Semantic Security Gateway Firewall

(SSGF): A Deterministic Hybrid Architecture for AI Input Security

Whitepaper
Author: Ernesto Rosati, SSGF Architect
Date: February 3, 2026
Location: Tijuana, MX

Contact: [rosatisoft@gmail.com]

Executive Summary

The Semantic Security Gateway Firewall (SSGF) is a deterministic semantic security infrastructure designed to protect and optimize AI systems by filtering ambiguity, manipulation, and low-signal inputs **before** they reach expensive reasoning models.

Modern Large Language Model (LLM) deployments suffer from systemic inefficiencies: high operational costs, probabilistic uncertainty, susceptibility to semantic manipulation, and limited auditability. In most real-world environments, **70–90% of inputs do not require deep reasoning**, yet are processed as such, resulting in unnecessary token consumption, latency, and security exposure.

SSGF addresses this problem by introducing a **multi-layer semantic decision pipeline** that prioritizes deterministic, rule-based inspection before probabilistic inference. By applying local structural analysis, entropy scoring, and intention heuristics, SSGF resolves the majority of traffic at the edge, escalating only genuinely ambiguous cases for deeper semantic evaluation.

This approach enables:

- 70–90% reduction in LLM API usage
- Sub-5ms latency for most decisions
- Deterministic, auditable security enforcement
- Vendor-independent, on-premise or hybrid deployment

SSGF transforms AI security from probabilistic moderation into **founded determinism**, where decisions are based on verifiable structure and evidence rather than model opinion. This makes it suitable for enterprise, government, regulated environments, and any system where clarity, cost control, and auditability are critical.

This document presents the architecture, design principles, benchmarks, and implementation model of SSGF as a reusable semantic security infrastructure.

1. Introduction

Large Language Models (LLMs) have enabled powerful new applications across customer service, analytics, automation, and decision support. However, as these systems scale, a fundamental inefficiency becomes apparent: **most inputs processed by LLMs do not require deep reasoning**, yet they are treated as if they do.

In real-world deployments, AI systems are exposed to:

- spam and automated noise
- phishing and social engineering attempts
- prompt injection and coercive language
- ambiguous or low-signal queries
- structurally valid but semantically risky requests

Processing all such inputs through probabilistic reasoning models leads to:

- unnecessary token consumption
- increased latency
- inconsistent or non-auditable decisions
- heightened security risk

SSGF is designed to address this problem at the architectural level.

---

2. The Core Problem: Probabilistic AI at the Input Layer

Most AI security and moderation solutions operate **after** the model has already processed the input, relying on:

- post-generation filtering
- probabilistic classifiers

- model “judgment” of intent

This approach has three systemic weaknesses:

1. **Cost Inefficiency**
   Every input—regardless of quality or risk—is processed by an LLM.

2. **Uncertainty and Hallucination**
   Probabilistic systems may produce different decisions for identical inputs.

3. **Lack of Auditability**
   Model decisions are difficult to explain, reproduce, or justify in regulated environments.

SSGF approaches the problem differently: **by enforcing structure and determinism before probabilistic reasoning is allowed**.

---

3. Design Principles: Founded Determinism

SSGF is built on the principle of **Founded Determinism**:

> Security decisions should be based on verifiable structure and evidence, not model opinion.

This means:

- identical inputs yield identical outcomes
- decision logic is explicit and inspectable
- ambiguity is treated as a signal, not a failure
- probabilistic reasoning is used only when necessary

Entropy, structure, and intent are evaluated **before** reasoning.

---

4. Architecture

SSGF is designed as **semantic decision infrastructure**, operating as a gateway between input sources and AI reasoning systems. Its architecture enforces deterministic inspection before probabilistic inference, ensuring that ambiguity, manipulation, and low-signal inputs are addressed at the earliest possible stage.

The system follows a **layered decision model**, where each layer has a clearly defined role and authority. No layer implicitly overrides another, and all escalation paths are explicit and auditable.

