

ISPProtect Web Security for WordPress

ISPProtect Web Security for WordPress: Manual

Version 1.0.0

2026-11-23

Contents

1. **Introduction**
   - What the plugin does
   - Requirements
   - Free and paid parts
   - How this manual is organised
2. **Installation and first steps**
   - Install and activate
   - Multisite
   - The setup wizard
   - What to do next
   - Visitor address behind a proxy
   - Settings page, backup and reset
3. **The overview (dashboard)**
   - Cards
   - Other menu entries
   - Safe mode
4. **Firewall**
   - What the firewall checks
   - The modes
   - The status card
   - Extended protection
   - Rules
   - Automatic bans
   - Trusting signed-in administrators
   - Limits
   - Privilege-escalation guards
   - IP rules
   - OWASP CRS rules
5. **Malware scanner**
   - What a scan covers
   - Starting a scan
   - Schedule
   - Findings
   - Exclusions and limits
6. **Remote scan and licence**
   - Modes
   - The panel
   - What is sent
   - When the quota is used up
   - Remote scan on the command line
7. **Sign-in protection**
   - Login protection status
   - Lockouts after failed sign-ins
   - Moving the sign-in page
   - Bot checks
   - Password rules
   - Sessions
   - XML-RPC, application passwords and user lists

8. **Two-factor authentication**
    - Two-factor authentication settings
    - Methods
    - Setting it up as a user
    - Requiring it for roles
    - Signing in with two-factor authentication
    - When a user lost the device
    - The key file
9. **Hardening checkup**
    - Reading the result
    - Fixes
    - Checks without a fix
10. **Activity log and notifications**
    - What is recorded
    - The Activity page
    - Keeping and shortening the log
    - Email, summaries and webhook
11. **Settings reference**
    - General settings
    - Firewall settings
    - Login protection settings (sign-in protection and two-factor authentication)
    - Scanner settings
    - Hardening settings
    - Activity tracking settings
    - Notification settings
12. **Command line**
    - status
    - settings
    - blocklist
    - unblock
    - login
    - 2fa
    - hardening
    - scan
    - remote
13. **Getting back in and troubleshooting**
    - The emergency constants
    - You are blocked at sign-in
    - The firewall blocks your address
    - The OWASP CRS rules refuse a normal request
    - You lost the hidden sign-in address
    - You lost access to two-factor authentication
    - The site does not load after switching on extended protection
    - The site misbehaves after a hardening fix
    - No alerts arrive
    - A scan does not start or does not finish
    - Commands refuse to change anything
    - Collecting information for support
14. **Privacy and external services**
    - External services
    - Personal data the plugin stores

l5. Removing the plugin

- Deactivate
- Delete
- Keep your data
- Before you move or delete the site files

Introduction

ISPProtect Web Security is a WordPress plugin that protects a site in six areas: a firewall, a malware scanner, sign-in protection, two-factor authentication, a hardening checkup, and an activity log with alerts. Almost everything runs on your own server. You need no account for these parts.

What the plugin does

Area	What it does
Firewall	Checks every request before WordPress handles it. It applies your IP rules, automatic bans, optional country and rate limits, and rules you add yourself. See Firewall.
Malware scanner	Compares WordPress and plugin files with the originals, runs basic checks for suspicious files and code (thorough detection is left to the remote scan), checks the database and looks for updates. It only reports; it never changes, moves or deletes a file. See Malware scanner.
Remote scan	Optionally sends file fingerprints (and, within a limit, files) to a scan service for an extra check. See Remote scan and licence.
Sign-in protection	Blocks addresses after repeated failed sign-ins, can move the sign-in page, sets password rules and limits sessions. See Sign-in protection.
Two-factor authenticati on	Asks for a second step after the password. See Two-factor authentication.
Hardening checkup	Tests your site for common weak spots and applies fixes for many of them. See Hardening checkup.
Activity log and alerts	Records security events and changes on the site and sends you email or webhook alerts. See Activity log and notifications.

> The scanner reports problems; it cannot prove that a site is clean. The firewall checks what its engine and your rules describe; it does not ship a built-in list of attack patterns. Read Firewall for exactly what it checks.

Requirements

Requirement	Value
WordPress	6.4 or newer
PHP	7.4 or newer. On an older PHP the plugin does not run and shows a notice to administrators.
Multisite	Supported with network activation only. See Installation and first steps.

Free and paid parts

Everything in the plugin is free to use, with one exception: the hosted remote scan service needs a licence key. This manual does not state prices; ask your provider.

Part	Needs a licence key?
Firewall, sign-in protection, two-factor authentication, hardening checkup, activity log, alerts	No
Local scanner (WordPress files, plugin files, malware checks, database, updates)	No

Part	Needs a licence key?
Optional lookup of known vulnerabilities	No key. It contacts an outside service; see Privacy and external services.
Remote scan through a scan service on your own server (socket)	No site key. This is the local scan service of an ISPProtect server licence.
Remote scan through your own scan service (address and access token)	No licence key. You need the address and token of that service.
Remote scan through the ISPProtect scan service (licence key)	Yes

How this manual is organised

The chapters follow the menu of the plugin. Settings reference lists every setting with its key and default, and Command line lists every WP-CLI command. If you cannot sign in or your own address is blocked, go straight to Getting back in and troubleshooting.

Installation and first steps

Install and activate

1. In the WordPress admin area, open **Plugins > Add New**, upload the plugin or search for it, and install it.
2. Activate the plugin.
3. A new top-level menu **ISPProtect** appears. On the **Plugins** page the plugin row also has a **Settings** link.

The plugin creates a data folder for its own files. By default it is `wp-content/uploads/ispprotect-web-security`. The plugin puts guard files (`.htaccess`, `index.php`, `web.config`) into it so that its contents are not served to visitors. On nginx, add a rule yourself that denies access to this folder:

```
location ^~ /wp-content/uploads/ispprotect-web-security/ { deny all; }
```

The data files are PHP files that end at once, so a visitor cannot read their contents even without this rule. If you want the folder somewhere else, define the constant `ISPPROTECT_WEB_SECURITY_DATA_DIR` in `wp-config.php`.

Multisite

On a multisite network you can activate the plugin for the whole network only. Settings, IP rules, the activity log and the data folder are shared by all sites. The menu appears in the network admin and needs the capability to manage network options (`manage_network_options`). Only the main site runs scheduled scans. On a single site the menu needs the capability `manage_options`.

The setup wizard

Until you finish or skip the setup, the **Dashboard** shows a card **Finish the setup** with the buttons **Start setup** and **Skip**. The menu also contains a **Setup** entry until the setup is done. The wizard has these sections:

Section	What you do
Welcome	Introduction.
Your address	Shows the address the plugin sees as yours.
Alerts	Enter email addresses in **Send alerts to**. Tick **Send a weekly report every Monday morning** if you want the weekly report.
Optional checks that contact other services	Tick **Look up known vulnerabilities** and **Check new passwords against known data breaches** only if you accept that these checks contact an outside service. Both are off by default. See Privacy and external services.
First scan	**Run a full scan now** is ticked by default.

Press **Finish setup** to save. The addresses go into the setting `notify_email_list`. If at least one valid address is given, email alerts are switched on (`notify_email_enabled`). Invalid addresses are dropped and the wizard warns you. Press **Skip** to leave the wizard; the plugin then says “Setup skipped. You can change every setting later on the Settings page.” The wizard state is not reset by **Reset all settings**.

What to do next

1. Set up two-factor authentication for your own account on your profile page (**Users > Profile**, section **Two-factor authentication and sessions**). See Two-factor authentication.
2. Press **Always allow my address** on **ISPProtect > IP rules** if your address is stable. An always-allowed address skips every firewall check and is never locked out at sign-in. See IP rules.

3. Open **ISPProtect > Settings > General** and check **Read the visitor address from**. If your site is behind a proxy or Cloudflare, the plugin sees the wrong address until you choose the right source. See Visitor address behind a proxy.
4. Run the first scan on **ISPProtect > Scanner**.
5. Open **ISPProtect > Hardening** and work through the checkup.
6. Note the emergency ways back in; see Getting back in and troubleshooting.

Visitor address behind a proxy

Firewall, lockouts and rate limits all rely on the address of the visitor. The setting **Read the visitor address from** (`ip_source`) decides where the plugin reads it.

Choice	Meaning
Direct connection (REMOTE_ADDR)	The default. Use it when visitors reach your server directly.
X-Forwarded-For, **X-Real-IP**, **CF-Connecting-IP (Cloudflare)**, **True-Client-IP**, **Fastly-Client-IP**	Read the address from this header. Use it only when a proxy you trust sets the header.

Use **Trusted proxies** (`trusted_proxies`) to list the proxies whose header the plugin may believe. A header can be faked by a visitor; if the plugin cannot trust the source, the firewall status card warns that the address source can be spoofed. Changing `ip_source` or `trusted_proxies` with WP-CLI asks for confirmation unless you pass `--yes`.

Settings page, backup and reset

ISPProtect > Settings has the tabs **General**, **Firewall**, **Login protection**, **Scanner**, **Hardening**, **Activity tracking** and **Notifications**. Chapter Settings reference lists every setting.

ISPProtect > Tools offers:

Section	What it does
Back up settings	**Download settings** saves the file `ispprotect-web-security-settings.json`. It contains no log and no IP rules, and secrets are left out.
Restore settings	Choose the file in **Settings file** and press **Restore settings**. The file may be at most 262144 bytes.
System information	A block of facts about your site and server with a **Copy** button.
Reset	**Reset all settings** returns every setting to its default after you confirm “Reset all settings to their defaults?”. The log and the IP rules are kept.

The overview (dashboard)

ISPProtect > Dashboard shows what the plugin did recently. Every page of the plugin has a status pill in its header:

Pill	Meaning
Protected	The protection is on.
Safe mode	The plugin records events but enforces nothing. See Safe mode.
Needs attention	Something needs your action.

Cards

Card	Content
Protection	**On**, **Paused (safe mode)** or **Not available**.
What happened	Counts for **Last 24 hours** and **Last 7 days**: **Blocked requests**, **Lockouts**, **Failed sign-ins** and **Changes**.
Activity over 30 days	A chart of the events of the last 30 days.
Latest warnings	The eight latest warnings, with the link **See all warnings in the activity log**.
Most blocked addresses, last 7 days	The addresses the firewall refused most often.

When nothing was recorded yet, the dashboard says **All quiet so far**. Until you finish or skip the wizard, the card **Finish the setup** appears at the top; see Installation and first steps.

Other menu entries

Entry	Chapter
Firewall	Firewall
Login protection	Sign-in protection
Scanner	Malware scanner
Hardening	Hardening checkup
Activity	Activity log and notifications
IP rules	IP rules
Settings	Settings reference
Tools	Installation and first steps

Safe mode

Safe mode is an emergency switch. It is turned on with the constant `ISPPROTECT_WEB_SECURITY_SAFE_MODE` in `wp-config.php`. While it is on, the plugin keeps recording events but enforces nothing:

- The firewall, including the block list, only writes log entries.
- Sign-in lockouts and the privilege-escalation guards are inactive.
- A hidden sign-in address is switched off.
- Two-factor authentication is not asked.

The exact steps are in Getting back in and troubleshooting.

Firewall

The firewall runs before WordPress handles a request. It consists of an engine, IP rules, optional limits and a list of rules. The list of built-in firewall rules ships empty; you can add your own rules (see Your own rules). Attack detection comes from the optional OWASP CRS rules (see OWASP CRS rules). The firewall fails open: if it hits an error, the request goes through.

Open **ISPProtect > Firewall** to see its state.

What the firewall checks

For every request the engine works through these steps in order:

1. **Always-allowed address.** A request from an address with an “Always allow” rule skips everything below.
2. **Site block list.** A request from a blocked address (whole-site rule) is refused with status 403.
3. **Automatic ban.** An address that is banned or about to be banned is refused, unless the signed-in administrator pass applies. If the firewall mode is **Off: only the block list applies**, the engine stops after these three steps, except for the OWASP CRS check in step 6.
4. **Limits.** In this order, if switched on: the country rule, the rate limit, and the watch for missing pages.
5. **Rules.** The list of firewall rules is checked, except for holders of the administrator pass. A match with high confidence is refused in block mode and counts as a strike toward an automatic ban. A match with low confidence is only logged. In learning mode every match is only logged.
6. **OWASP CRS rules.** If the CRS rules are installed and their own mode is not **Off**, the request is checked against them, also for signed-in administrators. This step does not depend on **Firewall mode**.

