Semantic Attack Inspection and GCRA Rate Limiting: The Shield Surface

STATE
published
TYPE
Architectural Specification
CREATED
2026-09-07
SUMMARY
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.

STATUS
Published
AUTHORS
Sibuna Contributors <team@sibuna.local>
UPDATED
2026-10-15

DISCUSSION
No external discussion URL assigned

LABELS
waf, security, rate-limiting, zero-alloc

Record status

The catalog defines the review process and the meaning of each state. This numbered record forms part of the catalog.

Contents

1. Revision note 1
2. Abstract 3
3. Context and problem statement 3
4. Engine choice: automata and tokenizers, not regular expressions 3
5. Design 4
6. Product boundary 5
7. Verification 5
8. Review corrections (2026-09-07) 5
9. References 5

1. Revision note

2026-10-15: Ported from the Typst original in the Zig repository. The record describes the Zig implementation as it was decided; SID 0010 maps its packages to the Rust crates of the port.

2026-10-15 (Rust port): the tagged signature automaton is built by `sibuna-patterns`, whose table holds one `u32` successor per state and byte class instead of 256 `u16` successors per state. Bytes that occur in no signature share one class, and with case folding both cases of a letter share a class, so a row is as wide as the number of distinct signature bytes plus one. A scan is still exactly one dependent load per input byte with failure

links folded in, and the state capacity is still derived from the signature lengths; the signature sets need several times less memory. Case folding is an option of the builder, which Sibuna turns on, instead of a fixed comptime table. Category tags run from 0 to 63, so that the set of categories found in a field is one 64-bit word; Sibuna's categories are below 4. Findings are unchanged: a differential test replays generated requests and payloads against the Zig library.
2026-10-15 (Rust port): the GCRA limiter (`sibuna-store`) keeps its decisions, its bound, its table layout and its probe rule. Zig's limiter already used saturating arithmetic; the port also saturates the conversion of a window in seconds to milliseconds in its convenience entry point, where Zig would have overflowed. The ban table keeps its versioned snapshots and lock-free readers. WAF denials and honeypot hits still enter the bounded incident ring; the storage layer behind it is SID 0011 instead of SID 0005.

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.

2026-10-15 (Rust port, WAF review): canonicalisation no longer stops at 8 KiB. Zig canonicalised into an 8 KiB stack buffer and inspected a longer field raw only, so a header, query string or cookie padded beyond 8 KiB hid a percent-encoded payload from the second pass. Every inspected field is now canonicalised whole: values from the request head up to the 16 KiB head bound, and body fields within the unchanged 8 KiB body prefix. Fields longer than 2 KiB use a 16 KiB buffer that each thread allocates once, so inspection still allocates nothing per request. Canonicalisation also decodes JSON string escapes (`\"`, `\\`, `\/`, `\'`, `\b`, `\f`, `\n`, `\r`, `\t` and `\uXXXX` below U+0080) and HTML character references (decimal and hexadecimal references below U+0080, and named references of ASCII punctuation and white space such as `<` and `:`), and it keeps the text of MySQL executable comments (`/*!50000select*/`), whose markers become spaces. A backslash, or an `&` together with a `;` or a `#`, now triggers the second pass. The decoded form, before whitespace folding, is searched for a line feed followed by a shell command (`%0a`), which folding used to turn into a space, and for a `javascript:` or `vbscript:` scheme split by tabs or line breaks. The structural detectors also find an event handler with white space before its `=` (`onerror =`), an event handler inside a tag of any name, tautologies with `<`, `>`, `<=`, `>=`, `<>` and `!=` (`or 2>1`), and `union` joined to `select` by parentheses (`union(select`). Body coverage changes in three ways. A body of a declared binary type is inspected as text unless its prefix looks binary, that is, unless at least one byte in sixteen is a control byte other than tab, line feed, vertical tab, form feed and carriage return. Multipart delimiters accept the transport padding of RFC 2046. A part with an empty `filename` is an ordinary field. Work stays linear in the input. The differential test lists the requests whose findings change for these reasons.
2026-10-15 (Rust port, security fix): the GCRA limiter places cells by a keyed hash (SipHash-1-3, key drawn per table from operating-system randomness) of the scope and the client, instead of Wyhash with seeds in the source, which let an attacker compute addresses sharing a victim's 16-slot probe window. Its size is configurable (`--rate-table-cells`, default 262,144 cells, 4 MiB), and a full probe window evicts the cell with the oldest TAT instead of refusing the new client; at worst the evicted client regains a full burst. The challenge budget has its own table. Measured cost: about 10 ns more per check (6.6 to 16.8 ns).
2026-10-15 (Rust port, security fix): the ban table places keys by a keyed hash drawn per table, holds `--ban-table-slots` slots (default 65,536), and never evicts a ban in effect: when every slot a key may take holds a live ban, the new ban is refused and counted (`sibuna_bans_refused_total`). A banned client therefore cannot evict its own ban by getting colliding keys banned, and floods cannot erase bans.
2026-10-15 (Rust port, security fix): limits and bans key on the client's canonical address (16-byte IPv4-mapped form), so different spellings of one address are one client; IPv6 clients are aggregated to `--ipv6-prefix` (default /64). The honeypot bans only requests whose `Sec-Fetch-Dest` is absent or `document` and whose

