

ISPProtect Web Security for Shopware

ISPProtect Web Security for Shopware: Manual

Version 1.0.0

2026-11-23

Contents

1. **Introduction**
   - What the plugin does
   - What the plugin does not do
   - Free and paid parts
   - Requirements
   - Who can use it
   - How this manual is arranged
2. **Installation and first steps**
   - Install the plugin
   - What installation creates
   - Open the module
   - First steps
   - Behind a reverse proxy or CDN
   - Scheduling
3. **The module in the Administration**
   - Menu entry and pages
   - Settings on the pages
   - Dashboard
4. **Firewall**
   - How the firewall decides
   - Firewall modes
   - What a refused visitor sees
   - Bans
   - The administrator’s own address
   - Address lists
   - Custom rules
   - Trap paths and paths without request limit
   - Protection against locking yourself out
   - Firewall settings
   - If the firewall itself fails
   - Limits of the firewall
   - OWASP CRS rules
5. **File scanner**
   - What a scan covers
   - Start and stop a scan
   - Findings
   - Scanner settings
6. **Remote scan and licence**
   - Remote scan modes
   - The card Remote scan
   - What is sent and what never is
   - Limits and refusals
   - Findings of the remote scan
   - Credentials and storage
   - Console
7. **Sign-in protection**
   - Which sign-ins are counted
   - How it works

- Lifting locks and bans
- Shortened addresses
- Sign-in settings

8. **Two-factor authentication**
   - How two-factor sign-in works
   - Signing in
   - Who can open the page
   - Set up your second step
   - Recovery codes and turning off
   - Requiring it for everybody
   - Limits of the second step
9. **Security checkup**
   - Results and score
   - Checks
   - Listed files and the permission fix
10. **Event log and notifications**
    - Reading the event log
    - What is recorded
    - Event levels
    - Event log settings
    - Alert mails
    - Summary mail
    - Notification settings
11. **Settings reference**
    - Changing settings
    - Value types
    - Group Firewall
    - Group Request limit
    - Group Missing pages
    - Group Trap paths
    - Group Custom rules
    - Group Bans
    - Group Sign-in protection
    - Group Two-factor authentication
    - Group Event log
    - Group Notifications
    - Group Scanner
    - Group Response headers
    - Group Remote scan
    - Group OWASP CRS rules
12. **Command line**
    - Overview of the commands
    - Status
    - Checkup
    - Scan
    - Commands for findings
    - Events
    - Commands for address lists
    - Unlock
    - Settings
    - Maintenance

- Remote

13. Getting back in and troubleshooting

- Your own address is banned or blocked
- A user name or an address is locked
- Held on the two-factor set-up page
- Lost device or locked out of two-factor
- The firewall refuses everything
- A rule or a trap path blocks the Administration
- All visitors have the same address
- Alerts, summary mails or scans do not run
- Mails do not arrive
- Messages from the plugin
- The OWASP CRS rules refuse a normal request

14. Privacy and external services

- What the plugin stores
- External services
- Switching off

15. Removing the plugin

- Deactivate
- Uninstall
- What happens to a connected licence
- Remove the files

Introduction

ISPProtect Web Security is a plugin for Shopware 6. It protects the shop and its Administration with a request firewall, sign-in protection, two-factor sign-in for Administration users, a file scanner, a security checkup, an event log and mail alerts.

What the plugin does

- **Firewall.** It checks every request before the shop answers it. It works with address lists, request rates, trap paths and rules you write yourself. See Firewall.
- **Sign-in protection.** It counts failed sign-ins to the Administration and of storefront customers, bans addresses and locks user names. See Sign-in protection.
- **Two-factor authentication.** Administration users can add a second step with a code from an authenticator app. You can require it for every Administration user. See Two-factor authentication.
- **File scanner.** It does basic checks of the files of the shop: changes since the last scan, a few generic suspicious-code checks, unexpected files, and installed Composer packages against published security advisories. The remote scan is the thorough one. See File scanner.
- **Remote scan.** Optionally, the scanner asks a scan service about code files. See Remote scan and licence.
- **Security checkup.** It checks the server, PHP and the shop and gives a score with advice. See Security checkup.
- **Event log and notifications.** It records security-relevant events and sends alerts and a summary by e-mail. See Event log and notifications.
- **Response headers.** It can remove the PHP version header and send the HSTS and Permissions-Policy headers. See Response headers.

What the plugin does not do

- The firewall does not look into the content of requests. It has no list of attack patterns and does not detect injected code or script in a request. It decides by address, request rate, path and the rules you write.
- The scanner only reads files. It never deletes, moves, repairs or quarantines a file. The local scanner does basic checks and finds changes. It is no guarantee that a shop is clean. A file without a match of its checks, and code that is stored in the database, are not found.
- The plugin gives no guarantee that a shop is clean or safe.
- The plugin changes no Shopware core file and no Shopware table.

Free and paid parts

All functions above work without a licence, with one exception. The remote scan in the mode **Hosted scan service (licence)** needs a licence key. The mode **Local scan service (socket)** needs no site key. See Remote scan and licence.

Requirements

- Shopware 6.7.16.0 or later in the 6.7 line (the plugin requires `shopware/core` `>=6.7.16.0 <6.8.0` and declares no other requirement).
- A MySQL or MariaDB database, as Shopware uses. The plugin creates its own tables, all with the prefix `ispprotect_`.
- For the check against known vulnerabilities, the server must be able to open HTTPS connections to `packagist.org`.
- For alerts and the summary mail, the mail settings of the shop must work. See Event log and notifications.
- A running scheduler and message consumer, so that the maintenance task runs. See Scheduling.

Who can use it

The module **ISPProtect Web Security** is shown in the Administration to administrators only (users with the administrator flag). It has no right that you can give to a role. Users who are not administrators see on every page of the module except **Two-factor** the text “This area is only available to administrators.” The page **Two-factor** is for the own account of every Administration user. See Two-factor authentication.

How this manual is arranged

1. Introduction (this chapter)
2. Installation and first steps
3. The module in the Administration
4. Firewall
5. File scanner
6. Remote scan and licence
7. Sign-in protection
8. Two-factor authentication
9. Security checkup
10. Event log and notifications
11. Settings reference
12. Command line
13. Getting back in and troubleshooting
14. Privacy and external services
15. Removing the plugin

Names of buttons, fields and tabs are written in bold, exactly as the Administration shows them. Setting keys, values, commands and file names are written as `code`. Console commands are written as you run them in the main folder of the shop, for example `bin/console ispprotect:status`.

Installation and first steps

Install the plugin

The plugin is distributed as a zip file by the vendor. You can install it in the Administration or on the console.

In the Administration:

1. Open **Extensions > My extensions** and upload the zip with **Upload extension**.
2. Install the plugin, then activate it.

On the console, with the plugin folder in `custom/plugins/ISPProtectWebSecurity` (or installed with Composer as `ispprotect/shopware-web-security`):

```
bin/console plugin:refresh
bin/console plugin:install --activate ISPProtectWebSecurity
bin/console cache:clear
```

What installation creates

Installing runs a database migration that creates the tables of the plugin. All of them start with `ispprotect_`. Activating and updating the plugin check that the tables exist, create missing ones, and clear the cache pool of the plugin. Installing a new version with `plugin:update` runs the same checks.

The plugin writes its own log file, `var/log/ISPProtectWebSecurity.log` (a rotating file, 14 files are kept, level info). Problems inside the plugin go there.

The plugin fails open: when one of its parts fails, the request or sign-in that is being checked is not held up. The exceptions are named in the chapters on Sign-in protection and Two-factor authentication.

Open the module

Open **Extensions > ISPProtect Web Security** in the Administration menu. Only administrators see the entry. The module opens on the tab **Dashboard**. See The module in the Administration.

First steps

1. Open the tab **Dashboard** to see the state of the protection.
2. Check the firewall mode on the tab **Firewall**. The default is **On**.
3. If the shop is behind a reverse proxy or CDN, set `SYMFONY_TRUSTED_PROXIES` before anything else. See Behind a reverse proxy or CDN. Otherwise all visitors look like one address.
4. Enter a receiver for alerts in **Notification address** on the tab **Settings**. If it stays empty, the e-mail address from the basic information of the shop (`core.basicInformation.email`) is used.
5. Start a first scan with **Start scan** on the tab **Scanner**, or run `bin/console ispprotect:scan`. The first complete scan records the current state of the files. Later scans report differences against it.
6. Set up two-factor sign-in for your own account on the tab **Two-factor** before you require it for everybody.
7. Open the tab **Checkup** and work through the results.
8. Make sure the scheduled task runs. See Scheduling.

`bin/console ispprotect:status` shows the state on the console.

Behind a reverse proxy or CDN

The plugin uses the client address that Symfony and Shopware report. Shopware reads the address from headers such as `X-Forwarded-For` only for requests that come from a trusted proxy. If the shop is behind a reverse proxy, a load balancer or a CDN, set the environment variable `SYMFONY_TRUSTED_PROXIES` to the addresses of those proxies, and only to those. Otherwise:

- All visitors share the address of the proxy. One bad visitor can then lead to a ban of the proxy and everyone behind it, and the failed sign-ins of all users add up.
- Or, if you trust every address (`0.0.0.0/0`, `::/0` or `REMOTE_ADDR` on a public shop), a visitor can choose their own address with a fake header.

The checkup has the check “Trusted proxies match the real setup” for both cases. See Security checkup. If you want to be sure that your proxies are never refused, also add their addresses to the allow list. See Address lists.

Scheduling

The plugin registers one Shopware scheduled task, `ispprotect.maintenance`, which runs every 5 minutes. It does this work:

- It sends pending alert mails.
- Once per hour, it removes old events, expired bans and expired locks.
- It starts the scheduled scan when it is due and advances a running scan by up to 20 seconds per run.
- It renews a connected licence when that is due. See Remote scan and licence.
- It sends the summary mail when it is due.

Runs that overlap skip each other (the lock lasts 15 minutes at most). Errors are logged and do not stop the scheduler.

Shopware runs scheduled tasks only when its scheduler and a message consumer are running, as for every Shopware task. For example, run `bin/console scheduled-task:run` and `bin/console messenger:consume` under a process manager or from cron. See the Shopware documentation. Without them, alerts, the summary mail, the cleanup and the scheduled scan do not run. In that case you can run the same work by hand or from cron with:

```
bin/console ispprotect:maintenance
```

A scan that you start on the tab **Scanner** keeps going while that page stays open, even without the scheduler.

The module in the Administration

Menu entry and pages

The plugin adds the module **ISPProtect Web Security** to the menu **Extensions** of the Administration. The module shows its pages as tabs in a bar. The bar is shown to administrators only. The address of a tab is `#/ispprotect/web-security/index/<tab>`. The module opens on **Dashboard**.

Page	What it shows
Dashboard	The state of the protection at a glance.
Firewall	Add an address, the block list, the allow list and the bans, the firewall settings and the card of the OWASP CRS rules. See Firewall and OWASP CRS rules.
Sign-in protection	Locked user names, sign-in locks of addresses, firewall bans, and the sign-in settings. See Sign-in protection.
Scanner	Start and stop a scan, the remote scan card and the findings. See File scanner.
Events	The event log with filters. See Event log and notifications.
Checkup	All check results. See Security checkup.
Settings	Every setting, grouped by area. See Settings reference.
Two-factor	The second step of your own account. See Two-factor authentication.
Manual	This manual, with a link to download it as a PDF.

Users who are not administrators see no tab bar. They see the text “This area is only available to administrators.” on every page except **Two-factor**.

Settings on the pages

Each group of settings is a card with the buttons **Save** and **Reset to defaults**. The tab **Settings** shows all groups. The tabs **Firewall** and **Sign-in protection** show the groups that belong to them:

Tab	Groups
Firewall	Firewall, Bans, OWASP CRS rules
Sign-in protection	Sign-in protection, Two-factor authentication
Settings	Firewall, Request limit, Missing pages, Trap paths, Custom rules, Bans, Sign-in protection, Two-factor authentication, Event log, Notifications, Scanner, Response headers, Remote scan, OWASP CRS rules

How to save and reset is described in Settings reference.