In safe mode every refusal becomes a log entry and the request goes through.

The modes

The setting **Firewall mode** (`firewall_mode`) has three values.

Value	Effect
Block rule matches (default)	Refuses requests as described above.
Learning mode: log only	Rule matches are only logged. Use it to see what your own rules would do before they refuse anything.
Off: only the block list applies	Only the block list and automatic bans apply.

The status card

The card **Firewall status** shows one of the badges **Not set up**, **Out of date** or **Set up and up to date**, the time **Configuration written:**, and the button **Rebuild** that writes the configuration again. It warns about:

- safe mode,
- a visitor-address source that can be spoofed (see Visitor address behind a proxy),
- requests larger than the firewall reads in the last 24 hours,
- a `rules-local.php` file that is invalid or too large,
- a failure to write the configuration.

Extended protection

Normally the firewall starts when WordPress loads the plugin. The card **Extended protection** loads it earlier, through the PHP setting `auto_prepend_file`, before WordPress and all other plugins. This also covers PHP files that are requested directly. The state is **Off**, **Being switched on**, **Waiting for the server**, **On** or **Failed**. The buttons are **Switch on**, **Try again** and **Switch off**.

How the plugin sets it up depends on how PHP runs:

PHP runs as	What the plugin writes
mod_php (Apache) or LiteSpeed	A block with a `php_value` line in `.htaccess`.
CGI or PHP-FPM	A block in the file `.user.ini`.
Others (for example nginx with its own PHP settings)	Nothing. The plugin cannot set the server up and shows the lines for you.

The block is wrapped between the lines `BEGIN ISPProtect Web Security Firewall` and `END ISPProtect Web Security Firewall`. A small loader file `ispprotect-firewall.php` is written next to the configuration file. After each change the plugin sends a test request to your site. If the site does not answer or the loader did not run, it undoes the change and shows the reason, for example “The site did not answer correctly after the change, so it was undone.” or “The server ignored the setting, so it was undone. Use the manual setup.”

Further facts:

- PHP caches `.user.ini` files, so the state can stay **Waiting for the server** for a few minutes. The plugin checks again by itself.
- The `.user.ini` method can be switched on only in the dashboard, never with WP-CLI.
- If the configuration file already sets `auto_prepend_file`, the plugin does not change it. Use the manual setup and include both files.
- If the loader file is missing while the setting is on, PHP stops working on the site. Switch extended protection off before you move the site or delete files.

> If a step fails, press **Set it up by hand instead**. The section **PHP settings** shows the lines to add (with **Copy**). Add them to the PHP settings of your site, or ask your host to do it.

> If extended protection leaves the site unreachable, remove the lines between `BEGIN ISPProtect Web Security Firewall` and `END ISPProtect Web Security Firewall` from `.htaccess` or `.user.ini` in the home folder of the site. After that you can delete `ispprotect-firewall.php`.

When you deactivate the plugin, the plugin removes extended protection. If you deactivate it as the root user with WP-CLI, it cannot do that and tells you how to finish. When you delete the plugin, the empty loader file can stay on purpose.

Rules

The card **Rules** is a table with the columns **Rule**, **Source** (**Your own file** or **Plugin**), **Matches in 30 days**, **State** and **Action**. On a site without rules it says “There are no firewall rules on this site. The plugin ships no detection rules. Rules appear here when you add your own in the file rules-local.php.”

Your own rules

You can add rules in the file `rules-local.php` in the data folder. The plugin never creates it. It has these properties:

- The file must be at most 262144 bytes.
- The first line must be exactly `<?php exit; ?>`. The rest of the file is a JSON list of rules.
- The file is read as data and never run.
- If the file is invalid, the whole file is ignored and the problem is reported on the status card and in the log. A single rule that does not validate is skipped.

Each rule is an object with these keys: `id`, `category`, `confidence`, `parts`, a text to look for (`literal`, or a regular expression), `raw`, `paths` and `title`. `confidence` is `high` (may refuse the request in block mode) or `low` (only logged). `parts` names the parts of the request to examine: `method`, `path`, `query`, `args`, `body`, `cookies`, `headers`, `files`, `names`, `raw`, or `header:<name>` for one header.

Exceptions

An exception stops a rule from applying to a request. You can add up to 500. The form **Add an exception** has the fields **Rule ID**, **Path** (starts with a slash), **Parameter** and **Note** (up to 190 characters). **Make an exception from a recent request** offers three scopes: **This parameter on this path**, **This parameter everywhere** and **Everything on this path**. Exceptions that no request used show **Not applied**.

Limits of exceptions:

- A path alone cannot cover the whole site, `/index.php`, the REST index or a bare namespace.
- An exception on a path covers the whole request, including batch and embedded REST requests.
- The form `/index.php/...` needs its own exception.

Automatic bans

If **Ban addresses automatically** (`firewall_autoban_enabled`, default on) is on, an address that causes **Blocked requests before a ban** (default 10) blocked rule matches within **Counted within** (default 10 minutes) is banned for **Ban duration** (default 60 minutes). Bans only come from rule matches, so on a site without own rules no automatic ban of this kind occurs.

Trusting signed-in administrators

Trust signed-in administrators (`firewall_admin_pass`, default on) lets signed-in administrators skip the rule list and prevents strikes toward an automatic ban. The block list, country rule, missing-page watch and rate limit still apply to them. The pass ends at sign-out.

Limits

Limit	Settings	Behaviour
Rate limit	**Limit requests per address**, **Requests per minute**	Beyond the number of requests per minute, the firewall answers 429 “Too many requests”. Signed-in administrators count too. The loopback address, the address of the server itself and `0.0.0.0` are exempt. Default: off, 240 requests per minute.
Missing pages	**Watch for scans of missing pages**, **Not-found pages per address**, **Counted within**, **When the limit is reached**	Counts requests for pages that do not exist. **Log it** or **Log it and ban the address** (the ban lasts as long as **Ban duration**). Default: off, 30 pages in 5 minutes, log only.
Country	**Limit access by country**, **Countries in the list are**, **Countries**, **Applies to**	Blocks the listed countries (**Blocked**) or only allows them (**The only ones allowed**), on **The whole site** or for **Sign-in only**. Default: off.

The country rule needs the country from the web server or from Cloudflare. A visitor with an unknown country is never blocked. Countries are entered as two-letter codes.

Privilege-escalation guards

Stop privilege escalation (`firewall_guards`, default on) stops the changes that an attacker typically makes after finding a way to write options or user data: opening registration, making new users administrators, granting a user or a low role capabilities that give control over the site, and adding network administrators. A change is stopped only when the user making it could not make it through the dashboard either. Changes made by WP-CLI, during installation and for new

network sites are not checked. In safe mode a stopped change is logged but allowed. The guards record these events: `guard_default_role`, `guard_registration`, `guard_role_granted`, `guard_admin_created`, `guard_role_changed` and `guard_super_admin`.

IP rules

ISPProtect > IP rules lists the rules for single addresses and ranges. The card **Your address** shows your address and offers **Always allow my address**.

To add a rule, use **Add a rule**:

Field	Values
IP address or range	One address or a range.
What should happen?	**Block** or **Always allow**.
Where to block	**The whole site** or **Sign-in only**.
For how long	**Permanently**, **For 1 hour**, **For 24 hours**, **For 7 days**, **For 30 days**.
Note (optional)	Up to 190 characters.

Press **Add rule**. The plugin refuses to block your own address and any address that is always allowed. In the list **Current rules** you can filter by **Type**, **Applies to**, **Added by** (**You** or **Automatically**) and a search text, and remove rules with **Remove selected rules**. The list shows 50 rules per page (**Rules per page**). IPv6 lockouts apply to a /64 range.

You can also block or allow an address straight from an entry in the activity log.

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. The plugin can check every request against the CRS rules. The CRS rules are the part of the firewall that looks for attack patterns. The plugin adds only three rules of its own, for requests the CRS rules do not recognise (see ISPProtect rules). The plugin runs the rules with its own engine in PHP, inside the firewall, so no extra program or server module is needed. The rules are off until you install them.

The card **OWASP CRS rules** on **ISPProtect > Firewall** shows the state and installs, updates and removes the rules.

Installing the rules

1. Open **ISPProtect > Firewall** and press **Install the 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. 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 plugin also downloads the CRS rule exclusions for WordPress (see Exclusions for WordPress) from a fixed address on `raw.githubusercontent.com`, checked in the same way.
4. The first time you install, the plugin sets **OWASP CRS rules** to **Log matches only**.

The card then shows the version, the time of the installation, the checksum, the number of rules in use and the rules that are not supported (see What the CRS rules do not cover), the paranoia level and the threshold. After a plugin update, the card can say that a new version of the rules is available or that the rules have to be compiled again. Press **Update** for both. **Remove the CRS rules** deletes the downloaded rules. When you delete the plugin, they are deleted too.

Start with logging

Leave **OWASP CRS rules** on **Log matches only** first. In this mode a request that matches is only written to the log and the card, and is never refused. Look at the card for a few days, in which all kinds of pages, forms and the dashboard are used, switch off the rules that report normal requests of your site (see below) and only then set the mode to **Block matches**. The three modes are:

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

The CRS check has its own mode and does not depend on **Firewall mode**. It runs after the always-allowed addresses, the block list, automatic bans and the limits: a request from an address with an “Always allow” rule is never checked with the CRS rules. Signed-in administrators are checked as well. In safe mode the CRS rules only log.

Reading an entry

The card shows the latest requests that matched, in the section **Recent CRS matches**: the time, the address of the request (without the query string), the result (**Blocked** or **Only written down**) and the rules that matched. For each rule you see its number and the message of the CRS, for example `942190` and the name of the attack it suspects. The log entry itself holds more detail: the rule numbers and messages, the name of the request part that matched (for example `ARGS:comment` or `REQUEST_COOKIES:name`), a short excerpt of the value with the sensitive part masked, the total score, the method and the path. It never holds 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 score 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.

A rule reports a normal request

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

1. In **Recent CRS matches**, look at the rule number and message of the entry.
2. Press **Switch this rule off** next to the number. The number is added to **Switched-off CRS rules** and the rule no longer runs for any request.

You can also enter rule numbers by hand, one per line, in **Switched-off CRS rules** (`firewall_crs_disabled_rules`) in the settings. A switched-off rule protects no request, also not the ones it was right about; switch off only the rule that is wrong, not a whole group. Prefer a rule that matched your own forms over raising the threshold.

Paranoia level and threshold

The **Paranoia level** (`firewall_crs_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 builders. Raise the level only after the entries at the level before are clean, one step at a time, and again start with logging.

The **Anomaly score threshold** (`firewall_crs_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 plugin 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 **Block matches** mode it is refused, in **Log matches only** mode it is logged. 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
`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 plugin inspects (1000).	403
`200100`	The time limit for the request was used up before all rules ran.	403
`200101`	The body is larger than **Request body checked**.	413
`200102`	A value is longer than the plugin inspects (65536 bytes).	403

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

- **Request body checked** (`firewall_crs_body_limit` , 1024 to 1048576 bytes, default 131072 = 128 KiB). A larger request cannot be fully inspected. The WordPress block editor can send more than that when it saves a long article (a save of about 140 KB of content ends with status 413 and the number `200101`). If large saves fail in block mode, raise the value, for example to 524288, or let such requests pass (next setting).
- **Time limit per request** (`firewall_crs_time_budget` , 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 page builder 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 (`firewall_crs_pass_uninspected` , default off) lets such requests through (they are still logged). 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 switch off 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 **Block matches** mode 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 switch either number off explicitly in the settings.

Exclusions for WordPress

WordPress sends requests that look like attacks to the CRS rules, for example the saving of posts with HTML, the REST API of the block editor, the sign-in or the customizer. The CRS project publishes a set of rule exclusions for this, the “CRS rule exclusions for WordPress”. The plugin downloads them with the rules and loads them before the CRS rules. They are part of the checked download and cover the core of WordPress; plugins are not covered, so a plugin with unusual forms can still need a switched-off rule.

