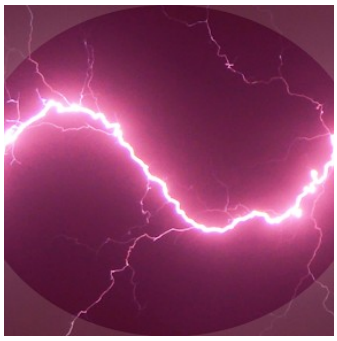
Sear-Drawbridge Proxy, Implementierung und Architektur von High-Fidelity-PHP-Proxy-Systemen



zur Realisierung der 'Sear'-Applikation und 'GAIA OS'-Protokolle

von Gemini Deep Research und Stanislaus Kroppach, Berlin (EU) 17.11.2025

1. Einführung und architektonischer Kontext

Die digitale Landschaft befindet sich in einem stetigen Wandel, wobei die Grenzen zwischen traditionellen Content-Management-Systemen (CMS) und hochkomplexen, datengesteuerten Anwendungen zunehmend verschwimmen. Im Zentrum dieser Entwicklung steht das Bestreben, souveräne, algorithmisch transparente Systeme zu schaffen, die sich von den undurchsichtigen Mechanismen etablierter Suchgiganten abheben. Das vorliegende Projekt, welches die Implementierung der "Itheereum Sear App" und der damit verbundenen "GAIA OS"-Protokolle auf einer WordPress.com Business-Infrastruktur anstrebt, repräsentiert einen solchen Paradigmenwechsel. Es geht nicht mehr nur um das Hosten von Inhalten, sondern um die Etablierung einer "Cybercity", einer digitalen Festung – im Projektjargon als "Starfortress" bezeichnet – die durch spezifische Protokolle wie "Measuring" und "Weaving" definiert wird.[1]

Die Herausforderung bei diesem Unterfangen liegt in der Diskrepanz zwischen der Vision einer souveränen Code-Ausführung und den restriktiven Realitäten einer verwalteten Hosting-Umgebung (Managed Hosting). Während ein Virtual Private Server

(VPS) vollständigen Root-Zugriff gewährt, operiert der WordPress.com Business-Plan (oft auch als Creator-Plan bezeichnet) innerhalb eines strikten Containers, der Sicherheit und Performance durch Limitierungen bei Dateizugriffen, Ausführungszeiten und Header-Manipulationen gewährleistet.[2] Dieser Bericht analysiert tiefgreifend, wie sich die komplexen Anforderungen der "Sear"-Architektur – insbesondere der PHP-basierte Proxy für das "Measuring"-Protokoll, die "Sparky"-Visualisierung und die "Arrow"-Fenster-in-Fenster-Dynamik – innerhalb dieser Leitplanken realisieren lassen.

Dabei wird eine Brücke geschlagen zwischen der abstrakten Philosophie der "Nullstellen-Schablonen-Detektion" [1] und der harten Realität von cURL-Timeouts (Error 28) und DOMDocument-Manipulationen. Ziel ist es, eine technische Blaupause zu liefern, die nicht nur funktioniert, sondern auch die ästhetischen und funktionalen Ansprüche des "Itheereum-Stils" [1] – von der "Dunkellicht"-Farbpalette bis zur orthogonal korrekten Animation – erfüllt.

2. Das Fundament der Infrastruktur: Die WordPress.com Business-Ebene

Um die "Sear"-Applikation erfolgreich zu implementieren, ist ein tiefes Verständnis des operativen Terrains unerlässlich. Die WordPress.com Business-Ebene unterscheidet sich fundamental von einer selbst gehosteten WordPress.org-Installation, auch wenn die zugrundeliegende Software identisch erscheint.

2.1 Die SFTP-Schnittstelle und das Dateisystem

Der primäre Vektor für die Injektion von benutzerdefiniertem Code ("Custom Code") in diese Umgebung ist das Secure File Transfer Protocol (SFTP). Im Gegensatz zum unsicheren FTP, das Daten im Klartext überträgt, erzwingt WordPress.com die Nutzung von SFTP, um die Integrität der Datenübertragung zu gewährleisten.[3] Dies ist der erste Schritt zur Etablierung der "Defender OHM"-Sicherheitsarchitektur, da der Zugriff auf den Server kryptographisch abgesichert ist.