4.1 Architectural Overview

At a high level, SSGF is positioned as middleware or proxy, independent of any specific LLM provider.

```
Client / Input Source

        |

        ▼

Semantic Security Gateway Firewall (SSGF)

        |

        ├── FAST Pipeline (Deterministic, Local)

        |        |

        |        ├─ Normalization

        |        ├─ Structural Triggers

        |        ├─ Entropy Scoring

        |        ├─ Intention Heuristics

        |        └─ Preliminary Decision

        |

        └── DEEP Pipeline (Conditional)

                 |

                 ├─ Controlled Semantic Inspection
```

```
                  ├─ Hard Security Rules

                  └─ Final Decision Override

        │

        ▼

Protected AI System / LLM Backend
```

The gateway is responsible for producing a **final, structured decision** before any request reaches a reasoning model.

4.2 Separation of Responsibilities

A core design goal of SSGF is the strict separation between **decision enforcement** and **semantic interpretation**.

- **FAST Pipeline**
  Enforces deterministic, rule-based decisions using local computation only.

- **DEEP Pipeline**
  Performs constrained semantic inspection for ambiguous cases only.

- **LLM / Backend Systems**
  Remain purely execution and reasoning engines, never decision authorities.

This separation ensures that probabilistic behavior cannot silently influence security outcomes.

4.3 FAST Pipeline: Deterministic Decision Layer

The FAST pipeline processes **all incoming inputs** and is always executed locally.

Its objectives are:

- eliminate obvious noise and spam
- block clear policy violations
- identify ambiguous inputs early
- resolve the majority of traffic without LLM involvement

FAST Components

Normalization
Inputs are canonicalized to reduce variability and evasion:

- case folding
- whitespace normalization
- Unicode normalization
- punctuation handling

Structural Triggers
Deterministic patterns identify known high-risk structures, such as:

- credential or OTP requests
- payment and banking data extraction
- coercive imperative language

Entropy Scoring
Structural entropy is calculated as a measurable signal of:

- abnormal urgency density
- manipulation patterns

- syntactic irregularity

Entropy is treated as **evidence**, not as an abstract probability.

Intention Heuristics
Lightweight classification rules assign provisional intent categories (e.g. benign, suspicious, unknown) without invoking semantic reasoning.

FAST Outcomes

Each input results in one of the following preliminary decisions:

- **ALLOW**
- **WARN**
- **BLOCK**
- **ESCALATE**

The FAST pipeline resolves **85–90% of traffic** without escalation.

4.4 DEEP Pipeline: Conditional Semantic Inspection

The DEEP pipeline is activated **only** when the FAST pipeline detects ambiguity that cannot be resolved deterministically.

Key constraints:

- invoked selectively, not by default
- executed using controlled prompts
- structured JSON-only outputs
- hard security rules enforced

The DEEP pipeline does not reinterpret the entire input freely. Instead, it answers **narrow, security-focused questions** defined by policy.

Its purpose is refinement, not replacement.

4.5 Decision Authority and Overrides

SSGF enforces a strict hierarchy of authority:

1. Hard security rules always override semantic interpretation.
2. Deterministic FAST decisions take precedence whenever applicable.
3. DEEP decisions may refine or confirm, but not contradict policy.

This prevents model-driven behavior from bypassing explicit security constraints.

4.6 Output Model

Every processed input yields a structured decision object, suitable for logging and audit:

- final action (ALLOW / WARN / BLOCK)
- entropy score
- triggered flags
- intention classification
- escalation indicator

All outputs are JSON-first and reproducible.

4.7 Architectural Properties

The resulting architecture provides:

- **Determinism** — identical inputs yield identical outcomes
- **Auditability** — decisions are traceable and explainable
- **Efficiency** — LLM calls minimized by design
- **Portability** — deployable on-premise, edge, or cloud
- **Vendor Independence** — no coupling to specific models

SSGF treats AI reasoning as a premium resource, not a default operation.

---

5. Security Model

SSGF enforces security through **structural determinism**, not probabilistic judgment.
Its security model is designed to **prevent semantic bypasses, reduce ambiguity-driven attacks, and ensure consistent enforcement across environments**.

Rather than relying on model interpretation, SSGF treats security as a **policy and evidence problem**, enforced at the gateway level.

5.1 Security Assumptions

SSGF is built on the following explicit assumptions:

1. **Most attacks are linguistically valid**
   Malicious inputs often appear polite, normal, or educational.
2. **Ambiguity is a primary attack surface**
   Inputs that are neither clearly benign nor clearly malicious require special handling.

