Partial Geometric Security

Structural Hardening of Unsafe Systems via Shape-Validated Ingress

Registry: [HOWL-COMP-5-2026]

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.

Abstract

This paper introduces **Partial Geometric Security**, a practical security model that significantly reduces attack surface in unsafe languages (e.g. C/C++) by enforcing **geometric invariants at system ingress**. Unlike full Geometric Security systems, which seal both interpretation and execution, Partial Geometric Security focuses on **structural closure at I/O boundaries**. By mapping all untrusted input into fixed-shape memory structures—or rejecting it outright—this approach eliminates entire classes of exploits (buffer overflows, parser confusion, injection) while remaining compatible with existing software stacks. Partial Geometric Security is inexpensive to implement, deployable as a shim, and represents a decisive improvement over policy-based defenses.

---

1. Introduction

Modern systems security remains dominated by **policy enforcement over arbitrary execution**: firewalls, permissions, sanitizers, and runtime checks layered atop fundamentally unbounded input handling. Decades of patching have demonstrated that this paradigm fails because the system retains the *vocabulary* necessary to express exploits.

Geometric Security reframes the problem: security arises not from policy, but from **anatomy**. A system is secure when it lacks the structural capacity to express escalation or exfiltration.

This paper presents **Partial Geometric Security**, a subset of this philosophy that can be adopted immediately—even in unsafe languages—by sealing the ingress boundary.

---

2. Threat Model

Partial Geometric Security targets the following attack classes:

- Buffer overflows
- Length confusion
- Parser differentials
- Injection attacks
- Type confusion
- Memory adjacency corruption

It explicitly **does not** claim to eliminate:

- Side-channel attacks (timing, cache, speculation)
- Denial of service (availability attacks)
- Logic flaws or bad business rules

The goal is **structural reduction of exploitability**, not semantic correctness.

---

3. Core Principle: Shape Before Meaning

The fundamental rule is:

> **Untrusted input is never interpreted as logic, only as geometry.**

This is enforced by a single invariant:

> **All ingress data must either conform exactly to a predefined memory shape or be dropped.**

There is no recovery path, partial acceptance, truncation, or negotiation.

---

4. The Ingress Shim

4.1 Fixed-Shape Buffers

All ingress enters the system through **fixed-size buffers**, for example:

```
struct InboundPacket {

uint32_t src_ip;

uint32_t dst_ip;
```

```
uint16_t src_port;

uint16_t dst_port;

uint8_t  payload[2048];

uint16_t payload_length;

};
```

Rules:

- Exact byte count required
- No variable-length fields
- No pointers
- No nested allocations
- No flexible array members

4.2 Reject-on-Mismatch

Ingress handling is binary:

- ✅ Struct maps exactly → copy into RAM
- ❌ Any mismatch → drop packet

This includes:

- wrong size (2047 or 2049 bytes)
- invalid ranges
- failed checksums
- malformed headers

Dropped input never touches application memory.

---

5. Post-Ingress Rule: Structs Only

After ingress:

- Raw byte buffers are never accessed again
- Logic may only read validated struct fields
- No re-parsing
- No string interpretation

- No pointer arithmetic derived from input

This single rule collapses the exploit surface.

---

6. Security Properties Achieved

6.1 Memory Safety (Structural)

- No buffer overflow is possible
- No adjacent memory exists to corrupt
- No heap interaction occurs during ingress

6.2 Injection Resistance

- Input cannot introduce new execution paths
- No input controls instruction pointers
- No input selects function calls or jumps

6.3 Authority Containment

- Input cannot widen privileges
- Input cannot access memory it does not already reach
- Escalation requires vocabulary that does not exist

6.4 Exfiltration Resistance

- Output paths are equally shape-restricted
- No reflective serialization exists
- Data not already visible cannot be leaked

---

7. What Still Remains Vulnerable

Partial Geometric Security intentionally leaves some surfaces open:

- Side channels (timing, cache effects)
- Data poisoning (valid but malicious values)
- Semantic manipulation (game logic abuse)
- Availability attacks (flooding)

These are **design and operations problems**, not structural security failures.

Crucially, none of them permit:

- arbitrary code execution
- memory disclosure
- privilege escalation

---

8. Comparison to Traditional Security

Aspect	Traditional Systems	Partial Geometric Security
Input Handling	Parse & sanitize	Shape-validate or drop
Buffer Safety	Runtime checks	Structural impossibility
Injection	Filtered	Unexpressible
Exploit Surface	Large	Narrow
Cost	High	Minimal
Deployability	Complex	Shim-level

---

9. Implementation Cost

Partial Geometric Security requires:

- One ingress shim per I/O boundary
- Fixed struct definitions
- Strict copy-or-drop semantics

It does **not** require:

- Language changes
- Memory-safe runtimes
- Kernel modifications
- Application rewrites

This makes it viable for legacy C/C++ systems.

---

10. Relationship to Full Geometric Security

Partial Geometric Security seals **ingress only**.

Full Geometric Security additionally seals:

- interpretation
- control flow
- state transitions
- execution vocabulary

Partial implementation is therefore **not perfect**, but it provides a massive security posture improvement with near-zero friction.

---

11. Conclusion

Partial Geometric Security demonstrates that **security does not require perfect systems to be effective**. By enforcing geometric invariants at ingress, unsafe languages can eliminate entire exploit classes with minimal effort.

This approach reframes security from *reactive defense* to *structural refusal*:

> **If an attack cannot be expressed, it cannot succeed.**

Partial Geometric Security is not the end state—but it is an immediately deployable, high-leverage step toward structurally safe systems.

References

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

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