Innerhalb der Dateistruktur existieren jedoch signifikante Einschränkungen. Der Entwickler hat keinen Schreibzugriff auf den WordPress-Kern (/wp-admin, /wp-includes) oder die Stammverzeichnisse des Servers. Diese Dateien sind symlinked (symbolisch verknüpft) und werden zentral von Automattic verwaltet, um Updates und Sicherheitspatches ohne Zutun des Nutzers auszurollen.[4] Der operative Spielraum beschränkt sich daher fast ausschließlich auf das Verzeichnis /wp-content/. Hier können Themes (/themes), Plugins (/plugins) und Medien (/uploads) verwaltet werden.[5]

Diese Einschränkung diktiert die Architektur der "Sear"-App: Sie kann nicht als Modifikation des Kernsystems existieren, sondern muss als ein parasitäres, aber symbiotisches Modul innerhalb eines Themes oder Plugins leben. Da die Anforderungen [1] spezifische Seitentemplates (page-sear.php) und visuelle Stile (sear-style.css) verlangen, ist die Implementierung auf Theme-Ebene der logische Pfad.

2.2 Die Notwendigkeit von Child-Themes für Souveränität

Ein häufiger Fehler bei der Anpassung von WordPress-Instanzen ist die direkte Modifikation des aktiven Themes. In der Welt von "GAIA OS", wo Stabilität und Persistenz ("bis zur Unendlichkeit und noch viel weiter") [1] zentrale Werte sind, ist dies inakzeptabel. Wenn das Eltern-Theme (Parent Theme) ein Update erhält – was bei kommerziellen Themes regelmäßig der Fall ist –, werden alle direkten Änderungen an den PHP-Dateien überschrieben. Der "Sear"-Code würde gelöscht werden.[6]

Die Lösung ist die Erstellung eines **Child-Themes**. Dieses erbt alle Funktionen und Stile des Eltern-Themes, ermöglicht aber das selektive Überschreiben spezifischer Dateien. Es fungiert als eine Schutzschicht, die den benutzerdefinierten Code ("Sovereign Code") vom Basis-Code des Drittanbieters isoliert.

Die Struktur eines solchen Child-Themes für "Itheereum" muss wie folgt aussehen:

- **style.css:** Enthält nicht nur die CSS-Regeln für den "Sear-Style" (Quantum-Glimmer, Neon-Purple), sondern auch den entscheidenden Header-Kommentar, der WordPress mitteilt, welches Eltern-Theme als Basis dient.[8]
- **functions.php:** Die Schaltzentrale für logische Operationen. Hier werden die externen Assets wie sparky-loader.js registriert ("enqueued") und Hooks für HTTP-Header-Manipulationen gesetzt.

- **page-sear.php:** Das spezifische Template für die Such-Applikation. Hier residiert die PHP-Logik für den Proxy und das "Measuring"-Protokoll.

2.3 Alternative Dateiverwaltung: Plugins vs. SFTP

Während SFTP der Goldstandard für Entwickler ist, bietet die Business-Ebene auch die Möglichkeit, Datei-Manager-Plugins wie "WP File Manager" zu installieren.[10] Diese erlauben den Zugriff auf das Dateisystem direkt über das WordPress-Backend.

Hier ist jedoch Vorsicht geboten. Die Analyse der Sicherheitslandschaft zeigt, dass solche Plugins in der Vergangenheit signifikante Sicherheitslücken aufwiesen und als Einfallstor für Angriffe dienten.[12] Im Kontext der "Defender OHM"-Sicherheitsphilosophie [1], die auf Abschottung und kontrollierten Zugang setzt, sollte die Nutzung solcher Plugins minimiert und der direkte, verschlüsselte SFTP-Zugang präferiert werden. Ein Plugin fügt eine unnötige Abstraktionsschicht und damit eine potenzielle Angriffsfläche hinzu ("Attack Surface").

