MCP HARDENING RECORD

SAM.gov MCP Testing Record

Entity, exclusion, opportunity, and award MCP hardened across four audit rounds plus a live-key audit

Skill
sam-gov-mcp v0.3.1
github.com/1102tools/sam-gov-mcp

Date
April 2026
1102tools.com

Executive Summary

This Model Context Protocol server exposes four SAM.gov REST APIs (Entity Management v3, Exclusions v4, Opportunities v2, Contract Awards v1) plus the PSC lookup as 15 callable tools. It was hardened across four audit rounds plus a live audit with a real SAM.gov API key. The live audit surfaced three catastrophic P1 silent-wrong-data bugs that could never have been caught with mocks, including a WAF filter that was locally rejecting McDonald's and L'Oreal when SAM.gov actually accepts them. This MCP is also where the `extra='forbid'` cross-fix pattern was invented, then back-ported to seven other MCPs in the suite. The MCP ships with 79 regression tests (73 offline plus 6 live-gated).

Metric	Value
MCP tools exposed	15
Total regression tests	79 (73 offline, 6 live-gated)
Audit rounds completed	4 plus a live-key audit round
Total items addressed	46 across multiple releases
P1 silent-wrong-data bugs (live-audit-only)	3
Current release	0.3.1
PyPI status	Published as `sam-gov-mcp`, auto-publishes via Trusted Publisher on tag push

What Was Tested

The MCP exposes 15 tools covering four SAM.gov REST APIs plus the PSC lookup.

Entity Management (v3): `lookup_entity_by_uei`, `lookup_entity_by_cage`, `search_entities`, `get_entity_reps_and_certs`, `get_entity_integrity_info`

Exclusions (v4): `check_exclusion_by_uei`, `search_exclusions`

Opportunities (v2): `search_opportunities`, `get_opportunity_description`

Contract Awards (v1): `search_contract_awards`, `lookup_award_by_piid`, `search_deleted_awards`

PSC lookup and workflow: `lookup_psc_code`, `search_psc_free_text`, `vendor_responsibility_check`

Each tool was exercised for argument validation, input sanitization, WAF filter behavior against real API responses, response-shape guarantees (especially XML-to-JSON collapse edge cases on SOAP-backed endpoints), error translation, pagination, and real-world data handling against the live production API with a real SAM.gov key.

How It Was Tested

Testing discipline

Prior unit tests in v0.2.x awaited raw coroutines and relied on mocks that guessed at SAM.gov's WAF and response shapes. The hardening program switched to invoking tools through `mcp.call_tool(name, kwargs)` the way a real MCP client does, paired with a live audit using a real SAM.gov API key. The live audit is the step that surfaced the three most dangerous bugs: mocked WAF rules do not match reality, and pydantic's silent extra-argument dropping is invisible to tests that only check the happy path.

Audit rounds

Release	Audit context	Findings class
0.2.0	Pre-session baseline hardening	Some baseline validation
0.2.1	Applied `extra='forbid'` cross-fix first time	Typo'd-parameter silent drops closed
0.3.0	Rounds 1 through 4: full audit covering WAF, response-shape, validation, integrity	28+ items including 5 response-shape crashes
0.3.1	Live audit with a real SAM.gov API key	3 P1 silent-wrong-data plus 1 P3

Live audit status

All rounds in 0.3.1 included live calls against the production SAM.gov API with a real API key. The repository includes 6 live-gated regression tests executable via `SAM_LIVE_TESTS=1 SAM_API_KEY=... pytest` covering real entity search, exclusion check, opportunity search, contract award search, vendor responsibility check, and the previously-rejected apostrophe case.

Issues Found and Fixed

Priority 1: Live-audit silent wrong data (the headliners)

Three bugs in this class, all surfaced only in the live audit with a real API key. These could never have been caught with mocks.