Dashboard

The dashboard shows these cards:

- **Checkup**: the score as a number out of 100 (“/ 100”), the counts “{fail} failed, {warn} need attention, {ok} fine”, or “Nothing needs your attention.” The link **All checks** leads to the tab **Checkup**.
- **Firewall**: the firewall mode and the numbers **Active bans**, **Blocked entries**, **Allowed entries** and **Locked user names**. The link **Firewall** leads to the tab **Firewall**.
- **Open findings**: the number of open findings of the file scanner, with the counts per level. The link **Scanner** leads to the tab **Scanner**.

- **Last scan**: the status, the date and “{n} files checked” of the last scan, or “No scan has run yet.” The button **Start scan** starts a scan. While a scan runs, the link “View the running scan” leads to the tab **Scanner**.
- **Events in the last 7 days**: the counts per level (**Info**, **Notice**, **Warning**, **Alert**) and a table of the latest events with the columns **Time**, **Level**, **Event** and **Address**. The link **Events** leads to the tab **Events**.

Firewall

The firewall checks every request to the storefront, the Store API and the Administration before the shop routes it, starts a session or checks a sign-in. It sits in front of Shopware's own handling. When Shopware's HTTP cache answers a request, the firewall checks the request when the answer is sent. Page parts that are included through ESI (path `/_esi/`) are never checked.

The firewall settings are on the tab **Firewall** and on the tab **Settings**. See Firewall settings.

How the firewall decides

The firewall decides in this order:

1. Mode **Off**: the request passes.
2. An address on the allow list: the request passes.
3. An address on the block list or the ban list: the request is refused with status 403.
4. A request to a trap path: the address is banned and the request is refused with status 403.
5. Your custom rules, in order. The first matching rule with the action `block` refuses the request with status 403. A rule with the action `log` only records the request.
6. The request rate limit: an address that sends more requests than allowed in the time window gets status 429. Sign-in requests have a lower limit of their own. When an address has been refused as often as **Ban after refusals** says within 10 minutes, it is banned. The value 0 switches this ban off.

After the response, the firewall counts "not found" answers (status 404) per address. An address that causes too many of them in the time window is banned.

The firewall itself works with addresses, request rates, paths and the rules you write yourself. The search for attack patterns such as injected SQL or script code is a separate part, the OWASP CRS rules. See OWASP CRS rules.

Which requests are sign-in requests

These three paths count as sign-in requests for the lower rate limit **Sign-in requests per window**:

- `/api/oauth/token`, the token request of the Administration and of API clients
- `/account/login`, the sign-in form of the storefront
- `/store-api/account/login`, the sign-in of the Store API

Requests to `/api/oauth/token` with the grant type `refresh_token` or `client_credentials` (token refreshes of the Administration and integrations) are not counted as sign-in attempts, so they do not use up the limit.

Paths that are not counted

The request rate limit and the "not found" count skip these path prefixes: `/bundles/`, `/theme/`, `/media/`, `/thumbnail/`, `/sitemap/`, `/_esi/`, `/api/_info/` and `/api/_admin/`. You can add more in **Paths without request limit**.

Firewall modes

The setting **Firewall mode** has three values:

Value	Meaning
On	Requests that break a rule are refused. This is the default.
Log only	The firewall decides and records, but never refuses and never bans. A refusal is recorded as the event `firewall.logged`. Use it to see what the firewall would do.
Off	The firewall does nothing.

What a refused visitor sees

A refused request gets status 403 or 429. The response is marked `Cache-Control: no-store, private` and carries a `Retry-After` header.

Bans

A ban is a block of an address that ends after some time. The first ban of an address lasts as many minutes as **Ban length (minutes)** says. Each repeated ban of the same address within 30 days lasts four times as long, up to **Longest ban (minutes)**.

- IPv6 addresses are counted and banned per /64 range.
- Loopback addresses (for example `127.0.0.1`) cannot be blocked or banned.
- A ban can be lifted on the tab **Firewall** with **Remove**, on the tab **Sign-in protection** with **Lift ban**, or on the console. See Address lists.

The administrator's own address

When an administrator signs in to the Administration, that address (IPv6: its /64 range) is protected for 24 hours, as long as **Protect administrator addresses** is on. A protected address skips only the automatic measures:

- Bans do not apply to it.
- It is not checked against the trap paths, the general rate limit and the "not found" count.

The block list and your custom rules apply to it like to any other address. The sign-in request limit applies to it too, but too many sign-in requests from it do not lead to a ban. Only the allow list lets an address skip every step. Too many failed sign-ins from a protected address lock the sign-in from that address for a while. See Sign-in protection.

If you turn **Protect administrator addresses** off and save, the page warns you that a block rule, a trap path or the rate limit can now lock you out of the Administration.

Address lists

The tab **Firewall** has the card **Add an address** and the card **Entries**.

In the card **Add an address** you add an address or a range, for example `203.0.113.7` or `203.0.113.0/24`:

1. Choose the list: **Block** or **Allow**. Bans are made by the firewall itself. You can add a ban on the console with `ispprotect:ip:add`.
2. Type the address or range in **Address or range**.
3. Optionally type a **Note**.
4. Choose **Expires**: **Never**, **After 1 hour**, **After 1 day**, **After 7 days** or **After 30 days**.
5. Click **Add**.

If the block covers your own address, the page asks "This block covers your own address" and says that saving it may lock you out of the Administration. Click **Block anyway** to save it, or **Cancel**.

The card **Entries** has three lists: **Block list**, **Allow list** and **Bans**. Each entry shows **Address or range**, **Note**, **Source**, **Created**, **Expires**, **Strikes** and **Status** (**Active** or **Expired**). **Source** says what created the entry: `manual` for an entry that you added, otherwise the automatic measure (for example `rate`, `notfound`, `trap` or `login`). **Strikes** counts repeated bans of the address. The button **Remove** deletes an entry. An empty list shows "There are no entries in this list."

Changes of the lists are recorded in the event log. On the console, `ispprotect:ip:list`, `ispprotect:ip:add` and `ispprotect:ip:remove` do the same. See Command line.

Custom rules

A custom rule tests one part of the request with one comparison, then blocks or logs.

Part	Meaning
`path`	The path of the request
`query`	The query string
`user_agent`	The browser identification
`referer`	The referrer
`method`	The HTTP method
`host`	The host name

The comparisons are `equals`, `prefix`, `suffix`, `contains` and `regex`. The first four ignore upper and lower case. A regular expression is written without delimiters and matches case-insensitively. A rule value must not be empty and can be up to 500 bytes. You can define up to 100 rules.

The field **Custom rules** holds the rules as a JSON list. On the console, `rules.custom` holds the same list. Each rule has these keys:

Key	Meaning
`id`	Optional. Letters, digits, `-` and `_`, up to 32 characters.
`field`	One of the parts in the table above
`op`	One of the comparisons above
`value`	The text to compare with
`action`	`block` or `log`
`enabled`	Optional. `true` to use the rule (the default), `false` to switch it off

Example:

```
bin/console ispprotect:settings:set rules.custom '[{"id":"no-env","field":"path","op":"suffix","value":".en
v","action":"block","enabled":true}]'
```

Trap paths and paths without request limit

Trap paths and **Paths without request limit** are lists of paths, one per entry. Paths are compared in lower case, with a leading `/` and without a trailing `/`. An entry matches the whole path, or the start of the path when it ends with `*`: `/backup/*` matches `/backup` and `/backup/x`, but not `/backups`. For **Paths without request limit**, every entry matches the start of the path. An entry that consists only of `/`, `*` and spaces (it would cover the whole site) is refused.

Protection against locking yourself out

After you save the settings on the tab **Firewall**, the page checks your own request and warns about these cases:

- “Protection of your own address is off. A block rule, a trap path or the rate limit can now lock you out of the Administration.”
- A trap path covers an Administration path: “The trap path {entry} covers the Administration path {path}. Opening that path would ban your address.” The paths checked are `/admin` and `/api/oauth/token`.
- A block rule matches an Administration path for your own request: “The block rule {rule} matches the Administration path {path} for your own request. It can lock you out.”

These are warnings. The settings are saved anyway. If you are locked out, see Getting back in and troubleshooting.

Firewall settings

The tab **Firewall** shows the groups **Firewall** and **Bans** of the settings. The groups **Request limit**, **Missing pages**, **Trap paths** and **Custom rules** are on the tab **Settings**. All keys, defaults and ranges are in Settings reference.

If the firewall itself fails

If the database or the cache fails, the firewall lets the request through and writes the problem to the log file of the plugin. It does not lock everybody out because of its own fault.

Limits of the firewall

- The firewall only knows the address that Shopware reports as the client address. Behind a reverse proxy or CDN, set `SYMFONY_TRUSTED_PROXIES`. See Behind a reverse proxy or CDN.
- Request counters are approximate when many requests arrive at the same time.
- Cached pages that a proxy in front of Shopware answers never reach the shop. To stop an address from fetching them, you need a rule in your proxy or web server.

OWASP CRS rules

The OWASP Core Rule Set (CRS) is a widely used, openly maintained set of rules that recognises typical attacks on web applications, for example SQL injection, cross-site scripting, attempts to read files or to run commands, and requests of scanners and attack tools. In the Administration they are a card, **OWASP CRS rules**, on the tab **Firewall**. The CRS rules are the part of the firewall that looks for attack patterns; the plugin adds only two rules of its own, for requests the CRS rules do not recognise (see ISPProtect rules). The rules run with an engine of the plugin written in PHP, so no extra program or server module is needed. They are off until you install them.

The card **OWASP CRS rules** shows the installed version, the time of the installation, the active rules, the rules that are not supported, whether your own rules are loaded and the disabled rules. It has the buttons **Install CRS 4.25.2** (or **Update to CRS 4.25.2**), **Remove rules** and **Show licence**, and the field **Disable a rule**. Below it are the settings of the CRS rules (see Settings reference).

Installing the rules

1. Open the tab **Firewall** and click **Install CRS 4.25.2** in the card **OWASP CRS rules**.
2. The plugin downloads one fixed version of the rules from GitHub (the CRS project, version 4.25.2 in this release of the plugin) over `https`. Your server needs outgoing `https` access to `github.com` and to the download host that GitHub redirects to, and PHP needs `allow_url_fopen` and OpenSSL. The plugin compares the download with a checksum (SHA-256) that is built into the plugin and refuses anything else. A newer version of the rules arrives only with a newer version of the plugin; nothing else is ever downloaded or run.
3. The first time you install, the plugin sets the mode (`waf.mode`) to **Log only**.

The page then shows the version, the checksum, the time of the installation, the number of rules in use and the numbers of the rules that are not supported (see What the CRS rules do not cover). After a plugin update, the page can say that a new version of the rules is available. After a plugin update the card says when a new version of the rules is available, and **Update to CRS** installs it. **Remove rules** deletes them again, after you confirm. The rules are stored in `var/ispprotect/waf`, outside the public folder.

Start with logging

Leave the mode on **Log only** first. In this mode a request that matches is only written to the event log and is never refused. Look at the entries for a few days, in which all kinds of pages, forms and the Administration and the storefront are used, disable the rules that report normal requests of your site (see below) and only then set the mode to **Block**.

Value	Effect
Off (`off`)	Requests are not checked with the CRS rules. This is the state until you install the rules.
Log only (`log`)	A request that reaches the threshold is recorded as **Request logged by the firewall** and goes through.
Block (`block`)	A request that reaches the threshold is refused with status 403 and recorded as **Request refused by the firewall**.

The CRS check has its own mode (`waf.mode`) and does not depend on the firewall mode. It runs after the address lists, bans and limits of the firewall: a request from an address on the allow list is never checked with the CRS rules. Everybody else is checked, signed-in administrators too.

Reading an entry

A request that reached the threshold is recorded in the event log with the event **Request logged by the firewall** or **Request refused by the firewall**. The details hold:

- the numbers and the messages of the rules that matched, for example `942190` and the name of the attack it suspects,
- for each rule the name of the part of the request that matched (for example `ARGS:comment` , `REQUEST_COOKIES:name` or `REQUEST_URI`) and a short excerpt of the value with the sensitive part masked,
- the total score, the threshold, the paranoia level, the method, the path and the answer (never the query string, the headers or the body as such).