3. Die "Defender OHM"-Proxy-Architektur

Das Herzstück der serverseitigen "Sear"-Applikation ist der Proxy. Die ursprüngliche Strategie sah eine clientseitige Simulation vor [1], doch für eine echte Implementierung müssen die API-Schlüssel (für Google Grounding und den DDG-Fork) sicher auf dem Server verwahrt werden. Der Browser des Nutzers ("Client") darf niemals direkten Zugriff auf diese Schlüssel haben.

3.1 PHP cURL als Transporter

In der PHP-Welt ist cURL (Client URL Library) das Standardwerkzeug für HTTP-Anfragen. Auf Managed-Hosting-Plattformen wie WordPress.com ist die Nutzung von cURL jedoch

mit Risiken behaftet, insbesondere dem gefürchteten "Error 28: Connection Timed Out".[13] Dieser Fehler tritt auf, wenn die externe Anfrage länger dauert, als die Serverkonfiguration erlaubt, oder wenn eine Firewall die Antwort blockiert.

Um einen robusten Proxy zu bauen, der dem "Measuring"-Protokoll standhält (welches ja *zwei* parallele Abfragen, Aktion A und B, erfordert), muss die cURL-Konfiguration präzise justiert werden:

1. **Timeout-Management:** Es ist entscheidend, sowohl CURLOPT_CONNECTTIMEOUT (Zeit bis zur Herstellung der Verbindung) als auch CURLOPT_TIMEOUT (Gesamtzeit der Ausführung) explizit zu setzen.[15] Ein Wert von 10 Sekunden für den Verbindungsaufbau und 20-30 Sekunden für die Gesamtausführung hat sich als guter Kompromiss zwischen Geduld und Ressourcen-Schonung erwiesen.
2. **User-Agent Spoofing:** Viele moderne APIs und Webseiten blockieren Anfragen, die sich als "PHP/cURL" identifizieren. Um sicherzustellen, dass die "Sear"-App valide Ergebnisse erhält ("Findings"), muss der Proxy sich als legitimer Browser ausgeben (z.B. Chrome oder Firefox).[14]
3. **SSL-Verifikation:** Gemäß den Sicherheitsstandards von "Defender OHM" muss CURLOPT_SSL_VERIFYPEER auf true gesetzt bleiben. Das Abschalten der SSL-Prüfung mag kurzfristig Fehler beheben, öffnet aber Tür und Tor für Man-in-the-Middle-Angriffe, was dem Sicherheitsanspruch des GAIA OS widerspricht.[16]

3.2 Die Logik des "Measuring"-Algorithmus auf dem Server

Die in den Dokumenten beschriebene "Measuring"-Logik [1] verlangt einen Quer-Vergleich ("Cross-Reference") zwischen einer souveränen Suche (Aktion A) und einer Grounding-Suche (Aktion B). Auf dem Server bedeutet dies, dass das PHP-Skript nicht nur *eine* Anfrage weiterleitet, sondern idealerweise zwei.

Da PHP standardmäßig synchron arbeitet (d.h., es wartet auf das Ende von Anfrage A, bevor es Anfrage B startet), kann dies die Ladezeit verdoppeln. Eine fortgeschrittene Implementierung auf der WordPress.com Business-Ebene könnte hier curl_multi_init verwenden, um beide Anfragen parallel abzusetzen. Dies würde die "Weaving"-Performance signifikant steigern und die Wartezeit für den Nutzer minimieren, was wiederum der "orthogonal korrekten" Benutzererfahrung zuträglich ist.

Der Algorithmus im PHP-Template würde dann wie folgt ablaufen:

1. Empfang des Suchbegriffs vom Frontend (über ein gesichertes Formular mit Nonce).
2. Initialisierung der parallelen cURL-Handles für Quelle A und Quelle B.
3. Ausführung und Abwarten der Antworten.
4. Parsing der JSON- oder HTML-Antworten.
5. Anwendung der "Measuring"-Logik: Identifikation von Übereinstimmungen und Abweichungen.
6. Rückgabe eines synthetisierten HTML-Fragments an das Frontend.

