JTL-Shop: ALTCHA Spamschutz (Registrierung & Newsletter)

Selbst gehostetes, quelloffenes Spamschutz-Plugin für JTL-Shop 5 auf Basis des ALTCHA Proof-of-Work-Verfahrens (MIT-Lizenz). Schützt Registrierung und Newsletter-Anmeldung vor automatisierten Fake-Anmeldungen, ohne Daten an Dritte weiterzugeben. Ziel ist dabei bewusst der Verzicht auf Google reCAPTCHA und ähnliche externe Dienste sowie auf eine Umleitung des Datenverkehrs über Cloudflare oder vergleichbare Drittanbieter – die Prüfung läuft vollständig auf dem eigenen Server, ohne Cookies.

Der Plugin-Quellcode wird direkt aus unserem Gitea-Repository synchronisiert (Repo: **JTL-Shop-ALTCHA-Spamschutz**), sodass Änderungen dort immer sofort auch hier im Guide aktuell sind.

1. Hintergrund

Anlass war ein Bot-Problem bei einem Kunden: Über Wochen liefen massenhaft Fake-Registrierungen und Newsletter-Anmeldungen mit immer demselben Muster auf – „Test“/„Test User“/„Test Test“ in Vorname, Nachname, Straße und Ort, aber echte, unterschiedliche Fremd-E-Mail-Adressen. Rate-Limiting und andere getestete Plugins konnten das Problem nicht zuverlässig lösen, da deren Prüfungen an anderer Stelle ansetzen (z. B. an Kommentar-/Nachrichtenfeldern, die weder das Registrierungs- noch das Newsletter-Formular besitzen). Dieses Plugin prüft stattdessen direkt beim Absenden von Registrierung und Newsletter-Anmeldung selbst – unabhängig davon, welche Felder das Formular sonst enthält – und kann bestehende Lösungen vollständig ersetzen.

2. Funktionsweise

Registrierung (`/Registrieren`) und Newsletter-Anmeldung (`/Newsletter`) erhalten eine unsichtbare Sicherheitsprüfung nach dem ALTCHA-Prinzip: Proof-of-Work, selbst gehostet, MIT-Lizenz. Der Browser des Besuchers muss vor dem Absenden eine kleine Rechenaufgabe lösen (meist unter 1 Sekunde, läuft im Hintergrund, kein Klicken/Kästchen nötig). Ein Skript, das das Formular direkt per HTTP-POST anspringt – wie bei den ursprünglichen Fake-Registrierungen offenbar geschehen – hat diese Lösung nicht und wird serverseitig abgelehnt.

Kann ergänzend zu einem bestehenden Rate-Limit eingesetzt werden oder andere Spamschutz-Plugins vollständig ersetzen.

3. Installation

1. Im Adminbereich unter *Plugins → Plugin-Verwaltung* prüfen, ob eine „Plugin hochladen“-Funktion für ZIP-Dateien existiert. Falls ja: Plugin-ZIP direkt dort hochladen und installieren.
2. Falls nicht: ZIP entpacken, Ordner `fp_altcha_spamschutz` per FTP/Dateimanager in das Plugin-Verzeichnis des Shops legen (`…/plugins/fp_altcha_spamschutz/`), danach in

Plugins → Plugin-Verwaltung auf „Neue Plugins suchen“ klicken.
3. Plugin öffnen → Einstellungen → Feld **HMAC-Geheimschlüssel** mit einem zufälligen, einmaligen Wert füllen (z. B. per `openssl rand -hex 32` erzeugt) und danach nicht mehr ändern, solange das Plugin aktiv genutzt wird.
4. Übrige Einstellungen können auf den Standardwerten bleiben (Sicherheitsstufe „Mittel“, beide Formulare aktiv, Debug-Logging aus).
5. Speichern.

4. Einstellungen

Einstellung	Bedeutung
HMAC-Geheimschlüssel	Signiert die Sicherheitsprüfung. Pro Shop-Installation einmalig erzeugen, danach nicht mehr ändern.
Sicherheitsstufe (Rechenaufwand)	Niedrig/Mittel/Hoch – wie viel Rechenarbeit der Browser leisten muss, i. d. R. unter 1 Sekunde.
Gültigkeit der Prüfung	Wie lange eine erzeugte Prüfung gültig bleibt (Standard 600 Sekunden).
Kundenregistrierung / Newsletter-Anmeldung schützen	Formulare einzeln aktivierbar.
Debug-Logging	Schreibt Diagnoseinformationen ins Shop-Errorlog, nur temporär zum Testen aktivieren.

5. Pflicht-Test vor Produktivbetrieb

Dieses Plugin greift aktiv in Registrierung und Newsletter-Anmeldung ein. Die Kern-Sicherheitsprüfung (Challenge erzeugen → lösen → verifizieren) wurde isoliert erfolgreich getestet, die Einbindung in JTL-Shop selbst basiert mangels Zugriff auf den JTL-Shop-Quellcode auf der offiziellen Hook-Dokumentation und echtem, öffentlichem Code vergleichbarer, quelloffener JTL-5-Plugins. Deshalb vor jedem Produktiveinsatz auf einem Testsystem prüfen:

1. **Normale Registrierung:** Formular ausfüllen und absenden, muss wie gewohnt funktionieren (kurzes „Sicherheitsprüfung wird vorbereitet …“ / „… bestätigt“).
2. **Normale Newsletter-Anmeldung:** muss wie gewohnt funktionieren.
3. **Bypass-Versuch (wichtigster Test):** In der Entwicklerkonsole `document.querySelector('input[name=altcha]').remove()` ausführen und absenden – muss mit Fehlermeldung abgelehnt werden.
4. **Direkter POST ohne JavaScript** (z. B. curl/Postman) ohne gültiges `altcha`-Feld – muss ebenfalls abgelehnt werden.
5. Einige Tage laufen lassen und die echten Fake-Registrierungszahlen beobachten, bevor produktiv übernommen wird.
6. Danach kurz Debug-Logging aktivieren, ein paar Testanmeldungen durchführen, im Shop-Fehlerprotokoll auf `[fp_altcha_spamschutz]`-Einträge prüfen, anschließend wieder deaktivieren.

