

Semantic Attack Inspection and GCRA Rate Limiting: The Shield Surface

STATE

published

CREATED

2026-09-07

LAST UPDATED

2026-09-07

INTENDED STATUS

Published

AUTHORS

Sibuna Contributors <team@sibuna.local>

DISCUSSION

Design and measured implementation of Sibuna's Shield surface: a single-pass tagged signature automaton, single-pass structural tokenizers for SQL injection, cross-site scripting, path traversal, and command injection, the byte-class pre-scan and canonicalisation gate, the GCRA rate limiter, the ban table, and incident recording.

LABELS

waf, security, rate-limiting, zero-alloc

Status of This Memo

This document is an internal Sibuna discussion record authored in Typst and tracked in git. It intentionally follows RFC-style structure so the design scope, rationale, trade-offs, and operational constraints remain explicit. Documents that still use the placeholder number XXXXX are provisional drafts. Numbered SID documents are part of the permanent project record.

Contents

Abstract	1
Introduction	1
The SID Lifecycle	1
Tooling and Numbering Workflow	1
Creating a Draft	1
Promoting a Draft	1
Listing Discussions	1
Engineering Principles and Code Standards	1
Hard Structural Limits	1
Core TigerStyle Priorities	1
Elm-Style Diagnostic and Error Reporting	1

The Three Elements of an Elm-Style Error	1
Concrete Diagnostic Example	1
Implementation Architecture	1
Recoverable Errors vs Invariant Assertions	1
Compilation Targets and Verification	1
Conclusion	1
Decision	1
Request contract	1
Authentication and work	1
Policy and memory	1
Measurement gates	1
Release toolchain	1
Native platforms and release boundaries	1
Remaining limits	1
1. Abstract	9
2. Introduction	9
3. The SID Lifecycle	9
4. Tooling and Numbering Workflow	10
4.1. Creating a Draft	10
4.2. Promoting a Draft	10
4.3. Listing Discussions	10
5. Engineering Principles and Code Standards	10
5.1. Hard Structural Limits	10
5.2. Core TigerStyle Priorities	11
6. Elm-Style Diagnostic and Error Reporting	11
6.1. The Three Elements of an Elm-Style Error	11
6.2. Concrete Diagnostic Example	11
6.3. Implementation Architecture	12
6.4. Recoverable Errors vs Invariant Assertions	12
7. Compilation Targets and Verification	12
8. Conclusion	12
9. Decision	9
10. Request contract	10
11. Authentication and work	11
12. Policy and memory	11
13. Measurement gates	11
14. Release toolchain	12
15. Native platforms and release boundaries	12
16. Remaining limits	12
Decision summary	1
Data model	1
Evaluation order	1
Built-in policy	1
Policy file	1
Dynamic policies	1
Verification	1
Milestones	1
Review corrections (2026-09-07)	1
17. Decision summary	9

18. Data model	9
19. Evaluation order	10
20. Built-in policy	10
21. Policy file	11
22. Dynamic policies	12
23. Verification	12
24. Milestones	12
25. Review corrections (2026-09-07)	12
Context and problem statement	1
Engine choice: automata and tokenizers, not regular expressions	1
Design	1
Byte-class pre-scan	1
26. Context and problem statement	9
27. Engine choice: automata and tokenizers, not regular expressions	9
28. Design	10
28.1. Byte-class pre-scan	10
Context and motivation	1
Zaxonlite foundations	1
Build and configuration	1
Schema	1
Hot-path isolation	1
Read-copy-update engine slot	1
Incident ring	1
Storage thread tick	1
Campaign clustering	1
Forensics	1
Product boundary	1
Delivery record	1
Verification	1
Open items	1
Review corrections (2026-09-07)	1
29. Context and motivation	9
30. Zaxonlite foundations	10
31. Build and configuration	10
32. Schema	10
33. Hot-path isolation	11
33.1. Read-copy-update engine slot	11
33.2. Incident ring	11
33.3. Storage thread tick	11
34. Campaign clustering	12
35. Forensics	12
36. Product boundary	12
37. Delivery record	12
38. Verification	12
39. Open items	13
40. Review corrections (2026-09-07)	13
Abstract	1
Introduction and Adversary Model	1
Setting	1