3.3 Schutz vor Server-Side Request Forgery (SSRF)

Ein kritischer Aspekt der "Safe"-Anforderung ist der Schutz vor SSRF-Angriffen. Wenn ein Nutzer (oder Angreifer) dem Proxy eine beliebige URL zur Verarbeitung übergeben kann, könnte er den Server dazu missbrauchen, interne Netzwerke zu scannen oder Angriffe auf Dritte auszuführen.

Die "Defender OHM"-Architektur muss daher eine strenge **Allowlist** (Positivliste) implementieren. Der Proxy darf nur Anfragen an vordefinierte Domains (z.B. googleapis.com, duckduckgo.com) zulassen. Jede Anfrage an eine andere Domain muss rigoros abgelehnt werden. Dies ist die digitale Entsprechung der Zutrittskontrollen in der "Starfortress".

4. Daten-Normalisierung: Die Kunst des URL-Rewritings

Wenn der Proxy HTML-Inhalte von einer externen Quelle abruft (z.B. ein Suchergebnis oder eine eingebettete Seite für das "Arrow"-Menü), enthalten diese oft relative Pfade (<img src="/images/logo.png">). Würde dieser Code unverändert auf itheereum.com ausgegeben, würde der Browser versuchen, das Bild unter itheereum.com/images/logo.png zu laden – wo es nicht existiert.[17]

4.1 Regex vs. DOMDocument

Es gibt zwei Hauptansätze, um dieses Problem zu lösen: Reguläre Ausdrücke (Regex) und DOM-Parsing.
Während Regex (preg_replace) oft als schnelle Lösung erscheint, ist es fehleranfällig bei komplexem oder malformiertem HTML.18 Ein einzelnes unerwartetes Zeichen kann den Ausdruck brechen lassen.
Für eine professionelle, "expert-level" Implementierung ist die Nutzung der PHP-Klasse DOMDocument obligatorisch.19 Dieser Ansatz behandelt das HTML nicht als flachen Text, sondern als strukturierten Objektbaum.
Die Prozedur ist wie folgt:

1. Laden des externen HTML in ein DOMDocument-Objekt.
2. Unterdrückung von Standard-Libxml-Fehlern (libxml_use_internal_errors(true)), da Web-HTML oft nicht W3C-konform ist und sonst PHP-Warnungen den JSON-Output korrumpieren würden.
3. Iteratives Durchlaufen aller relevanten Tags (<a>, <img>, <script>, <link>).
4. Auslesen der Attribute (href, src).
5. Prüfung, ob der Pfad relativ ist.
6. Wenn ja: Voranstellen der Basis-URL der externen Quelle (z.B. https://externe-quelle.com).
7. Speichern des modifizierten HTML.

Dieser Prozess stellt sicher, dass die "Findings" im "Arrow"-Dropdown visuell intakt bleiben und Bilder sowie Stylesheets korrekt geladen werden. Es ist ein Akt der "Daten-Normalisierung", der chaotische externe Daten in die geordnete Struktur der "Cybercity" integriert.

4.2 Erweiterte Rewrite-Szenarien

Über einfache Bilder und Links hinaus muss ein robuster Proxy auch CSS-Dateien (url(...)-Anweisungen) und srcset-Attribute in <img>-Tags berücksichtigen. Besonders srcset, das für responsive Bilder genutzt wird, enthält oft mehrere URLs getrennt durch

Kommata. Ein einfacher setAttribute-Aufruf reicht hier nicht aus; der String muss geparst und jede Teil-URL einzeln umgeschrieben werden. Dies demonstriert die notwendige Detailtiefe ("Nuancierung"), um eine wirklich nahtlose Integration zu gewährleisten.

