Sicherheit in PHP-Code

Die Sicherheitsregeln für PHP-Code in Exponential an einem Ort - Eingaben validieren, Zahlen casten, SQL und Ausgaben je nach Kontext maskieren, POSTs mit dem Formular-Token schützen, Berechtigungen an jedem...

Source file: `doc/ger-de/hilfe/php-entwickler/sicherheit-in-php-code.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/php-developers/security-in-php-code.md

Diese Seite richtet sich an PHP-Entwickler, die Code schreiben, der Anfragen, Daten oder Dateien verarbeitet. Sie sammelt die Regeln, die Exponential-Code sicher halten, und verweist auf die Seite, die jede davon erklärt. Der Sicherheitsstandard des Projekts ist der Sicherheitsstandard; Administratoren härten die Installation mit Sicherheitshärtung.

Inhalt

1. Die Regeln
2. Woher Eingaben kommen
3. Berechtigungen an jedem Einstiegspunkt
4. Weiterleitungen
5. Geheimnisse
6. Dateien und Uploads
7. Was der Kernel für Sie tut
8. Eine Schwachstelle melden

Die Regeln

#	Regel	Seite
1	Validieren Sie Eingaben dort, wo sie hereinkommen: Zahlen mit `(int)` und `(float)` casten, Auswahlwerte gegen Allow-Listen prüfen, Längen begrenzen	Eingaben validieren und casten
2	Maskieren Sie Strings für SQL mit `$db->escapeString()`, oder verwenden Sie Bedingungen von `eZPersistentObject`	SQL mit eZDB, sicher
3	Maskieren Sie Ausgaben für ihren Kontext: `|wash` in Templates, `htmlspecialchars()` mit ausdrücklichen Flags in PHP, `json_encode()` für JSON	Ausgaben aus PHP maskieren
4	Ändern Sie Daten nur per POST, geschützt durch das Formular-Token	CSRF-Schutz und ezformtoken
5	Prüfen Sie Berechtigungen in jeder View, Fetch-Funktion, Serverfunktion, REST-Aktion und jedem Befehl	Berechtigungen an jedem Einstiegspunkt
6	Leiten Sie nur an Adressen weiter, die die Regeln des Kernels akzeptieren	Weiterleitungen
7	Halten Sie Geheimnisse in den Einstellungen der Installation, niemals in Code, URLs oder Logs	Geheimnisse
8	Speichern Sie Dateien über den Kernel; prüfen Sie Uploads nach Inhalt und Größe	Dateien und Uploads

#	Regel	Seite
9	Fügen Sie in von Ihnen erzeugtes Markup keine Inline-Skripte und kein `style=""` ein	Content Security Policy für Ihren Code

Woher Eingaben kommen

Behandeln Sie all das als nicht vertrauenswürdig:

- `$Params` und ungeordnete Parameter einer View: Strings aus der Adresse.
- `eZHTTPTool::instance()->postVariable()`, `getVariable()`, `variable()`: die Anfrage. Lesen Sie Eingaben über `eZHTTPTool`, nicht direkt aus `$_POST` und `$_GET`, damit die Anfragevorbereitung des Kernels greift (unter Velocity bereitet er sie pro Anfrage vor).
- Argumente von ezjscore-Serverfunktionen und REST-Routen.
- Einstellungen: von Menschen geschrieben, manchmal von anderen Erweiterungen.
- Die Datenbank: früher von jemandem geschrieben, vielleicht von einem Import.
- Header wie `Referer`, `Host` und `X-Forwarded-For`; verwenden Sie die Behandlung vertrauenswürdiger Proxys des Kernels, nicht den rohen Header (Vertrauenswürdige Proxys).

Berechtigungen an jedem Einstiegspunkt

PHP

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

- `accessWord` ist `yes`, `no` oder `limited`; bei `limited` prüfen Sie die Einschränkung gegen das Objekt (oder verwenden Sie für Inhalte `$node->canRead()`, `$object->canEdit()`).
- Die `functions` einer View in `module.php` prüft der Kernel, bevor Ihr Code läuft, aber Fetch-Funktionen, ezjscore-Serverfunktionen, REST-Aktionen und Befehle werden von Stellen aufgerufen, die nicht prüfen: Sie müssen selbst prüfen (Zugriffskontrolle im Code).
- Inhaltslisten aus `eZContentObjectTreeNode::subTreeByNodeID()` wenden `content/read` des aktuellen Benutzers an, sofern Sie nicht `'Limitation' => array()` übergeben; übergeben Sie das niemals für Ergebnisse, die Benutzer sehen.