A request counts as an attack when the scores of all rules that matched add up to the **Anomaly threshold**. Every rule adds a score by its severity: a critical match adds 5, an error 4, a warning 3 and a notice 2. With the default threshold 5, one critical match is enough; with a higher value, several rules have to match.

If the CRS rules themselves have a problem (for example the compiled rules cannot be read), the request goes through and the event **Problem with the CRS rules** is recorded, at most once an hour per kind of problem.

A rule reports a normal request

If a rule reports requests that are normal for your site (a false positive), disable that rule:

1. Open the entry in the event log and note the rule number and the message.
2. In the card **OWASP CRS rules**, enter the number in the field **Disable a rule** and click **Disable**.

You can also enter rule numbers by hand under **Disabled rules** (`waf.disabled_rules`), one per line. A disabled rule no longer runs for any request, also not for the ones it was right about; disable only the rule that is wrong, not a whole group. If you disabled a rule by mistake, remove its number from the list.

Paranoia level and threshold

The **Paranoia level** (`waf.paranoia` , 1 to 4, default 1) decides how many rules run. Level 1 contains the rules with very few false positives and is recommended for most sites. Each higher level adds stricter rules that find more attacks and also report more normal requests, mainly in forms with free text, in editors and in page content. Raise the level only after the entries at the level before are clean, one step at a time, and start with logging again.

The **Anomaly threshold** (`waf.threshold` , 1 to 100, default 5) is the score at which a request counts as an attack. Raise it only if you accept that single rule matches no longer count.

Requests that cannot be fully inspected

Some requests cannot be checked completely: the body is larger than the engine reads, the request has more fields than it looks at, the body cannot be read, or the check takes too long. An attacker could use such a request to get past the rules, so by default the plugin treats it like an attack: in mode **Block** it is refused, in mode **Log only** it is recorded. The entry shows one of these numbers:

Number	Meaning	Answer when refused
`200002`	The body could not be read (for example invalid JSON or XML).	400

Number	Meaning	Answer when refused
`200004`	The body is a form upload without its closing boundary.	400
`200005`	A pattern of a rule ran into the regular expression limits of PHP.	403
`200007`	More fields, headers, cookies or parts than the engine inspects (1000).	403
`200100`	The time limit for the request was used up before all rules ran.	403
`200101`	The body is larger than **Inspected body size**.	413
`200102`	A value is longer than the engine inspects (65536 bytes).	403

You can disable these numbers like any other rule. Two settings are for the two usual cases:

- **Inspected body size** (`waf.body_limit` , 1024 to 1048576 bytes, default 131072 = 128 KiB). A larger request cannot be fully inspected. A save with a large amount of content can be bigger than that and is then answered with status 413 and the number `200101` . If large saves fail in mode **Block**, raise the value, for example to 524288, or let such requests pass (see below).
- **Time limit per request** (`waf.time_budget_ms` , 5 to 10000 milliseconds, default 200). The time the rules may take for one request. A normal page or save needs 2 to 10 milliseconds, a save of 42 KB of content about 100 milliseconds on the server where it was measured. A request near the body limit at paranoia level 2 can reach the default. If requests stop with `200100` , raise the value. Keep it as low as your site allows: the limit is also what an attacker can make your server spend on one request.
- **Program code in a form field.** Program code pasted into a form field (indented PHP or Python, about 20 to 40 KB) can use up the default time limit of 200 milliseconds, and the request is refused with `200100` in block mode. For a site where program code is pasted into forms, raise the time limit, switch off `200100` , or let requests pass that cannot be fully inspected. Normal text, HTML and the editor saves that the exclusions cover are not affected.
- **Pattern limits.** Rarely, a long field with program code makes a pattern of a CRS rule exceed the regular expression limits of PHP, and the request is refused with `200005` . The answers are the same: switch off `200005` or let requests pass that cannot be fully inspected.

Let requests pass that cannot be fully inspected (`waf.pass_uninspected` , default off) lets such requests through (they are still recorded). Big saves keep working then, but a crafted request that is too large or too slow is no longer checked completely. Use it only when you cannot raise the limits, or disable the single number that causes the trouble, for example `200100` .

An XML request body that declares an entity or names an external DTD is refused with `200002` (HTTP 400) in mode **Block** even when **Let requests pass that cannot be fully inspected** is on. Downloaded CRS exclusions cannot remove `200002` or the missing-multipart-boundary rule `200004` ; you can still disable either number explicitly in the settings.

Exclusions for Shopware

The plugin comes with its own exclusion rules in `rules/shopware-exclusions.conf` . They are loaded before the CRS rules and have the numbers 10000 to 10999. Each one is limited to a route and to named fields, and none of them switches off a whole kind of attack. They remove false positives that normal work in the Administration and the storefront caused at paranoia levels 1 and 2:

- the cookie `bearerAuth` of the Administration, as long as it has the form of the token cookie,
- the fields `redirectParameters` , `forwardParameters` and `errorParameters` of the storefront forms, as long as they hold a plain JSON object,
- the address fields of registration, guest checkout and the address book, and the message of the contact form and of product reviews (only the SQL injection rules skip these),
- the mail templates, the mail headers and footers (Twig code),
- the content of Shopping Experiences (text elements with HTML, embeds),
- the font stacks of the theme configuration, the media upload (content type of the file, address of an upload by URL) and the rule builder,
- the text settings of the pages of ISPProtect Web Security, on their save route only.

The CRS rules run before Shopware checks the token or the session, so an unauthenticated request can name these routes too. Shopware itself refuses such a request, so the skipped fields are only processed for signed-in users. All other fields, headers and cookies are still checked.

> Paths are matched at their end, so a shop in a subfolder and sales channel domains with a path keep their exclusions. Two rules, for mail templates and for the settings of the plugin, are anchored at the start of the path. A shop in a subfolder does not get these two. You then see more entries, but the protection is not weaker.

ISPProtect rules

The plugin ships two rules of its own in `rules/ispprotect-shopware.conf`, with the numbers 12000 to 12999. They report requests that the CRS rules do not recognise at paranoia level 1. They run with the CRS rules, so only while the CRS rules are installed and the mode is **Log only** or **Block**: after the CRS rules, before the rules of your own file and before the decision on the threshold. Each match adds a critical score (5), so with the default threshold one match is enough. The card **OWASP CRS rules** shows how many of them are in use in the line **ISPProtect rules in use**.

Rule	Reports
`12001`	The version control folders of Bazaar (`.bzr/`) and Darcs (`_darcs/`) and the files `CVS/Entries`, `CVS/Root` and `CVS/Repository`. The CRS rules report `.git/`, `.svn/`, `.hg/` and `.env` themselves.
`12002`	Copies of PHP and YAML files (also `.phtml`, `.phar`, `.inc`) with the ending `.orig` or `.rej`, a number (`shopware.yaml.1`) or a numbered `.bak`, `.old` or `.save` (`index.php.bak1`), and the Emacs files `#name#` and `.#name`. The CRS rules report `.bak`, `.old`, `.save`, `.swp`, `.pem` and `~` themselves.

When the rules help:

- The rules see only requests that reach Shopware's front controller (`public/index.php`). With the `.htaccess` file of Shopware, a request for a file that does not exist in the public folder goes to Shopware, which answers "not found". The rules refuse such a request and record it with their number, so a search for leftover files shows in the log as what it is.
- A file that exists in the public folder is sent by the web server without Shopware, and a PHP script that exists is run by PHP without Shopware. The rules never see these requests and cannot stop them, also not a real copy `index.php.orig`.
- Requests for the folders `media`, `thumbnail`, `theme`, `bundles`, `css`, `fonts` and `js` never reach Shopware: its `.htaccess` file and the nginx configuration of the Shopware documentation answer them without Shopware. For this reason the plugin has no rule for PHP scripts in the media folders; only the server can refuse them.

The real fix is the configuration of the web server and a clean public folder: the server refuses version control folders and copies of files and runs no PHP in `media` and `thumbnail`. The check **No backups or debugging files in the public folder** of the Security checkup lists backups and debugging files it finds in the public folder.

The rules are disabled like CRS rules: one rule by its number (the field **Disable a rule**, or **Disabled rules** in the settings), both with the line `SecRuleRemoveByTag ispprotect` in your own rules file (see below), or with the tag `ispprotect/source-disclosure`. A plugin update can bring changed rules; the plugin compiles the rules again on the next request.

Your own CRS rules file

You can add your own rules in SecLang, the language of the CRS, in the file `var/ispprotect/rules/own.conf` of your installation. The plugin never creates the file. The page shows whether it is in use. Rules in it:

- need a number from 11000 to 11999; a rule with another number is skipped,
- run after the CRS rules and before the decision on the threshold, so a rule can add to the score,
- use the subset of SecLang that the engine supports: `SecRule`, `SecAction`, `SecMarker`, `SecRuleRemoveById`, `SecRuleRemoveByTag`, `SecRuleUpdateTargetById` and `SecRuleUpdateTargetByTag`, and chains. Rules that need something else are skipped and counted as not supported.

An example that adds a critical score for a PHP tag in a parameter:

```
SecRule ARGS "@rx (?i)<\?php" \
    "id:11001,phase:2,block,t:none,t:urlDecodeUni,\
    msg:'PHP open tag in a parameter',severity:'CRITICAL',\
    setvar:'tx.inbound_anomaly_score_pl1=+%{tx.critical_anomaly_score}'"
```

The plugin notices a changed file on the next request and compiles the rules again. If they cannot be compiled, the CRS check does not run, requests go through and the problem is recorded as **Problem with the CRS rules**.

What the CRS rules do not cover

- The plugin checks requests only. Rules for responses (the CRS “response” phases) are not run.
- Rules that need the libinjection library are not run, because PHP has no such library: `941100` and `941101` (XSS) and `942100` and `942101` (SQL injection). The other CRS rules for these attacks do run. The page lists the rules that are not supported.
- The part headers of a form upload (`MULTIPART_PART_HEADERS`) are not available, because PHP reads the upload before the plugin sees it. The file names and sizes are checked.
- The CRS check only looks at the request. It does not scan files and does not replace updates.

Cost of the check

On a normal request the check costs 2 to 10 milliseconds. A save with 42 KB of content took about 100 milliseconds. At paranoia level 2 and above, and for large bodies, the check takes longer; the time limit stops it at the value you set. Program code pasted into a form field (about 20 to 40 KB of indented PHP or Python) can use up the default time limit of 200 milliseconds; normal text, HTML and the editor saves that the exclusions cover do not.

If the CRS rules lock you out

Switch the CRS check off on the console:

```
bin/console ispprotect:settings:set waf.mode off
```

Then sign in again, find the rule that caused the trouble in the event log, disable it and set the mode back to **Log only**. See also Getting back in and troubleshooting.

Licence and attribution

Uses the OWASP® Core Rule Set (CRS) 4.25.2, © Trustwave Holdings, Inc. and the OWASP CRS project contributors, licensed under the Apache License 2.0. The rules are downloaded directly from the CRS project on GitHub. OWASP® is a registered trademark of the OWASP Foundation; ISPProtect is not affiliated with or endorsed by OWASP.

The page shows the same text. The licence text of the CRS is part of the download; the button **Show licence** shows it.

File scanner

The scanner does basic checks of the files of the shop: changes, PHP in upload folders, leftovers and a few generic suspicious-code checks. The remote scan is the thorough one. The scanner only reads. It never deletes, moves, repairs or quarantines a file. You decide what to do with each finding. The scanner is on the tab **Scanner** and on the console. See Scanner settings.

What a scan covers

A scan has these steps, which the card **Scan progress** shows as phases: “Reading files”, “Looking for removed files”, “Checking files with the scan service” (only when the remote scan is on), “Checking the public directory”, “Checking known vulnerabilities” and “Finishing”.