Axioms	1
Cost calculus	1
Part A: The Hashcash Tier	1
Part B: Proof of Sequential Work	1
Why sequential work	1
The construction	1
Theorems	1
Why not the alternatives	1
Implementation and calibration	1
Part C: Stateless Challenges and Symmetric Tokens	1
Pseudorandom functions and message authentication	1
Stateless challenge identifiers	1
Session tokens	1
Part D: Rate Limiting by the Generic Cell Rate Algorithm	1
Part E: Robin Hood Hashing for the Spent Set	1
Part F: Linear-Time Inspection	1
Multi-pattern matching	1
The semantic layer	1
Vectorised matchers	1
Part G: Read-Copy-Update Engine Slots	1
Part H: Load-Adaptive Difficulty	1
Part I: Campaign Clustering by Feature Hashing	1
Verification and Benchmark Gates	1
Security Considerations	1
Open Problems	1
References	1
41. Abstract	9
42. Introduction and Adversary Model	10
42.1. Setting	10
42.2. Axioms	10
42.3. Cost calculus	11
43. Part A: The Hashcash Tier	11
44. Part B: Proof of Sequential Work	12
44.1. Why sequential work	12
44.2. The construction	12
44.3. Theorems	13
44.4. Why not the alternatives	14
44.5. Implementation and calibration	15
45. Part C: Stateless Challenges and Symmetric Tokens	15
45.1. Pseudorandom functions and message authentication	15
45.2. Stateless challenge identifiers	16
45.3. Session tokens	17
46. Part D: Rate Limiting by the Generic Cell Rate Algorithm	17
47. Part E: Robin Hood Hashing for the Spent Set	18
48. Part F: Linear-Time Inspection	19
48.1. Multi-pattern matching	19
48.2. The semantic layer	20
48.3. Vectorised matchers	20
49. Part G: Read-Copy-Update Engine Slots	21

50. Part H: Load-Adaptive Difficulty	21
51. Part I: Campaign Clustering by Feature Hashing	22
52. Verification and Benchmark Gates	22
53. Security Considerations	23
54. Open Problems	23
55. References	24
Abstract	1
Introduction and Motivation	1
Operator Workflows	1
Design Principles	1
Don't make me think	1
Fast judgement: System 1	1
Careful judgement: System 2	1
Elegance	1
Principle audit by page	1
Terminology and Scope	1
Goals and Non-Goals	1
Goals	1
Non-Goals	1
Architecture Overview	1
Modules and directories	1
Process model and the isolation contract	1
Startup from the command line	1
The Serve Kernel	1
Authentication and Authorization	1
The Telemetry Pipeline	1
The sampler	1
The incident tap	1
The challenge funnel	1
Traffic sampling and top-k rankings	1
Data-plane changes this record requests	1
Cluster aggregation	1
The Real-Time Protocol	1
Data Model	1
GeoIP	1
Live earth globe on the landing page	1
Cluster Management	1
The Interface Module	1
Pages	1
Wireframes	1
The Build Pipeline	1
Security Considerations	1
Performance Budget	1
Delivery Plan	1
Verification	1
Resolved Design Decisions	1
Operational Contracts and Trade-offs	1
Retained observations	1
Rule observations	1