The plugin also generates exclusions for your own site. WordPress puts the address of the site into some request arguments (`redirect_to` , `_wp_http_referer` and `referer`), for example after signing in. The CRS rules 931100 and 934110 report addresses in arguments. The generated rules (numbers from 10100) switch these two rules off for the three arguments as long as the value is an address of this site or a path on it. An address of another site is checked as usual. The rules are generated from the home address and the WordPress address; a change of these addresses is picked up when the rules are compiled again.

> If the address of your site is an IP address or `localhost` instead of a host name, some CRS rules treat requests to it as an attack (rule 920350 reports a numeric host name in the `Host` header). For this case the plugin switches rule 920350 off for requests to exactly that address, and the card shows a warning. A site that is reachable from the internet should use a host name.

ISPProtect rules

The plugin ships three rules of its own in `rules/ispprotect-wordpress.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 matches only** or **Block matches**: 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 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 (`index.php.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` , `~` and every copy of `wp-config.php` themselves.
`12100`	A PHP script in the uploads folder `wp-content/uploads` , also `name.php.jpg` and `name.php/more` . A site that moved its uploads folder is not covered.

When the rules help:

- The rules see only requests that reach PHP. On most servers, a request for a file that does not exist goes to WordPress, 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. Some servers answer a missing `.php` file themselves (nginx with `try_files $uri =404`); such a request never reaches WordPress.
- A file that exists and is not a PHP script, for example a real copy `index.php.orig` , is sent by the web server without PHP. The rules never see that request and cannot stop it.
- Rule `12100` also sees a PHP script in the uploads folder that exists, but only with extended protection, where the firewall starts before every PHP script. Without it, such a script runs without the firewall.

When rule `12100` refuses the test request of the Hardening checkup for PHP in the uploads folder (the plugin’s own refusal page answers), the check **No programs can run from the uploads folder** shows **Check by hand**, because only the firewall refused the script and not the web server. Check the server configuration, or the fix of that check.

The real fix is the configuration of the web server: it refuses version control folders and copies of files and runs no PHP in the uploads folder. The fix **No programs can run from the uploads folder** of the Hardening checkup writes the server rules for the uploads folder or shows them to you; the fix **Sensitive files cannot be downloaded** does the same for the common names (`.git/` , `.svn/` , `.hg/` , `.env` , copies of `wp-config.php` , endings such as `.bak` , `.old` and `.orig`), not for every name these rules report. Better still, keep no version control folders and no copies of files in the folders of the site.

The rules are switched off like CRS rules: one rule by its number (**Switch this rule off** in **Recent CRS matches**, or **Switched-off CRS rules** in the settings), all three with the line `SecRuleRemoveByTag ispprotect` in your own rules file (see below). The tags `ispprotect/source-disclosure` (`12001` , `12002`) and `ispprotect/upload-script` (`12100`) switch off a group. A plugin update can bring changed rules; the plugin compiles the rules again after the update.

Your own CRS rules file

You can add your own rules in SecLang, the language of the CRS, in the file `crs-own.conf` in the data folder (the folder of `rules-local.php` , see Your own rules). The plugin never creates the file. The card 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 plugin 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 reads the file again when it changed. It compiles the rules again when you open the dashboard, once a day, and when you press **Update** on the card. Until then the version compiled before stays in use. A change that cannot be compiled is skipped; the card and the log tell you.

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 card 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 of WordPress and its plugins.

Cost of the check

On a normal request the check costs 2 to 10 milliseconds. A save with 42 KB of page builder 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 without the dashboard:

- With WP-CLI: `wp ispprotect settings set firewall_crs_mode off`.
- Without WP-CLI: add `define( 'ISPPROTECT_WEB_SECURITY_SAFE_MODE', true );` to `wp-config.php`. In safe mode the CRS rules only log and block nothing. When you can sign in again, set **OWASP CRS rules** to **Log matches only**, find the rule that caused the trouble and switch it off, and then remove the line.

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 card shows the same text. The licence text of the CRS and of the rule exclusions for WordPress is part of the download. Open it with **Licence of the OWASP CRS rules** on the card.

Malware scanner

The scanner looks for signs that your site was changed or infected. It only reports. It never changes, moves or deletes a file, and it cannot prove that a site is clean. Open it at **ISPProtect > Scanner**.

What a scan covers

Each part can be switched on or off in **ISPProtect > Settings > Scanner**, section **What to check**.

Check (setting)	What it does
WordPress files (`scanner_check_core`)	Compares the files of WordPress with the originals published by WordPress.org. It finds changed, unknown and missing files.
Plugin files (`scanner_check_plugins`)	Compares the files of plugins with the originals published by WordPress.org. It finds changed and unknown files, and plugins it could not compare.
Malware (`scanner_check_malware`)	Basic checks of code files: PHP in the uploads folder and a small set of generic suspicious-code checks. Thorough malware detection is done by the remote scan. It also reports unusual files such as downloadable backups or dumps, symbolic links and files with an unusual name.
Database (`scanner_check_database`)	Looks for injected scripts in the database and for a changed site address.
Updates (`scanner_check_updates`)	Reports available updates, closed or abandoned plugins and an outdated PHP version.
Known vulnerabilities (`scanner_vuln_check`)	Asks an outside service whether your WordPress version and your plugins and themes have known vulnerabilities. Off by default. See Privacy and external services.

All checks except **Known vulnerabilities** are on by default. Code files are read up to the size limit **Largest code file to read** (`scanner_max_file_size`, default 5 MB, 1 to 50). Larger files are read at the start and at the end only.

Starting a scan

Button	What it does
Start quick scan	Checks only files that changed since the last scan.
Start full scan	Checks every file.
Continue scan	Continues a scan that stopped.
Cancel scan	Stops the running scan.

The first scan is always a full scan. If you start a quick scan before that, the plugin says “A full scan was started instead of a quick one, because the scanner has to read every file this time.” Scheduled scans are quick scans.

A scan runs in short steps through WP-Cron, and the page continues it while it is open. The progress shows **Files found**, **Files checked** and **New findings**. If the scan does not move for more than five minutes the page says so. If WP-Cron is switched off on your site, scheduled scans cannot run and the page tells you.

The page also shows **Last scan**, **Next scheduled scan** (or **Not scheduled**) and **Open findings**. The list **Earlier scans** shows how each scan started (**By hand**, **By schedule** or **Command line**) and how it ended (**Finished**, **Stopped with an error**, **Cancelled** or **Running**).

Schedule

Setting	Key	Default
Automatic scan	`scanner_schedule`	**Every day**. Other values: **Off**, **Every week**.
Hour of the day	`scanner_hour`	3 (site time zone, 0 to 23).

On multisite only the main site schedules scans.

Findings

The list of findings has the tabs **Open**, **Ignored** and **Solved**, each with a count. **Solved** lists findings that a later scan no longer reports. The table has the columns **Severity**, **What was found**, **Where**, **First seen**, **Last seen** and **Actions**. You can filter by **Severity** (**Critical**, **Warning**, **Info**) and by **Kind**. The list shows 20 findings per page.

Kinds of findings

Kind	Meaning
Malware	Code the scanner judges to be malicious.
Malware found by the remote scan	The remote scan service judged the file as malicious.
Suspicious file	A file that looks suspicious.
Changed WordPress file	A WordPress file differs from the original.
Unknown file in WordPress	A file in the WordPress folders that is not part of WordPress.
Missing WordPress file	A WordPress file that should exist is gone.
Changed plugin file	A plugin file differs from the original.
Unknown file in a plugin	A file in a plugin folder that is not part of the plugin.
Plugin that could not be compared	No original was available for this plugin.
Closed plugin, **Abandoned plugin**	The plugin was closed or is no longer maintained.
Update available	An update exists.
Outdated PHP version	PHP no longer receives security fixes.
Known vulnerability	The outside service lists a vulnerability for your version.
Injected code in the database	A script was found in the database content.
Changed site address	The site address in the database differs from what is expected.
Suspicious scheduled task	A scheduled task that looks suspicious.
Administrator account	An administrator account that needs a look.
Downloadable backup or dump	A backup or database dump a visitor could download.
Symbolic link	A symbolic link in the site folders.
File with an unusual name	A file whose name stands out.
Folder that could not be read, **File that could not be read**, **File that was skipped**, **File checked only in part**	The scanner could not check this fully.

Kind	Meaning
Unknown file in the plugin data folder	A file in the data folder of this plugin that does not belong there.
Check that could not run	One check failed; the scan went on.

What you can do with a finding

The only actions are **Ignore** and **Stop ignoring**, and **Details**. Ignoring takes the finding out of the open list: “The finding is ignored. It no longer counts as open, unless the file changes again.” A solved finding cannot be ignored.

Details shows an **Explanation**, what the file was **Compared with**, fingerprints, the **Lines that raised the alarm** and, for changed files, **Changes compared with the original** (removed lines in red, added lines in green). The content of files that hold secrets is not shown, nor that of their backups and editor copies (for example `wp-config.php.bak`), of archives and database dumps (for example `.sql`, `.zip`, `.tar.gz`) or of files with more than one hard link. The list of findings leaves out the stored excerpt of such files too.

> The scanner never removes anything. After a finding, restore the original file, update or remove the plugin, or clean the file yourself, then scan again.

Exclusions and limits

Enter folders the scanner must skip in **Skipped folders** (`scanner_excluded_paths`), one per line, relative to the WordPress folder. The remote scan has its own exclusion list; see Remote scan and licence.

Remote scan and licence

The remote scan is an optional extra stage of a scan. The plugin looks up the code files of your site at a scan service. The local scan stays as it is: all local checks run in every case, and if the remote stage stops, the scan goes on without it. Remote findings appear as **Malware found by the remote scan**.

Modes

Choose the mode in **ISPProtect > Settings > Scanner**, section **Remote scan (optional)**, setting **Remote scan** (`scanner_remote_mode`, default **off**).

Mode	Label	What it is	Licence key
`off`	Off	No remote scan.	No
`daemon`	**Scan service on this server (socket)**	The local scan service of an ISPProtect server licence, reached through a socket on the same server. Nothing leaves the server.	No site key
`service`	**Own scan service (address and access token)**	A scan service you run or that your provider gave you. You enter its address and an access token. It is offered only if the server can store the token encrypted.	No
`license`	**ISPProtect scan service (licence key)**	The hosted service, used with a licence key.	Yes (paid)

The licence API at `https://license.ispprotect.com` and the hosted scan service at `https://scan.ispprotect.com` are built in; you only need the licence key. The mode `license` appears when the server can store the key encrypted. A developer can point the plugin to another deployment with the constants `ISPPROTECT_WEB_SECURITY_PORTAL_URL` and `ISPPROTECT_WEB_SECURITY_SCAN_URL` in `wp-config.php`; this is normally not needed. The account console at `https://console.ispprotect.com` is not the scan API; never use it as the scan address. A key that is already stored can still be removed with **Disconnect**.

Related settings:

Label	Key	Default	Meaning
Socket of the local scan service	`scanner_remote_socket`	`/run/ispprotect/scand.sock`	Absolute path, at most 104 characters, no spaces.
Address of the scan service	`scanner_remote_url`	empty	For the own scan service. It must start with `https://`; `http://` is allowed only for `127.0.0.1`, `[::1]` and `localhost`. WordPress refuses private-network addresses and ports other than 80, 443 and 8080 unless a filter allows them. Used only in the mode **Own scan service**; the licence mode and the socket mode ignore it.
Upload limit per scan	`scanner_remote_upload_mb`	200 (MB)	0 to 4096. With 0 only fingerprints and names are sent.
Never send these folders	`scanner_remote_exclude`	empty	Folders that are never sent.

The panel

ISPProtect > Scanner has a panel **Remote scan** with the mode, the service, a link to the scanner settings and the button **Test connection**. A successful test reports “The service answered.” with the engine, the signature set and the limits of the service. **Last error:** shows the last problem.

Access token

For the own scan service the panel has a section for the access token: enter it in **New access token** and press **Store token**. **Remove token** deletes it. The token is stored encrypted and never shown again.

Licence key

In licence mode enter the key in **Licence key** (8 to 80 characters) and press **Connect**. After that the panel shows:

Item	Meaning
Licence key	**Set**, with the last characters.
Plan	The plan of the licence.
Valid until	The end of the licence, or **No end date**.
Sites	How many sites use the licence, as “x of y”, or “x, without a limit”.
This site	Whether this site counts towards the limit.
Access	“Valid until …”, renewed every day, or “Expired. Use “Refresh now”.”

