Example Shop — EXTERNAL SECURITY DIAGNOSTIC

Security Posture Assessment

External, publicly-observable security review
WooCommerce / WordPress storefront

Client: Example Shop · **Domain:** example-shop.com

Date: 18 July 2026

Author: Yassin BELHAJ SALAH

E-commerce Solutions Architect

Scope: Security posture observable from the public internet

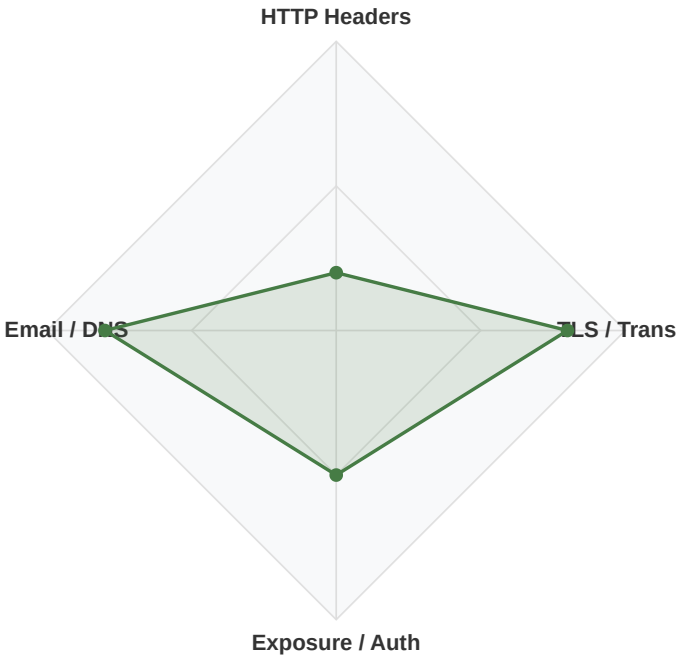
Confidentiality: Internal use — Example Shop

Executive summary

> **How to read this report.** This is a **preliminary external assessment** based only on signals that anyone on the public internet can observe — HTTP response headers, TLS handshakes, cookies, exposed paths, and DNS (email) records. It does **not** include access to your code, your WordPress admin, your server configuration, your web-application firewall rules, or your internal patching. The work is produced with an AI-assisted workflow, with **manual validation of the findings we retained**. Every conclusion that would require internal access is therefore stated as an **observation → hypothesis to validate**, never as a proven internal fact.

Example Shop runs a **WooCommerce / WordPress** storefront serving multiple regions (IP-based geo-redirection detected). The picture that emerges is **two-sided and, importantly, not alarmist**: your **transport and email-authentication foundations are genuinely strong** — among the best of the brands we benchmarked — while your **HTTP security-header layer is essentially empty**, which is the single most impactful and lowest-risk area to fix.

Security radar — indicative scores (/10), derived from measured signals



HTTP Headers **2/10** · TLS/Transport **8/10** · Exposure/Auth **5/10** · Email/DNS **8/10** — scores are a judgement summary of the measured facts detailed below, not an external grade.

MOZILLA OBSERVATORY

F

0/100 · 4 OF 10 TESTS PASSED

BASELINE HEADERS PRESENT

0/6

ON THE TESTED RESPONSES

TLS 1.3

✓

1.0/1.1 DISABLED

DMARC POLICY

reject

STRONGEST ANTI-SPOOFING

The three priorities (measured impact, low regression risk)

#	Priority	Why it is first	Effort
CT-01	Add the baseline security headers (nosniff, X-Frame-Options, Referrer-Policy, Permissions-Policy)	Absent on every tested response; safe, non-breaking; your benchmarked competitors already expose several of these	1 man/day
CT-02	Enable HSTS (max-age only) after a readiness audit	No HTTP-Strict-Transport-Security today; closes the downgrade gap while your TLS is already modern	1 man/day
CT-03	Content-Security-Policy — staged (report-only → enforce)	Highest-value control for a card-taking checkout; directly supports PCI DSS 4.0 (6.4.3 / 11.6.1) against e-skimming	6 man/days

Backlog identified: ≈ 17.5 man/days (12 workstreams) · **Prioritized short/mid-term roadmap: ≈ 15.5 man/days** (Phases 1–2). Prioritization is a strategic judgement, not a measurement.