- **Changed files.** The scanner records a baseline of the files in the main folder of the shop (the project root): their size and checksum. Later scans compare against it. This is on when **Detect changed files** is on.
- **Suspicious content.** A few basic, generic checks look in the code files for common signs of malware. They do not recognise known malware; use the remote scan for that. This is on when **Search for malicious code** is on. It reads the content of the folders `public/theme`, `public/bundles`, `public/media`, `public/thumbnail`, `public/sitemap` and `files`, and of the code files of the project root. Template files (`twig`, `tpl`, `stub`, `dist`, `template`) count as code files too.
- **Unexpected uploads.** PHP files in the upload folders `public/media`, `public/thumbnail` and `files`.
- **Leftover files.** Files in the public folder that do not belong there.
- **Known vulnerabilities.** The installed Composer packages are checked against the published security advisories of Packagist. This is on when **Check for known vulnerabilities** is on. See Privacy and external services.
- **Remote scan.** Optionally, a scan service checks the code files. See Remote scan and licence.

The scanner always skips the folder `var`, the folders `node_modules` and `.git`, and the paths in **Excluded paths**. The plugin does not look into the database.

The first scan

The first complete scan with **Detect changed files** on records the current state of the files. It reports no changed or new files. Later scans report the differences. Files of Composer packages that you update are not reported as changed.

Excluded paths

Excluded paths takes paths relative to the main folder of the shop, one per entry, for example `public/uploads-test`. An entry covers the folder and everything below it. Leading `./` and `/` are ignored. **Largest scanned file (KB)** sets the size above which a file is not read. The default is 5120.

Start and stop a scan

On the tab **Scanner**, the card **Scan** has the button **Start scan**. While a scan runs, the button is **Stop scan** and the card shows the status (**Running**, **Complete**, **Failed** or **Stopped**) and the counts **Files**, **Size**, **Scripts** and **Findings**. A stopped scan shows “The scan was stopped.”

A scan runs in steps of up to 200 files. While the tab **Scanner** is open, the page advances the scan. When the page is closed, the scheduled task advances it by up to 20 seconds per run. See Scheduling. A scan whose position has not moved for 6 hours counts as dead.

On the console, `bin/console ispprotect:scan` starts a scan and shows its progress. See Command line.

Scheduled scan

With **File scan schedule** set to **Daily** or **Weekly**, the scheduled task starts a scan when it is due. **File scan hour** is the hour from which a scan may start. The hour counts in UTC. A daily scan starts at most once per calendar day (UTC). A weekly scan starts when no scan started in the last 7 days. The value **Off** switches the schedule off.

Findings

The card **Findings** lists the findings. You can filter by **Level from** and **Kind**. Each row shows the kind, the file or package (**File or package**), a line number (“Line {line}”) when there is one, **First seen** and the status.

Kinds and levels

Kind	Level	Meaning
Changed file	2	A file differs from the baseline.
New file	2	A file is not in the baseline.
Removed file	1	A file of the baseline is gone.
Suspicious content	3 or 2	A code file has a match of the content check. Level 3 for malware, level 2 for suspicious.
Unexpected upload	3	A PHP file in an upload folder.
Known vulnerability	3 to 1	A package has a published advisory. Critical and high is level 3, low is level 1, all others level 2.
Leftover file	2	A file in the public folder that does not belong there.
Malware found by the scan service	3	The scan service reports malware.

The levels of the plugin are 1 to 3. For each level 3 finding, the scan records an event `scan.finding` (at most 20 per scan), and the plugin sends an alert mail. See Event log and notifications.

Status of a finding

A finding is **Open**, **Ignored** or **Fixed**. A finding becomes **Fixed** when a later scan no longer reports it, or when you accept it.

Actions on a finding

- **Ignore** hides an open finding from the open count. For a finding about a file, the finding stays ignored as long as the content of the file does not change. When the file changes, the finding is open again.
- **Stop ignoring** makes an ignored finding open again.
- **Accept** is offered for changed, new and removed files. It makes the current state the new baseline. A changed or new file is accepted only when it still has the content that the scan saw. If it changed after the scan, nothing is accepted. A changed file keeps its old baseline entry until you accept it.
- **Accept all changes** accepts all open and ignored findings of the kinds changed, new and removed, in batches of 200. A confirmation asks “Accept all changed, new and removed files?” and says “{count} findings become the new baseline. Accept them only if you know the changes are yours.” Click “Accept {count} findings”. The result says “{accepted} accepted, {skipped} skipped because the file changed after the scan.”
- **Show lines** shows the lines of the file of a finding below its row, read as the file is now: at most 4 lines before and after the line of the finding, with the line of the finding marked. When the finding names no line, the first lines of the file are shown. **Hide lines** closes them. Nothing is changed.

Accept a change only when you know it is yours, for example after an update or a deployment.

Lines of a file

Instead of the lines, a note is shown when the file was removed or moved after the scan, when it can hold passwords or keys, when the path is a symbolic link, is not a regular file, has more than one hard link or leads out of the main folder of the shop, or when the file is larger than 5 MB. Files that can hold passwords or keys are, for example, `.env` files, `.htaccess`, `config/jwt`, key files, database dumps and the folder `var`: the files that the remote scan never sends or sends only in redacted form, see Remote scan and licence. Backups and editor copies of such files count as well, for example `.env.bak` or `.env.swp`. The content of archives and database dumps (for example `.sql`, `.sql.gz`, `.zip`,

`.tar.gz`) and of leftover files is never shown. A note also says when the file was changed after the finding was last seen, and when a line was cut after 500 characters. Characters that cannot be shown safely (control characters, invisible characters and characters that change the direction of the text) appear as `␀`. Only administrators can see the lines.

Scanner settings

The settings are in the group **Scanner** on the tab **Settings**. All keys, defaults and ranges are in Settings reference.

Remote scan and licence

The remote scan is optional and off by default. With it, the file scanner also asks a scan service about the code files of the shop. The service knows checksums of files that are known to be harmless or malicious, so it can find malware that the basic local checks do not. The remote scan is the thorough one. Everything else of the scanner works without it. The local scan is described in File scanner.

A scan never depends on the remote part. If the service cannot be reached or refuses, the local scan finishes all the same, and the scan records that the remote part failed or was partial. After an internal error of the remote part, the event log also gets a notice with the reason `remote_failed`.

You set the mode on the tab **Settings** (**Remote scan**, key `remote.mode`) and look after the connection in the card **Remote scan** on the tab **Scanner**.

Remote scan modes

Value	What it does
`off`	Sends nothing. This is the default.
`daemon`	Uses a scan service on the same server, through the socket in **Socket of the local scan service** (default `/run/ispprotect/scand.sock`). It needs no access token and sends nothing to another computer.
`service`	Uses a scan service that you run yourself, at the address in **Address of the scan service**. It needs an access token. The address must use HTTPS; plain HTTP is accepted on the loopback address only.
`license`	Uses the hosted scan service. It needs a connected licence, which is a licence key from the licence portal.

Only the hosted scan service (`license`) needs a licence key. The local scan service (`daemon`) needs no key for the site.

The hosted scan service at `https://scan.ispprotect.com` and the licence API at `https://license.ispprotect.com` are built in; you only need the licence key. The setting **Address of the hosted scan service** (`remote.hosted_url`) is an optional override for testing and is normally left empty. `remote.portal_url` overrides it only for a different portal deployment, using `ispprotect:settings:set` on the command line. The account console at `https://console.ispprotect.com` is not the scan API.

The remote scan needs the content check of the scanner (`scanner.content`) to be on. The remote part runs between the check for removed files and the check of the public folder. The mode is read when the scan starts. If you change the mode while a scan runs, the remote part of that scan ends, and the next scan uses the new mode.

The card Remote scan

The card on the tab **Scanner** shows the mode and a short help text for it, and the link **Change the mode under Settings**. When the mode is not `off`, it also shows the address in use. In mode `daemon` or `service` it shows the socket or the address of the scan service, or **Not set** when the setting is empty.

The button **Test connection** appears when the mode is not `off`. It asks the service for its status and shows these facts, as far as the service reports them:

Fact	Meaning
Engine	The scan engine of the service.
Signatures	The number of signatures.
Signature set	The set in use.
Loaded	When the service loaded it.
Largest file	The largest file the service accepts.

Fact	Meaning
Upload per day	How much the service accepts per day.
Requests per minute	How many requests per minute the service accepts.

The test changes nothing. The three limits at the end are those of the service, not of the plugin.

Access token

In mode `service` the card has the section **Access token**. It shows one of three states:

- **Stored, ends in** and the last four characters.
- **No access token stored**.
- **Stored, but it cannot be read; enter it again**.

To store a token, type it into **New access token** and click **Save token**. The token is stored encrypted and is never shown again. Leave the field empty to keep the current token. **Remove token** deletes it. The field takes up to 4112 characters. You can do the same on the console with `ispprotect:remote token:set` and `ispprotect:remote token:clear`, see Command line.

Licence

In mode `license` the card has the section **Licence**. It shows **Connected** or **Not connected**.

The built-in licence API address is available by default. If an invalid `remote.portal_url` override has replaced it, the section cannot offer **Connect**; correct the override on the console first.

When the licence is not connected, type the key into **Licence key** (up to 96 characters) and click **Connect**. The key is sent to the licence portal once and stored encrypted. It is never shown again.

When the licence is connected, the section shows:

Fact	Meaning
Licence key	The key is shown as **ends in** and its last four characters.
Plan	The plan of the licence.
Licence valid until	The end of the licence.
Sites	The sites in use and the sites the licence allows, as “used of limit”.
Host of this site	The host name under which this site is counted.
Access valid until	The end of the current access to the service. It is marked **Expired** when the time has passed.
Renewal	Whether the access is renewed automatically, and when it was last renewed.

A connected site is counted at the licence portal.

The plugin renews its access by itself from the scheduled maintenance, see Scheduling. It renews about once a day, and again when the access runs out within two days. A temporary failure is tried again later. It does this in every mode while a licence key is stored. The line **Renewal** says one of these:

- **Renewed automatically about once a day.**
- **The last renewal failed; the next attempt follows automatically.**
- **Automatic renewal is stopped. Use “Check now”.**

After a clear refusal by the licence portal, the automatic renewal stops until you click **Check now** or enter a new key. **Check now** (console: `ispprotect:remote refresh`) renews the access at once.

If the clock of the server is wrong, the portal sends an access that is already expired by the clock of the server. The card then says that the clock seems to be wrong. Correct the system time and click **Check now**.

Disconnect asks for a confirmation with the question **Disconnect the licence?**. If you confirm, the site is released at the licence portal and the key is removed from this server. The hosted scan service cannot be used until you connect a licence again. The key is removed in every case, also when the portal cannot be reached. Uninstalling the plugin with its data also releases the site, see Removing the plugin.

What is sent and what never is

For each non-empty code file in the folders that the scanner reads for content, the plugin sends two things: the SHA-256 checksum, and a name hint. The name hint is the file name without its path, or only `file.` and the extension when the name is too long or unusable. Only files with a name in UTF-8 are sent.

Only files that the service does not know are uploaded. They are uploaded only within the **Upload limit per scan (MB)** (`remote.upload_mb`, default 200, up to 4096) and within the limits of the service. With the value `0` the plugin sends checksums only and uploads nothing. Files larger than **Largest scanned file (KB)** of the scanner (`scanner.max_file_kb`) are not sent. In mode `daemon` all of this stays on the server.

The following is never sent, although the local scanner still examines these files:

- the configuration files `.htaccess`, `.user.ini` and `.env*`;
- files with the name of a configuration or credential file, such as `parameters.yml`, `auth.json`, `.htpasswd`, `.npmrc`, `.netrc` and private key files such as `id_rsa`, and copies whose name starts with `wp-config`, `localconfig`, `parameters.y` or `config.local`, except the PHP files named below;
- files with the extension `pem`, `key`, `crt`, `p12`, `pfx`, `jks`, `kdbx`, `sql`, `sqlite`, `db` or `ini`;
- everything in the folders `config/jwt`, `config/secrets`, `var`, `.git` and `.ssh`, and everything below a folder called `.git` or `.ssh` anywhere;
- the whole Shopware folder `config` and the file `deploy.php`, because the plugin marks them as protected on this platform;
- files in `config/packages` with “secret” in the name;
- files whose name starts or ends with a space or another blank, such as `wp-config.php `;
- files whose name only looks like a configuration file and that are not PHP files, such as `local_config.yml`, `wp_config.php.bak` or `wpconfig.txt`, also with an invisible or non-Latin character in the name;
- the paths you list in **Paths not sent to the scan service** (`remote.exclude`).