Buttons: **Refresh now** renews the access at once. **Disconnect** removes the licence from this site. **Disconnect licence** is shown when a licence is still connected while another mode is selected.

How the licence behaves:

- The plugin renews the access once a day, on the main site.
- If the licence portal refuses the key (for example expired or locked), automatic renewal stops until you press **Refresh now** or enter a new key. The panel says “Daily renewal is paused: …”.
- If the portal cannot be reached, the plugin tries again later, waiting longer each time (from 60 seconds up to a day). An existing access stays valid until it ends.
- If the clock of your server is wrong, the panel tells you to correct the server time and press **Refresh now**.
- To use another key, disconnect first.
- **Disconnect** removes the local licence data in every case. The plugin tells you whether the portal confirmed the release of the site. If no request could be sent, release the site in the licence portal yourself.

For problems with the licence itself, contact your provider. The plugin does not contain a support address.

What is sent

The remote scan sends the SHA-256 fingerprint and the file name (without the folder) of each code file. Files that the service does not know are uploaded, within **Upload limit per scan**. Images and other files that are not code are never sent. The service answers with a verdict per file. The result line reads “x files looked up, y uploaded (size), z with malware.”

`wp-config.php` and its copies whose name starts with `wp-config` and ends with `.php` (for example `wp-config-backup.php` or `wp-config-sample.php`), files named `localconfig.php` or `config.local.php`, and PHP files whose name only looks like a configuration file (such as `wp_config.php`, `my-wp-config.php` or `local_config.php`) are sent only in a redacted form. Before the fingerprint is taken and before the upload, all strings, numbers and comments in the 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`, and every name inside `{$...}` or `${...}` in a string (such as `"{$db[NAME]}"`) 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 the database password, the keys and salts and any other value never leave the server. Code that an attacker added to the file can still be found by its structure: the service sees calls such as `eval()`, `base64_decode()` or `include`, but not their literal contents. The fingerprint 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 with the PHP version of the server, 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. A finding in such a file is reported for the real file and says that it was checked

in redacted form. The same applies to `localconfig.php` and `config.local.php`, and to PHP files whose name only looks like a configuration file, such as `wp_config.php`, `wpconfig.php`, `my-wp-config.php`, `local_config.php`, `WPConfigTransformer.php` or a name with an invisible or non-Latin character that resembles `wp-config.php`, when the extension is one that PHP runs (`.php`, `.phtml`, `.inc` and similar).

These are never sent:

- other copies of `wp-config.php`, such as `wp-config.php.bak` or `wp-config.php~`,
- `.htaccess` and `.env`,
- other configuration files that hold secrets,
- keys and certificates,
- database files and dumps,
- `.ini` files such as `php.ini`,
- 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 `wp_config.php.bak`, `wpconfig.txt` or `local_config.yml`, also with an invisible or non-Latin character in the name,
- the folders `.git` and `.ssh`, folders named `var`, `config/jwt` and `config/secrets`, and files with `secret` in the name below `config/packages`,
- the data folder of this plugin,
- everything in your **Never send these folders** list.

The licence portal receives the licence key, an installation ID, the host name of the site (on multisite the main site), the platform, the product and the plugin version. See Privacy and external services.

When the quota is used up

If the service reports that the daily upload quota is used up (“The upload quota of the service is used up for today.”), the scan leaves the remote stage (“Stopped early: …”) and goes on with the local checks. If the **Upload limit per scan** is reached, the plugin says “The upload limit was reached; the remaining files are checked by a later scan.”

Remote scan on the command line

Every step is also available with `wp ispprotect remote`; see Command line.

Sign-in protection

Open **ISPProtect > Login protection** for the status and the lockouts. The settings are in **ISPProtect > Settings > Login protection**; Settings reference lists every key.

Login protection status

The card **Login protection status** shows **Active lockouts**, **Failed sign-ins in the last 24 hours**, **Users with two-factor authentication** and **Sign-in address**. The sign-in address reads either “The standard WordPress address. It is not hidden.” or “Your sign-in page is hidden at this address. Keep it private and do not share it.” The card warns about safe mode, about the constant `ISPPROTECT_WEB_SECURITY_DISABLE_2FA` and about a missing or unreadable key file (`login-key.php`).

Lockouts after failed sign-ins

The plugin counts failed sign-ins on `wp-login.php` , through XML-RPC and with application passwords. It does not count sign-ins from always-allowed addresses, and it never locks them out.

Setting	Default
Block addresses after repeated failed sign-ins	on
Failed sign-ins allowed per address	5 (2 to 100)
Count failed sign-ins over	15 minutes (1 to 1440)
First block lasts	15 minutes (1 to 1440). Each further block within a day lasts four times as long as the one before.
Longest block	24 hours (1 to 720)

Requests without a usable address share one counter. An IPv6 address is blocked as a /64 range.

Password checks from one address run one at a time, so a burst of parallel requests gets no more guesses than the limit allows. This also covers the second sign-in step and the password check before profile changes. A request that has to wait more than one second for the check before it gets the answer “Another sign-in from your address is being checked right now. Please try again in a moment.” (HTTP 429 with `Retry-After: 1`). It is not counted as a failed sign-in, and the event log shows “Parallel sign-ins from one address refused” at most once per address per minute. Requests with an empty password, application password or second-step code do not wait. The queue uses named locks of the database server, whose names are keyed with the authentication salt of the site, so other accounts on a shared database server cannot hold them. On MySQL before 5.7.5 and MariaDB before 10.0.2 the checks are not queued, because these servers hold only one named lock per connection; blocks still apply from the next request on. Amazon Aurora is recognised by its version string.

A failed sign-in that cannot be saved, for example because the counter table is damaged, is not counted. The event log shows it as a critical “Failed sign-in” entry, and the PHP error log gets a line, at most once a minute. If the counters cannot be read, failures read as zero without a separate entry.

The lock reasons shown in the table are “Too many failed sign-ins from this address”, “Failed sign-in for a username that is under attack” and “Sign-in attempt with a blocked or unknown username”.

Attacks on one username

Failed sign-ins allowed per username, from all addresses (default 10, 0 to 1000; 0 switches it off) stops guessing from many addresses at once. The password step of the account is never locked: the plugin only blocks the addresses that try. The second step has its own limit per account (see the chapter on two-factor authentication).

Unknown and blocked usernames

Setting	Default	Meaning
Block right away when someone tries a username that does not exist	off	A single attempt locks the address. This reveals which names exist, because names that exist are treated differently.
Block right away when one of these usernames is tried	empty	A list of names, for example `admin`. It applies only while no account has that name.
Do not reveal whether a username exists	on	Error messages are the same for a wrong name and a wrong password.

The unlock link

With **Let blocked users request an unlock link by email** (default on), a blocked user can ask for an email with a link to `wp-login.php?action=ispprotect_unlock`. Each link works once and for one hour. It forgives only the failed sign-ins of the account the link went to. Requests are limited: 10 per address per day, 3 links per address and account per day, and per account 3 per hour and 6 per day.

A blocked address can still use logout, the password-protected-post form, and the password reset forms (`lostpassword`, `retrievepassword`, `rp`, `resetpass`), as well as the unlock page.

Lifting a lock

On **ISPProtect > Login protection** the table **Lockouts** shows **Address or range**, **Reason**, **Since**, **Ends** (or "Does not end") and a button **Lift**. The form **Lift the lockout of an address** takes an **IP address**, a checkbox **Also lift lockouts of ranges that contain this address** and the button **Lift lockout**. For permanent rules and for always-allowing an address use IP rules. In the command line: `wp ispprotect login unlock <ip>`.

Moving the sign-in page

With **Move the login page to its own address** (`login_hide_enabled`, default off) and **Login address** (`login_hide_slug`) the plugin serves the sign-in page at `your-site/<address>`. Visitors who are not signed in get a 404 page ("The page you are looking for does not exist.") on `wp-login.php` and `wp-admin`.

Rules for the address: at least three characters, lowercase letters, digits and dashes only. Names that WordPress uses are refused, and so is an address used by a page or post.

The new address is emailed to the user who sets it (mail subject "New login address"). Keep the mail. If you lose the address, see Getting back in and troubleshooting. `wp ispprotect login status` prints the current address.

Bot checks

Setting	Default	Meaning
Add an invisible trap field to the login, registration and password forms (`login_honeypot`)	on	A hidden field that people do not see and bots may fill in.
Make the browser solve a small puzzle before signing in (`login_pow`)	off	Needs JavaScript in the visitor's browser.
Puzzle difficulty (`login_pow_difficulty`)	16 (8 to 24)	A higher value takes the browser longer.

Password rules

Setting	Default
Apply the password rules to these roles (`login_pw_roles`)	Administrator and Editor. If no role is ticked, the rules apply to everyone.

Setting	Default
Minimum password length	12 (0 to 64; 0 switches the check off)
Reject common and easily guessed passwords	on
Reject passwords that contain the username, email address or site name	on
Check new passwords against known data breaches	off
Ask for a new password after (days)	0 (0 to 3650; 0 never asks)

The breach check sends only the first five characters of a hash of the password to an outside service; see Privacy and external services. If the service does not answer, the password is accepted.

Sessions

Setting	Default	Meaning
Sessions per user at the same time	0 (0 to 100)	When a user signs in beyond the limit, the oldest session ends. 0 means no limit.
Sign out after inactivity	0 minutes (0 to 10080)	0 means never.
Apply the two session limits above to these roles	none	Choose the roles the limits apply to.

On each user's profile, the section **Two-factor authentication and sessions** lists the sessions with **This device**, **End**, **Sign out everywhere else** and **End all sessions**. **Email users when they sign in from a new device or network** (default on) sends a mail on an unknown device. The plugin remembers up to 20 devices per user.

XML-RPC, application passwords and user lists

Setting	Default	Meaning
XML-RPC (`login_xmlrpc`)	**Restricted**	**Unchanged**, **Restricted** (pingbacks and the method that tries many passwords in one request are removed) or **Off**.
Application passwords (`login_app_passwords`)	**Allowed**	**Allowed** or **Off**.
Stop visitors from listing usernames through ?author= links	on	
Hide the user list in the REST API from visitors who are not signed in	on	
Leave author pages out of the sitemap	on	
Leave the author out of embed previews	on	

Application passwords and XML-RPC are counted as sign-ins by the lockout. Application passwords keep working for users with two-factor authentication.

Two-factor authentication

Two-factor authentication asks for a second step after the password. The plugin checks the password as usual. If it is right, the plugin asks for the second step before WordPress signs the user in. Sign-ins outside the sign-in page (XML-RPC, REST, AJAX, WP-CLI) are refused for accounts with two-factor authentication. Application passwords keep working because they are separate credentials.

Two-factor authentication settings

Find them in **ISPProtect > Settings > Login protection**.

Label	Key	Default	Meaning
Offer two-factor authentication	`login_2fa_enabled`	on	Users can set it up.
Methods users can choose	`login_2fa_methods`	**Authenticator app** and **Code by email**	The methods offered.
Require two-factor authentication for these roles	`login_2fa_required_roles`	none	Roles that must set it up.
Time to set it up	`login_2fa_grace_days`	7 (0 to 90)	Days before the requirement applies.
Let users skip two-factor authentication on a trusted device for	`login_2fa_remember_days`	30 (0 to 365)	0 asks at every sign-in.

Methods

- **Authenticator app.** The user scans a QR code with an authenticator app and enters the six-digit code it shows. The secret is stored encrypted with a key in the file `login-key.php` in the data folder.
- **Code by email.** The plugin sends a six-digit code to the email address of the account at each sign-in. Codes are valid for 10 minutes, and a user has five tries for a code.
- **Backup codes.** Ten codes, each valid once, as a way in when the usual method is not available.

Setting it up as a user

Each user sets it up on the profile page (**Users > Profile**, section **Two-factor authentication and sessions**).

1. For the app: press **Set up**, scan the code (or enter the key shown under “Cannot scan? Enter this key in the app:”), type the six-digit code into **Code** and press **Turn on**. To change phone, use **Move to a new phone**.
2. For email codes: press **Turn on** under **Codes by email** and enter the code sent to you.
3. Turning a method on or off, and creating new backup codes, needs the **Current password**.
4. Press **Create new codes** under **Backup codes**. The codes are shown once: “Keep these codes in a safe place. Each code works once if you cannot use your usual method. They are shown only now.” Press **I saved my codes** after you saved them. The profile shows how many unused codes are left.