What is already working — real strengths

A credible security review starts with the controls you have got **right**. These are measured, not assumed, and several of them put you **ahead** of well-known brands in your category.

> **1 — Modern transport (TLS).** Your edge negotiates **TLS 1.3** (cipher `TLS_AES_256_GCM_SHA384`, confirmed on a clean handshake) and **TLS 1.2**, while the deprecated **TLS 1.0 and 1.1 are disabled** (explicit handshakes fail with "no protocols available"). The certificate is a valid Let's Encrypt certificate covering the apex, `www` and `webmail`, and is not near expiry.

2 — Excellent email authentication. Your domain publishes **SPF**, a responding **DKIM** selector (`default`), and — notably — a **DMARC policy at `p=reject` (pct=100)**, the strongest anti-spoofing posture available. In our benchmark this is **stronger than every competitor we measured** (KeepCup was quarantine; Frank Green and Chilly's published none). This protects your customers from phishing that impersonates your domain, and helps your transactional email deliverability.

3 — An active server-side defensive layer. During testing, repeated automated requests were **blocked by Imunify360 bot-protection** (`{"message":"Access denied by Imunify360 bot-protection"}`). This is a genuine defensive layer in front of the application. (It also means some of our probes were edge-filtered, so where we could not reach a definitive answer we say so rather than guess.)

4 — Transport hygiene on the tested pages. No mixed content (no `http://` resources on the HTTPS homepage), and sensitive files probed (`/.git/config`, `/.env`, `/wp-config.php.bak`) were **not served** (HTTP 403) on the tested responses. The observed cookie carried `Secure` and `HttpOnly`.

These strengths are why the roadmap below is about *hardening a fundamentally sound site*, not rescuing a broken one. The gaps are concentrated in one layer — HTTP response headers — which is precisely the cheapest and safest layer to improve.

HTTP security headers — the main gap

HTTP response headers are the instructions your server sends to every visitor's browser telling it how to behave safely. They are public, anyone can read them, and automated scanners grade them. On the tested responses, the baseline set is **absent**.

Header	Role (plain language)	Status on tested responses
Strict-Transport-Security (HSTS)	Forces browsers to only use HTTPS	**Absent**
Content-Security-Policy (CSP)	Controls which scripts/resources may load — the main defence against injected/skimming code	**Absent (not even report-only)**
X-Frame-Options	Blocks clickjacking (your pages framed by a malicious site)	**Absent**
X-Content-Type-Options: nosniff	Stops the browser guessing file types (MIME sniffing)	**Absent**
Referrer-Policy	Limits what URL data leaks to third parties	**Absent**
Permissions-Policy	Disables unneeded browser features (camera, geolocation…)	**Absent**

> **Independent corroboration.** Mozilla Observatory (a public scanner, its own requests) returned **grade F, score 0/100, 4 of 10 tests passed** for example-shop.com on 18 July 2026 — consistent with the missing headers above. Per the 2026 OWASP Secure Headers baseline, these controls are expected on any production HTTPS site; *Permissions-Policy* in particular is the most commonly absent header across the web, which is exactly why adding it is a reliable, low-effort improvement. (Sources: OWASP Secure Headers Project; "Security Headers in 2026", competitor-a.example.)

CT-01 — Add the baseline security headers

Finding: none of the four safe baseline headers (nosniff, X-Frame-Options, Referrer-Policy, Permissions-Policy) are present. **Risk:** clickjacking, MIME-sniffing and referrer leakage remain un-mitigated, and the site scores at the bottom of any public header scan a customer or partner could run. **Solution:** add them once at the web-server layer (nginx, under Plesk). These four are non-breaking and can ship immediately.

```
# nginx (Plesk) — safe, non-breaking baseline. Add in the site's vhost/nginx directives.
add_header X-Content-Type-Options "nosniff" always;
add_header X-Frame-Options "SAMEORIGIN" always;
add_header Referrer-Policy "strict-origin-when-cross-origin" always;
add_header Permissions-Policy "camera=(), microphone=(), geolocation=()" always;
```

Profile: DevOps · **Seniority:** Confirmed · **Effort:** 1 man/day · **Phase 1**

Expected gain: closes four of the Observatory failing controls and removes the clickjacking / MIME-sniffing surface. Re-measure Observatory after deployment (we do not project a post-fix grade — it is measured once live).

