Security log

The security signals an Exponential site writes (security.log from the request classifier, auth.log, the CSP report log and the audit trail), how to read them, which settings tune them, and how a host forwards them to a…

Source file: `doc/eng-us/help/hosting/operations/security-log.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/hosting/operations/security-log.md

This page is for system administrators and security staff at a host who watch many Exponential sites for attacks and abuse. It answers: which logs carry security signals, what a `security.log` line means, how the classifier is tuned, what it does not do (it never blocks), and how to get the signals off the machine into your own monitoring.

The details are in Request logs, The audit trail and Security hardening: watching.

Contents

1. The security sources at a glance
2. security.log: the request classifier
3. Tuning the classifier
4. Blocking is not the classifier's job
5. auth.log and csp.log
6. The audit trail and its sinks
7. A weekly routine
8. Checklist

The security sources at a glance

Source	What it records	Where to read it
`var/log/security.log`	the request classifier's verdicts: probes, traversal, injection, attack tools, bursts of 404	**Setup > Log files**, `./console exp:logs`
the audit trail	sign-ins and failures, password changes, role and setting changes, removals, commands run; hash-chained	**Audit** tab, `./console exp:audit tail`
`var/log/auth.log`	every step of two-step sign-in (never a code or a secret)	**Setup > Log files**
`var/log/csp.log`	Content Security Policy violation reports	**Setup > Log files**
`error.log`, the web server's logs	errors with a reference id	**Setup > Log files**
request-shield's log	its decisions, when the shield is installed	its own live view

Source: Security hardening.

security.log: the request classifier

Every request is classified once, at its end, whatever its status. A request a rule recognizes is written to `security.log` instead of `404.log` or `notice.log`:

```
[ Oct 17 2026 09:00:00 ] [203.0.x.x] probe/wp-login confidence=high 404 GET /wp-login.php siteacc
ess=site namespace=/wp-login.php ua=3f2a9c1b "curl/8.5.0"
```

Field	Meaning
`probe/wp-login`	`<category>/<rule>`: built-in categories are `probe`, `traversal`, `injection`, `scanner` and `threshold`; extensions may add more
`confidence=`	`low`, `medium` or `high`
`404`	the final HTTP status (a 200 is possible: an injection attempt against a search page answers normally)
`GET /wp-login.php`	method and URI; unlike `404.log` the query values are kept as evidence, with secrets, e-mail and IP addresses masked
`namespace=`	the path below the siteaccess, so one rule sees the same path under every siteaccess

The **threshold** rule flags a client that receives more than `Max404PerMinute=30` not-found answers within `WindowSeconds=60` (`requestclassifier.ini [ThresholdSettings]`); its counter is keyed by a salted hash of the address, never the address itself.

The **error digest** (**Setup > Log files > Error digest**) groups `security.log` by `<category>/<rule>`, so a week of probes reads as a few lines such as `probe/wp-login 312`.

Tuning the classifier

`settings/requestclassifier.ini`, overridden in `settings/override/requestclassifier.ini.append.php`, a siteaccess or an extension:

Block and key	Shipped	Meaning
`[ClassifierSettings] Classification`	`enabled`	`disabled` classifies nothing
`[ClassifierSettings] LogFile`	`security.log`	the file in the log directory
`[ClassifierSettings] RuleSets[]`	`probes`, `traversal`, `injection`, `scanner`, `threshold`	leave one out to switch it off
`[ClassifierSettings] Listeners[]`	none	classes told about every verdict
`[ThresholdSettings] Max404PerMinute`	`30`	`0` switches the threshold rule off
`[RuleSet-<id>] Disabled[]`, `Excludes[]`, `Patterns[]`, `Confidence[]`	per set	switch off a rule, exclude a path, add a rule, change a confidence

A customer whose own application uses a path a rule flags (for example a legitimate `/administrator/` path) gets an `Excludes[]` entry rather than a disabled rule set. The rule names of each set are listed in the shipped file.