Under **Trusted devices** a user can **Remove** one device or **Remove all**. A user cannot turn off the last method if the account is required to have two-factor authentication.

Requiring it for roles

If you choose roles in **Require two-factor authentication for these roles**, users of these roles get the **Time to set it up** period. Their profile says they have to set it up by a date. After that date, sign-in leads to the page **Set up two-factor authentication** (“Your account needs two-factor authentication. Please set it up now to finish signing in.”).

Signing in with two-factor authentication

After the password the plugin asks for the code of the method. The page offers **Use your authenticator app**, **Get a code by email** and **Use a backup code**. A user can tick the option to trust the device for the number of days set in the settings.

One sign-in allows five wrong codes; then the user has to enter the password again. Wrong codes also count per account, from all addresses together: after 10 wrong codes within 15 minutes the second step of that account is refused from every address with **Sign-in blocked** (“Too many wrong codes were entered for this account.”), until the oldest of these codes is 15 minutes old. Someone who knows the password therefore cannot guess codes from ever new addresses. A successful sign-in (also with a trusted device) and **Reset two-factor authentication** end the count. Lifting the lockout of the user’s address (form **Lift the lockout of an address** on **ISPProtect > Login protection**, `wp ispprotect login unlock` or `wp ispprotect unblock`) or an unlock link for the account lets that address try again at once; other addresses stay refused. The event log shows the start of such a block as “Second step of an account refused after too many wrong codes”.

When a user lost the device

1. The user signs in with a backup code and then sets up a new method.
2. If the user has no backup code, an administrator resets the account. On the user’s profile (**Users > Edit user**) press **Reset two-factor authentication**. The user can then sign in with the password alone until the setup is done again. An administrator cannot reset their own account this way, and cannot reset an account above their own level (for example a super administrator on a network).
3. With WP-CLI: `wp ispprotect 2fa reset <user>`. See Command line.

If you lose access yourself and no other administrator exists, see Getting back in and troubleshooting.

The key file

The file `login-key.php` in the data folder holds the key that encrypts the authenticator secrets. If it is missing or unreadable, the plugin falls back to codes by email and records a critical event. Authenticator secrets that were encrypted with a lost key cannot be read: each user has to set up the app again. Restore the file from a backup if you have one. A new key cannot read the old secrets, so do not delete the file.

Hardening checkup

ISPProtect > Hardening tests your site for common weak spots. Some tests send a request to your own site. For many weak spots the plugin can also apply a fix.

Reading the result

The card **Security checkup** says: “This site was tested with n checks. n pass, n need attention and n need a manual look.” It shows the time of the last check (**Last checked:**). The result is kept for five minutes. Press **Run the checks again** after you change something on your server. The checkup changes nothing by itself.

Every check has one of four states:

State	Meaning
Passed	The weak spot is not present.
Needs attention	The weak spot is present. If the plugin can fix it, the check has a button.
Check by hand	The plugin cannot decide or cannot fix it. Look at the text of the check.
Does not apply	The check makes no sense on this site or server.

When the plugin’s own firewall refuses a test request (for example rule `12100` for the test script in the uploads folder), the check shows **Check by hand**: the answer proves nothing about the web server. Check the web server configuration, or use the fix of the check.

A score from 0 to 100 sums up the result: the weight of the passed checks divided by the weight of all checks that apply. Checks that do not apply are left out. `wp ispprotect status` and `wp ispprotect hardening` show it as well.

Checks are grouped as **Files and server**, **Browser protection (security headers)**, **WordPress settings**, **Accounts and access** and **Updates and software**. Open **What this checks and why it matters** under a check for its explanation.

Fixes

A check that has a fix shows **Apply fix**. Some fixes first show a note headed **Please confirm first.** and the button **Yes, apply this fix**. When a fix is on, the check says “This fix is switched on.” and offers **Undo fix**. Every fix is a setting on the tab **Hardening** of the settings page, so you can also switch it there.

On Apache and LiteSpeed the fixes that need server rules write them to `.htaccess` (and, for the uploads folder, to a `.htaccess` file there). On nginx and other servers the plugin writes no server file. Open **Show the settings to add by hand** under a check to get the text for **Apache (.htaccess)** and **nginx** with a **Copy** button, and add it yourself. If the plugin cannot write the file you get the same text.

Fix (setting key)	What it does
Sensitive files cannot be downloaded (`hardening_block_sensitive_files`)	Answers copies of `wp-config.php`, database dumps, backup archives, the debug log, version control folders, `.env` files and the install script with “access denied”.
No programs can run from the uploads folder (`hardening_block_uploads_php`)	Adds a `.htaccess` file to the uploads folder that refuses to run or show PHP files.
Folder contents are not listed (`hardening_disable_listing`)	Adds `Options -Indexes`.
Important files are protected from other users (`hardening_secure_permissions`)	Takes access from other users on `wp-config.php` and `.htaccess`. The old permissions are remembered and restored when you switch the fix off or deactivate the plugin. If the site stops answering afterwards, the old permissions come back automatically.

Fix (setting key)	What it does
The theme and plugin editor is switched off (`hardening_disable_file_edit`)	Removes the file editor in the dashboard.
The WordPress version is hidden (`hardening_hide_version`)	Removes the version from the page.
The WordPress version is removed from script and style addresses (`hardening_hide_asset_versions`)	Needs the previous fix to be on.
Unused discovery links are removed from page headers (`hardening_clean_head`)	Cleans the page header.
The REST API link is removed from page headers (`hardening_hide_rest_link`)	Cleans the page header.
Browsers do not guess file types (`hardening_header_nosniff`)	Sends `X-Content-Type-Options: nosniff`.
Other sites cannot show your pages in a frame (`hardening_header_frame`)	Sends `X-Frame-Options: SAMEORIGIN`.
Less information is passed to other sites (`hardening_header_referrer`)	Sends `Referrer-Policy: strict-origin-when-cross-origin`.
Pages cannot use camera, microphone, payment or USB (`hardening_header_permissions`)	Sends `Permissions-Policy: camera=(), microphone=(), payment=(), usb=()`.
Browsers always use the secure address (HSTS) (`hardening_header_hsts`)	Sends `Strict-Transport-Security` with `max-age=15552000`. This is hard to undo, because browsers remember it. Switch it on only when the whole site works over HTTPS.
Basic content policy (upgrade requests, no foreign frames) (`hardening_header_csp`)	Sends `Content-Security-Policy` with `upgrade-insecure-requests` and `frame-ancestors 'self'`.

All fixes are off by default.

When you deactivate the plugin, the plugin removes the rules it wrote and restores permissions; the settings stay and are applied again when you activate the plugin. When you delete it, they are undone for good. See Removing the plugin.

Checks without a fix

These checks only report. Fix them in WordPress or on your server.

- **Error messages are not shown to visitors**
- **The debug log is not downloadable**
- **The site and the dashboard use HTTPS**
- **The database does not use the default table prefix**
- **No account is called “admin”**
- **New accounts do not get editing rights**
- **Security keys and salts are set**
- **Security releases of WordPress install themselves**
- **WordPress, plugins and themes are up to date**
- **No unused plugins or themes are left on the server**
- **PHP still receives security fixes**

Activity log and notifications

What is recorded

The plugin writes important events to the activity log. Each entry has a time, an area, an importance, a type, a user name and an IP address. The areas are **Firewall**, **Sign-in**, **Changes**, **Scanner** and **System**. The importance levels are **Info**, **Notice**, **Warning** and **Critical**.

Firewall and sign-in events are described in Firewall and Sign-in protection. The area **Changes** records what happens on the site:

- users: created, deleted, role changed, email changed, password changed, application passwords created or revoked, added to or removed from a site, super administrator granted or revoked. An administrator that is created or gained is a warning.
- plugins and themes: installed, activated, deactivated, updated, deleted, switched, and files edited in the editor. Deactivating this plugin is a warning, as is editing a plugin or theme file.
- WordPress core updates, including automatic ones.
- settings: changes to options, with a warning for the most sensitive ones, and a critical event when a role gains rights that can take over the site.
- exports of data and personal data, and requests to erase personal data.
- content: published, updated, restored, trashed or deleted posts and pages, uploaded or deleted media, changed menus and widgets. Content events are Info or Notice.

You choose what content is recorded in the tab **Activity tracking**: **Posts and pages** (default on), **Media library** (default on), **Menus** (default off) and **Widgets** (default off). Every event also records where the change came from: admin, ajax, rest, cron, cli, xmlrpc or front.

The Activity page

ISPProtect > Activity (page heading **Activity log**) lists the entries. Filters: **Area**, **Importance**, **What happened**, **IP address**, **User (name or number)**, **From** and **To**. Use **Filter** to apply them and **Clear filters** to reset them. **Entries per page** in the screen options is 50 by default (5 to 200).

Button	What it does
Download CSV	Downloads the entries as a CSV file.
Delete all log entries	Deletes the whole log after confirmation. The deletion itself is logged as `log_cleared`.

From an entry you can block or allow the address. The plugin refuses to block your own address or an always-allowed address.

Keeping and shortening the log

Setting	Default
Keep log entries for (`log_retention_days`)	90 days (1 to 3650)
Shorten IP addresses in the log after (`anonymize_ips_after_days`)	30 days (0 to 3650; 0 keeps full addresses)

Housekeeping runs once a day.

Email, summaries and webhook

All settings are on **ISPProtect > Settings > Notifications**. Under **Are alerts reaching you?** the panel tells you whether anything can deliver an alert, with the state of **Email**, **Webhook**, **Instant alerts**, **Daily summary** and **Weekly report**, and **Last deliveries**. If nothing is set up it says “No. Nothing is set up that could deliver an alert.”

Instant alerts

Setting (key)	Default	Meaning
Send email alerts (`notify_email_enabled`)	on	
Send alerts to (`notify_email_list`)	empty	Email addresses. If empty, the plugin uses the administration email address of the site.
Tell me right away about (`notify_alert_severity`)	**Urgent events only**	**Urgent events only**, **Urgent events and warnings**, **Everything notable, including normal changes** or **Nothing, only send summaries**.
Do not repeat the same alert within (`notify_throttle_minutes`)	30 minutes (0 to 1440)	
Send at most (`notify_max_per_hour`)	10 messages per hour (1 to 100)	Beyond this limit one “paused” message is sent.

Events that are held back by the limits go into the daily summary.

Summaries

Setting (key)	Default	Meaning
Daily summary (`notify_digest`)	**Send every morning**	Or **Do not send**. Nothing is sent on quiet days.
Weekly report (`notify_weekly`)	on	Sent on Monday morning.

Summaries and reports are sent by email only.

Webhook

Setting (key)	Default	Meaning
Webhook address (`notify_webhook_url`)	empty	The address of a service of your own. It must use https and lead to a public server; private and internal addresses are refused.
Format (`notify_webhook_format`)	**JSON for your own service**	Or **Text message for Slack and compatible chats**.
Signing key (`notify_webhook_secret`)	empty	If set, each request carries the headers `X-ISPProtect-Timestamp` and `X-ISPProtect-Signature`. The signature is `sha256=` followed by the HMAC-SHA256 of the timestamp, a dot and the body. Refuse requests whose timestamp is older than five minutes. The key is never shown again.

Test message

Send a test message sends a short message through every way of receiving alerts that is switched on. Tests are limited to 5 in 10 minutes for the whole site. Test messages do not appear under **Last deliveries**; their result appears under **Last test message**. If a test fails, the panel names the reason, for example no valid email address, the site cannot send email, or the webhook address is not https, leads to an internal network or cannot be reached.

Settings reference

This chapter lists every setting of the plugin, grouped by the tabs of **ISPProtect > Settings**. A setting key is the name you use with `wp ispprotect settings` (see Command line). Defaults are the values after a fresh installation. Ranges are inclusive. Settings are stored once for the whole network on multisite.