`Sec-Fetch-Site` is not `cross-site`; others are answered 404 and counted, so an embedded link cannot ban a page's visitors.
2026-10-15 (Rust port, performance): placement keeps its secrecy at one keyed hash per request. The daemon hashes the canonical client key once with SipHash-1-3 under a key drawn at start (`ClientHasher`); each limiter and the ban table draw a 64-bit secret when built and derive the placement key as the SplitMix64 finaliser of (hash xor table secret xor scope). Since the attacker cannot compute the keyed hash, every derived key is unpredictable, and the per-table secret makes a colliding set found in one table scatter in another. Measured: GCRA check 7.6 ns and ban lookup 2.8 ns on a precomputed hash (6.2 and 5.5 ns unkeyed), plus 8.7 ns for the one SipHash per request; the store path of a request fell from 43 to 26 ns. The limiter and spent set are allocated zeroed (an empty slot is all zero bits), so idle tables are not resident: daemon idle RSS fell from 42 MiB to 5.4 MiB.
2026-10-15 (Rust port, false positive fix): a command name counts only when it is a command. A name followed by = or by a bracketed index and = (`id=`, `type[]=`) is a shell assignment word, and in a query string it is a parameter, so `a=1&id=5`, `page=2&type=book` and `q=x;env=prod` are no longer denied as `waf:rce`. Zig denied them, which blocked ordinary requests whose parameters are named `id`, `type`, `set`, `dir`, `cat` or `env`. The lone `&` stays a separator, so `ping 127.0.0.1&id` is still denied. The index is looked at for at most 64 bytes.
2026-10-15 (Rust port, inspection modes): when some category is audited or disabled rather than enforced, each field is inspected per category, so that an audited finding cannot hide an enforced one. Zig then scanned the field with the automaton once per category. The port scans it once for the set of categories whose signatures occur in it, and then runs the tagged search only for the wanted categories among them; a category without a signature hit goes to its structural detector, as before. The findings are the same.
2026-10-15 (Rust port, correction): the section on rate limiting stated Zig's table size, 16 shards of 512 slots. It now states the port's: the 16 shards share `--rate-table-cells` cells (default 262,144, so 16,384 per shard), a value clamped to 4,096 to 16,777,216 and rounded up to a power of two. The probe window is still 16 slots. The ban table's writers are serialised by a `std::sync::Mutex` instead of a spin flag; readers still validate a versioned snapshot without taking a lock.

2. Abstract

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.

3. 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.

4. 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 μs (about 2.9 ns per byte); SIMD prefiltering remains an open item for body-heavy deployments (SID 0006 records the analysis).

5. Design

5.1. Byte-class pre-scan

One pass over each field records which byte classes occur: quote characters, `=`, `<`, shell separators (`;`, `|`, `&`, `$`, the backtick, and `\n`), `%`, NUL, canonicalisation triggers (`%`, `+`, `\t`, `\r`, `\n`, `/*`), and double spaces. Every subsequent detector consults the classes instead of rescanning, and the second (canonicalised) pass runs only when a trigger byte was seen.

5.2. Tagged signature automaton

All strong signatures of every category (traversal targets such as `../` and `/etc/passwd`, SQL constructs such as `union select` and `xp_cmdshell`, script vectors such as `<script` and `javascript:`, command vectors such as `/bin/sh` and `${jndi:`) live in one dense Aho-Corasick automaton sized from the sum of signature lengths with an 8-bit category tag per pattern. A field is scanned exactly once regardless of the number of signatures; a hit yields a `Decision` of `deny` named `waf:<category>`. The automaton is case-folded through a comptime table.

> **Rust port note (2026-10-15)**
> The Rust automaton indexes its rows by byte class rather than by byte, and its tags run from 0 to 63. A scan is still one load per byte. See the revision note.

5.3. Structural tokenizers

Strong signatures alone cannot catch `id=5' or 1=1` without also catching prose, so each category has a single-pass structural detector gated by the byte classes:

- **SQL injection**: a word scanner looks up every alphanumeric word (at most 7 bytes) in the keyword table and, for `or`/`and`, checks for a `literal = literal` tautology. A tautology is terminal. Otherwise a quote is mandatory, and a score of quote (1) + up to two keywords + comment marker (1) + `=` (1) must reach 4; `admin'--` and `admin'/*` are terminal on their own. "it's a group order from the shop" scores 3 and passes.
- **Cross-site scripting**: requires `<`; fires when a tag that can host script (`svg`, `img`, `body`, `iframe`, …) co-occurs with an `on<event>=` attribute in attribute position (preceded by whitespace, `/`, or a quote). `onboarding=1` in a query string does not count.
- **Command injection**: requires a separator; fires when `;`, `|`, `&`, `&&`, `||`, `$(`, a backtick, or a newline is followed by a known command name that is not an assignment (`id=5`) (`cat`, `wget`, `curl`, `nc`, `bash`, …). A bare `;` is punctuation.
- **Path traversal**: signatures plus the NUL byte and `%00`.