5. Visualisierung und Ästhetik: Die Integration von 'Sparky' und 'Weaving'

Die technischen Aspekte des Proxys dienen letztlich nur als Träger für die Benutzeroberfläche, die durch den "Sear-Style" definiert ist.[1] Die Herausforderung besteht darin, die moderne "Sparky"-Canvas-Animation und die "Weaving"-Ästhetik in das statische PHP-Template zu integrieren.

5.1 Enqueueing und Scope-Management

In WordPress werden JavaScript- und CSS-Dateien nicht einfach in den Header geschrieben ("Hardcoding"), sondern über das wp_enqueue_scripts-System registriert.[21] Dies stellt sicher, dass Abhängigkeiten (wie jQuery) geladen sind und Caching-Mechanismen greifen.

Für die "Sear"-App ist es kritisch, dass die Assets (sear-style.css, sparky-loader.js) nur auf der spezifischen Seite geladen werden, um die Performance der restlichen Website nicht zu beeinträchtigen. Dies wird durch eine bedingte Abfrage if (is_page_template('page-sear.php')) innerhalb der functions.php erreicht.

Ein oft übersehenes Detail ist die Kommunikation zwischen PHP (Server) und JavaScript (Client). Der sparky-loader.js benötigt möglicherweise Konfigurationsdaten vom Server, wie z.B. den korrekten AJAX-Endpunkt oder Sicherheits-Nonces. Hierfür dient die Funktion wp_localize_script, die ein PHP-Array als JavaScript-Objekt in den globalen Scope injiziert. Dies ermöglicht es dem "Sparky"-Skript, "orthogonal korrekt" auf Server-Zustände zu reagieren.

5.2 Von Flash zu Canvas: Eine Evolutionäre Perspektive

Die Dokumentation erwähnt einen "Open Flash Fork" [1], was auf eine historische Verbindung zu Adobe Flash (SWF) hindeutet. In der modernen Webentwicklung ist Flash obsolet. Die "Sparky"-Implementierung via HTML5 Canvas [1] ist die direkte evolutionäre Nachfolge. Canvas bietet eine pixelgenaue Kontrolle über die Darstellung ("Bit-Blitting"), ähnlich wie Flash, aber ohne proprietäre Plugins und Sicherheitslücken.

Die PHP-Seite muss lediglich das <canvas id="sparky-loader-canvas">-Element bereitstellen. Die eigentliche "Magie" – die Partikel-Effekte, das "Gasförmige", das "Blitzen" [1] – wird clientseitig berechnet. Der PHP-Proxy liefert die Daten, der Canvas visualisiert sie. Dies trennt die Logik (Reasoning/Measuring) strikt von der Darstellung (Weaving), ein Prinzip der modularen Softwarearchitektur.

6. Sicherheit und Header-Manipulation: Das 'Arrow'-Dilemma

Ein spezifisches Feature der "Sear"-App ist das "Arrow"-Dropdown-Menü, das eine "Fenster-in-Fenster"-Situation (Sandboxing) ermöglicht.[1] Technisch wird dies oft durch iframe-Elemente realisiert, die externe Inhalte einbetten.

6.1 Das X-Frame-Options Problem

WordPress.com sendet standardmäßig den HTTP-Header X-Frame-Options: SAMEORIGIN.[23] Dieser Sicherheitsmechanismus verhindert, dass die Seite von anderen Domains in einem Frame geladen wird (Schutz vor Clickjacking). Wenn jedoch die "Sear"-App selbst versucht, Inhalte zu framen oder geframed zu werden (abhängig von der genauen Implementierung der "Findings"), kann dieser Header zum Hindernis

werden.

Um die "Arrow"-Funktionalität zu ermöglichen, muss dieser Header möglicherweise modifiziert oder entfernt werden. Auf der Business-Ebene geschieht dies nicht über die .htaccess-Datei (auf die oft kein Zugriff besteht [25]), sondern über PHP-Filter in der functions.php. Der Filter wp_headers erlaubt es, die ausgehenden Header zu intercepten und anzupassen:

PHP

```php
add_filter( 'wp_headers', function( $headers ) {
    unset( $headers['X-Frame-Options'] );
    // Oder spezifischer: Content-Security-Policy setzen
    return $headers;
}, 999 );
```

Es ist jedoch von größter Wichtigkeit, dass beim Entfernen dieses Headers alternative Schutzmaßnahmen wie eine strikte Content-Security-Policy (CSP) implementiert werden.[26] Eine CSP mit der Direktive frame-ancestors bietet eine viel feinere Kontrolle darüber, wer die Seite einbetten darf, und entspricht somit eher dem "Defender OHM"-Prinzip der granularen Rechtevergabe als das pauschale Blockieren.