6. Technischer Hintergrund

6.1 Warum ALTCHA V1 statt des aktuellen Widgets (v3)?

Das offizielle ALTCHA-JS-Widget v3 nutzt ein neueres, komplexeres Protokoll (PBKDF2/Argon2id/Scrypt mit Key-Prefix-Matching), für das es keine robust dokumentierte „Challenge direkt einbetten“-Variante ohne zusätzlichen Netzwerk-Endpunkt gibt. Das klassische V1-Protokoll (SHA-256 Hashcash: Client sucht per Brute-Force eine Zahl n, für die SHA256(salt+n) == challenge gilt) ist dagegen vollständig dokumentiert, einfach zu prüfen und wird hier zusammen mit einem kleinen eigenen JS-Löser eingesetzt. Serverseitig kommt der offizielle, unveränderte altcha-org/altcha-lib-php-Code zum Einsatz (MIT-Lizenz).

Die Kryptografie wurde Cross-Language getestet: Der echte, unveränderte `fp-altcha.js` wurde unter Node.js gegen eine von der echten PHP-Bibliothek erzeugte Challenge gelöst und das Ergebnis erfolgreich mit derselben PHP-Bibliothek verifiziert.

6.2 Verwendete JTL-Shop-Hooks

Hook	Zweck
HOOK_SMARTY_OUTPUTFILTER	Fügt Widget-Container und Skript in das gerenderte HTML von Registrierungs- und Newsletter-Formular ein (phpQuery-DOM-Filter).
HOOK_REGISTRIEREN_PAGE_REGISTRIEREN_PLAUSI	Plausibilitätsprüfung nach Absenden des Registrierungsformulars.
HOOK_NEWSLETTER_PAGE_EMPFAENGEREINTRAGEN	Zusätzliche Absicherung kurz vor dem Speichern des Newsletter-Empfängers (defense in depth).

Registriert per EventDispatcher in `Bootstrap.php` (moderner Mechanismus), nicht über die veraltete XML-<Hooks>-Datei.

6.3 Defensives Design (fail-open)

Jede Stelle, an der sich das Plugin in den Shop einklinkt, ist mit try/catch abgesichert. Bewusste Entscheidung: Schlägt eine Prüfung aus unerwartetem Grund fehl (z. B. Plugin nicht vollständig konfiguriert), wird die Aktion durchgelassen statt den gesamten Shop lahmzulegen. Ein Bot mehr ist besser als ein Shop, bei dem sich niemand mehr registrieren kann. Die eigentliche Sicherheitsprüfung (`AltchaService::verifyPost()`) lehnt dagegen bei fehlender Konfiguration sicherheitshalber ab (fail-closed) – die beiden Ebenen ergänzen sich.

7. Bekannte Einschränkung

Falls das Registrierungs- oder Newsletterformular im verwendeten Template abweichende CSS-Klassen/Feldnamen hat als im Template „NOVA“ ermittelt (`form.register-form` bzw. ein Formular mit `input[name=„abonnieren“]`), erscheint das Widget dort nicht – der Shop bleibt aber unverändert nutzbar (kein Fehler, das Plugin erkennt das Formular einfach nicht). In dem Fall bitte über den Issue-Tracker melden (siehe unten), dann werden die Selektoren angepasst.

8. Fehlerbehebung

Problem	Ursache	Lösung
Widget erscheint nicht	Formular-Selektoren passen nicht zum Template	Siehe Abschnitt 7, Issue öffnen
Registrierung/Newsletter immer abgelehnt	Kein oder falscher HMAC-Geheimschlüssel hinterlegt	Einstellungen prüfen, Schlüssel neu setzen
Keine Fehlermeldung, Formular tut einfach nichts	JavaScript blockiert/sehr alter Browser ohne Web Crypto API	Erwartetes, sicheres Verhalten – Prüfung kann dann nicht bestätigt werden

9. Quellcode

Wird als reguläre Plugin-Dateien im Shop-Plugin-Verzeichnis abgelegt (siehe Installation). Die vendorierte ALTCHA-PHP-Bibliothek (`src/Vendor/AltchaOrg/Altcha/V1/`, MIT-Lizenz) ist Teil der Plugin-ZIP, wird hier aber nicht dupliziert – siehe altcha-org/altcha-lib-php.