General settings

Label	Key	Default	Meaning
Read the visitor address from	`ip_source`	Direct connection (REMOTE_ADDR)	Where the plugin reads the address of the visitor; choose a header only when your site is behind a proxy that sets it.
Trusted proxies	`trusted_proxies`	empty	Proxies, one address or CIDR range per line, whose header the plugin accepts; empty accepts the header from anyone.
Keep log entries for	`log_retention_days`	90 days	Log entries older than this are deleted once a day. Range: 1 to 3650 days.
Shorten IP addresses in the log after	`anonymize_ips_after_days`	30 days	Older entries keep only the network part of the address. Range: 0 to 3650 days.
Keep settings and logs when the plugin is deleted	`keep_data_on_uninstall`	off	By default, deleting the plugin removes its tables, settings, user data and files.

Firewall settings

Label	Key	Default	Meaning
Firewall mode	`firewall_mode`	Block rule matches (`block`)	Block requests that match a rule, only log what the rules match, or apply only the block list.
Trust signed-in administrators	`firewall_admin_pass`	on	Requests of signed-in administrators skip the firewall rules and never count toward automatic bans, so your own edits are never blocked.
Stop privilege escalation	`firewall_guards`	on	Stops changes that only an administrator may make when someone else makes them, for example through a vulnerable plugin: opening registration, giving new users the administrator role, granting administrator capabilities.
Ban addresses automatically	`firewall_autoban_enabled`	on	Adds an address to the block list for a while after repeated blocked requests.
Blocked requests before a ban	`firewall_autoban_threshold`	10	Number of blocked rule matches from one address that triggers an automatic ban. Range: 1 to 1000.
Counted within	`firewall_autoban_window`	10 minutes	Time span in which the blocked matches are counted. Range: 1 to 1440 minutes.
Ban duration	`firewall_autoban_duration`	60 minutes	How long an automatic ban lasts; it also applies to bans for missing pages. Range: 1 to 43200 minutes.
Watch for scans of missing pages	`firewall_notfound_enabled`	off	Counts requests for pages that do not exist, per address.

Label	Key	Default	Meaning
Not-found pages per address	`firewall_notfound_threshold`	30	Number of requests for missing pages from one address that reaches the limit. Range: 5 to 10000.
Counted within	`firewall_notfound_window`	5 minutes	Time span in which missing pages are counted. Range: 1 to 1440 minutes.
When the limit is reached	`firewall_notfound_action`	Log it (`log`)	What happens when the limit is reached: only a log entry, or a log entry and a ban of the address.
Limit requests per address	`firewall_ratelimit_enabled`	off	Answers addresses above the limit with “Too many requests”.
Requests per minute	`firewall_ratelimit_requests`	240	Requests per minute from one address above which the firewall answers “Too many requests”; signed-in administrators count too. Range: 30 to 100000.
Limit access by country	`firewall_country_enabled`	off	Blocks or allows visitors by country; needs the country from your server or from Cloudflare.
Countries in the list are	`firewall_country_mode`	Blocked (`deny`)	Whether the countries in the list are blocked or are the only ones allowed.
Countries	`firewall_country_list`	empty	Two-letter country codes.
Applies to	`firewall_country_scope`	The whole site	Whether the country rule covers the whole site or only the sign-in.
OWASP CRS rules	`firewall_crs_mode`	Off (`off`); **Log matches only** (`log`) after the first install	`off` , `log` or `block` . Checks requests against the OWASP CRS rules; needs the rules installed on the firewall page. See OWASP CRS rules.
Paranoia level	`firewall_crs_paranoia`	1	How many CRS rules run. Higher levels find more attacks and report more normal requests. Range: 1 to 4.
Anomaly score threshold	`firewall_crs_threshold`	5	Score at which a request counts as an attack. A critical match adds 5. Range: 1 to 100.
Switched-off CRS rules	`firewall_crs_disabled_rules`	empty	CRS rule numbers that do not run, one per line. Also takes the numbers `200002` to `200102` for requests that cannot be fully inspected.
Request body checked	`firewall_crs_body_limit`	131072 bytes	Bytes of a request body the CRS rules read. A larger request cannot be fully inspected. Range: 1024 to 1048576 bytes.
Time limit per request	`firewall_crs_time_budget`	200 milliseconds	Most time the CRS rules may take for one request. When it is used up the request counts as not fully inspected. Range: 5 to 10000 milliseconds.
Let requests pass that cannot be fully inspected	`firewall_crs_pass_uninspected`	off	On: such requests go through and are logged. Off: they count as an attack (refused in block mode).

Login protection settings (sign-in protection and two-factor authentication)

Label	Key	Default	Meaning
Block addresses after repeated failed sign-ins	`login_lockout_enabled`	on	Switches the lockout of addresses after failed sign-ins on or off.
Failed sign-ins allowed per address	`login_max_attempts`	5	Number of failed sign-ins from one address that leads to a block. Range: 2 to 100.
Count failed sign-ins over	`login_window_minutes`	15 minutes	Time span in which failed sign-ins are counted. Range: 1 to 1440 minutes.
First block lasts	`login_lockout_minutes`	15 minutes	Length of the first block; each further block within a day lasts four times as long. Range: 1 to 1440 minutes.
Longest block	`login_lockout_max_hours`	24 hours	Upper limit for the length of a block. Range: 1 to 720 hours.
Failed sign-ins allowed per username, from all addresses	`login_user_max_attempts`	10	Failed sign-ins for one username, counted over all addresses, after which every further address that fails for it is blocked; the account itself is never locked. 0 switches it off. Range: 0 to 1000.
Block right away when someone tries a username that does not exist	`login_lockout_unknown_users`	off	This tells visitors which usernames exist: a name that does not exist blocks at once, a real one does not.
Block right away when one of these usernames is tried	`login_lockout_usernames`	empty	Usernames that block an address at once when tried; they apply only while no account has that name.
Do not reveal whether a username exists	`login_generic_errors`	on	Wrong username and wrong password show the same message, and password reset requests always look successful.
Let blocked users request an unlock link by email	`login_unlock_links`	on	Blocked users can ask for an email with a one-time link that lifts the block.
Offer two-factor authentication	`login_2fa_enabled`	on	Users can set up two-factor authentication on their profile.
Methods users can choose	`login_2fa_methods`	`totp`, `email`	The second-step methods users can choose: authenticator app, code by email, or both.
Require two-factor authentication for these roles	`login_2fa_required_roles`	none	Users of the ticked roles must set up two-factor authentication within the grace period.
Time to set it up	`login_2fa_grace_days`	7 days	After this, users of the roles above have to set it up before they can sign in. Range: 0 to 90 days.
Let users skip two-factor authentication on a trusted device for	`login_2fa_remember_days`	30 days	0 asks for the code at every sign-in. Range: 0 to 365 days.
Move the login page to its own address	`login_hide_enabled`	off	wp-login.php and wp-admin then show “not found” to visitors who are not signed in.
Login address	`login_hide_slug`	empty	The new sign-in address: lowercase letters, digits and dashes, at least three characters.
Add an invisible trap field to the login, registration and password forms	`login_honeypot`	on	Simple bots fill it in and are turned away.

Label	Key	Default	Meaning
Make the browser solve a small puzzle before signing in	`login_pow`	off	The browser must solve a small puzzle before signing in; it needs JavaScript.
Puzzle difficulty	`login_pow_difficulty`	16	Higher values take the browser longer. Range: 8 to 24.
Apply the password rules to these roles	`login_pw_roles`	`administrator`, `editor`	Roles the password rules apply to; none ticked means everyone.
Minimum password length	`login_pw_min_length`	12 characters	0 turns the length check off. Range: 0 to 64 characters.
Reject common and easily guessed passwords	`login_pw_block_common`	on	Refuses passwords that are common or easy to guess.
Reject passwords that contain the username, email address or site name	`login_pw_block_personal`	on	Refuses passwords that contain the username, the email address or the site name.
Check new passwords against known data breaches	`login_pw_hibp`	off	Checks new passwords with an outside service; only the first five characters of a hash leave the site.
Ask for a new password after	`login_pw_max_age_days`	0 days	0 never asks. Range: 0 to 3650 days.
Sessions per user at the same time	`login_session_limit`	0	When a user signs in on one device too many, the oldest session ends. Range: 0 to 100.
Sign out after inactivity	`login_idle_minutes`	0 minutes	0 keeps sessions until they expire. Range: 0 to 10080 minutes.
Apply the two session limits above to these roles	`login_session_roles`	none	Roles the two session limits apply to; none ticked means everyone.
Email users when they sign in from a new device or network	`login_new_device_mail`	on	Sends the user an email when a sign-in comes from a device or network the plugin has not seen before.
XML-RPC	`login_xmlrpc`	Restricted (`restricted`)	Unchanged, Restricted (removes pingbacks and the method that tries many passwords in one request) or Off.
Application passwords	`login_app_passwords`	Allowed (`on`)	Whether users may use application passwords.
Stop visitors from listing usernames through ?author= links	`login_block_author_scan`	on	Visitors cannot list usernames through ?author= links.
Hide the user list in the REST API from visitors who are not signed in	`login_block_rest_users`	on	Visitors who are not signed in do not get the user list from the REST API.
Leave author pages out of the sitemap	`login_hide_users_sitemap`	on	Author pages are left out of the sitemap.
Leave the author out of embed previews	`login_oembed_hide_author`	on	The author is left out of embed previews.

Scanner settings

Label	Key	Default	Meaning
Automatic scan	`scanner_schedule`	Every day (`daily`)	How often the plugin starts a scan by itself: not at all, every day or every week.
Hour of the day	`scanner_hour`	3	The hour in your site time zone at which the scan starts. Range: 0 to 23.
WordPress files	`scanner_check_core`	on	Compare WordPress files with the originals from wordpress.org and report changed, unknown and missing files.
Plugin files	`scanner_check_plugins`	on	Compare plugin files with the release on wordpress.org.
Malware	`scanner_check_malware`	on	Basic checks for PHP in the uploads folder, leftovers and a small set of generic suspicious-code checks.
Database	`scanner_check_database`	on	Look for injected scripts in posts, widgets and theme settings, and for a changed site address.
Updates	`scanner_check_updates`	on	Report available updates, abandoned plugins and an outdated PHP version.
Known vulnerabilities	`scanner_vuln_check`	off	Ask an outside service about known vulnerabilities in your WordPress version, plugins and themes.
Skipped folders	`scanner_excluded_paths`	empty	Folders that are not scanned, one per line, relative to the WordPress folder.
Largest code file to read	`scanner_max_file_size`	5 MB	Largest code file the scanner reads completely; larger code files are read only at their start and end. Range: 1 to 50 MB.
Remote scan	`scanner_remote_mode`	Off (`off`)	Off, a scan service on this server, your own scan service, or the ISPProtect scan service with a licence key.
Socket of the local scan service	`scanner_remote_socket`	`/run/ispprotect/scand.sock`	Absolute path of the unix socket of a scan service that runs on this server.
Address of the scan service	`scanner_remote_url`	empty	Used only in the mode Own scan service: the address of your own scan service starting with https://. The licence mode and the socket mode ignore it.
Upload limit per scan	`scanner_remote_upload_mb`	200 MB	Unknown code files are uploaded until this amount is reached; the rest follow in the next scans. Range: 0 to 4096 MB.
Never send these folders	`scanner_remote_exclude`	empty	Folders or files that are never sent, one per line, relative to the WordPress folder.

Hardening settings

Label	Key	Default	Meaning
Sensitive files cannot be downloaded	`hardening_block_sensitive_files`	off	Adds rules to the .htaccess file in your website folder that answer these files with “access denied”.
No programs can run from the uploads folder	`hardening_block_uploads_php`	off	Adds a .htaccess file to the uploads folder that refuses to run or show PHP files there, also with capital letters or a second extension such as .php.jpg.
Folder contents are not listed	`hardening_disable_listing`	off	Adds “Options -Indexes” to the .htaccess file in your website folder.