Management consistency	1
Browser and peer state	1
Acceptance and Open Questions	1
Query and authentication observation bounds	1
Launch scope (2026-09-11)	1
Post-launch completion (2026-09-11)	1
Interface budget review (2026-09-11)	1
References	1
56. Abstract	9
57. Introduction and Motivation	10
58. Operator Workflows	10
59. Design Principles	11
59.1. Don't make me think	11
59.2. Fast judgement: System 1	12
59.3. Careful judgement: System 2	13
59.4. Elegance	13
59.5. Principle audit by page	13
60. Terminology and Scope	14
61. Goals and Non-Goals	15
61.1. Goals	15
61.2. Non-Goals	15
62. Architecture Overview	16
62.1. Modules and directories	16
62.2. Process model and the isolation contract	17
62.3. Startup from the command line	18
63. The Serve Kernel	19
64. Authentication and Authorization	20
65. The Telemetry Pipeline	21
65.1. The sampler	21
65.2. The incident tap	22
65.3. The challenge funnel	23
65.4. Traffic sampling and top-k rankings	23
65.5. Data-plane changes this record requests	24
65.6. Cluster aggregation	24
66. The Real-Time Protocol	24
67. Data Model	26
68. GeoIP	28
68.1. Live earth globe on the landing page	29
69. Cluster Management	30
70. The Interface Module	30
70.1. Pages	31
71. Wireframes	33
72. The Build Pipeline	43
73. Security Considerations	44
74. Performance Budget	45
75. Delivery Plan	46
76. Verification	47
77. Resolved Design Decisions	48
78. Operational Contracts and Trade-offs	49

78.1. Retained observations	49
78.2. Rule observations	49
78.3. Management consistency	50
78.4. Browser and peer state	50
79. Acceptance and Open Questions	51
79.1. Query and authentication observation bounds	51
79.2. Launch scope (2026-09-11)	51
79.3. Post-launch completion (2026-09-11)	52
79.4. Interface budget review (2026-09-11)	53
80. References	53
Abstract	1
Introduction and Motivation	1
Terminology and Scope	1
Problem Statement and Architectural Challenges	1
The Gap in the Current Codebase	1
The Asymmetric Challenges of AI Bot Identification	1
System Invariants and Design Principles	1
Detailed Design	1
1. The Multi-Tier Bot Classification Engine	1
1.1 Provider and Bot Taxonomy	1
1.2 Comprehensive AI Bot Signatures Database	1
1.3 Sub-Microsecond Radix CIDR Verification	1
1.4 Behavioral and Structural Heuristics	1
2. High-Efficiency Streaming Analytics: Algorithms, Mathematics, and Storage	
Decoupling	1
2.1 The Analytics Dilemma: Why Traditional OLAP Fails Edge WAFs	1
2.2 Axiomatic Foundations of Edge Analytics	1
2.3 Contention-Free Cache-Line Striped Accumulation	1
2.4 Bounded Stochastic Sampling and Poisson Ring Dynamics	1
2.5 Space-Saving Heavy-Hitters Algorithm for Path Scraping	1
2.6 Micro-RRD Pre-Aggregated Rollups and Zaxonlite Storage Decoupling	1
2.7 Long-Term Telemetry Egress Strategy	1
3. Operator Console Dashboard Interface	1
3.1 Operator Screen Wireframes	1
3.2 Visual Components	1
3.3 Granular Tabular Components	1
3.4 Actionable Policy Integration	1
4. Elm-Style Operator Error and Diagnostic Reporting	1
Verification and Testing Matrix	1
Security Considerations	1
Rollout and Operational Plan	1
References	1
81. Abstract	9
82. Introduction and Motivation	9
83. Terminology and Scope	10
84. Problem Statement and Architectural Challenges	11
84.1. The Gap in the Current Codebase	11
84.2. The Asymmetric Challenges of AI Bot Identification	12
85. System Invariants and Design Principles	12