CT-02 — Enable HSTS (max-age only), after a readiness audit

Finding: no `Strict-Transport-Security` header, so browsers are not instructed to refuse plain HTTP. **Risk:** a first insecure request can be intercepted/downgraded. **Solution:** start with `max-age` only. **Do not add `includeSubDomains` or `preload` until an HSTS readiness audit** confirms every subdomain (you have at least `webmail.example-shop.com`) is HTTPS-ready — those flags are near-irreversible and can break subdomains still served over HTTP.

```
# Step 1 now: max-age only. includeSubDomains/preload ONLY after the readiness audit.
add_header Strict-Transport-Security "max-age=31536000" always;
```

Profile: DevOps · **Seniority:** Confirmed · **Effort:** 1 man/day (incl. readiness audit) · **Phase 1**

Expected gain: removes the protocol-downgrade window on repeat visits; prerequisite to a clean transport posture.

CT-04 — Reduce technology & version disclosure

Finding: responses advertise `X-Powered-By: PHP/8.2.32` and `X-Powered-By: PleskLin`, and the HTML exposes `<meta name="generator">` for WordPress and All-in-One-SEO (4.9.5.1). **Risk:** this fingerprints your exact stack, letting an attacker match published CVEs to your precise versions

before touching the site. **Solution:** suppress the `X-Powered-By` headers and the generator meta (native/free).

```
# nginx: stop advertising the backend
proxy_hide_header X-Powered-By;   fastcgi_hide_header X-Powered-By;
# PHP: expose_php = Off (php.ini)   |   WordPress:
remove_action('wp_head','wp_generator');
```

Profile: DevOps · **Seniority:** Confirmed · **Effort:** 0.5 man/day · **Phase 1**
Expected gain: raises the cost of targeted reconnaissance; standard hardening hygiene.

Content-Security-Policy & payment-page integrity

Your storefront takes card payments and loads scripts from **several third parties** — we detected origins for Klaviyo, Facebook, Instagram, Google Analytics and Jetpack in the page (detected — active use not confirmed). A Content-Security-Policy is the browser-level control that decides which of those scripts are allowed to run. Today there is **no CSP at all** (neither enforced nor report-only) on the tested responses.

Why this matters for a checkout: e-skimming (Magecart). The dominant attack against online stores injects malicious JavaScript that quietly copies card data at the payment step. A well-scoped CSP is one of the few controls that meaningfully constrains this. **Honesty note:** we tested the homepage, not the payment page — a CSP absent on the tested responses is *not* proof that anti-skimming defence is off at checkout; the checkout may carry its own policy. **Checkout-specific enforcement must be verified before concluding on payment-page protection.**

Current compliance context (sourced). PCI DSS 4.0 **Requirement 6.4.3** (inventory, authorize and integrity-check every script on the payment page) and **Requirement 11.6.1** (detect and alert on unauthorized changes to payment-page scripts *and* HTTP security headers) have been **mandatory since 31 March 2025**, created specifically to counter e-skimming. These map almost one-to-one onto CSP + header integrity. (Sources: PCI SSC, "Payment Page Security and Preventing E-Skimming", March 2025; Foregenix; cside.)

CT-03 — Deploy CSP in stages (report-only → review → enforce)

Finding: no CSP present. **Risk:** if any third-party script is compromised, nothing at the browser level constrains what it can do — the classic skimming path. **Solution:** the safe, industry-standard rollout is to ship CSP **in report-only first** (it breaks nothing, only reports), review the reports to build the real allow-list of your legitimate third parties, then switch to enforcement — prioritising the checkout. Avoid `'unsafe-inline'` where possible; a report-only phase is how you get there without breaking the store.

```
# Phase A — observe only (does NOT block anything), collect violation reports:
add_header Content-Security-Policy-Report-Only "default-src 'self'; report-uri
/csp-report" always;
# Phase B — after reviewing reports and allow-listing your real third parties
(Klaviyo, GA, Meta...),
# switch the header to Content-Security-Policy (enforced), starting with the
checkout template.
```

Profile: Backend DevOps · **Seniority:** Senior · **Effort:** 6 man/days (inventory + report-only + review + staged enforcement) · **Phase 2**
Expected gain: introduces browser-level control over third-party scripts on the payment path and directly advances PCI DSS 4.0 (6.4.3 / 11.6.1). Effectiveness is validated by the violation reports and post-enforcement testing.