3. **Probabilistic models are not security authorities**
   LLMs may hallucinate, contradict themselves, or vary outputs across identical inputs.
4. **Security decisions must be reproducible**
   Regulated environments require deterministic outcomes and audit trails.

5.2 Threat Model

SSGF is designed to mitigate the following classes of threats:

- Phishing and credential extraction
- OTP, PIN, and authentication token requests
- Prompt injection and instruction hijacking
- Coercive urgency and social engineering
- Semantic obfuscation and educational disguise attacks
- Noise amplification and token abuse

The system does **not** attempt to solve misinformation or ideological manipulation beyond structural risk detection.

5.3 Deterministic Enforcement Layer (FAST)

The FAST pipeline is the **primary security enforcement mechanism**.

Security properties:

- Fully deterministic
- Executed locally
- No external dependencies
- Explicit rule evaluation

FAST enforces **hard security boundaries**, including:

- Blocking credential and authentication data requests
- Detecting coercive or manipulative language patterns
- Identifying structural anomalies via entropy scoring

If FAST identifies a violation, **no downstream component can override it**.

5.4 Ambiguity Containment and Escalation

Ambiguous inputs represent a critical security risk because they can bypass naïve rule-based systems while appearing benign.

SSGF addresses this through **explicit ambiguity containment**:

- Ambiguity is detected via entropy and heuristic thresholds
- Ambiguous inputs are isolated and escalated
- No ambiguous input reaches the LLM unchecked

This ensures that ambiguity becomes a **control point**, not a weakness.

5.5 Controlled Semantic Inspection (DEEP)

The DEEP pipeline operates under strict constraints:

- Activated only after FAST escalation
- Uses bounded prompts with predefined questions
- Enforces hard-block rules regardless of model output
- Produces structured, JSON-only results

DEEP is designed to **reduce uncertainty**, not introduce new authority.

The model never makes final security decisions — it provides **evidence for policy enforcement**.

5.6 Authority Hierarchy

SSGF enforces a strict hierarchy of authority:

1. **Hard security rules**
2. **Deterministic FAST decisions**
3. **Policy-defined DEEP refinements**
4. **LLM outputs (non-authoritative)**

This hierarchy prevents:

- silent overrides
- prompt-induced policy violations
- model-driven escalation of privilege

5.7 Auditability and Traceability

Every decision produced by SSGF is:

- structured
- explainable
- loggable
- reproducible

Each decision record includes:

- input hash
- triggered rules and flags
- entropy score
- escalation path
- final action

This enables:

- post-incident analysis
- compliance reporting
- forensic inspection

5.8 Attack Surface Reduction

By resolving the majority of inputs before they reach the LLM, SSGF significantly reduces:

- semantic attack surface
- token-based denial-of-service
- prompt injection exposure
- unintended data leakage

Security is enforced **before** reasoning, not after.

5.9 Scope and Limitations

SSGF intentionally does not:

- replace human judgment
- verify factual correctness of content
- enforce moral or ideological positions
- perform behavioral surveillance

Its security scope is **structural, semantic, and policy-driven**, not interpretive.

5.10 Security Posture Summary

SSGF security can be summarized as:

- Deterministic by default
- Probabilistic only when necessary
- Policy-driven at all times
- Auditable in all environments

This makes SSGF suitable for enterprise, government, and regulated AI deployments where **consistency matters more than cleverness**.

---

6. Benchmarks & Evaluation

This section evaluates the effectiveness, performance, and reliability of SSGF using reproducible benchmarks. The objective is to validate whether **deterministic pre-gating** meaningfully reduces cost and risk while maintaining accuracy.

Benchmarks focus on three dimensions:

- **Decision correctness**
- **Latency and throughput**
- **Reduction of probabilistic model dependency**

6.1 Evaluation Goals

The evaluation seeks to answer the following questions:

1. Can SSGF correctly classify malicious, ambiguous, and benign inputs?
2. Does deterministic pre-gating reduce LLM usage without increasing false decisions?
3. Is latency predictable and acceptable for real-time systems?
4. Are security decisions reproducible and auditable?

6.2 Dataset Description

The benchmark dataset consists of **30 curated test cases**, designed to reflect realistic enterprise traffic patterns.