Blocking is not the classifier's job

"Nothing is blocked. The classifier only classifies: the request gets the answer it would get anyway." Blocking belongs to:

- **a shield in front of the kernel**, request-shield, which runs before Exponential starts and can refuse or log a request in its own log (A shield in front, Request shield and request rules);
- **the web server or a firewall** of the host;
- **a listener** registered in `[ClassifierSettings] Listeners[]` (a class implementing `expRequestVerdictListenerInterface`), the hook "where a ban list, a rate limit or a notification can be added later".

> **Note**
>
> A request the shield refuses never reaches the kernel, so it is not in `security.log`. In the shield's monitor mode the same probe can be in both logs. Do not count them twice when you report attack figures to a customer.

auth.log and csp.log

- `auth.log` records every step of two-step sign-in, never a code or a secret. A burst of failed second steps on one account is worth a look (Two-factor sign-in).
- `csp.log` holds the browser reports of the Content Security Policy, grouped in the digest by directive and blocked source. "A flood of reports is one `abuse/csp-report-flood` line in security.log a minute" (Request logs, Content Security Policy).

The audit trail and its sinks

The audit trail is a signed, tamper-evident record of who did what, on by default from 6.0.15, in `var/<site>/log/audit/` (The audit trail):

Shell

```
./console exp:audit status --allow-root-user
./console exp:audit tail --channel=access --lines=20 --allow-root-user
./console exp:audit verify --allow-root-user     # exit 1 when a chain is broken
```

A copy on the same machine can be destroyed together with it. Forward the trail with a **sink**: syslog (RFC 5424; with `Transport=local`, journald's native socket, so `journalctl -t exponential` finds the records), a signed webhook (HMAC-SHA-256, spooled with retries) or e-mail on critical events. The sinks are configured in `settings/audit.ini` (`[AuditSink_syslog]`, `[AuditSink_webhook]`, `[AuditSink_mail]`); the full reference is the audit guide. The host-side view, including who may read the trail, is on Audit trail.

> **Warning**
>
> `settings/override/audit.ini.append.php` holds the signing and pseudonym keys. Back it up with every backup and keep it as private as the database password (Backups: what to back up).

A weekly routine

From the project's own checklist (Security hardening):

1. Run the `frequent` cronjob group, so the audit's alert rules run, and check who receives alert mail (`./console exp:audit alerts recipients`).
2. Forward the audit trail to a central log or SIEM with a sink, and run `exp:audit verify` regularly.
3. Look at the error digest and `security.log` weekly; "a sudden rise of one rule is worth a look".
4. Give `audit/read` and the log viewer only to administrators.

Checklist

☐ `security.log` is listed in **Setup > Log files** and rotated (shipped: 5 archives, pruned after 90 days).

☐ Rule exclusions for customer applications are made with `Excludes[]`, documented per customer.

☐ Blocking is done in front of the kernel (shield, web server, firewall), not expected from the classifier.

☐ The audit trail is forwarded off the machine and verified on a schedule.

☐ The audit key file is in every backup.

☐ Someone reviews the digest of `security.log` weekly.

`demo/mode` **lines.** Every process that starts with the environment variable `EXP_DEMO_MODE=1` writes one line: `demo/mode on pid=<pid> sapi=<sapi> EXP_DEMO_MODE=1: ...`. On a public demo that is expected; on any other site it means demo mode reached it: alert on it. Demo mode problems.

Related pages

- Request logs
- Log viewer and rotation
- Hardening for hosts
- Compromised site (runbook)
- Security hardening
- Free documentation: this page is free to copy, translate and improve under the GNU FDL 1.3.

> **Become an Exponential Partner**
>
> Partners share their information, provide funding, contribute back and develop the products with us, and that is how Exponential grows. A partner website and dashboard are planned for mid-2027: please become a partner today to speed up our growth rate and land new features faster.
>
> Partners: start here | How to apply

Exponential Copyright © 1998-2026 Exponential Foundation and 7x
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".
This documentation is free documentation, licensed under the GNU Free Documentation License 1.3 or later:
https://www.gnu.org/licenses/fdl-1.3.html.