9.1 info.xml (Plugin-Manifest & Einstellungen)

```
<?xml version="1.0" encoding="UTF-8"?>
<jtlshopplugin>
    <Name>ALTCHA Spamschutz</Name>
    <Description>Selbst gehosteter, quelloffener Spamschutz auf Basis des
ALTCHA Proof-of-Work-Verfahrens (MIT-Lizenz). Schuetzt Registrierung und
Newsletter-Anmeldung ohne Datenweitergabe an Dritte -- bewusst ohne Google
reCAPTCHA und ohne Umleitung ueber Cloudflare oder aehnliche Dienste. Fuer
beliebige JTL-Shop-5-Installationen geeignet.</Description>
    <Author>falk.plus</Author>
    <URL>https://falk.plus</URL>
    <PluginID>fp_altcha_spamschutz</PluginID>
    <XMLVersion>101</XMLVersion>
    <MinShopVersion>5.2.0</MinShopVersion>
    <CreateDate>2026-09-21</CreateDate>
    <Version>1.0.0</Version>
    <Install>
        <FlushTags>CACHING_GROUP_TEMPLATE</FlushTags>
        <!-- Hooks werden nicht ueber XML, sondern modern per
EventDispatcher in Bootstrap.php registriert. -->
        <Locales>
            <Variable>
                <VariableLocalized iso="GER">Sicherheitspruefung wird
vorbereitet ...</VariableLocalized>
                <VariableLocalized iso="ENG">Preparing security check
...</VariableLocalized>
                <Description></Description>
                <Name>fp_altcha_preparing</Name>
            </Variable>
```

```
            <Variable>
                <VariableLocalized iso="GER">Sicherheitspruefung
bestaetigt</VariableLocalized>
                <VariableLocalized iso="ENG">Security check
passed</VariableLocalized>
                <Description></Description>
                <Name>fp_altcha_verified</Name>
            </Variable>
            <Variable>
                <VariableLocalized iso="GER">Sicherheitspruefung
fehlgeschlagen. Bitte Seite neu laden und erneut
versuchen.</VariableLocalized>
                <VariableLocalized iso="ENG">Security check failed. Please
reload the page and try again.</VariableLocalized>
                <Description></Description>
                <Name>fp_altcha_failed</Name>
            </Variable>
            <Variable>
                <VariableLocalized iso="GER">Bitte einen Moment warten, die
Sicherheitspruefung laeuft noch.</VariableLocalized>
                <VariableLocalized iso="ENG">Please wait a moment, the
security check is still running.</VariableLocalized>
                <Description></Description>
                <Name>fp_altcha_wait</Name>
            </Variable>
            <Variable>
                <VariableLocalized iso="GER">Sicherheitspruefung konnte
nicht bestaetigt werden. Bitte Seite neu laden.</VariableLocalized>
                <VariableLocalized iso="ENG">Security check could not be
verified. Please reload the page.</VariableLocalized>
                <Description></Description>
                <Name>fp_altcha_serverfehler</Name>
            </Variable>
        </Locales>
        <Adminmenu>
            <Settingslink sort="0">
                <Name>Grundeinstellungen</Name>
                <Setting type="text" initialValue="" sort="0" conf="Y">
                    <Name>HMAC-Geheimschluessel</Name>
                    <Description>Zufaelliger, geheimer Schluessel zur
Signierung der Sicherheitspruefung. Nach der Installation automatisch
vorbelegt -- nicht aendern, solange das Plugin aktiv ist (sonst schlagen
laufende Pruefungen fehl). Niemals weitergeben.</Description>
                    <ValueName>fp_altcha_hmac_secret</ValueName>
                </Setting>
                <Setting type="selectbox" initialValue="150000" sort="10"
conf="Y">
                    <Name>Sicherheitsstufe (Rechenaufwand)</Name>
                    <Description>Wie viel Rechenarbeit der Browser des
Besuchers vor dem Absenden leisten muss. Hoeher = mehr Schutz, aber minimal
laengere Wartezeit (i.d.R. unter 1 Sekunde).</Description>
```

```
                    <ValueName>fp_altcha_max_number</ValueName>
                    <SelectboxOptions>
                        <Option value="60000" sort="1"><![CDATA[Niedrig
(schnell)]]></Option>
                        <Option value="150000" sort="2"><![CDATA[Mittel
(empfohlen)]]></Option>
                        <Option value="400000"
sort="3"><![CDATA[Hoch]]></Option>
                    </SelectboxOptions>
                </Setting>
                <Setting type="text" initialValue="600" sort="20" conf="Y">
                    <Name>Gueltigkeit der Pruefung (Sekunden)</Name>
                    <Description>Wie lange eine erzeugte Sicherheitspruefung
gueltig bleibt, bevor sie abgelaufen ist. Sollte deutlich laenger sein als
eine typische Formularausfuellzeit. Standard: 600 (10
Minuten).</Description>
                    <ValueName>fp_altcha_expiry_seconds</ValueName>
                </Setting>
            </Settingslink>
            <Settingslink sort="10">
                <Name>Formulare</Name>
                <Setting type="checkbox" initialValue="on" sort="0"
conf="Y">
                    <Name>Kundenregistrierung schuetzen</Name>
                    <Description>Sicherheitspruefung im
Registrierungsformular (/Registrieren) aktivieren.</Description>
                    <ValueName>fp_altcha_protect_register</ValueName>
                </Setting>
                <Setting type="checkbox" initialValue="on" sort="10"
conf="Y">
                    <Name>Newsletter-Anmeldung schuetzen</Name>
                    <Description>Sicherheitspruefung im Newsletter-
Anmeldeformular (/Newsletter) aktivieren.</Description>
                    <ValueName>fp_altcha_protect_newsletter</ValueName>
                </Setting>
            </Settingslink>
            <Settingslink sort="20">
                <Name>Diagnose</Name>
                <Setting type="checkbox" initialValue="" sort="0" conf="Y">
                    <Name>Debug-Logging aktivieren</Name>
                    <Description>Schreibt zusaetzliche Informationen
(erkannte Formulare, Pruefungsergebnisse) in das JTL-Shop Errorlog. Nur zum
Testen aktivieren, danach wieder ausschalten.</Description>
                    <ValueName>fp_altcha_debug</ValueName>
                </Setting>
            </Settingslink>
        </Adminmenu>
    </Install>
</jtlshopplugin>
```