86. Detailed Design	13
86.1. 1. The Multi-Tier Bot Classification Engine	13
86.1.1. 1.1 Provider and Bot Taxonomy	13
86.1.2. 1.2 Comprehensive AI Bot Signatures Database	14
86.1.3. 1.3 Sub-Microsecond Radix CIDR Verification	15
86.1.4. 1.4 Behavioral and Structural Heuristics	16
86.2. 2. High-Efficiency Streaming Analytics: Algorithms, Mathematics, and Storage Decoupling	16
86.2.1. 2.1 The Analytics Dilemma: Why Traditional OLAP Fails Edge WAFs	16
86.2.2. 2.2 Axiomatic Foundations of Edge Analytics	17
86.2.3. 2.3 Contention-Free Cache-Line Striped Accumulation	17
86.2.4. 2.4 Bounded Stochastic Sampling and Poisson Ring Dynamics	18
86.2.5. 2.5 Space-Saving Heavy-Hitters Algorithm for Path Scraping	19
86.2.6. 2.6 Micro-RRD Pre-Aggregated Rollups and Zaxonlite Storage Decoupling . . .	20
86.2.7. 2.7 Long-Term Telemetry Egress Strategy	21
86.3. 3. Operator Console Dashboard Interface	22
86.3.1. 3.1 Operator Screen Wireframes	22
86.3.2. 3.2 Visual Components	24
86.3.3. 3.3 Granular Tabular Components	25
86.3.4. 3.4 Actionable Policy Integration	25
86.4. 4. Elm-Style Operator Error and Diagnostic Reporting	26
87. Verification and Testing Matrix	26
88. Security Considerations	26
89. Rollout and Operational Plan	27
90. References	27
Abstract	1
Introduction	1
Terminology and Scope	1
Problem Statement	1
Goals and Non-Goals	1
Goals	1
Non-Goals	1
State of the Art	1
Protocol Rules	1
Detailed Design	1
The decoder	1
Admission: decoding into the head buffer	1
Forwarding	1
Cost	1
Verification and Testing	1
Security Considerations	1
Rollout and Operational Plan	1
Implementation Record (2026-10-02)	1
References	1
91. Abstract	9
92. Introduction	9
93. Terminology and Scope	10
94. Problem Statement	10
95. Goals and Non-Goals	10

95.1. Goals	10
95.2. Non-Goals	11
96. State of the Art	11
97. Protocol Rules	11
98. Detailed Design	12
98.1. The decoder	12
98.2. Admission: decoding into the head buffer	13
98.3. Forwarding	14
98.4. Cost	15
99. Verification and Testing	15
100. Security Considerations	15
101. Rollout and Operational Plan	16
102. Implementation Record (2026-10-02)	16
103. References	17

Revision note (2026-09-07)

Revised after the implementation review of 2026-09-07. The first implementation matched a flat list of substrings in every header; because /* was on that list, every browser's Accept: text/html, ..., */;q=0.8 header was classified as SQL injection and every real browser received 403. The review replaced the flat list with the tagged automaton and the structural tokenizers described here, added a test that passes a complete Chrome request untouched, and corrected the "sliding window" description of the rate limiter to what the code implements: the Generic Cell Rate Algorithm.

26. Context and problem statement

An edge proxy faces two threat classes at once: automated harvesting that looks like ordinary traffic, and application-layer exploitation (SQL injection, cross-site scripting, path traversal and local file inclusion, command injection, and request floods). Deployments typically chain a challenge proxy in front of a separate application firewall and pay for two processes, two parsers, and an inter-process hop on every request. Sibuna's **Shield** surface (- -shield, the default) performs both roles inside the same zero-allocation pass; the **Gate** surface (- -gate) switches the semantic inspection and the flood limiter off for deployments that only want proof-of-work admission.

27. Engine choice: automata and tokenizers, not regular expressions

Three families of engine were considered.

- **Regular-expression rule sets** (backtracking engines) execute hundreds of backtracking patterns per request, allocate match contexts, and are notoriously prone to false positives.
- **SIMD literal engines** (Hyperscan, Vectorscan) prefilter with shuffle-based literal matchers and verify with automata; they win on long inputs and very large literal sets, at the cost of a multi-megabyte C++ dependency, a compile-time pattern database, and no portability to the browser side.
- **Aho-Corasick automata with structural tokenizers** (the libinjection lineage) scan each field once in linear time with a fixed table and detect structure with small deterministic scanners.

Sibuna's inputs are dominated by short header fields; a User-Agent scans in 85 ns and an entire browser request is measured by the standalone policy benchmark. The third family was chosen. The measured cost on an 8 KB body is 23.7 μ s (about 2.9 ns per byte); SIMD prefiltering remains an open item for body-heavy deployments (SID 0006 records the analysis).

28. Design

28.1. Byte-class pre-scan

One pass over each field records which byte classes occur: quote characters, =, <, shell separators (`;` `|` `&` `$` `\n`), %, NUL, canonicalisation triggers (`%+trn`