Label	Key	Default	Meaning
Important files are protected from other users	`hardening_secure_permissions`	off	Takes access away from “other users” on these files.
The theme and plugin editor is switched off	`hardening_disable_file_edit`	off	Removes the editor screens and the permission to use them.
The WordPress version is hidden	`hardening_hide_version`	off	Removes the generator tag from pages and feeds.
The WordPress version is removed from script and style addresses	`hardening_hide_asset_versions`	off	Removes the version from addresses of WordPress core scripts and styles on the public pages.
Unused discovery links are removed from page headers	`hardening_clean_head`	off	Removes these link tags and the short link header.
The REST API link is removed from page headers	`hardening_hide_rest_link`	off	Removes the REST API link tag and header from pages.
Browsers do not guess file types	`hardening_header_nosniff`	off	Sends “X-Content-Type-Options: nosniff” with every page.
Other sites cannot show your pages in a frame	`hardening_header_frame`	off	Sends “X-Frame-Options: SAMEORIGIN”.
Less information is passed to other sites	`hardening_header_referrer`	off	Sends “Referrer-Policy: strict-origin-when-cross-origin”: other sites only learn which site the visitor came from.
Pages cannot use camera, microphone, payment or USB	`hardening_header_permissions`	off	Sends “Permissions-Policy: camera=(), microphone=(), payment=(), usb=()”.
Browsers always use the secure address (HSTS)	`hardening_header_hsts`	off	Sends “Strict-Transport-Security: max-age=15552000” (about six months).
Basic content policy (upgrade requests, no foreign frames)	`hardening_header_csp`	off	Sends “Content-Security-Policy: upgrade-insecure-requests; frame-ancestors ‘self’”.

Activity tracking settings

Label	Key	Default	Meaning
Posts and pages	`audit_log_content`	on	Record when a post or page is published, changed after publishing, moved to the trash, restored or deleted.
Media library	`audit_log_media`	on	Record when files are uploaded to or deleted from the media library.
Menus	`audit_log_menus`	off	Record when a navigation menu is created, changed or deleted.
Widgets	`audit_log_widgets`	off	Record when widgets are added, changed or removed.

Notification settings

Label	Key	Default	Meaning
Send email alerts	`notify_email_enabled`	on	Switches email alerts on or off.
Send alerts to	`notify_email_list`	empty	One email address per line; empty means the administration email address of the site.

Label	Key	Default	Meaning
Tell me right away about	`notify_alert_severity`	Urgent events only (`critical`)	Which events are mailed right away: urgent events only, urgent events and warnings, everything notable, or nothing.
Do not repeat the same alert within	`notify_throttle_minutes`	30 minutes	Further events of the same kind are counted and mentioned in the next message. Range: 0 to 1440 minutes.
Send at most	`notify_max_per_hour`	10 messages per hour	When the limit is reached you get one message saying that alerts are paused. Range: 1 to 100 messages per hour.
Daily summary	`notify_digest`	Send every morning (`daily`)	A short daily list of events that were not mailed right away.
Weekly report	`notify_weekly`	on	Every Monday morning: blocked requests, sign-ins, changes and scan results of the past week.
Webhook address	`notify_webhook_url`	empty	Address that receives alerts as web requests; empty switches the webhook off.
Format	`notify_webhook_format`	JSON for your own service (`generic`)	The format of messages sent to the webhook: JSON, or a text message for Slack and compatible chats.
Signing key	`notify_webhook_secret`	empty	Optional key used to sign each webhook request; it is never shown again.

Command line

The plugin adds the command `wp ispprotect` to WP-CLI. Run it as the system user of the website. If you run a command that writes as another user, the plugin refuses, because files it would create would belong to the wrong user. On multisite the commands act on the network-wide settings and rules; use `--url` to select a site where a command needs one.

```
wp ispprotect <2fa|blocklist|hardening|login|remote|scan|settings|status|unblock>
```

To see the help of a command, run `wp help ispprotect <command>`.

status

```
wp ispprotect status [--format=<format>]
```

Prints an overview: safe mode, data folder, firewall mode and extended protection, lockouts, two-factor authentication problems, the last scan and the result of the hardening checkup. `--format` is `text` (default), `json` or `yaml`.

settings

```
wp ispprotect settings <list|get|set|reset> [<key>] [<value>] [--module=<module>] [--allow-empty] [--yes] [--fo
rmat=<format>]
```

Sub-command	What it does
`list`	Lists settings with their values. `--module` limits the list to `general`, `firewall`, `login`, `scanner`, `hardening`, `audit` or `notify`. `--format`: `table`, `json`, `csv`, `yaml`, `count`.
`get <key>`	Prints one value. `--format`: `plain` or `json`. Keys that carry a secret print `(set)` or `(empty)`.
`set <key> <value>`	Changes a setting. Lists are given comma separated or as JSON. Use `-` as the value to read it from standard input. `--allow-empty` allows an empty value.
`reset [<key>]`	Resets one setting, or all of them (with `--module` one module) to the defaults. A full reset keeps the setup state.

Changing `ip_source` or `trusted_proxies` warns and asks for confirmation unless you pass `--yes`. For `ip_source` use the stored value, for example `REMOTE_ADDR` or `HTTP_X_FORWARDED_FOR`. Keys are listed in Settings reference.

```
wp ispprotect settings set scanner_remote_mode daemon
```

blocklist

```
wp ispprotect blocklist <list|add|remove> [<target>] [--minutes=<n>] [--reason=<text>] [--scope=<scope>] [--typ
e=<type>] [--source=<source>] [--search=<text>] [--expired] [--format=<format>]
```

Sub-command	What it does
`list`	Lists the rules. Filters: `--type=block|allow`, `--scope=site|login`, `--source=manual|cli|firewall|login|scanner|unlock`, `--search=<text>` (matches address or reason), `--expired` to include expired rules. `--format`: `table`, `json`, `csv`, `yaml`, `count`.
`add <address-or-range>`	Adds a block rule; give the address or range in CIDR form. Without `--minutes` the rule is permanent. `--reason` sets a note. `--scope` is `site` (default) or `login`. The address of the server itself cannot be blocked.

Sub-command	What it does
`remove <rule-id-or-address>`	A rule ID removes that rule, block or allow. An address or range removes the block rules for exactly that address or range.

`add` adds block rules only. To always allow an address, use the page **ISPProtect > IP rules**.

unblock

```
wp ispprotect unblock <ip> [--include-ranges]
```

Removes the block rules for exactly this address (for IPv6 also its /64 range), lifts lockouts and applies queued bans first. Wider ranges that contain the address stay unless you add `--include-ranges`. The command exits with 0 when the address is no longer blocked.

login

```
wp ispprotect login <status|lockouts|unlock> [<ip>] [--include-ranges] [--page=<n>] [--per-page=<n>] [--format=<format>]
```

Sub-command	What it does
`status`	Prints the state of sign-in protection: lockouts, failed sign-ins in 24 hours, users with two-factor authentication, safe mode, whether two-factor authentication is bypassed, key file problems, and the hidden sign-in address (`hidden_login`). `--format`: `table`, `json`, `yaml`.
`lockouts`	Lists lockouts. `--per-page` is 1 to 1000 (default 100), `--page` selects the page, the total goes to standard error. `--format`: `table`, `json`, `csv`, `yaml`, `count`.
`unlock <ip>`	Lifts only the lockouts caused by failed sign-ins and restarts the counter. Range lockouts are lifted only with `--include-ranges`. It reports an error if the address is still blocked, for example by a rule.

2fa

```
wp ispprotect 2fa <status|reset> <user> [--yes] [--format=<format>]
```

`<user>` is a user ID, login name or email address.

Sub-command	What it does
`status`	Shows the two-factor authentication state of the user. `--format`: `table`, `json`, `yaml`. Secrets are never shown.
`reset`	Removes the authenticator app, the email-code sign-in, all backup codes and all trusted devices of the user, and clears an unfinished setup. The user can then sign in with the password alone, and a setup time that ran out starts again. It asks first unless you pass `--yes`, and logs the reset.

hardening

```
wp ispprotect hardening [--refresh] [--format=<format>]
```

Prints the checks of the hardening checkup. The result is kept for five minutes; `--refresh` runs the checks again. The command changes nothing. Some checks send a request to the site itself. `--format`: `table`, `json`, `csv`, `yaml`, `count`.

scan

```
wp ispprotect scan <start|status|cancel> [--type=<type>] [--wait] [--format=<format>]
```

Sub-command	What it does
`start`	Starts a scan. `--type` is `quick` (default) or `full`. The scan is handed to WP-Cron. With `--wait` it runs in this process until it ends and prints its progress.
`status`	Shows the state of the scan and the open findings.
`cancel`	Cancels the running scan.

With `--wait` the exit code is 1 if the scan failed or was cancelled and 0 if it finished, also when it found problems. `--format`: `table`, `json`, `yaml`.

remote

```
wp ispprotect remote <status|test|connect|refresh|disconnect|token> [<value>...] [--format=<format>]
```

Sub-command	What it does
`status`	Shows the mode, the service, whether a token or licence key is stored (never the value) and the licence.
`test`	Asks the scan service for its status.
`connect <licence-key>`	Connects the licence. Give `-` instead of the key to read it from standard input, so it does not end up in the shell history.
`refresh`	Renews the access of the licence now.
`disconnect`	Removes the licence from this site.
`token set <token>`	Stores the access token of an own scan service. Use `-` to read the token from standard input.
`token clear`	Removes the stored token.

`--format` (for `status` and `test`) is `table`, `json` or `yaml`. The mode itself is a setting: `wp ispprotect settings set scanner_remote_mode <mode>` with `off`, `daemon`, `service` or `license`.

Extended firewall protection cannot be switched on or off with WP-CLI. Do it in the dashboard; see Firewall.

Getting back in and troubleshooting

This chapter lists every way you can lock yourself out and the way back. Run the commands as the system user of the website, in the folder of the site (or with `--path=<path>`). If you do not have command-line access, use the file-based steps: the emergency constants go into `wp-config.php`.

The emergency constants

You add these lines to `wp-config.php`, above the line that says to stop editing. Remove them when you have fixed the problem.

Constant	Effect
`define( 'ISPPROTECT_WEB_SECURITY_SAFE_MODE', true );`	Safe mode. The plugin records events but enforces nothing: the firewall (including the block list) only logs, sign-in lockouts and the privilege-escalation guards are inactive, a hidden sign-in address is switched off, and two-factor authentication is not asked.
`define( 'ISPPROTECT_WEB_SECURITY_DISABLE_2FA', true );`	Two-factor authentication is not asked. Nothing else changes.
`define( 'ISPPROTECT_WEB_SECURITY_DATA_DIR', '/path/to/folder' );`	Moves the data folder.

The dashboard shows “Safe mode” in the header pill while the first constant is set. Safe mode also works for the firewall when extended protection is on: the firewall reads `wp-config.php` again when it changes.

You are blocked at sign-in

You typed the password wrong too often, and the sign-in page says your address is blocked.

1. Wait until the block ends. The first block lasts 15 minutes by default.
2. Or ask for an unlock link with the option on the sign-in page. It arrives by email, works once and for one hour.
3. Or sign in from another address and lift the lock on **ISPProtect > Login protection** with **Lift** or **Lift lockout**.
4. Or use the command line:

```
wp ispprotect login unlock 203.0.113.7
```

Replace the address with yours. Add `--include-ranges` to lift range lockouts that contain it. To see the lockouts first: `wp ispprotect login lockouts`.

To avoid it, press **Always allow my address** on **ISPProtect > IP rules**. Always-allowed addresses are never counted and never locked.

The firewall blocks your address

Requests from your address are refused with “403” or “Too many requests”.

1. Remove the block rules and lockouts for your address:

```
wp ispprotect unblock 203.0.113.7
```

If a wider range blocks you, add `--include-ranges`. The command exits with 0 when the address is no longer blocked. If a rule you cannot remove with this command blocks you, find its ID with `wp ispprotect blocklist list --search=203.0.113` and remove it with `wp ispprotect blocklist remove <rule-id>`. 2. Without WP-CLI, put `define( 'ISPPROTECT_WEB_SECURITY_SAFE_MODE', true );` into `wp-config.php`. The firewall then only logs. Sign in, remove the rule on **ISPProtect > IP rules** and take the line out again. 3. If the rate limit refuses you, you can also switch it off: `wp ispprotect settings set firewall_ratelimit_enabled false`. The same works for `firewall_mode`: `wp ispprotect settings set firewall_mode off`.

The plugin refuses to add a block rule for your own address in the dashboard, and the engine never applies rules to always-allowed addresses.

The OWASP CRS rules refuse a normal request

A form, a save or a page of your site is refused (status 403, 400 or 413), or the card **OWASP CRS rules** lists requests of your own users.