2026/09/21 15:35 · jf

Fragen & Feedback

Haben Sie Fehler gefunden, Verbesserungsvorschläge oder Fragen zu dieser Datei? Hier ein neues Gitea-Issue öffnen

9.2 Bootstrap.php (Hook-Registrierung)

```php
<?php

declare(strict_types=1);

namespace Plugin\fp_altcha_spamschutz;

use JTL\Events\Dispatcher;
use JTL\Plugin\Bootstrapper;
use JTL\Plugin\PluginInterface;
use JTL\Shop;
use Plugin\fp_altcha_spamschutz\src\Handler\TemplateHandler;
use Plugin\fp_altcha_spamschutz\src\Handler\ValidationHandler;
use Plugin\fp_altcha_spamschutz\src\Service\AltchaService;

require_once __DIR__ .
'/src/Vendor/AltchaOrg/Altcha/V1/Hasher/HasherInterface.php';
require_once __DIR__ .
'/src/Vendor/AltchaOrg/Altcha/V1/Hasher/Algorithm.php';
require_once __DIR__ . '/src/Vendor/AltchaOrg/Altcha/V1/Hasher/Hasher.php';
require_once __DIR__ .
'/src/Vendor/AltchaOrg/Altcha/V1/BaseChallengeOptions.php';
require_once __DIR__ .
'/src/Vendor/AltchaOrg/Altcha/V1/ChallengeOptions.php';
require_once __DIR__ .
'/src/Vendor/AltchaOrg/Altcha/V1/CheckChallengeOptions.php';
require_once __DIR__ . '/src/Vendor/AltchaOrg/Altcha/V1/Challenge.php';
require_once __DIR__ . '/src/Vendor/AltchaOrg/Altcha/V1/Payload.php';
require_once __DIR__ . '/src/Vendor/AltchaOrg/Altcha/V1/Solution.php';
require_once __DIR__ . '/src/Vendor/AltchaOrg/Altcha/V1/Obfuscator.php';
require_once __DIR__ .
'/src/Vendor/AltchaOrg/Altcha/V1/ServerSignaturePayload.php';
require_once __DIR__ .
'/src/Vendor/AltchaOrg/Altcha/V1/ServerSignatureVerificationData.php';
require_once __DIR__ .
'/src/Vendor/AltchaOrg/Altcha/V1/ServerSignatureVerification.php';
require_once __DIR__ . '/src/Vendor/AltchaOrg/Altcha/V1/Altcha.php';
require_once __DIR__ . '/src/Service/AltchaService.php';
require_once __DIR__ . '/src/Handler/TemplateHandler.php';
require_once __DIR__ . '/src/Handler/ValidationHandler.php';

/**
 * Bootstrap-Klasse fuer das Plugin fp_altcha_spamschutz.
```

```
 *
 * Setzt einen selbst gehosteten, quelloffenen Proof-of-Work-Spamschutz
(ALTCHA, MIT-Lizenz) fuer
 * die Kundenregistrierung und die Newsletter-Anmeldung ein. Keine Daten
verlassen den eigenen
 * Server -- bewusst ohne Google reCAPTCHA oder aehnliche externe Dienste
und ohne Umleitung
 * des Datenverkehrs ueber Cloudflare oder vergleichbare Drittanbieter.
 *
 * Hintergrund/Anlass: ein Bot-Problem bei einem Kunden (massenhafte Fake-
Registrierungen und
 * Newsletter-Anmeldungen, u. a. "Test"/"Test User" in
Vorname/Nachname/Ort/Strasse, aber echte
 * Drittanbieter-E-Mail-Adressen). Andere getestete Ansaetze konnten das
Muster nicht
 * zuverlaessig erkennen; dieses Plugin prueft stattdessen direkt beim
Absenden von
 * Registrierung und Newsletter-Anmeldung selbst.
 */
class Bootstrap extends Bootstrapper
{
    public function boot(Dispatcher $dispatcher): void
    {
        parent::boot($dispatcher);

        if (!Shop::isFrontend()) {
            return;
        }

        // Die komplette Einrichtung ist defensiv umschlossen: boot() laeuft
bei JEDEM
        // Frontend-Aufruf, noch bevor irgendeine Seite gerendert wird. Ein
Fehler hier (z. B.
        // weil das Plugin gerade erst installiert und noch nicht
konfiguriert wurde) darf
        // niemals den gesamten Shop lahmlegen.
        try {
            /** @var PluginInterface $plugin */
            $plugin = $this->getPlugin();

            $altchaService = new AltchaService($plugin);
            $templateHandler = new TemplateHandler($plugin, $altchaService);
            $validationHandler = new ValidationHandler($altchaService);

            // Frueheste moegliche Pruefung der Newsletter-Anmeldung: laeuft
direkt hier, noch
            // bevor JTL-Shop das $_POST verarbeitet. Siehe
ValidationHandler::guardNewsletterSubmission().
            $validationHandler->guardNewsletterSubmission();
```

```
            // Widget (Container + Skript) in Registrierungs- und
Newsletter-Formular einfuegen.
            $dispatcher->listen(
                'shop.hook.' . \HOOK_SMARTY_OUTPUTFILTER,
                [$templateHandler, 'generalTemplateIntegration']
            );

            // Registrierung: Plausibilitaetspruefung nach
Formularabsendung.
            $dispatcher->listen(
                'shop.hook.' . \HOOK_REGISTRIEREN_PAGE_REGISTRIEREN_PLAUSI,
                [$validationHandler, 'checkRegistrationPlausibility']
            );

            // Newsletter: zusaetzliche Absicherung kurz vor dem Speichern
des Empfaengers.
            if (\defined('HOOK_NEWSLETTER_PAGE_EMPFAENGEREINTRAGEN')) {
                $dispatcher->listen(
                    'shop.hook.' .
\HOOK_NEWSLETTER_PAGE_EMPFAENGEREINTRAGEN,
                    [$validationHandler, 'checkNewsletterRecipient']
                );
            }
        } catch (\Throwable $e) {
            error_log('[fp_altcha_spamschutz] Fehler beim Initialisieren des
Plugins: ' . $e->getMessage());
        }
    }
}
```