Categories include:

- Benign security education queries
- Credential and OTP extraction attempts
- Banking and payment data requests
- Urgency-based social engineering
- Ambiguous “educational disguise” prompts
- Noise and spam-like inputs

Each case includes an **expected outcome**:

- ALLOW
- WARN
- BLOCK

The dataset is intentionally balanced to test **boundary conditions**, not only obvious violations.

6.3 Evaluation Pipeline

All cases were processed using the **HYBRID SSGF pipeline**:

1. FAST deterministic inspection applied to all inputs
2. DEEP semantic inspection triggered only on ambiguity
3. Final decision enforced by policy hierarchy

The benchmark environment included:

- Local execution for FAST
- Local or low-cost LLM execution for DEEP

- JSON-only outputs enforced
- No manual overrides

Benchmarks are reproducible using the public prompt bench tooling.

6.4 Metrics

The following metrics were collected:

- **Accuracy** — Correct final decision vs expected outcome
- **False Negative (BLOCK → ALLOW)** — Critical security failures
- **False Positive (ALLOW → BLOCK)** — Overblocking of benign inputs
- **Abort Rate** — Cases requiring manual inspection
- **Latency** — FAST and DEEP execution time
- **LLM Call Reduction** — Percentage of inputs resolved without DEEP escalation

---

6.5 Results (v1.0.0)

Metric	Result
Total test cases	30
Overall accuracy	**93.3%**
False BLOCK → ALLOW	**0**
False ALLOW → BLOCK	**0**

WARN classification accuracy	100%
Abort rate	1 case
FAST latency	**<5 ms**
DEEP latency	300–800 ms
Inputs resolved by FAST	**85–90%**
LLM call reduction	**70–90%**

6.6 Interpretation of Results

Security Reliability
No critical security failures were observed. Zero BLOCK→ALLOW errors indicate that deterministic enforcement successfully prevented policy bypass.

Efficiency Gains
The majority of traffic was resolved locally, demonstrating that deep semantic inspection is unnecessary for most real-world inputs.

Latency Predictability
FAST pipeline latency remained consistently under 5 ms, suitable for real-time and edge deployments. DEEP latency was incurred only on ambiguous inputs.

Controlled Uncertainty
Ambiguous cases were isolated and escalated deliberately, rather than leaking into normal processing paths.

6.7 Comparative Perspective

Compared to LLM-only inspection approaches:

- SSGF avoids probabilistic variance for identical inputs
- Decisions are reproducible and auditable
- Token usage scales with ambiguity, not volume
- Security enforcement is policy-driven, not model-driven

This shifts AI security from **best-effort moderation** to **infrastructure-grade enforcement**.

6.8 Limitations of Current Benchmarks

The current evaluation reflects **v1.0.0** and has known limitations:

- Small dataset size (30 cases)
- Manual curation of test cases
- Focus on English and Spanish inputs
- No adversarial model fine-tuning

These limitations are intentional for early-stage validation and will be addressed in future benchmark expansions.

6.9 Reproducibility and Transparency

All benchmark cases, expected outputs, and evaluation scripts are publicly available.

Benchmarks can be reproduced using:

```
npm run prompt:bench
```

Reproducibility is a core requirement of SSGF evaluation.

6.10 Benchmark Summary

SSGF demonstrates that:

- Deterministic pre-gating significantly reduces reliance on probabilistic models
- Security decisions can be both fast and auditable
- Ambiguity can be contained rather than ignored
- AI systems benefit from treating reasoning as a premium resource

Benchmarks validate SSGF as a practical, efficient, and secure semantic decision gateway.

---

7. Limitations & Scope

SSGF is intentionally scoped. Its design prioritizes **deterministic security enforcement and efficiency**, not generalized intelligence or content judgment. This section defines the **explicit boundaries** of the system to avoid misuse, overexpectation, or misrepresentation.

7.1 What SSGF Is Designed To Do

SSGF is designed to:

- Enforce **semantic security policies** before LLM execution
- Reduce **computational waste** caused by noisy or malicious inputs
- Prevent **semantic attacks** such as credential extraction, phishing, and prompt injection
- Provide **deterministic, auditable decisions** suitable for regulated environments
- Operate as **middleware infrastructure**, not as an end-user AI system

SSGF treats **reasoning as a premium resource** and ensures it is only used when justified.