Sent only in redacted form: PHP configuration files named `wp-config.php`, `localconfig.php` or `config.local.php`, copies whose name starts with `wp-config` and ends with `.php`, and PHP files whose name only looks like a configuration file (such as `local_config.php`, `wp_config.php` or `my-wp-config.php`, also with an invisible or non-Latin character, with an extension that PHP runs such as `.php`, `.phtml` or `.inc`), when they are not in one of the folders above. Shopware keeps no such file outside `config`. A `wp-config.php` or `config.local.php` left in the shop folder, for example after a move from another system, is sent this way. Before the checksum is taken and before the upload, all strings, numbers and comments in such a file are replaced: every string by `ISPP_REDACTED`, every number by `0`, the text of every comment by `ISPP_REDACTED`, a variable name inside a string (such as `"pa$word"`) by `$ISPP_REDACTED`, every name inside `{$...}` or `${...}` in a string by `ISPP_REDACTED`. Only the code structure, the names (functions, variables, constants) and the constant names in `define()` stay (in a file with a `namespace` declaration, a `use` import or PHP 8 attributes those names are replaced too), so passwords and other values never leave the server. Code that an attacker added can still be found by its structure: the service sees calls such as `eval()`, `base64_decode()` or `include`, but not their literal contents. The checksum that is looked up is the one of the redacted form; the one of the original file is never sent. The service can still tell an empty string or the leading blanks of a line in a string, but not read any value. A file that cannot be redacted safely (it does not parse, it is larger than 1 MB, it contains `__halt_compiler`, or it uses a short open tag `<?` or starts with `<?xml`, whose meaning depends on the PHP setting `short_open_tag`) is not sent at all. Malware found in the redacted form is reported for the real file, with the note that it was checked in redacted form.

Names and paths are compared in lower case, so a different spelling does not get through.

The card shows a line about what leaves the server for the mode in use. The privacy side of this chapter is in Privacy and external services.

Limits and refusals

The plugin works within these limits. The service can set stricter ones, and **Test connection** shows them.

- **Upload limit per scan (MB)**: at most this much file content leaves the server in one scan. Files that do not fit stay waiting, and the next scan sends them.
- The service has its own limits for the size of one file, for the upload per day and for the requests per minute.

When the service refuses or cannot be used, the scan goes on locally and the card **Last scan** shows a state and a reason. The states are:

State	Meaning
Checked by the scan service	The remote part ran to the end.
Partly checked by the scan service	The remote part ended early, with a reason.
Scan service not available	The remote part could not run.
Remote scan off	The mode is `off`.

The reasons include:

- The scan service could not be reached.
- The scan service did not accept the access credentials.
- The scan service is not ready yet, for example because it is still loading its signatures.
- The scan service is busy, or asked to wait. The scan tries again later. It waits a limited time and leaves the remote part when the waits do not end.
- The quota of the scan service is used up. The plugin then sends no further file in this scan, but still looks up the checksums of the remaining files, so known verdicts are reported. The files not sent follow in a later scan.
- A file was too large for the scan service.
- The upload limit of this scan was reached. The remaining files follow in the next scan.
- The scan service refused access: the daemon does not allow this server user, or the access credentials do not include the remote scan.
- No valid address or socket is set, or an access token or a connected licence is missing.
- The remote scan setting was changed while the scan was running.

Last scan

The section **Last scan** of the card shows the state and the counts of the last scan:

Count	Meaning
Looked up	Files whose checksum was asked for.
Clean	Files the service knows as harmless.
Uploaded	Files uploaded because the service did not know them.
Malware	Files the service reported as malware.
Skipped	Files that were not sent, for example because they were too large, unreadable or changed while they were sent.
Not examined	Files that were not examined by the service in this scan.
Uploaded size	The size of what was uploaded.

The same counts are shown by `ispprotect:remote status`.

Findings of the remote scan

Malware reported by the service is a finding of the kind **Malware found by the scan service**, with the name of the detection, at level 3. It cannot be accepted as a new baseline, so **Accept all changes** does not cover it. It is closed when a later scan finds the file clean or gone. See File scanner for the list of findings.

Credentials and storage

The access token and the licence key are stored encrypted in the database. The key for this is derived from the application secret (`APP_SECRET`) of the shop. They are never shown again, never written to a log, to an event or to an answer of the Administration API, and never sent back to the browser. The page shows only whether they are set and the last four characters.

If the application secret changes, the stored token and licence data can no longer be read. The card then says so, and you enter them again. If the shop cannot store encrypted values, for example because the application secret is too short, the card says so and no token or key can be saved.

Changes to the token and the licence are recorded in the event log, never with a secret. The events are `remote.token_set` , `remote.token_cleared` , `remote.license_connected` , `remote.license_refreshed` , `remote.license_disconnected` and `remote.license_failed` .

Console

The command `ispprotect:remote` does everything the card does. Its actions, the way a secret is passed in and the exit codes are in Command line. If a connection does not work, see Getting back in and troubleshooting.

Sign-in protection

Sign-in protection counts failed sign-ins and reacts to them. It covers the Administration (users) and the storefront (customers). The settings are on the tab **Sign-in protection** and on the tab **Settings**. See Sign-in settings.

Which sign-ins are counted

The plugin keeps the two areas apart. In the tables of the Administration and in the event log, the area is shown as `admin` or `customer`.

- **Administration** (`admin`): every sign-in to the Administration through the token request `/api/oauth/token` with a user name and password.
- **Customers** (`customer`): the credential sign-ins of the storefront form (`/account/login`) and of the Store API (`/store-api/account/login`). Guest orders, impersonation by an administrator and a sign-in by customer ID are not counted. Shopware's own rate limit for customer sign-ins keeps working.

How it works

Failed sign-ins are counted per address (IPv6: per /64 range) and per user name. User names are compared in a normalised form and are at most 190 characters.

- When one address reaches **Failed sign-ins per address** failures within **Sign-in time window (minutes)**, the address is banned in the firewall. See Bans.
- When one user name reaches **Failed sign-ins per user name** failures within the window, the user name is locked for **Lock length (minutes)**. While it is locked, even the correct password is refused. A refused customer sign-in looks like a wrong password.
- There is one exception for the lock of a user name: an address range that the account signed in from within the last 90 days (as far as the event log holds it). From there the name stays open until **Failed sign-ins per user name** more failures have come from that range during the lock.
- Addresses on the allow list are never counted, banned or locked. An allow entry lets a locked address pass while it exists, but it does not remove the lock.
- A successful sign-in resets the failure count of that user name. A successful Administration sign-in also resets the failure count of the address. A customer sign-in does not, because anybody can register a customer account.
- When customers are bound to sales channels, each account is counted on its own: the user name then shows as `<address> [sales channel <id>]` in the lock list and the event log, and a sign-in through one sales channel never resets the failure count of an account with the same address in another sales channel. Unlock such an account with that full name.
- A typed address that has no account in the sales channel but ends like such a name (`... [sales channel <id>]`) is counted as `<typed text> [unknown account]`, so it never counts against the bound account.
- A sign-in without a client address is still counted and locked by user name, as in the Administration.
- A refused attempt is recorded as `login.refused`, at most once per minute.

Protected administrator addresses

For 24 hours after an administrator signed in, the address is protected (if **Protect administrator addresses** was on at that sign-in). Too many failed sign-ins from a protected address do not ban it in the firewall. They lock signing in from that address instead. The lock applies to the area where the failures happened, so the rest of the shop stays reachable. It lasts as long as **Lock length (minutes)**. When the previous lock of the same address began within 30 days, the lock lasts four times as long, up to **Longest ban (minutes)**. It holds until it ends, even when the 24 hours of protection end first. Entries of the block list still apply to the address.

The plugin keeps only a hash of such an address, so the address is not shown in a list. The event `login.address_locked` names it, in shortened form when **Shorten addresses in the log** is on. See Event log and notifications.

What the visitor sees

- A locked user name in the Administration: Shopware's own message that there were too many attempts.
- A refused address in the Administration: the error 1103 with status 403.
- A refused customer: the same message as for wrong credentials.

If the plugin cannot read or write its data, the sign-in is let through. The exception is the check of the second step for a user who has one. See Two-factor authentication.

New addresses

After an administrator signs in from an address that the same account has not used in the last 90 days, the plugin records the event `login.new_address`. With **Alert on sign-ins from new addresses** on (the default), the event has the level **Alert** and an alert mail is sent. With the setting off, it has the level **Notice**. It is only recorded for an account that has signed in before, not for its first sign-in. Customers have no such event.

Lifting locks and bans

The tab **Sign-in protection** has these cards:

- **Locked user names**: the columns **User name**, **Area**, **Locked since** and **Locked until**. The button **Unlock** lifts the lock. The message says "{name} was unlocked."
- **Sign-in locks of addresses**: the locks of protected addresses. Because only a hash of the address is kept, you unlock by typing the address. Enter the **IP address** (IPv4 or IPv6; an IPv6 lock covers its /64 range), choose the area (**All areas** or one area) and click **Unlock address**. This also resets the failure count of the address. A firewall ban of the address stays. The card shows at most 200 locks, the ones that end first ("Shown are the {shown} locks that end first, of {total}.") with the columns **Area**, **Locked since**, **Locked until** and **Minutes**. The message says "Signing in from {address} is possible again." or "No sign-in lock was found for {address}."
- **Firewall bans**: bans of addresses. The button **Lift ban** ends a ban.

Unlocking a user name on this tab also resets the count of wrong two-factor codes of that user. Unlocking on the console does not. Each lifted lock is recorded as `login.unlocked` or `login.address_unlocked`.

On the console:

```
bin/console ispprotect:unlock <user name>
bin/console ispprotect:unlock --address=<address>
bin/console ispprotect:unlock <user name> --realm=admin
```

See Command line for the options and exit codes.

Shortened addresses

With **Shorten addresses in the log** on, the plugin shortens addresses before it stores them in the event log: IPv4 addresses lose the last number (it becomes 0), IPv6 addresses keep their first 48 bits. The event of a user name lock then holds no range. The comparison with known addresses uses the shortened ranges. See Privacy and external services.

Sign-in settings

The settings are in the groups **Sign-in protection** and **Two-factor authentication** on the tab **Sign-in protection** and on the tab **Settings**. All keys, defaults and ranges are in Settings reference.

Two-factor authentication

Administration users can protect their sign-in with a second step. You can also require it for every Administration user. Customers of the storefront have no second step.

How two-factor sign-in works

- The second step is a time-based code (6 digits, changing every 30 seconds) from an authenticator app. A code from the neighbouring time step is accepted, to allow for clock differences. Each code works once.
- At set-up, the user gets 10 recovery codes. Each is made of 4 groups of 4 characters. They are shown once. Each works once, in place of an app code. The plugin stores only hashes of them.
- The key of the app is stored encrypted, with a key derived from `APP_SECRET` of the shop.
- After 5 wrong codes within 15 minutes, every code of that user is refused until the 15 minutes have passed. A right code resets the count. A wrong code is also a failed sign-in: it counts toward **Failed sign-ins per address** and **Failed sign-ins per user name** and can lock the user name. See Sign-in protection.

Signing in

After the password, a user with a second step is asked for a code in a dialog. The dialog accepts the 6-digit code or a recovery code. The same code is needed for Shopware's confirm-password dialog.

The sign-in request answers with these plugin errors:

Code	Status	Meaning
1101	401	A code is required. This is not counted as a failure.
1102	401	The code is wrong. It counts as a failure of the address and of the user name.
1103	403	The address is refused.
1104	503	The second step is unavailable.
1105	429	Too many wrong codes.

Who can open the page

The module entry is shown to administrators only. Users who are not administrators have no menu entry. They reach their own page at `#/ispprotect/web-security/index/twofactor`, or they are sent there when the shop requires the second step. On every other page of the module they see "This area is only available to administrators."

Set up your second step

Open the tab **Two-factor**. The card "Your account" shows **Two-factor sign-in** as **On**, **Off** or "Set-up not finished", the number of recovery codes left ("Recovery codes left") and the **Shop rule** ("Required for every user" or "Not required").