2026/09/21 15:36 · jf

Fragen & Feedback

Haben Sie Fehler gefunden, Verbesserungsvorschläge oder Fragen zu dieser Datei? Hier ein neues Gitea-Issue öffnen

9.3 AltchaService.php (Challenge & Verifikation)

```
<?php

declare(strict_types=1);

namespace Plugin\fp_altcha_spamschutz\src\Service;

use AltchaOrg\Altcha\V1\Altcha;
use AltchaOrg\Altcha\V1\ChallengeOptions;
use AltchaOrg\Altcha\V1\Hasher\Algorithm;
use JTL\Plugin\PluginInterface;
```

```
/**
 * Kapselt die ALTCHA V1 Proof-of-Work Bibliothek (MIT-Lizenz,
https://github.com/altcha-org/altcha-lib-php).
 *
 * Warum V1 statt der aktuellen ALTCHA-Widget-Version (v3)?
 * Das offizielle JS-Widget v3 nutzt ein neueres, komplexeres Protokoll
(PBKDF2/Argon2id/Scrypt mit
 * Key-Prefix-Matching), fuer das es keine einfache, robust dokumentierte
"Challenge direkt einbetten"-
 * Variante ohne zusaetzlichen Netzwerk-Endpunkt gibt. Das klassische V1-
Protokoll (SHA-256 Hashcash:
 * Client sucht per Brute-Force eine Zahl n, fuer die SHA256(salt+n) ==
challenge gilt) ist dagegen
 * vollstaendig dokumentiert, einfach zu pruefen und wird hier zusammen mit
einem kleinen eigenen
 * JS-Loeser (assets/fp-altcha.js) eingesetzt. Serverseitig kommt weiterhin
der offizielle,
 * unveraenderte ALTCHA-Code zum Einsatz (siehe
src/Vendor/AltchaOrg/Altcha/V1).
 */
class AltchaService
{
    private PluginInterface $plugin;
    private Altcha $altcha;

    public function __construct(PluginInterface $plugin)
    {
        $this->plugin = $plugin;
        $this->altcha = new Altcha($this->getHmacSecret());
    }

    public function isConfigured(): bool
    {
        return $this->getHmacSecret() !== '';
    }

    public function isEnabledForRegistration(): bool
    {
        return $this->getConfigValue('fp_altcha_protect_register', 'on') ===
'on';
    }

    public function isEnabledForNewsletter(): bool
    {
        return $this->getConfigValue('fp_altcha_protect_newsletter', 'on')
=== 'on';
    }

    public function isDebug(): bool
```

```
    {
        return $this->getConfigValue('fp_altcha_debug', '') === 'on';
    }

    /**
     * Erzeugt eine neue Pruefung und liefert sie als Array, das 1:1 als
JSON in die Seite
     * eingebettet werden kann (siehe TemplateHandler).
     *
     * @return array<string, string|int>
     */
    public function createChallengeArray(): array
    {
        $maxNumber = (int) $this->getConfigValue('fp_altcha_max_number',
'150000');
        if ($maxNumber < 1000) {
            $maxNumber = 150000;
        }

        $expirySeconds = (int)
$this->getConfigValue('fp_altcha_expiry_seconds', '600');
        if ($expirySeconds < 30) {
            $expirySeconds = 600;
        }

        $expires = new \DateTimeImmutable('+' . $expirySeconds . '
seconds');

        $challenge = $this->altcha->createChallenge(new ChallengeOptions(
            algorithm: Algorithm::SHA256,
            maxNumber: $maxNumber,
            expires: $expires,
        ));

        return [
            'algorithm' => $challenge->algorithm,
            'challenge' => $challenge->challenge,
            'maxnumber' => $challenge->maxNumber,
            'salt' => $challenge->salt,
            'signature' => $challenge->signature,
        ];
    }

    /**
     * Prueft das per POST["altcha"] gesendete, base64-kodierte Loesungs-
Payload.
     */
    public function verifyPost(): bool
    {
        $field = $_POST['altcha'] ?? null;
```

```
        if (!\is_string($field) || $field === '') {
            $this->log('kein altcha Feld im POST gefunden');

            return false;
        }

        if (!$this->isConfigured()) {
            // Kein HMAC-Secret hinterlegt -> Plugin ist nicht korrekt
eingerichtet.
            // Sicherheitshalber ablehnen statt durchzulassen.
            $this->log('kein HMAC-Secret konfiguriert, Pruefung wird
abgelehnt');

            return false;
        }

        $verified = $this->altcha->verifySolution($field, true);
        $this->log('Pruefungsergebnis: ' . ($verified ? 'erfolgreich' :
'fehlgeschlagen'));

        return $verified;
    }

    private function getHmacSecret(): string
    {
        return $this->getConfigValue('fp_altcha_hmac_secret', '');
    }

    private function getConfigValue(string $name, string $default): string
    {
        $value = $this->plugin->getConfig()->getValue($name);

        if (!\is_string($value) || $value === '') {
            return $default;
        }

        return $value;
    }

    private function log(string $message): void
    {
        if (!$this->isDebug()) {
            return;
        }

        error_log('[fp_altcha_spamschutz] ' . $message);
    }
}
```