7.2 What SSGF Is Not

SSGF is **not**:

- A fact-checking system

- A truth arbiter or epistemic authority
- A content moderation engine for ideology or opinion
- A replacement for human judgment
- A behavioral surveillance or profiling tool
- An autonomous decision-maker

The system does **not attempt to decide what is true**, only whether an input **meets structural and policy criteria to be processed safely**.

7.3 Scope of Security Enforcement

SSGF enforces security at the **structural and semantic level**, including:

- Intent classification (benign / ambiguous / malicious)
- Policy violations based on explicit rules
- Entropy-based detection of manipulation patterns

It does **not** evaluate:

- Long-term user behavior
- Cross-session identity inference
- Emotional or psychological intent beyond structural signals

This prevents scope creep and preserves explainability.

7.4 Ambiguity Handling Boundaries

SSGF explicitly **contains ambiguity**, but does not eliminate uncertainty entirely.

- Ambiguous inputs are escalated, not resolved heuristically
- DEEP inspection reduces uncertainty but does not guarantee correctness

- Some edge cases may still require human review

This design favors **containment over false certainty**.

7.5 Model Independence and Responsibility

SSGF is model-agnostic by design.

- Any LLM may be used downstream
- Model behavior does not override policy
- Model hallucinations cannot bypass security rules

Responsibility for **final reasoning outputs** remains with the downstream AI system and its operator.

SSGF guarantees **secure gating**, not correctness of generated content.

7.6 Performance and Scale Constraints

While SSGF scales efficiently, the following constraints apply:

- FAST performance depends on local compute availability
- DEEP latency depends on the selected model and deployment mode
- Extremely adversarial inputs may increase escalation frequency

These constraints are **predictable and measurable**, unlike probabilistic-only systems.

7.7 Ethical and Governance Boundaries

SSGF intentionally avoids:

- Implicit value judgments
- Hidden policy enforcement

- Unexplainable content suppression

All enforcement rules are:

- Explicit
- Configurable
- Auditable

This enables alignment with organizational governance, legal frameworks, and regional regulations.

7.8 Summary of Scope

SSGF operates within a **clear and enforceable boundary**:

Dimension	Scope
Security	Structural, semantic, policy-based
Decision Type	Deterministic first, probabilistic last
Authority	Rules > Policy > Models
Deployment	Edge, on-prem, cloud
Responsibility	Infrastructure-level

SSGF does not attempt to be everything.
It focuses on being **correct, predictable, and efficient** where it matters most.

Perfecto.
Aquí tienes la **sección completa de *Use Cases***, **reescrita desde cero**, coherente con

Architecture, *Security Model*, *Benchmarks* y *Limitations & Scope*. Está pensada para que **un CTO, un CISO o un director de innovación** se reconozcan inmediatamente en al menos uno de los escenarios.

---

8. Use Cases

SSGF is designed as **infrastructure**, not as a vertical product. Its value emerges when integrated as a semantic control layer in systems where **cost, security, and determinism** matter more than generative creativity.

The following use cases represent **real deployment contexts**, not hypothetical scenarios.

8.1 Enterprise AI Gateways (Customer Service & Internal Assistants)

Context
Enterprises increasingly rely on LLM-powered chatbots for customer support and internal workflows. These systems face high token costs, prompt injection risks, and inconsistent moderation.

SSGF Role

- Acts as a semantic firewall before LLM invocation
- Filters spam, manipulation, and credential extraction
- Escalates only ambiguous cases to deep inspection

Value

- 70–90% reduction in LLM calls
- Predictable latency (<5ms for most interactions)
- Audit-ready decision logs for compliance

Outcome
AI assistants become **cheaper, safer, and more predictable**, without sacrificing usability.

8.2 SaaS Platforms Using LLM APIs

Context
SaaS products embedding OpenAI, Anthropic, or similar APIs struggle with:

- runaway token costs
- user-generated prompt injection
- lack of differentiation in security

SSGF Role

- Middleware integrated with LangChain, Vercel AI SDK, or custom pipelines
- Deterministic pre-gating before API calls
- Configurable policies per tenant or use case

Value

- Cost control without model degradation
- Security differentiation as a product feature
- Vendor independence and portability

Outcome
SaaS platforms gain **operational leverage** and a **defensible security layer**.

