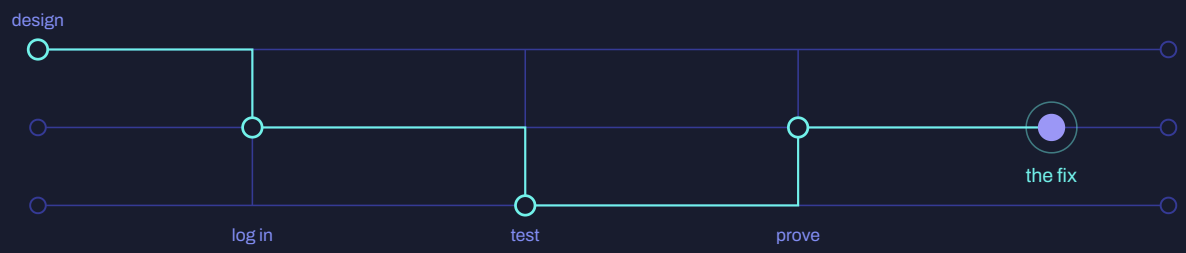
AppSentinels | Solution Brief

API Red-Teaming That Runs Itself

How AppSentinels takes on the work around a test: designing it, keeping it logged in, proving what it found and confirming the fix, so your team spends its time on the fix.



AppSentinels

Finding the flaw is half the job. Running the test is the other half.

Most API security red-teaming programs we see do not stall on detection. They stall on the work around it: writing auth scripts, keeping tokens alive, arguing over findings, and re-testing fixes by hand.

That work lands on a small number of senior security engineers. When they are busy, the complex tests (authorization across users, blind SSRF, multi-step login flows) run rarely, or not at all.

AppSentinels is built so that any AppSec or DevOps engineer can run deep, stateful tests, and every finding arrives ready for a developer to act on.

This brief walks through seven areas. Each starts with the pain we hear from AppSec and DevOps teams, then shows what the platform does about it.

124
Test cases across 27 suites, generated from what the platform learns

2,577
Tests generated automatically in 24 hours in a fintech proof of concept

4
Deployment models, from SaaS to fully air-gapped

AppSentinels

Tests that design themselves

The pain. A good BOLA test needs two real users, objects created by one, and a call sequence that reaches them from the other. A good SSRF test needs a reachable callback server. Teams wait for a senior engineer to build them.

What the platform does. It learns your APIs from traffic or a spec, then builds the tests from what it learned.

AUTHORIZATION	Uses the roles, privileged APIs and object ownership it has mapped. It picks attacker and victim users, creates genuine objects in the right order, and tries to reach them across users and roles.
AUTHENTICATION	Detects whether each API uses cookies, session IDs or JWT, and generates token tests to match: expired, replayed, tampered, and reused after logout.
OUT-OF-BAND	SSRF and other blind tests run against AppSentinels-hosted OAST servers that capture DNS and HTTP callbacks. There is nothing to stand up in your network.
TARGETING	Every parameter is classified, so tests go to the fields that can actually carry the attack.

In practice. At a US retailer, a blind SSRF sat deep inside a workflow, reachable through a single nested parameter. The platform mapped the workflow, sequenced the steps, injected at the right parameter and correlated the out-of-band callback, end to end, with no one scripting it.

Authentication that keeps up

The pain. Real logins chain several steps and often ask for a one-time code. Test environments expire tokens in minutes, so long runs fail halfway and someone has to start again.

Complex login flows. Configure the login once per role. The platform replays it whenever it needs a token:

· Multi-step sequences, values carried between steps
· Cookies, session IDs, bearer tokens and JWT
· MOTP and TOTP, so APIs behind MFA can be tested
· A test user for each role

Short token expiry. The platform tells a real failure apart from one caused by an expired token. It fetches a new token and resumes the test from where it stopped.

The run finishes without anyone watching it.

AppSentinels

Reading what the API really said

The pain. Status codes often do not tell the truth. A tool that trusts them calls a blocked attack a success, or misses a real one.

RESPONSE A

```
HTTP/1.1 200 OK
{"status": "error",
 "message": "not authorized"}
```

status code: success | effective: failed

RESPONSE B

```
HTTP/1.1 500 Internal Server Error
{"error": "field 'qty'
 must be a number"}
```

status code: server error | effective: rejected

What the platform does. It learns what success and failure look like for each API. It reads the status code and the payload together and works out the **effective API response**: did the call actually succeed? Every verdict is based on that, not on the raw code in the packet.

The result is far fewer false positives, and findings that match how your application really behaves. Where your API has its own conventions, admins can add status-code evaluation rules.

Findings no one disputes

The pain. The most expensive part of a finding is often the meeting about whether it is real.

Every finding carries its proof.