2026/09/21 15:37 · jf

Fragen & Feedback

Haben Sie Fehler gefunden, Verbesserungsvorschläge oder Fragen zu dieser Datei? Hier ein neues Gitea-Issue öffnen

9.4 TemplateHandler.php (Widget-Einbindung)

```php
<?php

declare(strict_types=1);

namespace Plugin\fp_altcha_spamschutz\src\Handler;

use JTL\phpQuery\phpQueryObject;
use JTL\Plugin\PluginInterface;
use JTL\Smarty\JTLSmarty;
use Plugin\fp_altcha_spamschutz\src\Service\AltchaService;

/**
 * Fuegt das ALTCHA-Widget (Container + Skript) in Registrierungs- und Newsletter-Formular ein.
 * Wird ueber HOOK_SMARTY_OUTPUTFILTER aufgerufen, also nach dem Rendern der Seite, auf dem
 * fertigen HTML-Dokument (phpQuery, jQuery-aehnliche PHP-DOM-API).
 */
class TemplateHandler
{
    private PluginInterface $plugin;
    private AltchaService $altchaService;

    public function __construct(PluginInterface $plugin, AltchaService $altchaService)
    {
        $this->plugin = $plugin;
        $this->altchaService = $altchaService;
    }

    /**
     * @param array<string, mixed> $args Enthaelt 'smarty' (JTLSmarty) und 'document' (phpQueryObject).
     */
    public function generalTemplateIntegration(array $args): void
    {
        // WICHTIG: Dieser Hook laeuft bei JEDEM Seitenaufruf im gesamten Shop. Ein Fehler hier
        // darf niemals die Anzeige irgendeiner Shop-Seite verhindern -- deshalb komplett
        // defensiv mit try/catch umschlossen. Im Zweifel wird einfach kein Widget angezeigt,
```

```
        // statt die Seite kaputt zu machen.
        try {
            $this->doTemplateIntegration($args);
        } catch (\Throwable $e) {
            if ($this->altchaService->isDebug()) {
                error_log('[fp_altcha_spamschutz] Fehler bei Template-
Integration: ' . $e->getMessage());
            }
        }
    }

    /**
     * @param array<string, mixed> $args
     */
    private function doTemplateIntegration(array $args): void
    {
        /** @var phpQueryObject $document */
        $document = $args['document'];

        $injected = false;

        if ($this->altchaService->isEnabledForRegistration()) {
            $registerForm = $document->find('form.register-form');
            if (\count($registerForm) > 0) {
                $this->injectWidget($document, $registerForm);
                $injected = true;
            }
        }

        if ($this->altchaService->isEnabledForNewsletter()) {
            $newsletterForm =
$document->find('input[name="abonnieren"]')->closest('form');
            if (\count($newsletterForm) > 0) {
                $this->injectWidget($document, $newsletterForm);
                $injected = true;
            }
        }

        if ($injected) {
            $this->includeScript($document);
        }
    }

    private function injectWidget(phpQueryObject $document, phpQueryObject
$form): void
    {
        // Nicht doppelt einfuegen (z. B. wenn die Seite mehrfach gefiltert
wird).
        if (\count($form->find('.fp-altcha')) > 0) {
            return;
```

```
        }

        $challenge = $this->altchaService->createChallengeArray();
        $challengeJson = htmlspecialchars(
            (string) json_encode($challenge, \JSON_UNESCAPED_SLASHES),
            \ENT_QUOTES,
            'UTF-8'
        );

        $isGerman = empty($_SESSION['cISOSprache']) ||
$_SESSION['cISOSprache'] === 'ger';
        $labelPreparing = $isGerman ? 'Sicherheitsprüfung wird vorbereitet
…' : 'Preparing security check …';
        $labelPreparing = htmlspecialchars($labelPreparing, \ENT_QUOTES,
'UTF-8');

        $html = '<div class="fp-altcha form-group" data-fp-altcha="' .
$challengeJson . '">'
            . '<span class="fp-altcha-status text-muted small"><i class="fa
fa-shield" aria-hidden="true"></i> '
            . $labelPreparing . '</span>'
            . '<input type="hidden" name="altcha" class="fp-altcha-input"
value="">'
            . '</div>';

        // Direkt vor dem Absende-Button einfuegen.
        $submitButton = $form->find('button[type="submit"],
input[type="submit"]');
        if (\count($submitButton) > 0) {
            $submitButton->first()->before($html);
        } else {
            $form->append($html);
        }
    }

    private function includeScript(phpQueryObject $document): void
    {
        if (\count($document->find('script[data-fp-altcha-script]')) > 0) {
            return;
        }

        // Bewusst analog zum bewaehrten Muster anderer, quelloffener JTL5-
Plugins direkt ueber
        // die globale URL_SHOP-Konstante aufgebaut, statt eine Plugin-Pfad-
API zu erraten.
        $assetUrl = \URL_SHOP . '/plugins/fp_altcha_spamschutz/assets/fp-
altcha.js';
        $assetUrl = htmlspecialchars($assetUrl, \ENT_QUOTES, 'UTF-8');

        $document->find('body')->append(
            '<script data-fp-altcha-script defer src="' . $assetUrl .
```

```
'"></script>'
        );
    }
}
```

2026/09/21 15:38 · jf

Fragen & Feedback

Haben Sie Fehler gefunden, Verbesserungsvorschläge oder Fragen zu dieser Datei? Hier ein neues Gitea-Issue öffnen

