	English, French, German, Hindi, Italian, Portuguese, Spanish, Thai.

example: experimental processor: Prompt Guard

Processor Types (current version)

- **GA-Processors:**
  - prompt-injection (detecting prompt-injection attacks) - english only
  - system-prompt (allowing tighter security guardrails) - language independent (should work with all languages)
  - language-id (predicting language for context-based routing)
  - repetition-detect (detecting attacks that rely on repeating strings - recognizes pattern)
  - watermark (unique identifiers into AI-generated responses, enables full traceability)
  - PII-Redactor: (automatically detects and removes/masks PII in requests or responses)

- **Experimental Processors:**
  - Prompt Guard (intended to cover OWASP Top Ten) - (EN, FR, DE, HI, IT, PT, ES, TH)

Processor example

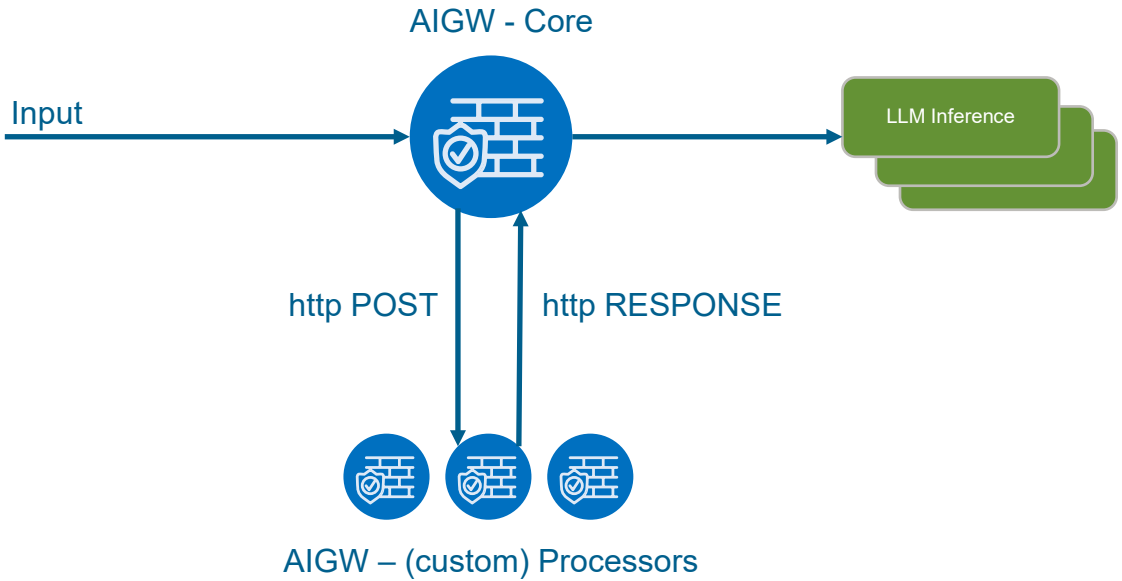
internal or external

```yaml
processors:
  - name: reject-processor
    type: reject        # Built-in processor, runs IN the Core
    params:
      message: "Request rejected"
```

```yaml
processors:
  - name: cat-classifier
    type: external      # ← External, HTTP-based processors
    config:
      endpoint: "cat-classifier-service:80"
      namespace: tutorial
      version: 1
    params:
      reject: true
      annotate: true
```

```python
class CatClassifierParameters(Parameters):
    """Custom parameters for the Cat Classifier processor"""
    enforce_cat_message: str = "Remember to talk about cats in your response"
    reject: bool = False
    annotate: bool = True

class CatClassifierProcessor(Processor):
    """
    A simple processor which adds a tag when a cat is found in the prompt or response
    """

    def __init__(self):
        super().__init__(
            name="cat-classifier",
            version="v1",
            namespace="tutorial",
            signature=BOTH_SIGNATURE,
            parameters_class=CatClassifierParameters
        )

    def process(self, prompt, response, metadata, parameters, request):
        """
        Main processing logic
        """
        my_tags = Tags()
        cat_found = False
        result = {}

        # Check if 'cat' is mentioned in the response
        if response:
            cat_found = any(
                "cat" in choice.message.content.lower()
                for choice in response.choices
                if choice.message.content
            )
            if cat_found:
                result["cat_detected_in_response"] = True
                my_tags.add("cat-response")

        # Check if 'cat' is mentioned in the prompt
        elif prompt:
            cat_found = any(
                "cat" in message.content.lower()
                for message in prompt.messages
                if message.content
```

AIGW Processor – flow (simplified)



AIGW – Demo

OWASP Top 10 LLM01
Prompt Injection

AIGW – Demo

OWASP Top 10 LLM02
Sensitive Information Disclosure

AIGW – Demo

OWASP Top 10 LLM07
System Prompt Leakage

what else…?

AIGW – Demo

System Guard Rails

AIGW – Demo

Identity Aware Model Routing

Controlware
Security Day

Danke für Ihre Aufmerksamkeit.
Wir freuen uns über Ihr Feedback!

Bitte geben Sie den ausgefüllten Bogen am Empfang ab und erhalten Sie als Dankeschön ein kleines Präsent.