· A clear description and why it matters
· The exact requests, with payloads and headers
· The responses, and the effective response behind the verdict
· A copy-ready curl sequence to replay it

Session attacks carry the whole session.
Authorization attacks span more than one user. The evidence covers the full session for both attacker and victim: who logged in, which objects the victim created, and each call the attacker made to reach them.

In practice. When the retailer asked for validation of the blind SSRF, the finding was re-run on demand and reproduced. The auth coverage report also lists the producer and consumer API pairs tested for BOLA, so reviewers see what was covered as well as what failed.

AppSentinels

From finding to fix

The pain. A finding only has value once it reaches the right developer and the fix is confirmed. Each hand-off in between costs days.

Tickets where developers work

Findings go to Jira and other ticketing systems, from the console or the API. Each ticket carries the full evidence, so developers start without access to the security tool.

Re-test in one click

Rerun repeats a test with the same setup. **Revalidate** re-checks only earlier issues, the fastest way to confirm a fix. **Test diff** shows new tests between two runs.

Reproduce anywhere

Export tests as a Postman collection or curl commands. Developers replay the attack locally, fix it, and check the fix before they push. Each team works at its own pace.

Test what changed, inside the pipeline

The pain. A full scan on every build is too slow for a pipeline, so teams leave security red-teaming out of it.

New and modified APIs only. The platform keeps discovering APIs as they appear in traffic or updated specs. An **incremental run** tests just the new and modified APIs and their relationships. You get a quick answer on what this release changed, and the full test runs on its own schedule.

API	Start, stop and view tests and test resources; everything in the console is also in the API.
CONTROL	Role-based access on every API call, with a full audit trail.
CI/CD	A ready-made Jenkins plugin, and simple API calls for GitHub Actions, GitLab CI, Azure DevOps and other tools.
STAGES	Test groups for each stage: a smoke test on every commit, OWASP API Top 10 before release, the full library overnight or weekly.

AppSentinels

Runs where your applications run

The test client runs the tests. You choose where it lives; the test flow is the same in every model.

Client in AppSentinels cloud

WHERE THE CLIENT RUNS
AppSentinels cloud

NETWORK NEEDS
App reachable from AppSentinels cloud

GOOD FIT FOR
Fast start, internet-facing apps

Client pools in AppSentinels cloud

WHERE THE CLIENT RUNS
AppSentinels cloud, as a pool

NETWORK NEEDS
App reachable from AppSentinels cloud

GOOD FIT FOR
Many apps, parallel red-teaming

Client in your environment

WHERE THE CLIENT RUNS
Your network, on Docker or Kubernetes

NETWORK NEEDS
Outbound HTTPS (443) only

GOOD FIT FOR
Internal apps

Fully on-prem or air-gapped

WHERE THE CLIENT RUNS
Your network, with the controller

NETWORK NEEDS
No internet link

GOOD FIT FOR
Regulated and air-gapped sites

Secure connections. The test client supports mutual TLS (mTLS), so APIs that require client certificates are tested as-is.

Light footprint. 2 vCPU, 6 GB memory, 30 GB disk per client. One client tests many apps in turn; add clients or Kubernetes pools for parallel runs.

At a glance

WHAT SLOWS API SECURITY RED-TEAMING DOWN	HOW APPSENTINELS HANDLES IT
Complex auth and SSRF tests need a senior engineer	Generated from learned roles, sequences and parameters; OAST servers hosted
Multi-step logins and MFA	Multi-step login flows, common auth schemes, MOTP and TOTP
Tokens expire mid-run	Detects expiry, fetches a new token, resumes where it stopped
`200 OK` on failure, errors in the body	Effective API response from status code and payload together
Debates over whether a finding is real	Evidence on every finding; full attacker and victim sessions
Getting findings to developers	Jira and other ticketing integrations, evidence attached
Confirming fixes and reproducing locally	Rerun, revalidate, test diff; export to Postman or curl
Full scans too slow for pipelines	Incremental runs on new and modified APIs; any CI/CD tool
Network and security constraints	Four deployment models, mTLS, outbound-only option

Less time running the test. More time on the fix.

Deep API red-teaming has always been possible for teams with the time and the specialists to build it by hand. AppSentinels makes it routine: the platform designs the tests, keeps them logged in, reads the responses the way your application means them, and hands developers proof they can replay.

See these workflows on your own APIs. Contact us at contact@appsentinels.ai or visit appsentinels.ai.

More than 20 enterprise deployments and 200,000+ APIs protected. Named a Leader and Outperformer by GigaOm, and a Representative Vendor by Gartner.

About AppSentinels

AppSentinels is the Business Logic Security platform for modern applications. Built around the Business Logic Graph, it delivers continuous discovery, automated stateful red-teaming, and runtime protection across AI agents, MCP servers, APIs, and the workflows that connect them. First-generation API security tools inventoried endpoints; AppSentinels secures the logic behind them, the layer where breaches happen.