1. Click **Start set-up**.
2. "Confirm your password" and click **Continue**. The password is used once and is not stored.
3. "Add the key to your authenticator app": the page shows the **Key** as text and as a **Set-up link**. It shows no QR code, so that nothing is requested from another site. The name of the shop is the issuer in the app.
4. Enter the first code in **Code from the app** and click **Turn on**.
5. "Your recovery codes": the page shows the recovery codes once. Click **Copy all codes**, store them safely, then click **I have saved them**. You cannot leave the page before.

You can discard an unfinished set-up with “Discard unfinished set-up”. That is not possible while the shop requires the second step. Starting the set-up again replaces an unfinished one.

Recovery codes and turning off

- **Create new codes** (“New recovery codes”) makes 10 new codes and replaces the old ones. You need a current code.
- **Turn off** (“Turn off two-factor sign-in”) removes your second step. You need a current code. In the field “Current code (from the app, or a recovery code)”, type a code from the app or a recovery code.
- The page warns “Only {count} recovery code(s) left.” when few are left.
- Errors you can see: “The code is wrong or was used already.” and “Too many wrong codes. Wait {seconds} seconds and try again.”

Turning off is not offered while the shop requires the second step. The page says “An administrator can lift the requirement under Sign-in protection.”

Requiring it for everybody

The setting **Require a second sign-in step** (`twofactor.required` , default off) is in the group **Two-factor authentication**, on the tab **Sign-in protection** and on the tab **Settings**.

While it is on:

- A user without a second step can still sign in with the password. The Administration then holds that user on the set-up page.
- Every other Administration API request of that user is refused with status 403 and the code `ISPPROTECT__TWOFACTOR_SETUP_REQUIRED` . Exceptions are the set-up routes of the plugin and five read-only start-up routes that the Administration needs to show the page (`api.info.entity-schema` , `api.info.config` , `api.info.me` , `api.config_me.get` and `api.consents.fetch`).
- A user who already has a second step is not held up.

Integrations and access keys are not affected, because their tokens belong to no user. The Store API is not affected either.

The form warns “You have not set up two-factor sign-in for your own account. If you turn this requirement on, you will be asked to set it up at once.” when you turn the requirement on while your own account has no second step. Set up your own second step first.

Next to the setting, the form always shows “Locked out of the Administration? On the server, run:” with the recovery command. See Lost device or locked out of two-factor.

Limits of the second step

- It covers the password sign-in of the Administration. The refresh token flow, client credentials (integrations and access keys), the authorization-code grant and Shopware’s SSO grant do not ask for a code. A user who signs in through single sign-on is decided by the identity provider.
- The requirement still applies to every request of a user, also with tokens from the refresh and SSO grants.
- When the setting cannot be read, the requirement is not enforced. When the state of the second step cannot be read while the requirement is on, the request gets status 503 with the code `ISPPROTECT__TWOFACTOR_UNAVAILABLE` .
- If you change `APP_SECRET` , the stored keys can no longer be decrypted, and users with a second step cannot pass the code check. Plan a change of `APP_SECRET` with this in mind.
- The plugin has no console command that removes the second step of one user. The Administration has no button for it.

Security checkup

The checkup tests the server, PHP, the shop and the plugin settings. It changes nothing, except with the button **Fix permissions** (see Listed files and the permission fix). It runs when you open the tab **Checkup** and with `bin/console ispprotect:checkup`.

Results and score

Each of the 22 checks has one of four results:

Result	Meaning
In order	The check passed.
Not known	The plugin could not tell. The check is not rated.
Needs attention	A weak point. It counts half in the score.
Problem	A serious weak point. It counts fully in the score.

Each result has advice on what to change. A check that stops with an error shows “A check could not be completed”.

The score is a number from 0 to 100. It is the weighted share of the checks that are in order among the rated ones. Every check has a weight from 1 to 5. A result **Needs attention** loses half of its weight, a result **Problem** all of it. **Not known** is ignored. The tab shows the score, and the counts “{n} failed”, “{n} need attention”, “{n} for information” and “{n} fine”. The dashboard shows a short form. See Dashboard.

On the console, the command prints a table and the score. It ends with exit code 1 when at least one check has the result **Problem**, otherwise with 0.

Checks

Server and PHP

Check	When it does not pass
PHP version {version} gets security fixes	**Problem** after the end of security support of the version, **Needs attention** in the last 90 days before the end. The end dates in the plugin are 2026-12-31 for PHP 8.2, 2027-12-31 for 8.3, 2028-12-31 for 8.4 and 2029-12-31 for 8.5.
PHP does not show error messages to visitors	`display_errors` is on.
PHP does not announce its version	`expose_php` is on. The advice names the option **Hide the PHP version**.
PHP does not include code from other servers	`allow_url_include` is on.
Enough free disk space	**Needs attention** below 5 percent free, **Problem** below 1 percent.

The three checks about PHP options can only read the settings of the web server when you open the tab **Checkup**. On the console, which can have other PHP settings, they show a hint instead of a result.

Shop and files

Check	When it does not pass
The application secret is long and not a sample value	**Problem** when `APP_SECRET` is shorter than 32 bytes or a sample value.

Check	When it does not pass
Production mode without debugging	**Needs attention** when debugging is on or the environment is not `prod`.
The site address uses HTTPS	**Needs attention** when it does not. **Not known** when no address is set.
Files with passwords and keys cannot be read by other users	**Needs attention** when other users can read them.
No file or folder in the code can be changed by everybody	**Needs attention** when something is writable for everybody. **Not known** when the search stopped at its limit.
No backups or debugging files in the public folder	**Needs attention** when it finds some.
The web installer is not left on the server	**Not known** (information) when `shopware-installer.phar.php` is in the public folder.
Trusted proxies match the real setup	**Problem** or **Needs attention** when every address is trusted or a forwarded header comes from an address that is not trusted. **Not known** when this cannot be told. See Behind a reverse proxy or CDN.

Accounts

Check	When it does not pass
Administrators use two-factor authentication	**Needs attention** when an administrator has no second step.
Few administrator accounts	**Needs attention** with more than 5 administrators.
No integration has administrator access	**Needs attention** when an integration is an administrator.
Administration passwords have a sensible minimum length	**Needs attention** when the minimum length is below 12. Shopware's default is 8.

Plugin

Check	When it does not pass
The firewall is on	**Needs attention** in mode **Log only**, **Problem** in mode **Off**.
Sign-in protection is on	**Problem** when it is off.
Alerts have a receiver	**Needs attention** when no receiver can be found. See Notification settings.
The file scan ran in the last {days} days	**Needs attention** when the last complete scan is older than 8 days or the schedule is off. **Not known** when no scan has run.
No open findings of a high level	**Problem** with an open finding of level 3, **Needs attention** with one of level 2.

Listed files and the permission fix

Three checks list what they found below their advice: **Files with passwords and keys cannot be read by other users**, **No file or folder in the code can be changed by everybody** and **No backups or debugging files in the public folder**. The names are relative to the project folder, for public leftovers to the public folder. The lists are shown on the tab **Checkup** and by `bin/console ispprotect:checkup`. The list of world-writable paths has at most 20 entries, the list of public leftovers at most 50. A longer list ends with "The list shows the first {count} entries only."

When the check **Files with passwords and keys cannot be read by other users** shows **Needs attention**, the button **Fix permissions** removes all permissions of "others" from the listed files, the same as `chmod o-rwx`. The owner and the group keep theirs. The button takes the file names from the check itself, never from the browser, and only changes a file that

- is a regular file directly in the project folder, not a link or a folder, with no second name (hard link), and
- belongs to the user PHP runs as.

The button is shown when at least one listed file can be changed. For every other file the message names the command to run in the project folder, for example `chmod o-rwx .env`. Afterwards the check runs again and shows the new result. The change is recorded in the audit log as “File permissions changed by the checkup”, with the names of the changed and of the unchanged files.

A program that reads these files under another user account, for example a cron job, needs to be in the group of the file after the change.

The other two checks have no button. Remove the write permission of “others” from the listed paths, for example with `chmod o-w`, and delete the listed leftovers or move them out of the public folder.

Event log and notifications

The plugin records security-relevant events in its own database table and sends alerts and a summary by e-mail.

Reading the event log

The tab **Events** shows the card **Event log** with 25 entries per page. The columns are **Time**, **Level**, **Event**, **User name**, **Address** and **Details**. When nothing matches, the page says “No events match.”

You can filter by **Type**, **Level from**, **From**, **To**, **User name** and **Address**. Click **Apply filter** to filter and **Reset** to clear the filter.

On the console, `bin/console ispprotect:events` shows the newest events:

```
bin/console ispprotect:events
bin/console ispprotect:events --type login.failure --since 60 --limit 100
bin/console ispprotect:events --since "2026-11-01 12:00"
```

`--since` takes a number of minutes or a date in UTC. `--limit` is 1 to 500 (default 50). See Command line.

What is recorded

- sign-ins, failures, refusals, sign-ins from new addresses, locks of user names and addresses, and lifted locks
- two-factor changes and failures
- bans and changes of the address lists
- firewall refusals, and decisions in the mode **Log only**
- changes of the settings of the plugin
- scans and findings
- changes in the shop that matter for security: Administration users, roles and role assignments, integrations and access keys, system configuration, installs, updates, activations and removals of plugins and apps, and password recovery requests for Administration users. Customers are not covered by these change events.

The log stores the names of changed fields only, never their values. A change of a setting lists the key names, not the values.

Event levels

Each event has a level: **Info**, **Notice**, **Warning** or **Alert**.

Event log settings

The group **Event log** on the tab **Settings** has two settings.

- **Keep log entries (days)**: how long an event is kept. The maintenance task removes older events once per hour.
- **Shorten addresses in the log**: when on, addresses are shortened before they are stored. IPv4 addresses lose the last number (it becomes 0). IPv6 addresses keep their first 48 bits. The setting does not change events that are already stored.

All keys, defaults and ranges are in Settings reference.

Alert mails

The plugin sends a mail for events of the level **Alert**. These are:

- a finding of level 3 of the scan (at most 20 events per scan)
- a sign-in from a new address, when **Alert on sign-ins from new addresses** is on

There is at most one mail per event type per hour. It says how many events are waiting and lists up to 10 of them. When a mail cannot be sent, the events stay unsent, that type waits 15 minutes, and the plugin records the event `system.notice` with the reason `mail_failed`.

Immediate alerts switches the alert mails on and off.

Summary mail

The summary mail covers a period and contains the events by type, bans and locks, open findings by level, the checkup score and problems. **Summary mail** sets it to **Off**, **Daily** or **Weekly**.

Notification settings

The group **Notifications** on the tab **Settings** has three settings: **Notification address**, **Immediate alerts** and **Summary mail**.

If **Notification address** is empty, the plugin uses the e-mail address from the basic information of the shop (`core.basicInformation.email`). If no receiver exists, alerts are marked as handled without a mail. The checkup then shows “Alerts have a receiver” as **Needs attention**.

The mails are sent through the mail service of Shopware, with the first active storefront sales channel and the name of the shop as sender name. Mail delivery therefore depends on the mail settings of the shop. The mails are plain text. The scheduled task sends them. See Scheduling.

Settings reference

Changing settings

On the tab **Settings**, every setting of the plugin is in a card per group. The tabs **Firewall** and **Sign-in protection** show the groups that belong to them. See The module in the Administration.

Each card has two buttons:

- **Save** saves the settings of the card. The message says “{group}: settings saved.” If a value is wrong, the message says “Nothing was saved. Correct the marked fields.” If nothing changed, it says “There are no changes to save.”
- **Reset to defaults** asks “Reset this area to defaults?” Click **Reset** to confirm. The message says “{group}: defaults restored.”

On the console, `ispprotect:settings:get [key]` shows one setting or all, and `ispprotect:settings:set <key> <value>` changes one:

```
bin/console ispprotect:settings:get firewall.mode
bin/console ispprotect:settings:set firewall.mode log
bin/console ispprotect:settings:set trap.paths '["/wp-login.php","/backup/*"]'
```

The value is read as JSON when it is valid JSON, otherwise as text. A setting that is a list or the rules is given as JSON. Every change is recorded as the event `settings.changed`, with the key names only.

