Geometric Security

Structural Security via Geometric Constraints

Registry: [HOWL-COMP-4-2026]

Series Path: [HOWL-COMP-1-2026] → [HOWL-COMP-2-2026] → [HOWL-COMP-3-2026] → [HOWL-COMP-4-2026] → [HOWL-COMP-5-2026]

Parent Framework: [HOWL-COMP-1-2026]

DOI: 10.5281/zenodo.zzz

Date: February 2026

Domain: Software Architecture / Systems Engineering / Real-Time Computing

Status: Architectural Blueprint for Independent Implementation

AI Usage Disclosure: Only the top metadata, figures, refs and final copyright sections were edited by the author. All paper content was LLM-generated using Anthropic's Claude 4.5 Sonnet.

---

1. Abstract

This paper describes a security model based on **Geometric Constraints**, implemented as a monotonic processing funnel within a single-address-space environment. Unlike conventional security models that rely on discretionary or mandatory access control policies applied to arbitrary execution, Geometric Security derives integrity from **anatomical limitation**. By restricting all state transitions to a closed set of six computational primitives and enforcing data I/O through a fixed-size relational Filter, the architecture eliminates broad classes of vulnerabilities including buffer overflows, privilege escalation, and unauthorized data exfiltration.

2. Technical Context: The Monotonic Funnel

In the Silo architecture, the distinction between "Kernel Space" and "User Space" is replaced by a **Relational Filter**. The system operates on a single, deterministic heartbeat (the Update Pump). All external inputs and internal state changes are processed as relational transforms of a memory-mapped database (MemDB).

The security of the system is an invariant of its geometry:

- **Vocabulary Restriction:** The execution engine only unifies logic that conforms to six defined primitives.
- **Zero-Negotiation I/O:** Network and memory operations are governed by strict schema alignment; data that does not conform to the expected geometric shape is discarded at the hardware/driver level.

3. The Six Computational Primitives

Integrity is maintained by ensuring that all system behavior is composed exclusively of these atomic operations:

1. **Finite State:** State is represented as a named identity in a pre-defined relational table. This prevents execution from entering undefined or "illegal" states.
2. **Explicit Transition:** State changes only occur across directed edges defined in the data schema. If a transition path is not explicitly asserted in the logic table, the transition is physically unreachable by the processor.
3. **Boolean Condition:** Predicate logic (SIQL) acts as a gate. Movement through the funnel requires absolute unification with local facts (Howland's Axiom).
4. **Utility Selection:** Multiplicative utility curves replace imperative branching logic. This eliminates the "if-else" jump-table surface area, as the path of execution is the result of a mathematical ranking of all valid candidates.
5. **Atomic Sequence:** Logic blocks are executed by an atomic stack machine with no support for indirect jumps or arbitrary memory pointers.
6. **Bounded Effect:** Transformations of numerical data (e.g., stats) are performed via time-bounded Envelopes. These ensure that all mutations occur within infrastructure-clamped ranges over a specific temporal curve.

4. Verification of Security Properties

4.1 Memory Integrity (Buffer Overflow Protection)

In a traditional OS, the system attempts to manage buffer sizes via software checks. In Silo, the `InboundPacket` struct is a fixed geometric requirement of the driver. A packet larger than the 2048-byte allocation is not "handled"; it is rejected by the DMA/Update Pump as a geometric mismatch. Because the system lacks a "Heap" in the traditional sense—relying instead on pre-sized Relational Columns—there is no adjacent memory for an overflow to corrupt.

4.2 Privilege Escalation (Vocabulary Limitation)

Privilege escalation requires a low-privilege process to trick the system into executing high-privilege instructions. In this model, there is no "Vocabulary" for escalation. An actor in a scene only has access to a whitelist of `[]i32` data paths. Because the "6 Primitives" are the only available operations, and none of them allow for the creation of new primitives or the widening of the path-whitelist, authority is a static property of the infrastructure.

4.3 Resilience to Exceptional States (Divide-by-Zero)

To prevent system crashes originating from mathematical outliers, the hardware spec is configured to handle arithmetic exceptions (like divide-by-zero) by returning defined

boundary values (`MAX` or `0`). This treats mathematical errors as **Data Outliers** rather than **Execution Faults**, ensuring the Monotonic Heartbeat remains uninterrupted.

5. Summary of Differentiators

Vector	Traditional Architecture	Geometric Security (Silo)
Integrity Enforcement	Policy-based (Permissions)	Anatomical (Vocabulary limit)
Error Handling	Exception/Crash	Boundary clamping (Burn dead wood)
Security Surface	Large (Arbitrary Code)	Minimal (6 Primitives)
I/O Processing	Flexible Negotiation	Strict Geometric Alignment
Observability	External (Debuggers/Logs)	Inherent (Text-Observable SIQL)

6. Conclusion

Geometric Security represents a shift from **Reactive Patching** to **Structural Invariance**. By reducing the computer to a Relational Funnel governed by fixed mathematical shapes, the architecture provides a deterministic guarantee of safety. The system does not "detect" attacks; it simply provides no vocabulary through which an attack can be expressed.

System Status: Invariant.

Unification: Bit-Perfect.

Authority: Structural.