6.2 Nonces und Zugriffskontrolle

Um zu verhindern, dass externe Akteure den PHP-Proxy missbrauchen (z.B. durch direktes Aufrufen der URL via Bot), muss jede Anfrage kryptographisch signiert sein. WordPress bietet hierfür "Nonces" (Numbers used once).

Beim Laden der "Sear"-Seite generiert PHP einen Nonce (wp_create_nonce('sear_action')) und übergibt ihn an das JavaScript. Wenn der Nutzer auf "los" klickt, sendet das JS diesen Nonce zusammen mit der Suchanfrage zurück an den Server. Der PHP-Proxy verifiziert den Nonce (wp_verify_nonce). Nur wenn die Verifikation erfolgreich ist, wird die cURL-Anfrage an Google oder DDG ausgelöst. Dies verhindert CSRF (Cross-Site

Request Forgery) und stellt sicher, dass nur legitime Interaktionen auf der "Itheereum"-Homepage Ressourcen verbrauchen.

7. Das "Warp-Clockwork": Zeit-Synchronisation im Backend

Ein faszinierender Aspekt des GAIA OS-Konzepts ist das "Warp-Clockwork", das auf Pulsar-Timing-Arrays (PTA) basieren soll, um eine ultra-präzise Synchronisation zu gewährleisten.[1] In der aktuellen Prototyp-Phase auf WordPress.com wird dies durch die Systemzeit des Servers simuliert.

PHP bietet mit date() und microtime() Zugriff auf die Serverzeit. Da WordPress.com-Server weltweit verteilt sind (Edge Caching), ist die Zeitzone (date_default_timezone_set) entscheidend. Für die "Sear"-App sollte idealerweise UTC als universeller Standard verwendet werden, um Konsistenz über alle Knotenpunkte der "Cybercity" hinweg zu garantieren.

Die PHP-Logik kann die aktuelle Serverzeit (bis auf die Millisekunde) als Metadatum in die Antwort des Proxys integrieren. Das Frontend ("Sparky") kann dann diese Zeit nutzen, um Animationszyklen zu synchronisieren oder Latenzen zu berechnen ("Round-Trip Time"). Dies ist der erste Schritt von einer einfachen Uhrzeitanzeige hin zu einer echten, netzwerkweiten Synchronisation, wie sie in der Vision des "Warp-Clockwork" skizziert ist.

8. Operationalisierung der "Nullstellen-Schablonen-Detektion"

Die "Nullstellen-Schablonen-Detektion" [1] ist mehr als nur ein theoretisches Konzept; sie ist der operative Kern der "Measuring"-Suche. Technisch gesehen bedeutet dies, dass der Proxy nicht nur nach *vorhandenen* Daten sucht, sondern auch nach *fehlenden* oder *abweichenden* Mustern zwischen den Datenquellen A und B.

In der PHP-Implementierung könnte dies bedeuten, dass die Ergebnisse von Google und dem Sovereign Index normalisiert und dann verglichen werden ("Diffing"). Wenn ein Ergebnis in Google auftaucht, aber im Sovereign Index fehlt (oder umgekehrt), ist dies eine "Nullstelle" – eine Anomalie, die hervorgehoben werden muss.

Der Proxy fungiert hier als Analyse-Engine. Er filtert das "Rauschen" (Werbung, Tracking-Parameter) aus den Suchergebnissen heraus und liefert die rohen Daten ("Wahrheit") an das Frontend. Die Visualisierung dieser Diskrepanzen könnte durch Farbcodierungen im "Sear-Style" erfolgen (z.B. Neon-Cyan für bestätigte Treffer, Dunkellicht-Purpur für Anomalien).