Value types

- A yes-or-no setting accepts `1`, `0`, `true`, `false`, `on`, `off`, `yes` and `no` on the console.
- A list (**Trap paths**, **Paths without request limit**, **Excluded paths**, **Paths not sent to the scan service**) holds at most 500 entries of at most 255 bytes each.
- A number outside its range is refused.

Group Firewall

Label	Key	Default	Meaning
Firewall mode	`firewall.mode`	`on`	`off`, `log` or `on`. See Firewall modes.
Protect administrator addresses	`firewall.allow_admin_ip`	on	Protects the address of a signed-in administrator for 24 hours from bans and automatic measures.

Group Request limit

Label	Key	Default	Meaning
Limit requests per address	`rate.enabled`	on	Switches the request limit on or off.
Requests per window	`rate.limit`	600	Requests one address may send in the time window (30 to 100000).
Time window (seconds)	`rate.window`	60	Length of the window (10 to 3600).
Sign-in requests per window	`rate.signin_limit`	30	Lower limit for sign-in requests (5 to 1000).
Paths without request limit	`rate.exempt_prefixes`	empty	Path prefixes that the limit does not count.

Label	Key	Default	Meaning
Ban after refusals	`rate.ban_after`	3	Refusals within 10 minutes that lead to a ban (0 to 100). 0 means no ban.

Group Missing pages

Label	Key	Default	Meaning
Watch “not found” answers	`notfound.enabled`	on	Counts “not found” answers (status 404) per address.
“Not found” answers per window	`notfound.limit`	40	Number that leads to a ban (5 to 10000).
“Not found” time window (seconds)	`notfound.window`	300	Length of the window (60 to 3600).

Group Trap paths

Label	Key	Default	Meaning
Trap paths	`trap.paths`	empty	Paths that no visitor should open. An address that opens one is banned. See Trap paths and paths without request limit.

Group Custom rules

Label	Key	Default	Meaning
Custom rules	`rules.custom`	empty	A JSON list of at most 100 rules. See Custom rules.

Group Bans

Label	Key	Default	Meaning
Ban length (minutes)	`ban.base_minutes`	15	Length of the first ban (1 to 10080).
Longest ban (minutes)	`ban.max_minutes`	10080	Upper limit for repeated bans and repeated address locks (1 to 525600).

Group Sign-in protection

Label	Key	Default	Meaning
Sign-in protection	`login.enabled`	on	Switches sign-in protection on or off.
Failed sign-ins per address	`login.ip_limit`	5	Failures that lead to a ban (2 to 1000).
Failed sign-ins per user name	`login.user_limit`	10	Failures that lead to a lock (2 to 1000).
Sign-in time window (minutes)	`login.window_minutes`	15	Counting window (1 to 1440).
Lock length (minutes)	`login.lock_minutes`	15	Length of a lock (1 to 1440).
Alert on sign-ins from new addresses	`login.alert_new_address`	on	Makes the event of a sign-in from a new address an alert. See Sign-in protection.

Group Two-factor authentication

Label	Key	Default	Meaning
Require a second sign-in step	`twofactor.required`	off	Requires a second step from every Administration user. See Two-factor authentication.

Group Event log

Label	Key	Default	Meaning
Keep log entries (days)	`audit.retention_days`	90	How long an event is kept (1 to 3650).
Shorten addresses in the log	`audit.anonymize_ip`	off	Stores shortened addresses.

Group Notifications

Label	Key	Default	Meaning
Notification address	`notify.email`	empty	Receiver of the mails. Empty uses the e-mail address of the shop.
Immediate alerts	`notify.alerts`	on	Sends alert mails.
Summary mail	`notify.digest`	`daily`	`off`, `daily` or `weekly`.

Group Scanner

Label	Key	Default	Meaning
File scan schedule	`scanner.schedule`	`daily`	`off`, `daily` or `weekly`.
File scan hour	`scanner.hour`	3	Hour (UTC) from which the scheduled scan may start (0 to 23).
Detect changed files	`scanner.baseline`	on	Compares with the baseline.
Search for malicious code	`scanner.content`	on	Reads the files for the basic suspicious-code checks.
Check for known vulnerabilities	`scanner.advisories`	on	Looks up advisories for the installed packages.
Excluded paths	`scanner.exclude`	empty	Paths relative to the main folder of the shop that are not scanned.
Largest scanned file (KB)	`scanner.max_file_kb`	5120	Largest file that is read (64 to 102400).

Group Response headers

The plugin can change the headers of the shop's answers.

Label	Key	Default	Meaning
Hide the PHP version	`headers.hide_php_version`	on	Removes the `X-Powered-By` header.
Enforce HTTPS (HSTS)	`headers.hsts`	off	Sends `Strict-Transport-Security` on HTTPS requests.
HSTS duration (seconds)	`headers.hsts_max_age`	31536000	Lifetime of the HSTS header (300 to 63072000).

Label	Key	Default	Meaning
Permissions-Policy header	`headers.permissions_policy`	empty	Value of the `Permissions-Policy` header, up to 1024 bytes. Empty sends none.

Turn on HSTS only when the whole shop, and all subdomains you rely on, work over HTTPS.

Group Remote scan

Label	Key	Default	Meaning
Remote scan	`remote.mode`	`off`	`off`, `daemon`, `service` or `license`. See Remote scan and licence.
Socket of the local scan service	`remote.socket`	`/run/ispprotect/scand.sock`	Absolute path of the socket (mode `daemon`), at most 107 bytes.
Address of the scan service	`remote.service_url`	empty	Address of your own scan service (mode `service`). HTTPS, or HTTP on the loopback address only.
Address of the hosted scan service	`remote.hosted_url`	empty	Optional override of the built-in hosted scan service `https://scan.ispprotect.com` (mode `license`). Normally left empty; set an HTTPS address only to use another deployment, for example for testing.
Address of the licence portal	`remote.portal_url`	empty (uses `https://license.ispprotect.com`)	Override the built-in licence API address only for a different portal deployment, on the command line.
Upload limit per scan (MB)	`remote.upload_mb`	200	Most file content one scan uploads (0 to 4096). 0 sends checksums only.
Paths not sent to the scan service	`remote.exclude`	empty	Paths whose files are never sent.

Group OWASP CRS rules

The group is shown on the tab **Firewall**, below the card **OWASP CRS rules**, and on the tab **Settings**. The rules must be installed before the mode **Log only** or **Block** has an effect. See OWASP CRS rules.

Label	Key	Default	Meaning
CRS rules mode	`waf.mode`	`off`	`off`, `log` or `block`. The first installation of the rules sets `log`. See Start with logging.
Paranoia level	`waf.paranoia`	`1`	1 to 4. Higher levels run more rules and report more normal requests. See Paranoia level and threshold.
Anomaly threshold	`waf.threshold`	`5`	Score at which a request counts as an attack (1 to 100).
Disabled rules	`waf.disabled_rules`	empty	Rule numbers that are not run, one per line (up to 500). A rule number is a whole number, for example `942190`.
Inspected body size	`waf.body_limit`	`131072`	Bytes of a request body that are inspected (1024 to 1048576). A larger body is refused with status 413 unless uninspected requests may pass.
Time limit per request	`waf.time_budget_ms`	`200`	Most milliseconds the rules may take for one request (5 to 10000).

Label	Key	Default	Meaning
Let requests pass that cannot be fully inspected	`waf.pass_uninspected`	off	Lets requests through that are too large, too slow or cannot be read. They are still recorded. See Requests that cannot be fully inspected.

Command line

The plugin adds 13 commands to the Shopware console. Run them in the main folder of the shop, as the user that runs the shop:

```
bin/console list ispprotect
bin/console help ispprotect:scan
```

All commands use the exit codes 0 (done), 1 (failure or a finding that needs attention) and 2 (wrong usage), as the sections below say.

Overview of the commands

Command	What it does
`ispprotect:status`	Shows firewall mode, bans, locks, findings and scans.
`ispprotect:checkup`	Runs the security checkup.
`ispprotect:scan`	Starts a file scan and shows its progress.
`ispprotect:findings`	Lists the open findings of the file scanner.
`ispprotect:events`	Shows the newest events of the event log.
`ispprotect:ip:list`	Lists allowed, blocked and banned addresses.
`ispprotect:ip:add`	Adds an address or range to the allow, block or ban list.
`ispprotect:ip:remove`	Removes an address or range from the allow, block and ban lists.
`ispprotect:unlock`	Lifts the sign-in lock of a user name or an address.
`ispprotect:settings:get`	Shows one setting or all settings.
`ispprotect:settings:set`	Changes a setting.
`ispprotect:maintenance`	Runs the maintenance once: alerts, cleanup, scan and summary mail.
`ispprotect:remote`	Remote scan: state, connection test, access token and licence.

Status

```
bin/console ispprotect:status
```

The output shows whether the firewall and the sign-in protection are on, the number of active bans and sign-in locks, the open findings (with those of level 3), the last complete scan, a running scan and the events of the last 24 hours with level 2 or higher. The exit code is 0.

Checkup

```
bin/console ispprotect:checkup
```

Prints the results and the score. It has no options. Exit code 1 when at least one check has the result **Problem**, otherwise 0. See Security checkup.

Scan

```
bin/console ispprotect:scan [--budget=SECONDS] [--no-wait]
```

Starts a scan, or continues a running one, and shows its progress. Without options it works until the scan is done.

Option	Meaning
`--budget=SECONDS`	Stops after about this many seconds (at most 86400). The scheduled task continues the scan.
`--no-wait`	Only starts the scan and returns.

Exit codes: 0 when the scan is done without findings of level 2 or higher, when `--no-wait` is used, and when the budget is used up. 1 when the scan failed or open findings of level 2 or higher exist. 2 for a wrong `--budget`.

Commands for findings

```
bin/console ispprotect:findings [--all] [--accept=ID] [--accept-all]
```

Option	Meaning
(none)	Lists the open findings, at most 500. Exit code 1 when open findings of level 2 or higher exist, otherwise 0.
`--all`	Also lists ignored and fixed findings.
`--accept=ID`	Accepts the changed, new or removed file of this finding as the new baseline. Exit code 0 when accepted. 1 when the file changed after the scan or the finding was not found. 2 for an invalid ID.
`--accept-all`	Accepts every changed, new and removed file. Exit code 1 when some were skipped, otherwise 0.

Events

```
bin/console ispprotect:events [--type=TYPE] [--since=SINCE] [--limit=LIMIT]
```

`--since` is a number of minutes or a date in UTC (for example `"2026-11-01 12:00"`). `--limit` is 1 to 500 (default 50). Exit code 2 for a wrong value, otherwise 0.

Commands for address lists

```
bin/console ispprotect:ip:list
bin/console ispprotect:ip:add <kind> <address> [--minutes=N] [--reason=TEXT]
bin/console ispprotect:ip:remove <address>
```

- `<kind>` is `allow`, `block` or `ban`. `<address>` is an IP address or a CIDR range.
- `--minutes=N` is the lifetime, 1 to 5256000. Without it, the entry never expires. `--reason=TEXT` is a note.
- `ispprotect:ip:add` exits with 2 for a wrong kind, address or number, 1 when the entry was not stored (for example a block of a loopback address), otherwise 0.
- `ispprotect:ip:remove` removes the address from all three lists. It exits with 2 for an invalid address, 1 when nothing was found, otherwise 0.
- `ispprotect:ip:list` exits with 0.

Unlock

```
bin/console ispprotect:unlock [<user name>] [--address=IP] [--realm=NAME]
```

Give either a user name or `--address`. `--realm` is the area: `admin` or `customer`. Without `--realm`, the lock is lifted in every area that has one. Exit code 0 when a lock was lifted, 1 when nothing was locked, 2 when both or neither of user name and `--address` were given or the address is invalid.

```
bin/console ispprotect:unlock admin
bin/console ispprotect:unlock --address=203.0.113.7
bin/console ispprotect:unlock --address=203.0.113.7 --realm=admin
```

The command does not lift a firewall ban. Use `ispprotect:ip:remove` for that. It does not reset the count of wrong two-factor codes. See Sign-in protection.