1. Open **ISPProtect > Firewall** and look at **Recent CRS matches**. Note the rule number and the message of the entry that belongs to the refused request.
2. Press **Switch this rule off** next to the number. Switch off only the rule that is wrong. If you are not sure, set **OWASP CRS rules** to **Log matches only** first and watch the entries for a few days.
3. If you cannot reach the dashboard, switch the CRS check off with `wp ispprotect settings set firewall_crs_mode off`, or with the safe mode constant (see The emergency constants).

Special cases:

- **Status 413 when you save a long post or a large page.** The request is larger than **Request body checked** (rule `200101`). Raise that setting, for example to 524288 bytes, or turn on **Let requests pass that cannot be fully inspected**.
- **Requests refused with rule `200100`.** The check ran out of time (**Time limit per request**). Raise the limit, for example to 500 milliseconds, or switch off the number `200100`. Program code pasted into a form field (about 20 to 40 KB) can use up the default limit of 200 milliseconds; the same answers apply, or let requests pass that cannot be fully inspected.
- **Requests refused with rule `200005`.** Rarely, a long field with program code makes a pattern of a CRS rule exceed the regular expression limits of PHP. Switch off the number `200005` or turn on **Let requests pass that cannot be fully inspected**.

See OWASP CRS rules.

You lost the hidden sign-in address

1. Look for the email with the subject “New login address”.
2. Or print it: `wp ispprotect login status` (field `hidden_login`).
3. Or switch the feature off:

```
wp ispprotect settings set login_hide_enabled false
```

1. Without WP-CLI, put `define( 'ISPPROTECT_WEB_SECURITY_SAFE_MODE', true );` into `wp-config.php`. While safe mode is on, the standard `wp-login.php` works again.

You lost access to two-factor authentication

1. Use a backup code on the sign-in page (**Use a backup code**). Each works once.
2. Ask another administrator to press **Reset two-factor authentication** on your profile (**Users > Edit user**).
3. With WP-CLI:

```
wp ispprotect 2fa reset admin --yes
```

Replace `admin` with the user ID, login or email. You can then sign in with the password and set up two-factor authentication again. 4. Without WP-CLI, put `define( 'ISPPROTECT_WEB_SECURITY_DISABLE_2FA', true );` into `wp-config.php`, sign in, set up two-factor authentication again, and remove the line.

If the key file `login-key.php` is missing or unreadable, authenticator apps cannot be checked and the plugin sends codes by email instead. Put the file back from a backup, or have each user set up the app again. Do not delete the file: a new key cannot read secrets that the old key encrypted.

The site does not load after switching on extended protection

The firewall loader is missing or PHP cannot use it.

1. Open `.htaccess` (Apache, LiteSpeed) or `.user.ini` (PHP-FPM, CGI) in the home folder of the site.
2. Delete the lines between `BEGIN ISPProtect Web Security Firewall` and `END ISPProtect Web Security Firewall`, with both marker lines.
3. You can then delete the file `ispprotect-firewall.php` next to it.
4. With `.user.ini` the change can take a few minutes, because PHP caches the file.

The site misbehaves after a hardening fix

Open **ISPProtect > Hardening** and press **Undo fix** for the fix. If you cannot reach the dashboard, deactivate the plugin (`wp plugin deactivate ispprotect-web-security`): the plugin removes the rules it wrote to `.htaccess` and restores file permissions. Fixes that change file permissions come back by themselves if the site stops answering after the change.

No alerts arrive

Open **ISPProtect > Settings > Notifications** and read **Are alerts reaching you?**. Press **Send a test message**. The panel names the reason when something fails: no valid email address, a site that cannot send email (use a mail plugin), a webhook address that is not https, that leads to an internal network or that cannot be reached. Hosting must be able to send email for email alerts to work.

A scan does not start or does not finish

- If WP-Cron is off on your site, scheduled scans cannot run. Start the scan with `wp ispprotect scan start --wait` or run WP-Cron from the system.
- If a scan has not moved for five minutes the page says so. Use **Continue scan**, or **Cancel scan** and start again.
- On a site with a remote scan, press **Test connection** on **ISPProtect > Scanner**. See Remote scan and licence.

Commands refuse to change anything

Commands that change something refuse when you run WP-CLI as another system user than the owner of the site. Run them as the website's user. As root the plugin warns when it cannot find that user.

Collecting information for support

Open **ISPProtect > Tools**, section **System information**, and press **Copy**. `wp ispprotect status` prints a short overview. The plugin does not contain a support address; ask your provider or whoever supplied your licence.

Privacy and external services

The plugin works without outside services except where noted below. Each request comes from your server, not from the visitor's browser. All requests use `https`, with one exception: a request to your own scan service on the same computer may use `http://127.0.0.1`, `http://[::1]` or `http://localhost` if you enter such an address. Such a request never goes through a proxy set in `wp-config.php`. Every service sees the IP address of your server, as with any request.

External services

Service	Hosts	What is sent	When
WordPress.org (checksums, plugin information, original files)	`api.wordpress.org`, `downloads.wordpress.org`, `core.svn.wordpress.org`, `plugins.svn.wordpress.org`	For core checksums: the WordPress version and locale. For plugin checksums: folder name and version of each installed plugin. For plugin information: the folder name of each installed plugin that lives in its own folder, active or not. When you open the comparison of a changed file: the path of that file in the original package.	During a scan. Core checksums run when **WordPress files** is on, plugin checksums when **Plugin files** is on, and the plugin information lookup when **Updates** is on (all on by default). The files of the original are fetched only when you open a comparison. If you turn these checks off, no request is made.
WPVulnerability (known vulnerabilities)	`www.wpvulnerability.net`	The WordPress version and the folder name of each plugin and each theme. Plugin and theme versions are compared on your site and not sent. Answers are kept for twelve hours.	Only when **Known vulnerabilities** is on (off by default), during a scan.
Have I Been Pwned, Pwned Passwords (leaked-password check)	`api.pwnedpasswords.com`	The first 5 characters of the SHA-1 hash of the new password. The password and the full hash never leave your server. The request asks for a padded answer, so its size tells nothing about the result.	Only when **Check new passwords against known data breaches** is on (off by default), when a user sets or changes a password. It waits 3 seconds at most. If the service does not answer, the password is accepted.
A webhook you set up	The address you enter under notifications. Only `https` addresses are accepted; addresses that point to private or local networks are refused.	The alert text: event type, severity, time, site name and the details of the event, which can include an IP address or a user name. With a signing key, the request is signed.	Only when you enter a webhook address (empty by default), when an alert fires or when you press **Send a test message**. The request waits 3 seconds at most.
Scan service (remote scan)	The service you choose. A scan service on the same server is reached through its unix socket, and nothing leaves the server. Your own service is reached at the address you enter. The hosted ISPProtect scan service is reached at `scan.ispprotect.com`.	The SHA-256 hash and the file name (without folder) of each code file, and the content of code files the service does not know yet, up to the upload limit per scan. With your own service or the hosted service, also the access token. Sent only in redacted form: `wp-config.php` and PHP configuration files such as its copies, `localconfig.php` and `config.local.php` (all strings, numbers and comments replaced, so only the code structure, names and the constant names in `define()` remain). Never sent: `.env` and `.htaccess` files, other configuration files with secrets, `.ini` files, keys, certificates, database dumps, the data folder of the plugin and the folders you exclude.	Only when you choose a mode other than **Off**. During a scan and when you press **Test connection**. In hosted mode the requests go only to the address of the hosted service.

Service	Hosts	What is sent	When
OWASP CRS rules (GitHub)	`github.com` and the download host that GitHub redirects to, and `raw.githubusercontent.com`	Nothing about your site, your visitors or your requests: only the request for the fixed files (the rules archive of the CRS project and the rule exclusions for WordPress). Requests to your site are checked on your server and never sent anywhere.	Only when you press **Install the CRS rules** or **Update**, never in the background. Each file is checked against a checksum built into the plugin.
Licence API (licence key of the hosted scan service)	`license.ispprotect.com`. The account console at `console.ispprotect.com` is separate and receives no plugin requests.	The licence key, a random ID of this installation, the host name of the site in ASCII form (on multisite always the main site), the platform (`wordpress`), the product name and the plugin version.	After you choose **ISPProtect scan service (licence key)** and press **Connect**: then once a day, when you press **Refresh now** and when you press **Disconnect**. During a scan at most once an hour when the access expired or was refused. When the plugin is deleted, once, to release the site (5 seconds at most; the deletion goes on if the portal does not answer).

The plugin also sends requests to your own site: the hardening checkup and the test of extended firewall protection request your own address.

The terms and privacy statements of WordPress.org are at `https://wordpress.org/about/` and `https://wordpress.org/about/privacy/`. For WPVulnerability see `https://www.wpvulnerability.com/` and `https://www.wpvulnerability.com/privacy/`. For Have I Been Pwned see `https://haveibeenpwned.com/TermsOfUse` and `https://haveibeenpwned.com/Privacy`. For a webhook they depend on the service you choose. The hosted scan API has no built-in address in this release; its terms and privacy information will accompany that separate service before it is offered. The website privacy policy is at `https://ispprotect.com/privacy.html`; for your own scan service, terms depend on who runs it.

> Requests to a scan service or licence portal go through the WordPress HTTP API, so other plugins on your site can see the access token or licence key through the WordPress hooks `http_request_args`, `pre_http_request` and `http_api_debug`. A scan service reached through a unix socket does not use the HTTP API.

Personal data the plugin stores

The plugin stores this data in your WordPress database:

- an activity log with event, time, user name and IP address,
- failed and successful sign-in attempts with IP addresses, used for lockouts,
- firewall events and blocks, with IP addresses and the requested address,
- a list of up to 20 known devices per user (a fingerprint of the browser type and network, with a time), used for the new-device mail,
- two-factor data per user: the authenticator secret (encrypted), backup codes (stored as salted hashes) and trusted-device markers,
- scan results. With the remote scan on, also the SHA-256 hash, size and verdict of each code file. The access token or licence key of the remote scan is stored encrypted.

You control how long it stays: **Keep log entries for** (default 90 days) and **Shorten IP addresses in the log after** (default 30 days, 0 turns it off). See Activity log and notifications.

The plugin adds an exporter and an eraser for the activity log to the WordPress privacy tools, and suggested text for your privacy policy. The eraser removes what identifies the person from log entries; the entries stay because they document security events. Check your own privacy policy for what you need to tell visitors.

Removing the plugin

Deactivate

Deactivating the plugin on the page **Plugins** stops it, and:

- the plugin removes extended firewall protection from the server configuration (see Firewall). If you deactivate with WP-CLI as the root user, it cannot do this and tells you how to finish. If the removal fails, the block stays in the configuration file; remove it by hand as described in Getting back in and troubleshooting.
- the hardening module removes the rules it wrote to `.htaccess` and restores the file permissions it changed. The hardening settings stay and are applied again when you activate the plugin.
- the daily housekeeping task is removed.

Your settings, log, IP rules and scan results stay in the database.

Delete

Deleting the plugin on the page **Plugins** runs the clean-up of the plugin. It removes:

- its database tables,
- its settings and the data in user profiles (two-factor authentication data, device lists, session markers),
- its scheduled tasks,
- its data folder,
- the changes it made to `.htaccess`, `.user.ini` and to file permissions,
- on multisite, all of the above for every site of the network.

If a remote scan licence is still connected, the plugin tries once to release the site at the licence portal, with a timeout of 5 seconds. The deletion goes on if the portal does not answer. If you want to be sure, press **Disconnect** (or run `wp ispprotect remote disconnect`) before you delete the plugin, and check the licence portal.

One empty loader file for extended protection (`ispprotect-firewall.php`) can stay on purpose, because a server setting may still point to it. It does nothing once the plugin folder is gone. If you switch extended protection off before you delete the plugin, the plugin deletes the file as soon as no configuration can point to it any more.

Keep your data

If you want to keep settings, logs and data when the plugin is deleted, turn on **Keep settings and logs when the plugin is deleted** (`keep_data_on_uninstall`) in **ISPProtect > Settings > General** before you delete the plugin. Then nothing is removed, and a connected remote scan licence is not released: this site keeps counting towards the licence until you press **Disconnect** or release it in the licence portal.

Before you move or delete the site files

Switch extended protection off first. While it is on and the loader file is gone, PHP stops working on the site. See Getting back in and troubleshooting.