Geometric Security: Safety by Anatomy

A Definitive Blueprint for Incorruptible Systems

---

1. Abstract

Traditional cybersecurity is a failed paradigm of "Policy over Behavior." We attempt to secure porous systems by layering rules, firewalls, and permissions on top of arbitrary code execution. This paper defines **Geometric Security**, a fundamental shift to "Safety by Anatomy." By restricting a system's computational vocabulary to a finite set of geometric primitives and forcing all I/O through a monotonic funnel, we move from a world where security is a *choice* to a world where exfiltration and escalation are *physiological impossibilities*.

2. The Concept: The Room with No Windows

To understand Geometric Security, imagine two different ways to build a secure bank vault.

The Traditional Way (Policy-Based)

You build a standard room with windows, doors, and vents. You then hire a thousand guards to watch every opening. You write a thick book of "Best Practices" for the guards to follow.

- **The Failure:** If a guard falls asleep, if a window is left unlatched, or if a guest "tricks" a guard into opening a door, the vault is compromised. This is the world of Unix, Windows, and the Web.

The Silo Way (Geometric Security)

You build a solid cube of reinforced concrete. You do not add windows. You do not add doors. Instead, you drill exactly six uniquely shaped holes in the ceiling (the **Six Primitives**). To interact with the vault, you must drop a matching shape through a hole.

- **The Success:** A "hacker" can scream at the vault, throw trash at it, or try to bribe it. But unless they provide a shape that matches the hole, nothing happens. The room physically lacks the "anatomy" for a break-in.

---

3. The Proof: The Six Primitives

Geometric Security is achieved by reducing all behavior to six invariant mathematical "shapes." In the Silo Operating System, these shapes are the only way data moves.

1. **State (The Point):** A finite, named identity. An actor is either "Alive" or "Dead." It cannot be "Half-Dead" or "Running Malicious Code."
2. **Transition (The Edge):** A fixed path between states. If no path exists from "Guest" to "Admin," the machine cannot make that jump.
3. **Condition (The Gate):** A boolean Filter. Only bit-perfect logic (SIQL/Prolog) can open the gate.
4. **Selection (The Shunt):** A mathematical utility score. Malicious input can't "hi-jack" a branch; it can only contribute a number to a formula.
5. **Sequence (The Pipe):** An atomic stack for math. No jumps, no returns, no hidden pointers.
6. **Effect (The Curve):** A time-bounded clamp. You cannot set "Health = -99999" instantly. You can only apply a curve that is clamped by the machine's physics.

---

4. Real-World Examples

Example A: The 2049-Byte Packet

- **The Attack:** A hacker sends a 2049-byte packet to a buffer designed for 2048 bytes, hoping to "overflow" into the system's memory and run a virus (RCE).
- **Geometric Response:** The Silo Master Network Scene defines the "Hole" as exactly 2048 bytes. The 2049-byte packet is a geometric violation. The **Update Pump** doesn't "truncate" it; it **Drops** it instantly. The data never enters the "Room." No negotiation, no crash. The 2047-byte packet is also rejected as malicious.

Example B: The Data Exfiltration

- **The Attack:** A malicious "Weather App" tries to read your "Bank Account" files and send them to a remote server.
- **Geometric Response:** The "Weather Scene" is assigned a `[]i32` path whitelist. It can only "see" memory addresses in its own sector. It physically lacks the **vocabulary** to ask for the Bank sector. To the Weather App, the Bank doesn't exist. There is no "Door" to knock on.

Example C: The Divide-By-Zero

- **The Attack:** Malicious math intended to crash the system.
- **Geometric Response:** Silo's hardware spec dictates that divide-by-zero returns `MAX`. The "Dead Wood" (the error) is burned, and the simulation continues. You cannot crash the machine because the machine treats errors as **Edge-Case Math**, not as **System Failures**.

---

5. The "Wall-Less" Paradox

Silo is often called "wall-less" because there are no traditional partitions between the Kernel and the App. However, this is an **Experiential** freedom, not an **Ontological** one.

The "Wall" has moved. It is no longer a fence around your data; it is the **Filter around your Logic.** You are free to move data anywhere inside the room, but you are trapped by the six shapes of the room itself.

---

6. The Conclusion for the First-Comer

Geometric Security means you stop worrying about **"Who"** is using the computer and start controlling **"What"** the computer is capable of doing.

In Silo, we do not use "Antivirus." We use **Anatomy**.

We do not use "Patches." We use **Invariants**.

We do not use "Permissions." We use **Geometry**.

"Nothing else ever happens" is not a policy; it is a physical law of the Silo universe.

The Funnel is Closed.

The Reality is Secure.

The Sovereign Author is Safe.

References

[**HOWL-COMP-4-2026**] Geometric Security. Github: https://github.com/ghowland/cks/tree/main/papers/COMP/HOWL
COMP-4-2026

[**HOWL-COMP-1-2026**] Implementing a Tall-Infra Data-Only Execution System . Github: https://github.com/ghowland/cks/tree/main/papers/COMP/HOWL-COMP-1-2026

[**HOWL-COMP-2-2026**] Tall-Infra, Data-Only Development. Github: https://github.com/ghowland/cks/tree/main/papers/
COMP-2-2026

[**HOWL-COMP-3-2026**] Silo OS. Github: https://github.com/ghowland/cks/tree/main/papers/COMP/HOWL-COMP-3-2026

[**HOWL-COMP-5-2026**] Partial Geometric Security . Github: https://github.com/ghowland/cks/tree/main/papers/COMP/
COMP-5-2026