CT-11 — Third-party script governance & PCI DSS 4.0 alignment

Finding: multiple third-party marketing/analytics scripts are detected in the storefront; there is no externally-visible evidence of a script inventory or tamper-detection on the payment page. **Risk:** each third party is both an attack surface and a compliance obligation under 6.4.3 / 11.6.1. **Solution:** build and maintain a **script inventory** (what loads on the checkout, from where, why), add change/tamper detection on the payment page, and use that inventory to scope the CSP allow-list from CT-03. This is a governance workstream, not a single toggle.

> **Scope note.** Subresource Integrity (SRI) is often proposed here, but dynamic tags (GTM, Meta, Klaviyo) do not have stable hashes — treat SRI as part of *third-party governance* (inventory + policy), not a blanket "all third-party code is integrity-checked" promise.

Profile: Backend Security · **Seniority:** Senior · **Effort:** 3 man/days · **Phase 2**
Expected gain: payment-page changes become inventoried and monitored, moving the checkout toward PCI DSS 4.0 6.4.3 / 11.6.1 — to be confirmed against your acquirer's assessment scope.

Transport (TLS) & certificate — strong, keep it that way

Check	Result (measured)	Verdict
TLS 1.3	Negotiated (`TLS_AES_256_GCM_SHA384`)	**Modern ✓**
TLS 1.2	Negotiated	**OK ✓**
TLS 1.0 / 1.1	Handshake fails ("no protocols available")	**Disabled ✓**
Certificate	Let's Encrypt; SAN: example-shop.com, www, webmail; expires 29 Sep 2026	**Valid ✓**
Mixed content (homepage)	0 `http://` resources	**Clean ✓**

> **This is a genuine strength.** There is no TLS finding to fix here. The one thing to keep in mind is that the certificate covers `webmail.example-shop.com` — evidence a mail/web subdomain exists — which is exactly why HSTS `includeSubDomains`/`preload` (CT-02) must wait for a readiness audit rather than being switched on blind.

Certificate renewal is expected to be automatic (Let's Encrypt). Worth a one-line confirmation that auto-renewal is active so the certificate never lapses during a high-traffic period.

Cookies, admin exposure & application surfaces

CT-05 — Harden cookies (SameSite) and verify the WooCommerce session cookies

Finding: the one cookie set on the homepage (`ip2location_redirection_first_visit`, a geo-redirection cookie) carries `Secure` and `HttpOnly` but **no `SameSite`** attribute. The WooCommerce **session / cart / login cookies were not observable** on the tested response, so their flags are **not externally verifiable**. **Risk:** a session cookie without `SameSite`/`HttpOnly` widens CSRF and theft surface. **Solution:** set `SameSite=Lax` on the geo cookie, and audit the cart/checkout/login cookies to confirm `Secure` + `HttpOnly` + `SameSite`.

Profile: Backend · **Seniority:** Confirmed · **Effort:** 1 man/day · **Phase 1** · **Gain:** narrows CSRF / session-theft surface on the authenticated and checkout paths.

CT-06 — Disable `xmlrpc.php` if unused

Finding: `xmlrpc.php` is **enabled** — the first clean probe returned the native WordPress response "XML-RPC server accepts POST requests only" (HTTP 405). **Risk:** XML-RPC is a classic vector for brute-force amplification and pingback abuse. Imunify360 sits in front and may throttle abuse, but disabling removes the surface entirely. **Solution:** if you do not use XML-RPC (most modern stores do not), block it at the server.

```
# nginx: return 403 for xmlrpc unless a specific integration needs it
location = /xmlrpc.php { deny all; return 403; }
```

Profile: DevOps · **Seniority:** Confirmed · **Effort:** 0.5 man/day · **Phase 1** · **Gain:** removes a well-known brute-force / amplification surface.