Issue	Fix
WAF filter was rejecting McDonald's, L'Oreal, and any apostrophe-containing company name with a local "WAF triggered" error. SAM.gov's actual API accepts all of them fine as literal search text. The MCP was guessing at WAF rules that did not exist. Even `<script>` returned 27 real entities from the API. Users could not search for any apostrophe-containing company name in 0.3.0.	WAF filter narrowed to null bytes plus tab, carriage return, and line feed only. Regression test includes "McDonald's" and "L'Oreal" as live probes.
Unknown parameter names silently dropped. `search_entities(keyword="Lockheed")` (the real param is `free_text`) silently dropped the typo'd argument and ran unfiltered, returning 736,007 entities with no indication anything was wrong. Same failure mode on `search_exclusions`, `search_opportunities`, `search_contract_awards`.	`extra='forbid'` applied to every tool's pydantic arg model. Typos now raise "Extra inputs are not permitted" before the HTTP call. Pattern cross-applied to every other MCP in the suite.
`lookup_award_by_piid` silently accepted empty PIID. The tool called the API with an empty string, received an empty result, and returned it with no warning.	Empty PIID raises a clear error with valid-PIID format examples.

Priority 1: Response-shape crashes

Five bugs in this class, all from round 4 (response-shape fuzzing):

Issue	Fix
`_normalize_awards_response`: `int(ar.get("totalRecords", 0))` crashed with `TypeError` when the API returned `totalRecords: null` (key exists with None). Same for `limit` and `offset`.	`_safe_int` helper added. Returns 0 on None or string "0". Pattern now reused across the suite.
`get_entity_reps_and_certs` (slim mode): the API returned `entityData` as a dict instead of a list (XML-to-JSON single-element collapse on the SOAP-backed endpoint). Slim-mode code iterated dict keys (strings) and called `.get()` on them.	`_as_list` normalizer wraps all collection fields. Single dicts are coerced to a length-1 list.
`vendor_responsibility_check`: `KeyError 'entityData'` when the API returned `{"totalRecords": 1}` without `entityData` (malformed partial response).	Guarded with `.get()` and a clear "partial SAM response" error.
`vendor_responsibility_check`: `excludedEntity` returned as a single dict (XML collapse). Code treated it as a list.	Same `_as_list` coercion.
`vendor_responsibility_check`: API returned `totalRecords` as string `"0"` instead of int. `if total == 0` was False (because `"0" != 0`), causing a crash on `entityData[0]`.	`_safe_int` normalizes both cases.

Priority 2: Validation gaps

Representative items from the 0.3.0 round 2 and 3 audits:

Issue	Fix
UEI and CAGE format were not validated on `check_exclusion_by_uei`, `get_entity_integrity_info`, and `search_contract_awards.awardee_uei` / `awardee_cage_code`. Bogus UEIs reached the API and wasted rate-limit tokens.	Format regex enforced: 12 alphanumeric for UEI, 5 alphanumeric for CAGE.
`search_opportunities`: 364-day `posted_from` to `posted_to` cap was not pre-checked (API has a documented hard limit).	Date span checked locally. Reversed ranges also raise actionable error.
`search_opportunities.title` / `free_text` / `legal_business_name` / `entity_name` had no length clamp. 2000+ character strings risked HTTP 414.	Capped at 500 characters with guidance.
`search_exclusions.country` lowercase ("usa") was accepted; API wants uppercase.	Normalized to uppercase.
`vendor_responsibility_check`: UEI was stripped but not format-validated.	Now stripped and format-validated.
WAF detection via substring match in the error body silently failed when SAM returned an empty error body.	WAF detection uses HTTP status code plus header inspection, not body substring.
Single quotes, angle brackets, SQL keywords, `../`, and null bytes in search parameters were not pre-rejected locally (they reached the API and got blocked remotely).	Calibrated WAF filter: null bytes plus tab, CR, LF are now the only pre-rejected characters. All other characters reach the API where SAM's actual behavior controls.
`check_exclusion_by_uei` and `get_entity_integrity_info` had no UEI validation at all.	Now format-validated.

Priority 3: Cleanup items

Ten-plus items including empty-string filters passing through to the API, business type and set-aside code fields not validated against the authoritative code sets in constants, NAICS length (6 digits) not validated with negative ints accepted, and case normalization missing on codes. All resolved. Also:

SAM's opaque "Entered search criteria is not found" 404 body is now translated into a helpful message with a PSC manual link.

Response-shape defense

The `_as_list` normalizer and `_safe_int` helper now wrap every SAM.gov response parsing path. SAM.gov's SOAP-backed Entity Management API occasionally returns single-element collapses and string-vs-int inconsistencies that previously produced type-confusion crashes. Both patterns are now reused across the other MCPs in the suite.

Test Coverage

The repo ships 79 regression tests across the test folder. All 79 pass on every release cycle.

