

Forum:[Hack Forums](#)

Thread: [\[Release\] AST Authorized Stress Tester source code \(PHP + MySQL + single-file SPA\)](#)

Clumsy - September 11, 2026, 11:14 pm

Hey HF,

Releasing the full source for AST (Authorized Stress Tester) — a self-hosted web platform for authorized network stress testing. Written from scratch: no frameworks, no dependencies, no build step. Just source. Nothing for sale, nothing hosted, no service.

??? WHAT'S IN THE SOURCE ???

```
Code ast/
+-- public/
| +-- index.html    # frontend SPA — single file, zero deps
| +-- index.php     # web entry point
+-- api/
| +-- index.php     # API dispatcher
| +-- auth.php      # login / register / logout / me
| +-- targets.php   # target CRUD
| +-- tests.php     # queue / list / stop
| +-- metrics.php   # time-series query
| +-- admin.php     # verification, users, bans, logs
+-- worker/run.php  # daemon worker
+-- cron/run_test.php # single-shot cron worker
+-- src/
| +-- bootstrap.php
| +-- Response.php
| +-- Csr.php
| +-- RateLimit.php
| +-- AuditLog.php
+-- config/config.php
+-- db.sql          # schema + procedures + triggers + events
+-- README.md
??? FRONTEND (public/index.html) ???
```

Single HTML file. No npm. No bundler. No CDN. No deps.
Vanilla JS SPA with client-side routing

4 themes (Midnight, Carbon, Blood, Matrix)

Ctrl/Cmd+K command palette with fuzzy search

Live canvas sparklines for latency + throughput

Terminal-style streaming log

Confetti + WebAudio sound effects (no assets)

Onboarding tour, session idle timer, keyboard nav

Every user string passes through `escapeHtml()`

??? BACKEND (api/*.php) ???

PHP 8.1+, no framework. PDO prepared statements throughout.

Session auth: `HttpOnly + Secure + SameSite=Lax`

CSRF on every mutating request

Per-IP / per-user token-bucket rate limiting

`password_hash()` / `password_verify()` with auto-logout

Ownership checks on every operation

Separate least-privilege DB users for app and worker

Code `auth.php` `login / register / logout / me`

`targets.php` `create / list / get / delete`

`tests.php` `queue / list / stop`

`metrics.php` `time-series query`

`admin.php` `verify / revoke / approve / promote / ban`

`index.php` `main dispatcher`

??? DATABASE (db.sql) ???

MySQL 8.0+ / MariaDB 10.5+, InnoDB, utf8mb4.

13 tables: `users`, `sessions`, `api_keys`, `targets`, `tests`, `test_metrics`, `logs`, `news`, `updates`, `config`, `rate_limits`, `workers`, `test_types`

4 views: `v_active_targets`, `v_running_tests`, `v_user_dashboard`, `v_latest_metrics`

7 procedures: `sp_create_test`, `sp_claim_next_test`, `sp_record_metric`, `sp_finish_test`, `sp_verify_target`, `sp_revoke_target`, `sp_purge_old_logs`

3 triggers: `trg_tests_before_insert`, `trg_targets_after_update`, `trg_users_after_update`

4 events: `retention cleanup`, `session expiry`, `rate-limit cleanup`, `stale-worker cleanup`

??? WORKER ???

Two modes: `systemd daemon` (throughput) or `cron single-shot` (low volume).

Strict argument whitelist — only 4 test types accepted

`escapeshellcmd` + `escapeshellarg` on every argument

Absolute binary path only (no PATH hijacking)

timeout wrapper enforces max duration

Re-validates target IP right before execution

Polls DB every 2s for external stop requests

SIGTERM/SIGINT handlers mark worker offline gracefully

??? ENFORCEMENT MODEL ???

Authorization checked at four independent layers:

Frontend input validation

API ownership / CSRF / rate-limit

Stored procedure re-validation in a transaction

DB trigger on the tests table

Plus a fifth: the worker re-validates the target right before spawning the child (defends against time-of-check vs time-of-use).

Every mutation is logged with user, IP, action, details, severity, timestamp.

??? SETUP ???

```
Code git clone <repo> /var/www/ast
cd /var/www/ast
mysql -u root -p < db.sql
cp config/config.example.php config/config.php
# edit DB creds + CSRF_KEY + RUNNER_BINARY
# point nginx/Apache docroot at public/
systemctl enable --now ast-worker
Cron alternative:
```

```
Code * * * * * /usr/bin/php /var/www/ast/cron/run_test.php --quiet >> /var/log/ast-worker.log 2>&1
```

Default credentials: root / root — change immediately after first login.

Full walkthrough in README.md.

??? INTENDED USE ???

Red-team engagements with signed scope

Load testing your own infrastructure

Bug bounty programs where DoS is in-scope

Security research with written authorization

You must have written permission for every target. The code enforces this at the DB layer — it can't verify your authorization document is real, that part is on you.

??? LICENSE ???

MIT. Do whatever you want with the code.

Provided as-is. Unauthorized use against systems you don't own or have written permission to test is illegal (CFAA, Computer Misuse Act, Directive 2013/40/EU, and equivalents). You are responsible for every test you launch.

??? DOWNLOAD ???

Mirror 1 (MediaFire):

<https://www.mediafire.com/file/6wicygtbqp9ezcr/AST+Authorized+Stress+Tester+By+Clumsy.zip/file>

Mirror 2 (direct):

https://download1348.mediafire.com/ieptanay2rwgHhgzM3TQqn6cu8zsLFOIYaaJ9LTaJYDTwXTrpc eo4hEgGOCyUR464fnPtzSnJvGbF7PIOXI2_J3tcHn-owwI3Dy5HZ1-5IXDNwjrDbNQQsVaqNBmwRt kyFDUbSJwlt3_wshcPmcEQEKaYGgR6Jbh6bS2nhZv0tqgSQrt/6wicygtbqp9ezcr/AST+Authorized+Stress+Tester+By+Clumsy.zip

Code SHA-256: <paste hash here>

Verify: sha256sum AST-Authorized-Stress-Tester-By-Clumsy.zip

??? CHANGELOG ???

Code v2.0 — full rewrite, layered authorization enforcement, single-file SPA
Feedback, bugs, and remakes welcome.