Weiterleitungen

Leiten Sie niemals ungeprüft an eine Adresse weiter, die aus der Anfrage stammt. `eZRedirectManager` (`kernel/classes/ezredirectmanager.php`) enthält die Regeln: Pfade derselben Site, der aktuelle Host oder Hosts aus `site.ini [SiteSettings] AllowedRedirectHosts`, keine Steuerzeichen oder Backslashes, kein `//` und keine fremden Schemata, auch nicht prozentkodiert, keine Zugangsdaten in der URL. `eZRedirectManager::safeURI( $uri )` prüft eine Adresse; `redirectTo()` und `returnURI()` wählen ein sicheres Ziel mit einem Standardwert (Sichere Weiterleitungen).

Geheimnisse

- API-Schlüssel, Passwörter und Tokens liegen in `settings/override/` der Installation oder in Umgebungsvariablen, niemals im Repository Ihrer Erweiterung und niemals in einer URL.
- Protokollieren Sie sie niemals; geben Sie sie niemals in der Debug-Ausgabe aus; maskieren Sie sie in Fehlermeldungen.
- Vergleichen Sie Geheimnisse mit `hash_equals()`; speichern Sie Tokens nach Möglichkeit als Hashes (die API-Schlüssel des Kernels speichern nur einen gesalzenen HMAC-SHA-256-Hash, expservices-Tokens nur einen SHA-256-Hash).
- Erzeugen Sie Zufallswerte mit `random_bytes()` oder `random_int()`.

Dateien und Uploads

- Lesen Sie Uploads über `eZHTTPFile::canFetch()` und `eZHTTPFile::fetch()`; prüfen Sie den Typ nach Inhalt und die Größe gegen eine Allow-Liste.
- Speichern Sie Dateien bevorzugt als Inhalte (die Klassen `file` und `image`), was Speicherung, Clustering und SVG-Bereinigung übernimmt (SVG-Bilder).
- Bauen Sie niemals einen Dateipfad aus einer Eingabe, ohne ihn zu normalisieren und zu prüfen, dass er im erwarteten Verzeichnis bleibt.
- `autoload.php` hebt bei jeder Anfrage die Registrierung des Stream-Wrappers `phar://` auf (sofern die Laufzeitumgebung nicht selbst aus einem Archiv läuft), sodass eine hochgeladene Datei, die zugleich ein gültiges phar ist, niemals als Code eingebunden werden kann. Registrieren Sie ihn nicht erneut.

Was der Kernel für Sie tut

Schutz	Wo
Formular-Token bei jedem POST angemeldeter Benutzer	`ezformtoken`
|wash maskiert einfache Anführungszeichen unter jeder PHP-Version	die Template-Engine, seit 6.0.15
Sicherheits-Header auf jeder Seite	`ezpKernelWeb::securityHeaders()`
eine Content Security Policy für die Administration	Content Security Policy
Einordnung von Sondierungen und Angriffen in `security.log`	Anfrage-Logs
ein Audit-Trail sensibler Aktionen	Audit-Trail

Eine Schwachstelle melden

Wenn Sie ein Sicherheitsproblem in Exponential finden, melden Sie es vertraulich an die Adresse in `SECURITY.md` im Wurzelverzeichnis des Repositorys, nicht in einem öffentlichen Issue.

Verwandte Seiten

- Sicherheit für Anwendungen
- Sicherheit in Erweiterungen
- Referenz zur Sicherheitshärtung
- Freie Dokumentation: diese Seite darf unter der GNU FDL 1.3 frei kopiert, übersetzt und verbessert werden.

> **Werden Sie Exponential-Partner**
>
> Partner teilen ihre Informationen, stellen Mittel bereit, geben Beiträge zurück und entwickeln die Produkte mit uns, und so wächst Exponential. Eine Partner-Website und ein Dashboard sind für Mitte 2027 geplant: Werden Sie noch heute Partner, damit wir schneller wachsen und neue Funktionen früher liefern.
>
> Partner: hier beginnen | Wie Sie sich bewerben

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.