CT-07 — Validate & restrict REST user enumeration

Finding — inconclusive, to validate. On first contact, `/wp-json/wp/v2/users` returned HTTP 200 with a user-shaped JSON signature; every reproduction attempt was then **blocked by Imunify360**, so we **could not confirm** whether author slugs are disclosed to an unauthenticated visitor. **Risk (if open):** leaked login names feed brute-force. **Solution:** validate `/wp-json/wp/v2/users` and `/?author=1` from a clean browser/IP; if they disclose usernames, restrict the REST users endpoint for anonymous callers (native filter, free).

Profile: Backend · **Seniority:** Confirmed · **Effort:** 0.5 man/day · **Phase 1** · **Gain:** removes login-name leakage if confirmed; otherwise closes the question with a definitive check.

CT-10 — Login hardening (rate-limit + 2FA for admins)

Finding: the default admin path was **not exposed** on the tested responses (`wp-login.php` → 404, `/wp-admin` → 301 → 404). This proves only "not exposed at the default path" — the actual administrative path and hardening are **not externally verifiable** (we do not infer that it was renamed). Login rate-limiting and 2FA are likewise not observable from outside. **Solution:** confirm and, if missing, enable login attempt-limiting and two-factor authentication for administrator accounts (native / free options exist; no paid plugin required).

Profile: Backend · **Seniority:** Confirmed · **Effort:** 1 man/day · **Phase 2** · **Gain:** reduces account-takeover risk on the admin surface.

Email authentication (SPF / DKIM / DMARC) — a highlight

This is a distinct security domain from HTTP headers, and it is where Example Shop is **strongest**.

Record	Value (measured)	Verdict
SPF	`v=spf1 +a include:_spf-us.competitor-b.example ~all`	**Present (softfail `~all`)**
DKIM	`default` selector responds (v=DKIM1)	**Confirmed present**
DMARC	`v=DMARC1; p=reject; pct=100;`	**Strongest policy ✓**
Aggregate reporting (rua)	Not present	**No visibility**

> **DMARC at `p=reject` is excellent** and rare — it actively rejects mail that spoofs your domain, protecting your customers from phishing. It also puts you ahead of the competitors we measured (quarantine / none). **Method note on DKIM:** DKIM selectors are not enumerable, so we confirm the `default` selector is present but cannot claim the configuration is complete — confirm via an outbound email header or your IONOS admin console.

CT-08 — Add DMARC aggregate reporting (rua) & document SPF/DKIM

Finding: `p=reject` is in place, but with no `rua` address you receive no aggregate reports — you are enforcing blind. **Risk:** no visibility into spoofing attempts, or into legitimate senders that might be failing. **Solution:** keep `p=reject`, add a `rua` mailbox to collect reports, and document the authorised senders. (SPF is `~all`; with `p=reject` already enforcing, tightening SPF is optional and low-priority.)

```
# DNS TXT _dmarc.example-shop.com — keep p=reject, add reporting
v=DMARC1; p=reject; pct=100; rua=mailto:dmarc-reports@example-shop.com; fo=1
```

Profile: DevOps · **Seniority:** Confirmed · **Effort:** 0.5 man/day · **Phase 1** · **Gain:** turns your already-strong DMARC into a monitored control with visibility into abuse and deliverability.

Additional surfaces & hygiene

CT-09 — Publish a `security.txt`

Finding: no `/.well-known/security.txt` was found. **Risk:** a security researcher who spots an issue has no clear, official channel to report it responsibly — increasing the chance it goes public first. **Solution:** publish a small static `security.txt` (a recognised standard, free).

```
# https://example-shop.com/.well-known/security.txt
Contact: mailto:security@example-shop.com
Expires: 2027-07-18T00:00:00Z
Preferred-Languages: en
```

Profile: DevOps · **Seniority:** Junior · **Effort:** 0.5 man/day · **Phase 1** · **Gain:** a documented responsible-disclosure channel; basic security hygiene expected of mature brands.