File	Purpose	Test count
`tests/test_validation.py`	Main regression suite covering every round-1 through live-audit finding, plus 6 live-gated integration tests	79
`tests/stress_test.py`	Round 1 through 4 scenario scripts (retained for reproducibility)	N/A (scenario scripts)
`tests/stress_test_live.py`	Live-key audit scenarios including McDonald's, L'Oreal, keyword-vs-free_text typo (retained for reproducibility)	N/A (scenario script)

Regression tests invoke tools through the FastMCP registry (`mcp.call_tool`) rather than awaiting decorated coroutines directly. An autouse fixture resets `srv._client` between tests so the shared httpx client does not leak across event loops.

Release History

Version	Focus	Outcome
0.2.x	Baseline with some validation already in place	Baseline coverage
0.2.1	First cross-fix release applying initial extra validation	Baseline hardening
0.3.0	Rounds 1 through 4 full audit: 28+ items including WAF calibration, response-shape crashes, input validation	5 P1 crashes, multiple P2 validation gaps resolved
0.3.1	Live audit with a real SAM.gov API key: 3 P1 silent-wrong-data plus 1 P3 translation fix	WAF filter recalibrated against reality; `extra='forbid'` invented and back-ported to all 7 sibling MCPs

Cross-MCP Context

This MCP is one of eight servers in the 1102tools federal-contracting MCP suite (`bls-oews-mcp`, `ecfr-mcp`, `federal-register-mcp`, `gsa-calc-mcp`, `gsa-perdiem-mcp`, `regulationsgov-mcp`, `usaspending-gov-mcp`, and this one). All eight were hardened under the same playbook. This MCP is where several cross-MCP patterns originated:

- **`extra='forbid'` on every tool's pydantic arg model** was invented here. Pydantic's default `extra='ignore'` silently drops typo'd parameters, and when a search tool drops a filter argument, the tool silently returns unfiltered default results. This fix was applied to all 8 MCPs via x.y.1 patches (`ecfr-mcp` 0.2.1, `usaspending-gov-mcp` 0.2.1, `gsa-calc-mcp` 0.2.1, `gsa-perdiem-mcp` 0.2.1, `bls-oews-mcp` 0.2.1, `federal-register-mcp` 0.2.1, `regulationsgov-mcp` 0.2.0).
- **WAF filter calibration against reality** was codified here: do not assume what the API blocks, probe it. SAM.gov was rejecting apostrophes locally when the API accepts them. Later, `gsa-calc-mcp` and others had their WAF filters tested against their actual API WAFs, not against guesses.
- **The `_as_list` normalizer** for XML-to-JSON single-element dict-vs-list collapse (common on SOAP-backed endpoints) was codified here and reused across the suite.

- **The `_safe_int` helper** for fields that might come back as null, `"0"`, or non-int was codified here and reused.

What Was Not Tested

- **Rate-limit behavior at scale.** SAM.gov has documented rate limits per key tier. The MCP surfaces 429s but does not implement client-side throttling.
- **OASIS+ and other special transactional endpoints.** This MCP covers the public REST endpoints, not the specialized transactional APIs that require elevated access.
- **SAM.gov SAML and login-required features.** The MCP covers the public REST endpoints only.
- **Multi-day opportunity span edge cases near the 364-day API cap.** The local cap check is bounds-safe but upstream behavior at exactly 364 days has not been live-audited.

Verification

All testing artifacts are in the repository. The methodology and fixes are reviewable commit-by-commit in git history. The regression test suite runs via `pytest` in the repo root and can be re-executed by anyone. The live suite runs with `SAM_LIVE_TESTS=1 SAM_API_KEY=... pytest` using a free SAM.gov API key.

> **Testing Methodology**
>
> Evaluators: James Jenrette, 1102tools, with Claude Code Opus 4.7 (1M context, max effort, Claude Max 20x subscription) during the hardening playbook execution.
>
> Testing spanned four audit rounds in 0.3.0 (WAF, response-shape, validation, integrity) plus a live-key audit round in 0.3.1 that surfaced three catastrophic silent-wrong-data bugs. The live regression suite runs against the production SAM.gov API when enabled with `SAM_LIVE_TESTS=1`.
>
> Test count: 79 regression tests. Total items addressed across releases: 46. P1 silent-wrong-data bugs surfaced only in live audit: 3. Response-shape crashes found and fixed: 5. Current version: 0.3.1. PyPI: `sam-gov-mcp`.
>
> Source: github.com/1102tools/sam-gov-mcp. License: MIT.