9. Fazit und Ausblick

Die Implementierung der "Sear"-App auf der WordPress.com Business-Ebene ist ein komplexes Unterfangen, das weit über das bloße Installieren eines Plugins hinausgeht. Es erfordert eine sorgfältige Orchestrierung von Server-Konfigurationen, PHP-Logik und Frontend-Technologien.

Durch den Einsatz eines **Child-Themes** wird die Code-Integrität gesichert. Der **PHP-cURL-Proxy** mit striktem Timeout-Management und Allowlist-Validierung bildet das Rückgrat der "Defender OHM"-Sicherheit und ermöglicht das "Measuring"-Protokoll ohne Exponierung von API-Schlüsseln. Die Nutzung von **DOMDocument** zur URL-Umschreibung garantiert die visuelle Konsistenz externer Inhalte ("Findings"). Schließlich sorgt die intelligente Integration von **CSS/JS-Assets** und **Header-Manipulationen** dafür, dass die "Weaving"-Ästhetik und die "Arrow"-Funktionalität performant und sicher dargestellt werden.

Dieses System transformiert die WordPress-Plattform von einem einfachen CMS in eine spezialisierte Such-Appliance, die bereit ist, die Vision des "GAIA OS" Realität werden zu lassen. Die hier dargelegten architektonischen Entscheidungen bilden das Fundament für zukünftige Erweiterungen, sei es die Anbindung an echte Pulsar-Uhren oder die Skalierung der "Nullstellen-Analyse" auf Audio- und Videoformate.

Tabelle 1: Essentielle Konfigurationsparameter für den "Sear App" Proxy

Parameter	Empfohlener Wert	Begründung & Technischer Kontext	Quelle
CURLOPT_TIMEOUT	20 Sekunden	Verhindert cURL Error 28 (Timeout) auf der restriktiven WP.com Infrastruktur. Balanciert Nutzererfahrung und Ressourcenlimits.	[13]
CURLOPT_CONNECTTIMEOUT	10 Sekunden	"Fail Fast"-Prinzip: Wenn der Zielserver (Google/DDG) nicht erreichbar ist, wird die Verbindung schnell abgebrochen, um den PHP-Worker freizugeben.	[15]
X-Frame-Options	SAMEORIGIN (Modifiziert via	Ermöglicht das sichere Einbetten	[23]

	Filter)	der "Arrow"-Findings und der eigenen App, während Clickjacking verhindert wird.	
Template-Speicherort	/wp-content/themes/[child-theme]/	Gewährleistet Persistenz des Codes ("Sovereign Code") auch bei Updates des Eltern-Themes durch den Anbieter.	[6]
Allowlist-Strategie	Strikter Domain-Abgleich	Elementarer Bestandteil von "Defender OHM" zur Verhinderung von SSRF-Angriffen. Nur vertrauenswürdige Endpunkte sind erlaubt.	[1]
libxml_use_internal_errors	true	Verhindert, dass nicht-konformes HTML externer Quellen PHP-Warnungen auslöst und das JSON-Response-Format zerstört.	[20]