CT-12 — (prospective) Evaluate cross-origin isolation & reporting endpoints

Context (prospective — hierarchy is a strategic judgement, not a measurement): the 2026 header baseline also discusses the cross-origin trio (COOP / COEP / CORP) and reporting endpoints. **These are advanced hardening to test, not a quick win**: on a storefront embedding Facebook, Instagram and Klaviyo, cross-origin isolation can break third-party embeds, so the benefit/regression trade-off must be evaluated in a staging environment first. **This deliberately sits below the measured priorities.**

Profile: DevOps · **Seniority:** Senior · **Effort:** 2 man/days (evaluation) · **Phase 3 / backlog** · **Gain:** potential defence-in-depth if it can be enabled without breaking embeds — to be proven in staging.

Competitive benchmark — where you stand

We measured three well-known reusable-cup brands with the **same method used on your site** (Mozilla Observatory, `curl` headers, `openssl` for TLS, `dig` for DMARC) so the comparison is homogeneous. KeepCup, Frank Green and Chilly's are the market references a buyer already recognises.

A — Measured comparison

Axis (same method both sides)	Example Shop	KeepCup	Frank Green	Chilly's
Mozilla Observatory grade	F (0/100)	F (0/100)	F (0/100)	F (0/100)
HSTS present	No	Yes	Yes	Yes
CSP present	No	Yes	Yes	Yes
X-Frame-Options present	No	Yes	Yes	Yes

Referrer-Policy present	No	No	No	No
Permissions-Policy present	No	No	No	No
TLS 1.3	Yes	Yes	Yes	Yes
DMARC policy	reject	quarantine	none	none

Read the grade carefully. Every brand here — including the leaders — scores Observatory **F**, because the 2026 scoring is strict. So the differentiation is *not* the grade; it is **which specific controls each site exposes**. That is where the real, honest gap sits.

B — Capability gaps & leads (observed both sides)

Capability the market treats as standard	You	Reference brands	What it means
HSTS header exposed	Behind	KeepCup, Frank Green, Chilly's — all yes	They instruct browsers to stay on HTTPS; you do not (yet)
CSP header exposed	Behind	All three — yes	They expose a script-control policy; you do not
X-Frame-Options exposed	Behind	All three — yes	They ship clickjacking protection; you do not
DMARC at reject	Ahead	Only KeepCup enforces (quarantine); the others none	You already lead on anti-spoofing — a real advantage
Referrer-Policy / Permissions-Policy	Open field	None of them ship these	Adding them lets you *lead* the category, not just catch up

The honest bottom line. On the header controls the market now treats as table stakes — **HSTS, CSP and X-Frame-Options — your site is measurably behind three brands your customers already know**, all of which expose these and you expose none. At the same time you are **ahead on email authentication**. So this is not a rescue; it is closing a visible, checkable gap on the header layer (Phase 1 does most of it in days) while keeping the lead you already hold — and, with Referrer-Policy/Permissions-Policy, moving from "catching up" to "setting the standard" in your category.

Business timing & seasonality

We are mid-July 2026. For a direct-to-consumer eco brand selling reusable cups, the demand calendar ahead is clear: **back-to-school (August–September)** as sustainable everyday kit, then the big

commercial peak — **Black Friday (27 November 2026) and Cyber Monday (30 November 2026)** — followed by **Q4 holiday gifting (December)**, where reusable cups are a natural eco-gift, and the **New-Year sustainability resolutions** in January. Earth Day (22 April) has already passed for this year.

Now → Weeks 1–2
Header quick wins (Phase 1)

Weeks 3–6
CSP + PCI + login (Phase 2)

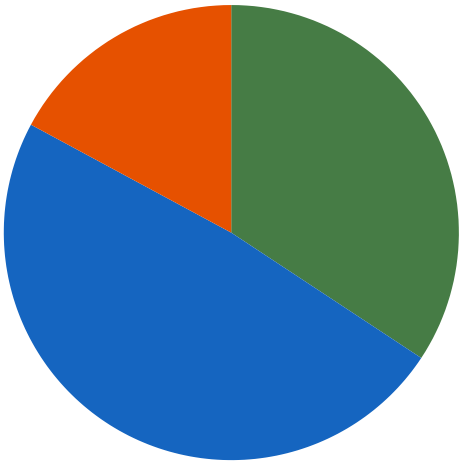
Before Black Friday
Hardened checkout for peak

Why now. Card-transaction volume, checkout traffic and attacker interest in payment pages all rise during high-traffic retail windows — the PCI SSC's own e-skimming guidance exists precisely because payment pages are the target. You have roughly **four months of runway** to bring the header layer and the payment path (CT-01→CT-03, CT-11) to a hardened, PCI-aligned state *before* the peak, rather than changing security-sensitive configuration in the middle of your busiest weeks.