5.4. Field coverage and canonicalisation

The path, the query string, the User-Agent, every header except the structural content-negotiation set (`Accept*`, `Content-Type`, `Sec-*`, `If-*`, `Cache-Control`, …), and the first 8 KB of the body are inspected. When the pre-scan saw percent escapes, plus signs, comment openers, or collapsible whitespace, the field is canonicalised (two rounds of percent decoding, `/* */` stripping, whitespace folding, lower-casing) into an 8 KB stack buffer and inspected again, so `1%27/**/UnIoN/**/SeLeCt` and `%252e%252e/` collapse onto the raw signatures.

5.5. GCRA rate limiter

The flood limiter is the Generic Cell Rate Algorithm in its virtual-scheduling form. A client's state is one 16-byte cell (keyed hash, theoretical arrival time). With emission interval $T = \max(1, \lceil W/N \rceil)$ and burst tolerance $\tau = (N-1)T$, an arrival at t conforms iff $\mathrm{TAT} \le t + \tau$, after which $\mathrm{TAT} = \max(\mathrm{TAT}, t) + T$. The guarantee is the token-bucket bound: at most $N + \lfloor L/T \rfloor$ requests conform in any interval of length L, with no fixed-window boundary artefact and no per-request log. Cells live in 16 lock-striped shards that share `--rate-table-cells` slots (default 262,144) with a 16-slot probe window; a cell whose TAT is older than

$t - \tau$ is drained and reclaimed in place. Defaults are 100 requests per 10 s (`--rate-limit`, `--rate-window`); a rejection answers `429` with `Retry-After`.

5.6. Ban table and incidents

Honeypot hits ban the client address for `--ban-seconds` (default 3600) in an open-addressed table with versioned atomic snapshots and retry on concurrent replacement. WAF denials and honeypot hits are handed to the storage layer's incident ring when a data directory is configured (SID 0005).

6. Product boundary

Gate provides admission, policy and local flood controls. Shield adds payload inspection. Distributed edge deployment adds replicated policy and reputation to either surface. No vendor feature-parity claim is made; coverage and limitations are defined by the code and tests.

7. Verification

- Unit tests: each detector on attack and benign inputs; evasion inputs (percent-encoded, comment-obfuscated, double-encoded); the automaton and the sequential scan agree on every signature; a complete Chrome request with `Accept`, `Accept-Language`, `Sec-Ch-Ua`, cookies containing `--`, and a JSON body containing "I'd select the second option – it's cheaper" passes untouched; custom headers, query strings, and bodies carrying attacks are caught.
- Rate limiter: exact burst admission and pacing, the $N + L/T$ bound across a window boundary, client independence, and reclamation of drained cells under a flood of keys.
- End-to-end: traversal in the path and SQL injection in the query string answer `403`; the honeypot answers `403` and the address is banned for subsequent requests; a burst of 12 requests against a limit of 8 yields exactly 4 `429` responses with `Retry-After`.

8. Review corrections (2026-09-07)

A valid session never bypasses inspection or a terminal policy denial. Canonicalisation now uses its caller's buffer in place and covers the full 8 KB inspection prefix; an encoded attack after byte 2048 is a regression case. Inputs beyond the bound remain a documented coverage gap. The signature automaton's capacity is derived from pattern lengths, preserving dense lookups with a smaller footprint. GCRA uses a ceiling-rounded interval and tolerance `(rate-1)*interval`, so non-divisible windows do not over-admit. Zero rate refuses requests. Hash zero alone is remapped: forcing every hash odd previously wasted half the shards. Saturated probe windows refuse new clients instead of evicting active quota state. Ban readers validate a versioned key/expiry snapshot, retrying concurrent replacements. These synchronization operations have real costs. Current measurements are in the benchmark result files; earlier numeric examples in this record are historical and must not be used as current performance gates.

9. References

The Typst original has no bibliography. This list collects the records that the text cites.

- SID 0005, Distributed Storage Architecture: Zaxonlite Integration for Dynamic Policies, Replicated Reputation, and Incident Forensics (the incident ring that receives WAF denials and honeypot hits).
- SID 0006, Mathematical Foundations of Sibuna: Sequential Work, Symmetric Authentication, Bounded State, and Linear-Time Inspection (the analysis of SIMD prefiltering).
- SID 0010, the mapping from the Zig packages to the Rust crates of the port.

Lifecycle event

```
2026-10-15: prediscussion -> discussion. Ported from the Zig repository's
Typst record
```

Lifecycle event

> 2026-10-15: discussion -> published. State carried over from the Zig repository's registry