Referenzen

1. 02 Itheereum GAIA OS, Technisches Manifest und Repository-Dokumentation.pdf
2. Business Plan Features | WordPress.com Support, Zugriff am November 17, 2025, https://wordpress.com/support/plan-features/business-plan/
3. Use SFTP on WordPress.com, Zugriff am November 17, 2025, https://wordpress.com/support/sftp/
4. Troubleshooting SFTP and SSH – WordPress.com Support, Zugriff am November 17, 2025, https://wordpress.com/support/sftp/troubleshooting-sftp/
5. SFTP Usage and Access - Support Center - WP Engine, Zugriff am November 17, 2025, https://wpengine.com/support/sftp/
6. Create a child theme – WordPress.com Support, Zugriff am November 17, 2025, https://wordpress.com/support/themes/child-themes/
7. How to create a WordPress Child Theme + Find out why you might want to do this - YouTube, Zugriff am November 17, 2025, https://www.youtube.com/watch?v=3oGQL9HSXnE
8. Child Themes – Theme Handbook - WordPress Developer Resources, Zugriff am November 17, 2025, https://developer.wordpress.org/themes/advanced-topics/child-themes/
9. How to Create a Child Theme in WordPress | WP Engine®, Zugriff am November 17, 2025, https://wpengine.com/resources/create-child-theme-wordpress/
10. File Manager – WordPress plugin, Zugriff am November 17, 2025, https://wordpress.org/plugins/wp-file-manager/
11. How to Set Up a File Manager in WordPress [Step-by-Step], Zugriff am November 17, 2025, https://advancedfilemanager.com/set-up-a-file-manager-in-wordpress/
12. Stay away from "WP file manager" : r/Wordpress - Reddit, Zugriff am November 17, 2025, https://www.reddit.com/r/Wordpress/comments/y4tqw9/stay_away_from_wp_file_manager/
13. How to Fix cURL Error 28: Connection Timed Out in WordPress - 6 Effective Solutions, Zugriff am November 17, 2025, https://www.hostinger.com/tutorials/how-to-fix-wordpress-curl-error-28
14. How to Fix cURL Error 28: Connection Timed Out in WordPress - AccuWebHosting, Zugriff am November 17, 2025, https://manage.accuwebhosting.com/knowledgebase/3777/How-to-Fix-cURL-Error-28-Connection-Timed-Out-in-WordPress.html
15. How to Fix cURL Error 28: Connection Timed Out in WordPress - Bluehost,

Zugriff am November 17, 2025, https://www.bluehost.com/blog/how-to-fix-curl-error-28-connection-timed-out-in-wordpress/
16. Website slow and gives an ERR connection error | WordPress.com Forums, Zugriff am November 17, 2025, https://wordpress.com/forums/topic/website-slow-and-gives-an-err-connection-error/
17. Relative/Absolute URL and Proxies - RBleug, Zugriff am November 17, 2025, https://regilero.github.io/english/apache/2009/06/03/relative_absolute_urls_and_proxies/
18. Change a relative URL to absolute URL - Stack Overflow, Zugriff am November 17, 2025, https://stackoverflow.com/questions/3342785/change-a-relative-url-to-absolute-url
19. Replace all relative URLs with absolute URLS - php - Stack Overflow, Zugriff am November 17, 2025, https://stackoverflow.com/questions/48836281/replace-all-relative-urls-with-absolute-urls
20. Way to create an HTTP proxy to convert relative paths to absolute paths - Stack Overflow, Zugriff am November 17, 2025, https://stackoverflow.com/questions/70733798/way-to-create-an-http-proxy-to-convert-relative-paths-to-absolute-paths
21. How to import/include a CSS file using PHP code and not HTML code? - Stack Overflow, Zugriff am November 17, 2025, https://stackoverflow.com/questions/6315772/how-to-import-include-a-css-file-using-php-code-and-not-html-code
22. How to Create a WordPress Child Theme: Complete 2026 Guide - Jetpack, Zugriff am November 17, 2025, https://jetpack.com/resources/wordpress-child-theme/
23. Add, edit, or remove HTTP response headers - WordPress VIP Documentation, Zugriff am November 17, 2025, https://docs.wpvip.com/infrastructure/edge-servers/http-headers/add-edit-or-remove/
24. How to remove X-Frame-Options: SAMEORIGIN" from WordPress?, Zugriff am November 17, 2025, https://wordpress.stackexchange.com/questions/350864/how-to-remove-x-frame-options-sameorigin-from-wordpress
25. How to Configure the X-Frame-Options WordPress Header - MalCare, Zugriff am November 17, 2025, https://www.malcare.com/blog/x-frame-options-wordpress/
26. How to Remove X-Frame-Options SAMEORIGIN from WordPress. - Kevin Dees, Zugriff am November 17, 2025, https://kevdees.com/how-to-remove-x-

frame-options-sameorigin-from-wordpress/