9.5 ValidationHandler.php (Prüfung bei Absenden)

```php
<?php

declare(strict_types=1);

namespace Plugin\fp_altcha_spamschutz\src\Handler;

use Plugin\fp_altcha_spamschutz\src\Service\AltchaService;

/**
 * Prueft die ALTCHA-Loesung bei Registrierung und Newsletter-Anmeldung.
 *
 * Registrierung: nutzt den dokumentierten Hook
HOOK_REGISTRIEREN_PAGE_REGISTRIEREN_PLAUSI (41),
 * der genau fuer diesen Zweck (Plausibilitaetspruefung nach
Formularabsendung) vorgesehen ist.
 * Bei fehlender/ungueltiger Pruefung wird ein Eintrag zu "fehlendeAngaben"
hinzugefuegt -- das ist
 * derselbe Mechanismus, den JTL-Shop fuer "Pflichtfeld nicht ausgefuellt"
nutzt, der Kunde bekommt
 * also die normale, vertraute Fehlermeldung des Shops.
 *
 * Newsletter: die Dokumentation zum passenden Hook
(HOOK_NEWSLETTER_PAGE_EMPFAENGEREINTRAGEN) ist
 * nicht eindeutig genug, um sich allein darauf zu verlassen. Deshalb wird
zusaetzlich ganz frueh
 * (direkt in Bootstrap::boot(), vor jeder Shop-Verarbeitung) geprueft:
schlaegt die Pruefung fehl,
 * wird das POST-Feld "abonnieren" entfernt, sodass JTL-Shop die Anmeldung
erst gar nicht verarbeitet.
 * Der Hook-Handler unten ist eine zusaetzliche Absicherung ("defense in
depth"), falls dieser fruehe
 * Filter aus irgendeinem Grund nicht greift.
 */
class ValidationHandler
{
```

```
    private AltchaService $altchaService;

    public function __construct(AltchaService $altchaService)
    {
        $this->altchaService = $altchaService;
    }

    /**
     * @param array<string, mixed> $args
     */
    public function checkRegistrationPlausibility(array $args): void
    {
        try {
            if (!$this->altchaService->isEnabledForRegistration()) {
                return;
            }

            if ($this->altchaService->verifyPost()) {
                return;
            }

            if (isset($args['fehlendeAngaben']) &&
\is_array($args['fehlendeAngaben'])) {
                $args['fehlendeAngaben'][] = 'Sicherheitspruefung';
            }

            if (\array_key_exists('nReturnValue', $args)) {
                $args['nReturnValue'] = 0;
            }
        } catch (\Throwable $e) {
            // Bewusste Entscheidung: bei einem unerwarteten Fehler (z. B.
Plugin nicht vollstaendig
            // konfiguriert) lieber die Registrierung durchlassen, als die
komplette Registrierungs-
            // seite fuer echte Kunden zu blockieren. Ein Bot mehr ist
besser als ein Shop, bei dem
            // sich niemand mehr registrieren kann.
            if ($this->altchaService->isDebug()) {
                error_log('[fp_altcha_spamschutz] Fehler bei
Registrierungspruefung: ' . $e->getMessage());
            }
        }
    }

    /**
     * Fruehe Pruefung fuer die Newsletter-Anmeldung. Wird direkt aus
Bootstrap::boot() aufgerufen,
     * nicht ueber einen Hook, damit sie garantiert vor der Verarbeitung
durch newsletter.php laeuft.
     */
    public function guardNewsletterSubmission(): void
```

```
    {
        try {
            if (!$this->altchaService->isEnabledForNewsletter()) {
                return;
            }

            $isNewsletterSubscribeSubmit = ($_SERVER['REQUEST_METHOD'] ??
'') === 'POST'
                && isset($_POST['abonnieren']);

            if (!$isNewsletterSubscribeSubmit) {
                return;
            }

            if ($this->altchaService->verifyPost()) {
                return;
            }

            // Feld entfernen, damit JTL-Shop die Anmeldung nicht als
"abonnieren"-Request erkennt
            // und stattdessen nur das (leere) Formular erneut anzeigt.
            unset($_POST['abonnieren']);
            $_SESSION['fp_altcha_newsletter_error'] = true;
        } catch (\Throwable $e) {
            // Wird direkt in Bootstrap::boot() aufgerufen (nicht ueber
einen Hook) -- ein
            // Fehler hier darf auf keinen Fall den kompletten Seitenaufbau
verhindern.
            if ($this->altchaService->isDebug()) {
                error_log('[fp_altcha_spamschutz] Fehler bei Newsletter-
Fruehpruefung: ' . $e->getMessage());
            }
        }
    }

    /**
     * Zusaetzliche Absicherung ueber
HOOK_NEWSLETTER_PAGE_EMPFAENGEREINTRAGEN, falls die fruehe
     * Pruefung in guardNewsletterSubmission() aus irgendeinem Grund nicht
gegriffen hat.
     *
     * @param array<string, mixed> $args Enthaelt 'oNewsletterEmpfaenger'
(stdClass, per Referenz).
     */
    public function checkNewsletterRecipient(array $args): void
    {
        try {
            if (!$this->altchaService->isEnabledForNewsletter()) {
                return;
            }
```

```
            if (!empty($_SESSION['fp_altcha_newsletter_error'])) {
                // Bereits in guardNewsletterSubmission() als ungueltig
erkannt: Empfaenger-E-Mail
                // leeren, damit JTL-Shops eigene Pflichtfeld-Pruefung den
Insert verhindert.
                if (isset($args['oNewsletterEmpfaenger']) &&
\is_object($args['oNewsletterEmpfaenger'])) {
                    $args['oNewsletterEmpfaenger']->cEmail = '';
                }
            }
        } catch (\Throwable $e) {
            if ($this->altchaService->isDebug()) {
                error_log('[fp_altcha_spamschutz] Fehler bei Newsletter-
Absicherung: ' . $e->getMessage());
            }
        }
    }
}
```