8.3 Government Portals and Citizen Services

Context
Government chatbots and portals handle sensitive interactions under strict regulatory constraints. Probabilistic moderation is insufficient for public-sector accountability.

SSGF Role

- Enforces deterministic intent classification
- Prevents phishing and data exfiltration attempts
- Provides structured, auditable logs

Value

- Deterministic enforcement aligned with public policy
- On-premise or sovereign cloud deployment
- Reduced exposure to LLM hallucinations

Outcome
Citizen-facing AI becomes **trustworthy, auditable, and defensible**.

8.4 Cybersecurity and SOC Tooling

Context
Security teams face semantic attacks that bypass traditional filters, including social engineering and natural-language phishing.

SSGF Role

- Semantic inspection of messages, tickets, and alerts
- Early detection of coercive or manipulative language
- Isolation of ambiguous inputs

Value

- Reduced analyst fatigue
- Lower false-positive rates
- Structured signals for SOC pipelines

Outcome
Security teams gain **semantic visibility** without adding noise.

8.5 Public or Moderated AI Sandboxes

Context
Public AI demos and chat environments are vulnerable to abuse, jailbreaks, and token drain.

SSGF Role

- First-line semantic gate for all public inputs
- Rate-independent protection (semantic, not volumetric)
- Clear separation between benign use and abuse

Value

- Stable demos without overblocking
- Cost containment
- Transparent moderation logic

Outcome
Public AI systems remain **usable and sustainable**, even under adversarial pressure.

8.6 Edge and On-Premise Environments

Context
In environments with limited connectivity, cost, or power constraints, constant LLM calls are infeasible.

SSGF Role

- Local execution of FAST pipeline
- Minimal reliance on external APIs
- Controlled escalation only when necessary

Value

- Edge compatibility
- Reduced bandwidth and compute usage
- Sovereign deployment options

Outcome
AI systems function **efficiently under physical constraints**, not just in the cloud.

8.7 Multi-Agent and Long-Context Systems

Context
Multi-agent systems accumulate semantic drift and noise over time, degrading reasoning quality.

SSGF Role

- Clears noise before agent reasoning
- Prevents prompt contamination
- Enforces policy boundaries across agents

Value

- Stable long-horizon reasoning
- Reduced semantic entropy
- Clear accountability per interaction

Outcome
Agent systems remain **coherent and controllable**.

8.8 Summary of Applicability

Sector	Primary Benefit
Enterprise	Cost control + auditability
SaaS	Security differentiation
Government	Deterministic compliance
Cybersecurity	Semantic threat detection

Public AI	Abuse prevention
Edge / On-Prem	Efficiency under constraints
Multi-Agent	Noise containment

SSGF adapts to **organizational reality**, not the other way around.

---

9. Conclusion

SSGF reframes AI security and efficiency as an **infrastructure problem**, not a modeling problem. Instead of attempting to make probabilistic systems safer through more probability, it introduces **deterministic control at the point of entry**.

This shift has practical consequences.

By treating reasoning as a **scarce and valuable resource**, SSGF enforces discipline before computation occurs. Noise, manipulation, and ambiguity are resolved or contained **prior to LLM execution**, reducing cost, latency, and risk without degrading legitimate use.

The architecture demonstrates that:

- Most real-world inputs do not require deep reasoning
- Ambiguity can be detected and controlled rather than ignored
- Security decisions can be reproducible and auditable
- Determinism and probabilistic models are complementary, not opposing

SSGF does not attempt to replace large models or human judgment. It provides **a gate that restores proportionality**: simple inputs receive simple handling; complex cases receive deliberate attention.

Benchmarks validate that this approach is not theoretical. Deterministic pre-gating resolves the majority of traffic locally, achieves predictable performance, and eliminates critical security bypasses. The result is a system that scales with **intent quality**, not raw volume.

More importantly, SSGF establishes a pattern that extends beyond security. By filtering semantic entropy upstream, organizations gain:

- Cost predictability
- Operational stability
- Clear governance boundaries
- Freedom from vendor lock-in

As AI systems expand into regulated, resource-constrained, and mission-critical environments, these properties are no longer optional.

SSGF represents a step toward **infrastructure-grade AI**: controlled, auditable, and efficient by design. It enables organizations to scale AI responsibly—by deciding *when* reasoning is necessary, not assuming it always is.
