Security in PHP code

The security rules for PHP code in Exponential in one place - validate input, cast numbers, escape SQL and output by context, protect POSTs with the form token, check permissions in every entry point, redirect safely…

Source file: `doc/eng-us/help/php-developers/security-in-php-code.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/php-developers/security-in-php-code.md

This page is for PHP developers who write code that handles requests, data or files. It gathers the rules that keep Exponential code safe and links to the page that explains each. The project's security standard is Security standard; administrators harden the installation with Security hardening.

Contents

1. The rules
2. Where input comes from
3. Permissions in every entry point
4. Redirects
5. Secrets
6. Files and uploads
7. What the kernel does for you
8. Reporting a vulnerability

The rules

#	Rule	Page
1	Validate input where it enters: cast numbers with `(int)` and `(float)`, check choices against allow lists, limit lengths	Validating input and casting
2	Escape strings for SQL with `$db->escapeString()`, or use `eZPersistentObject` conditions	SQL with eZDB, safely
3	Escape output for its context: `|wash` in templates, `htmlspecialchars()` with explicit flags in PHP, `json_encode()` for JSON	Escaping output from PHP
4	Change data only on POST, protected by the form token	CSRF protection and ezformtoken
5	Check permissions in every view, fetch function, server function, REST action and command	Permissions in every entry point
6	Redirect only to addresses the kernel's rules accept	Redirects
7	Keep secrets in the installation's settings, never in code, URLs or logs	Secrets
8	Store files through the kernel; check uploads by content and size	Files and uploads
9	Do not put inline scripts or `style=""` into markup you generate	Content Security Policy for your code

Where input comes from

Treat all of these as untrusted:

- `$Params` and unordered parameters of a view: strings from the address.
- `eZHTTPTool::instance()->postVariable()`, `getVariable()`, `variable()`: the request. Read input through `eZHTTPTool`, not from `$_POST` and `$_GET` directly, so the kernel's request preparation applies (under Velocity it prepares them per request).
- Arguments of ezjscore server functions and REST routes.
- Settings: written by people, sometimes by other extensions.
- The database: written earlier by someone, maybe by an import.
- Headers such as `Referer`, `Host` and `X-Forwarded-For`; use the kernel's trusted proxy handling, not the raw header (Trusted proxies).

Permissions in every entry point

PHP

```php
$user = eZUser::currentUser();
$access = $user->hasAccessTo( 'acmereport', 'read' );
if ( $access['accessWord'] === 'no' )
    return $Module->handleError( eZError::KERNEL_ACCESS_DENIED, 'kernel' );
```

- `accessWord` is `yes`, `no` or `limited`; with `limited`, check the limitation against the object (or use `$node->canRead()`, `$object->canEdit()` for content).
- A view's `functions` in `module.php` are checked by the kernel before your code runs, but fetch functions, ezjscore server functions, REST actions and commands are called from places that do not check: they must check themselves (Access control in code).
- Content lists from `eZContentObjectTreeNode::subTreeByNodeID()` apply the current user's `content/read` unless you pass `'Limitation' => array()`; never pass it for results users see.

Redirects

Never redirect to an address taken from the request without checking it. `eZRedirectManager` (`kernel/classes/ezredirectmanager.php`) holds the rules: same-site paths, the current host or hosts listed in `site.ini [SiteSettings] AllowedRedirectHosts`, no control characters or backslashes, no `//` or foreign schemes even when percent-encoded, no credentials in the URL. `eZRedirectManager::safeURI( $uri )` checks one address; `redirectTo()` and `returnURI()` pick a safe target with a default (Safe redirects).

Secrets

- API keys, passwords and tokens live in `settings/override/` of the installation or in environment variables, never in your extension's repository and never in a URL.
- Never log them; never print them in the debug output; mask them in error messages.
- Compare secrets with `hash_equals()`; store tokens as hashes where you can (the kernel's API keys store only a salted HMAC-SHA-256 hash, expservices tokens only a SHA-256 hash).

- Generate random values with `random_bytes()` or `random_int()`.

Files and uploads

- Read uploads through `eZHTTPFile::canFetch()` and `eZHTTPFile::fetch()`; check the type by content and the size against an allow list.
- Prefer storing files as content (the `file` and `image` classes), which handles storage, clustering and SVG sanitizing (SVG images).
- Never build a file path from input without normalizing it and checking it stays inside the expected directory.
- `autoload.php` unregisters the `phar://` stream wrapper for every request (unless the runtime itself runs from an archive), so an uploaded file that is also a valid phar can never be included as code. Do not register it again.

What the kernel does for you

Protection	Where
form token on every POST by signed-in users	`ezformtoken`
|wash escaping single quotes on every PHP version	the template engine, since 6.0.15
security headers on every page	`ezpKernelWeb::securityHeaders()`
a Content Security Policy for the admin	Content Security Policy
classification of probes and attacks into `security.log`	Request logs
an audit trail of sensitive actions	Audit trail

Reporting a vulnerability

If you find a security problem in Exponential, report it privately to the address in `SECURITY.md` at the root of the repository, not in a public issue.

Related pages

- Security for applications
- Security in extensions
- Security hardening reference
- Free documentation: this page is free to copy, translate and improve under the GNU FDL 1.3.

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

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