How to quantify the value (with your data, not ours). We do not invent business percentages. The value of hardening is expressed as: *(1)* measured control coverage (Observatory tests closed, headers added); and *(2)* a client-variable formula for risk-reduction on revenue — **protected checkout value = checkout traffic × conversion × average order value**, which your Analytics can populate in a 30-minute scoping workshop. A single successful skimming incident on a payment page during peak is the downside this spend insures against.

Effort distribution by profile

The full backlog is **≈ 17.5 man/days** across 12 workstreams. It leans toward application/backend work (the staged CSP), with a solid block of DevOps (headers, TLS/DNS, server config) and a governance slice for PCI alignment.



DevOps — infra, headers, TLS/DNS — 6.0 man/days

Backend / app (WordPress · WooCommerce) — 8.5 man/days

Security governance (PCI · third-party) — 3.0 man/days

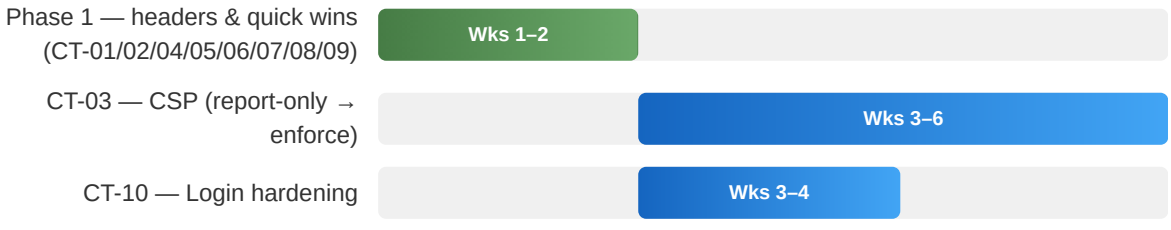
DevOps 34% · Backend/app 49% · Security governance 17% — total 17.5 man/days.

Action plan & roadmap

#	Workstream	Profile	Effort	Phase
CT-01	Baseline security headers (nosniff, X-Frame-Options, Referrer-Policy, Permissions-Policy)	DevOps	1	1
CT-02	HSTS (max-age only) + readiness audit	DevOps	1	1
CT-04	Reduce technology/version disclosure (X-Powered-By, generator)	DevOps	0.5	1
CT-05	Harden cookies (SameSite) + verify Woo session cookies	Backend	1	1
CT-06	Disable xmlrpc.php if unused	DevOps	0.5	1
CT-07	Validate & restrict REST user enumeration	Backend	0.5	1
CT-08	DMARC aggregate reporting (rua) + document SPF/DKIM	DevOps	0.5	1
CT-09	Publish security.txt	DevOps	0.5	1
CT-03	CSP staged (report-only → enforce), PCI 6.4.3/11.6.1	Backend+DevOps	6	2
CT-10	Login hardening (rate-limit + 2FA for admins)	Backend	1	2
CT-11	Third-party script governance + PCI alignment	Backend+Security	3	2
CT-12	(prospective) Evaluate cross-origin isolation (COOP/CORP)	DevOps	2	3

≈ 17.5 man/days identified (full backlog) · **≈ 15.5 man/days prioritized (Phases 1–2)**. CT-12 stays in the backlog as prospective/advanced hardening to test — never ahead of the measured priorities.

Schedule (indicative, ~6 weeks to a hardened baseline)



Suggested next step. A 30-minute scoping workshop to (1) validate the two inconclusive items from a clean IP (REST user enumeration, checkout-page CSP), (2) confirm the HSTS subdomain readiness, and (3) populate the risk-reduction formula with your Analytics. Phase 1 can then ship within days and be re-measured on Observatory.

External preliminary assessment — publicly observable signals only, no internal access. AI-assisted workflow with manual validation of retained findings. Observations are hypotheses to validate, not proven internal facts; no post-remediation score is projected.