Settings

```
bin/console ispprotect:settings:get [<key>]
bin/console ispprotect:settings:set <key> <value>
```

`settings:get` without a key shows every setting. An unknown key gives exit code 2. `settings:set` reads the value as JSON when it is valid, otherwise as text. An unknown key or a refused value gives exit code 2. See Settings reference.

Maintenance

```
bin/console ispprotect:maintenance
```

Runs once what the scheduled task runs every 5 minutes: it sends pending alerts, cleans up old data (once per hour), renews a connected licence, starts and advances the scan and sends the summary mail. Exit code 1 when one of the steps fails, otherwise 0. See Scheduling.

Remote

```
bin/console ispprotect:remote [<action>]
```

Action	Meaning
`status`	Shows the mode, the address of the scan service, whether secrets can be stored, the access token and the licence key (only their last four characters), whether the site is connected, the licence data and the counts of the last remote scan. This is the default.
`test`	Asks the scan service of the current mode for its state. Exit code 1 when it fails.
`token:set`	Stores the access token of your own scan service.
`token:clear`	Removes the access token.
`connect`	Connects this site to a licence of the hosted scan service.
`refresh`	Renews the access of the licence now.
`disconnect`	Releases this site at the licence portal and removes the local licence data.

The access token and the licence key are never taken from the command line, because the command line is visible to other users of the server and lands in the shell history. Pipe them in, or type them at the hidden prompt:

```
bin/console ispprotect:remote connect < key.txt
```

Exit codes: 0 when done, 1 when the action failed (also when there is no hidden prompt and no piped input), 2 for an unknown action, extra arguments or a secret in a wrong format. `disconnect` exits with 0 even when the release at the portal fails. It then prints a warning, because the local data is removed in every case. See Remote scan and licence.

Getting back in and troubleshooting

Every lock-out can be undone on the console of the server. Run the commands in the main folder of the shop. If you cannot sign in to the Administration, you need access to the server.

Your own address is banned or blocked

Signs: pages answer with status 403 or 429 for you, or the Administration does not load. Possible causes: a ban by the firewall, an entry in the block list, or a block rule.

1. Find the entry:

```
bin/console ispprotect:ip:list
```

1. Remove it. This removes the address from the allow list, the block list and the ban list:

```
bin/console ispprotect:ip:remove 203.0.113.7
```

If the entry is a range that covers your address, remove the range exactly as the list shows it. Bans end by themselves after their length. If your address was banned because of a proxy, see All visitors have the same address.

To make your address safe from automatic measures, add it to the allow list:

```
bin/console ispprotect:ip:add allow 203.0.113.7 --reason="office"
```

A user name or an address is locked

Signs: the Administration answers “too many attempts” for a user name, or error 1103 with status 403 for an address, even with the correct password. A customer sees the message for wrong credentials.

```
bin/console ispprotect:unlock admin
bin/console ispprotect:unlock --address=203.0.113.7
```

Add `--realm=admin` or `--realm=customer` to lift only one area. A firewall ban of the address is lifted with `ispprotect:ip:remove`. Or open the tab **Sign-in protection** from another account. See Lifting locks and bans.

Held on the two-factor set-up page

Signs: after the sign-in, the Administration shows only the page **Two-factor**, and other requests answer with the code `ISPPROTECT__TWOFACTOR_SETUP_REQUIRED`. The shop requires a second step and your account has none, and you cannot set it up. Lift the requirement:

```
bin/console ispprotect:settings:set twofactor.required false
```

Lost device or locked out of two-factor

If you still have a recovery code, use it in the place of the code from the app. Each recovery code works once.

Without a recovery code, and with a lost app, there is no console command that removes the second step of one user. Another administrator can turn it off for that user through the Administration API route `POST /api/_action/ispprotect/twofactor/user/{userId}/disable`. The caller needs the right to update users and a recently confirmed password, and cannot use it for the own account. The Administration has no button for it. The call is recorded as the event `account.changed`.

Lifting the requirement with `twofactor.required` does not remove the second step of a user who already has one.

If you changed `APP_SECRET` of the shop, the stored keys of the second step cannot be decrypted, and users with a second step cannot pass the code check. Restore the old `APP_SECRET`.

The firewall refuses everything

If you cannot find the cause, switch the firewall off, sign in, and then repair the settings:

```
bin/console ispprotect:settings:set firewall.mode off
```

Set the mode to `log` to see what the firewall would do without refusals. See Firewall modes.

A rule or a trap path blocks the Administration

Signs: after you saved a block rule or a trap path, the sign-in page or the Administration is refused. Remove the setting:

```
bin/console ispprotect:settings:set rules.custom '[]'
bin/console ispprotect:settings:set trap.paths '[]'
```

Then remove your own address from the ban list, as in Your own address is banned or blocked.

All visitors have the same address

Signs: the firewall bans an address and nobody can reach the shop, or the Events tab shows the same address for every visitor. The shop is behind a reverse proxy or CDN and Shopware does not know the proxy. Set `SYMFONY_TRUSTED_PROXIES` to the addresses of your proxies. See Behind a reverse proxy or CDN. Then lift the ban of the proxy address, as in Your own address is banned or blocked.

Alerts, summary mails or scans do not run

The scheduled task `ispprotect.maintenance` runs only when the Shopware scheduler and a message consumer run. Test it once with:

```
bin/console ispprotect:maintenance
```

If this works, the scheduler or the consumer is not running. See Scheduling. The check “The file scan ran in the last {days} days” in the checkup shows the same problem. See Security checkup.

Mails do not arrive

Check **Notification address**, **Immediate alerts** and **Summary mail**, and the mail settings of the shop. A failed mail is recorded as the event `system.notice` with the reason `mail_failed`, and the events stay unsent. See Notification settings.

Messages from the plugin

- Status 503 with the code `ISPPROTECT__API_UNAVAILABLE`: the plugin could not read or write its data for a request of the Administration. Check the database and the file `var/log/ISPProtectWebSecurity.log`.
- Status 503 with the code `ISPPROTECT__TWOFACTOR_UNAVAILABLE` or sign-in error 1104: the state of the second step cannot be read. Check the database and the log file.
- The plugin writes its own problems to `var/log/ISPProtectWebSecurity.log`.

The OWASP CRS rules refuse a normal request

The OWASP CRS rules can refuse a normal request (a false positive). The request is answered with status 403, or with 413 or 400 for a request that cannot be inspected completely. See OWASP CRS rules for the background.

1. Open the tab **Events** and find the entry **Request refused by the firewall** (or **Request logged by the firewall**, in mode **Log only**). Note the rule numbers in the details.
2. Disable the rule that is wrong: enter its number in the field **Disable a rule** on the card **OWASP CRS rules** on the tab **Firewall**. A rule that you disable no longer runs for any request, so disable only the rule that caused the trouble.
3. If a save of a large amount of content fails with status 413 and the number `200101`, raise **Inspected body size** (`waf.body_limit`), for example to 524288.
4. If requests stop with the number `200100`, raise **Time limit per request** (`waf.time_budget_ms`), or disable the number `200100`. Program code pasted into a form field (about 20 to 40 KB) can use up the default limit of 200 milliseconds. Requests refused with the number `200005` (rarely, a long field with program code exceeds the regular expression limits of PHP) are answered the same way: disable the number `200005` or let requests pass that cannot be fully inspected.

To find the cause without losing visitors, set **CRS rules mode** to **Log only**: requests are then only recorded.

If you cannot sign in any more, switch the CRS rules off on the console, then sign in and disable the rule:

```
bin/console ispprotect:settings:set waf.mode off
```

Privacy and external services

What the plugin stores

The plugin stores its data in its own tables in the database of the shop, all with the prefix `ispprotect_`. It stores:

- the address lists, bans and sign-in locks
- failure counters
- the event log with addresses and user names, for **Keep log entries (days)**. With **Shorten addresses in the log** on, addresses are shortened before they are stored. See Event log settings.
- the baseline and the findings of the file scanner
- the second step of Administration users: the key (encrypted) and the hashes of the recovery codes
- an access token and a licence key of the remote scan, encrypted

Of an address with a sign-in lock for a protected administrator address, only a hash is stored.

The event log stores the names of changed fields, never their values. The plugin does not change tables of Shopware.

External services

These are the outside services the plugin can contact. The page for the second step shows no QR code, so that nothing is requested from another site.

Packagist

When **Check for known vulnerabilities** is on, each scan asks `https://packagist.org/api/security-advisories/` about the installed Composer packages. The plugin sends package names. The answers are cached. When Packagist cannot be reached, the check is skipped and the plugin records the event `system.notice`. Switch the check off to prevent the request.

Scan service and licence portal (remote scan)

The remote scan is off by default. Only when you choose a mode other than **Off**, the plugin contacts a scan service.

- **Local scan service (socket)**: a service on the same server. Nothing leaves the server.
- **Own scan service**: your own service at the address you set.
- **Hosted scan service (licence)**: the hosted scan service at `https://scan.ispprotect.com`, and the licence API at `https://license.ispprotect.com`. The account console at `https://console.ispprotect.com` receives no plugin requests.

To a scan service the plugin sends the SHA-256 checksum and a name hint of each non-empty code file it checks, and uploads a file only when the service does not know its checksum, up to **Upload limit per scan (MB)**. Files with secrets, such as keys, `.env` files and database dumps, are never sent. A few PHP configuration files are sent only in redacted form, with all strings, numbers and comments replaced. For the exact list and for what the licence portal receives, see What is sent and what never is.

OWASP CRS rules (GitHub)

The plugin downloads the OWASP CRS rules from `github.com` (and the download host that GitHub redirects to) only when you click **Install CRS 4.25.2** or **Update to CRS**, never in the background. The request asks for one fixed file, the rules archive of the CRS project. Nothing about your shop, its visitors or their requests is sent. The download is checked against a checksum built into the plugin. Requests to your shop are checked on your server and never sent anywhere.

Mail

Alert mails and the summary mail go through the mail service of Shopware to the **Notification address**. What happens to them after they leave the shop depends on the mail settings of the shop.

Switching off

You can switch each use off: **Check for known vulnerabilities** off, **Remote scan Off**, **Immediate alerts** off and **Summary mail Off**. See Settings reference.

Removing the plugin

Deactivate

Deactivating the plugin stops the firewall, the sign-in protection and the second-step check, because they run as part of the plugin. The data stays. Activating the plugin again checks that the tables exist and clears the cache pool of the plugin.

Uninstall

Uninstalling asks whether to keep the user data. In the Administration, open **Extensions > My extensions** and uninstall the plugin. On the console:

```
bin/console plugin:uninstall --keep-user-data ISPProtectWebSecurity
bin/console plugin:uninstall ISPProtectWebSecurity
```

- **Keep the data** (the dialog of the Administration without “delete all data”, or `--keep-user-data`): the tables stay. All settings, address lists, bans, locks, the event log, scan results and the second-step secrets of users stay. They are used again when you install the plugin again. Nothing is sent to the licence portal, and a connected licence stays connected.
- **Delete the data** (the dialog with “delete all data”, or `plugin:uninstall` without `--keep-user-data`): the plugin drops all its tables and clears its cache pool. Settings, lists, the event log, findings and the second step of every user are gone. Afterwards, users sign in with the password only.

The plugin changes no table and no file of Shopware, so nothing else needs to be cleaned up. Shopware removes its own migration records on uninstall. The log file `var/log/ISPProtectWebSecurity.log` stays in both cases. Delete it yourself if you do not need it.

What happens to a connected licence

When you delete the data and a licence key is stored, the plugin first releases the site at the licence portal. This happens in every remote scan mode. It is one call with a short timeout, so the site no longer counts towards the site limit of your licence.

If the portal cannot be reached or refuses, the uninstall goes on, and the plugin shows no message. Release the site in the licence portal yourself, at the address you received with your licence.

To release a site before you uninstall, use **Disconnect** on the tab **Scanner** or `bin/console ispprotect:remote disconnect`. See Remote scan and licence.

Remove the files

After the uninstall, remove the code:

- If you installed the plugin with Composer: `composer remove ispprotect/shopware-web-security`.
- Otherwise delete the folder `custom/plugins/ISPProtectWebSecurity`.

Then run `bin/console cache:clear`.