2026/09/21 15:39 · jf

Fragen & Feedback

Haben Sie Fehler gefunden, Verbesserungsvorschläge oder Fragen zu dieser Datei? Hier ein neues Gitea-Issue öffnen

9.6 fp-altcha.js (Client-seitiger Löser)

```
/*!
 * fp-altcha.js -- kleiner, eigenstaendiger Loeser fuer das klassische
ALTCHA V1
 * Proof-of-Work-Verfahren (SHA-256 Hashcash-Stil).
 *
 * Sucht fuer eine vom Server vorgegebene Aufgabe {algorithm, challenge,
salt, maxnumber, signature}
 * eine Zahl n mit 0 <= n <= maxnumber, fuer die SHA-256(salt + n) ==
challenge gilt, und traegt das
 * Ergebnis base64-kodiert in ein verstecktes Formularfeld "altcha" ein. Die
Verifikation auf dem
 * Server erfolgt mit der offiziellen ALTCHA-PHP-Bibliothek (siehe
src/Vendor/AltchaOrg/Altcha/V1).
 *
 * Laeuft vollstaendig lokal im Browser -- es wird keine Anfrage an einen
Drittanbieter gesendet.
 */
(function () {
    'use strict';
```

```
    function bufferToHex(buffer) {
        var bytes = new Uint8Array(buffer);
        var hex = '';
        for (var i = 0; i < bytes.length; i++) {
            var h = bytes[i].toString(16);
            hex += h.length === 1 ? '0' + h : h;
        }
        return hex;
    }

    function base64Encode(str) {
        var bytes = new TextEncoder().encode(str);
        var binary = '';
        for (var i = 0; i < bytes.length; i++) {
            binary += String.fromCharCode(bytes[i]);
        }
        return btoa(binary);
    }

    async function solve(algorithm, salt, target, maxNumber) {
        if (algorithm !== 'SHA-256') {
            // Nur SHA-256 wird von diesem kleinen Loeser unterstuetzt.
            return null;
        }

        var encoder = new TextEncoder();

        for (var n = 0; n <= maxNumber; n++) {
            var data = encoder.encode(salt + n);
            var digest = await crypto.subtle.digest('SHA-256', data);
            if (bufferToHex(digest) === target) {
                return n;
            }
        }

        return null;
    }

    async function processWidget(container) {
        var raw = container.getAttribute('data-fp-altcha');
        if (!raw) {
            return;
        }

        var task;
        try {
            task = JSON.parse(raw);
        } catch (e) {
            return;
        }
```

```
        var status = container.querySelector('.fp-altcha-status');
        var input = container.querySelector('.fp-altcha-input');
        if (!input) {
            return;
        }

        try {
            var number = await solve(task.algorithm, task.salt,
task.challenge, task.maxnumber);

            if (number === null) {
                if (status) {
                    status.textContent = container.getAttribute('data-label-
failed') || 'Sicherheitspruefung fehlgeschlagen. Bitte Seite neu laden.';
                }
                container.setAttribute('data-fp-altcha-state', 'failed');
                return;
            }

            var payload = {
                algorithm: task.algorithm,
                challenge: task.challenge,
                number: number,
                salt: task.salt,
                signature: task.signature
            };

            input.value = base64Encode(JSON.stringify(payload));
            container.setAttribute('data-fp-altcha-state', 'verified');

            if (status) {
                status.innerHTML = '<i class="fa fa-check" aria-
hidden="true"></i> ' +
                    (container.getAttribute('data-label-verified') ||
'Sicherheitspruefung bestaetigt');
            }
        } catch (e) {
            container.setAttribute('data-fp-altcha-state', 'failed');
            if (status) {
                status.textContent = container.getAttribute('data-label-
failed') || 'Sicherheitspruefung fehlgeschlagen. Bitte Seite neu laden.';
            }
        }
    }

    function guardForm(form) {
        form.addEventListener('submit', function (event) {
            var widgets = form.querySelectorAll('.fp-altcha');
            for (var i = 0; i < widgets.length; i++) {
                var state = widgets[i].getAttribute('data-fp-altcha-state');
```

```
                if (state !== 'verified') {
                    event.preventDefault();
                    var status = widgets[i].querySelector('.fp-altcha-
status');
                    if (status) {
                        status.textContent = 'Bitte einen Moment warten, die
Sicherheitspruefung laeuft noch.';
                    }
                    return;
                }
            }
        });
    }

    function init() {
        if (!window.crypto || !window.crypto.subtle) {
            // Sehr alte Browser ohne Web Crypto API: Pruefung kann nicht
durchgefuehrt werden.
            // Das Formular bleibt dann serverseitig blockiert (kein
gueltiges "altcha" Feld).
            return;
        }

        var containers = document.querySelectorAll('.fp-altcha[data-fp-
altcha]');
        containers.forEach(function (container) {
            processWidget(container);

            var form = container.closest('form');
            if (form && !form.hasAttribute('data-fp-altcha-guarded')) {
                form.setAttribute('data-fp-altcha-guarded', '1');
                guardForm(form);
            }
        });
    }

    if (document.readyState === 'loading') {
        document.addEventListener('DOMContentLoaded', init);
    } else {
        init();
    }
})();
```

2026/09/21 15:40 · jf

Fragen & Feedback

Haben Sie Fehler gefunden, Verbesserungsvorschläge oder Fragen zu dieser Datei? Hier ein neues Gitea-Issue öffnen

Anleitung erstellt auf Basis einer echten Kundenumsetzung. Plugin-Quellcode wird live aus Gitea (Repo: JTL-Shop-ALTCHA-Spamschutz) synchronisiert