metlabs

AUDITORÍA DE SEGURIDAD — APPWALLET

Septiembre 2026 | Versión 2.0 — Cierre de remediación

Preparado por Metlabs para Latin Link

Índice

INFORME TÉCNICO · ANÁLISIS ESTÁTICO · CIERRE DE REMEDIACIÓN

1. Resumen ejecutivo

appWallet es una wallet non-custodial con DEX (agregador de swaps) construida sobre Next.js 16.2.6 / React 19, con Reown AppKit 1.8.20 + wagmi 2.19 + viem 2.52 para la conexión de wallet, y OpenOcean como agregador de swaps. El backend son 18 route handlers de Next que actúan como proxy hacia APIs de terceros (OpenOcean, Zerion, CoinMarketCap, CoinGecko).

Este documento es la versión de cierre de la auditoría de seguridad realizada por Metlabs el 6 de julio de 2026 (versión 1.0). Recoge íntegramente los hallazgos originales y documenta, para cada uno de ellos, la corrección aplicada y su estado final. A fecha de emisión de esta versión, la totalidad de los 17 hallazgos detectados (A–Q) han sido corregidos.

1.1 Alcance

Campo	Detalle
Proyecto	appWallet — wallet non-custodial + DEX (agregador de swaps)
Repositorio	rama main (commit auditado: 05a466b)
Fecha de la auditoría inicial	6 de julio de 2026
Fecha de cierre de remediación	Septiembre de 2026
Auditor	Metlabs — análisis estático del código fuente
Alcance	Código fuente (src/, ~31.000 LoC, 215 archivos) + árbol de dependencias
Fuera de alcance	Runtime desplegado, infraestructura, contratos on-chain de terceros (OpenOcean/Zerion), pentest dinámico
Estado final	17/17 hallazgos corregidos (4 Alto, 4 Medio, 9 Bajo/Info)

Tabla 1. Ficha de la auditoría.

1.2 Veredicto general

El diseño non-custodial está correctamente implementado: la aplicación nunca posee claves privadas ni frases semilla, todas las claves de API son server-side, y no existen sumideros de XSS. En la auditoría inicial no se encontraron vulnerabilidades que permitieran a un atacante remoto tomar el control ni exfiltrar secretos.

El riesgo real detectado en julio de 2026 se concentraba en tres áreas, hoy resueltas:

- Seguridad del flujo de swap — falta de fijación de chainId antes de firmar (riesgo de pérdida de fondos por ejecución cross-chain). Corregido.
- Rutas API sin protección de abuso — proxies públicos y sin límite de tasa hacia APIs de pago (DoS financiero). Corregido.
- Dependencias transitivas vulnerables — mayoritariamente heredadas del stack de wallet. Corregido.

Tras la aplicación de las medidas de remediación descritas en la sección 3, no queda ningún hallazgo abierto. El nivel de riesgo residual conocido de appWallet, dentro del alcance de esta auditoría estática, es el correspondiente a una aplicación sin hallazgos pendientes.

1.3 Cuadro de severidad y estado

Severidad	Nº	Hallazgos	Estado
Crítico / Alto	4	A (cross-chain), B (proxy sin rate-limit), C (Zerion sin validar), D (dependencias)	4/4 corregidos
Medio	4	E (comisión manipulable), F (allowlist router), G (cabeceras), H (fuga de errores)	4/4 corregidos
Bajo / Info	9	I–Q (ver detalle)	9/9 corregidos

Tabla 2. Distribución de hallazgos por severidad y estado de remediación.

1.4 Registro de remediación por hallazgo

ID	Sev	Hallazgo	Estado
A	Alto	Ejecución de swap cross-chain — chainId no fijado antes de firmar	Corregido
B	Alto	Proxies públicos a APIs de pago sin rate-limiting	Corregido
C	Alto	Rutas Zerion con parámetros sin validar ni codificar	Corregido
D	Alto	Dependencias vulnerables (4 high, 11 moderate, 1 low)	Corregido
E	Medio	Comisión de referral manipulable desde el cliente	Corregido
F	Medio	to/callData de OpenOcean sin allowlist de router	Corregido
G	Medio	Ausencia de cabeceras de seguridad	Corregido
H	Medio/Bajo	Reenvío de cuerpos de error del upstream al cliente	Corregido
I	Bajo	validUntil fabricado y no aplicado	Corregido
J	Bajo	Precompilado nativo de Polygon no normalizado	Corregido
K	Bajo	toWei sin control de errores / «Max» con aritmética float	Corregido
L	Bajo	Historial de swaps en localStorage	Corregido

M	Bajo	Log de address + calldata en consola y POST a ruta dev	Corregido
N	Info	cookieStorage de wagmi no HttpOnly	Corregido
O	Info	Código muerto (fetchOneInchQuote / ONEINCH_API_KEY)	Corregido
P	Info	URLs placeholder / http:// en listas de tokens	Corregido
Q	Info	console.error en boundary de cliente	Corregido

Tabla 3. Estado de remediación de la totalidad de los hallazgos.

2. Metodología

Auditoría estática por dimensiones, con verificación adversarial de cada hallazgo (lectura directa del código citado, no solo coincidencias de grep). Dimensiones cubiertas:

- Secretos y exposición de claves de API (cliente vs. servidor).
- Rutas API: SSRF, inyección, validación de entrada, autorización, abuso de cuota.
- Manejo de claves de wallet / firma / almacenamiento en cliente.
- Flujo de ejecución de swap y lógica financiera/de negocio.
- XSS y renderizado peligroso.
- Open redirect, CSRF, i18n, SSR/RSC, prototype pollution, ReDoS.
- Cabeceras de seguridad y scripts de terceros.
- Dependencias (npm/pnpm audit) y supply-chain (scripts de instalación).
- Manejo de .env y secretos en el repositorio.

2.1 Verificación de la remediación

Para el cierre de la auditoría se ha revisado la corrección de cada hallazgo contra la remediación prescrita en la versión 1.0 del informe: ubicación afectada, medida aplicada y ausencia del patrón vulnerable en el código resultante. El estado «Corregido» que figura en este documento indica que la medida recomendada ha sido implementada y que el hallazgo original ya no es reproducible en el código fuente.

3. Hallazgos detallados y corrección aplicada

Cada hallazgo se presenta con su descripción original (julio 2026), la remediación prescrita y la corrección finalmente aplicada. Todos los hallazgos de esta sección figuran en estado Corregido.

3.1 [A] Alto (potencial crítico): ejecución de swap cross-chain — chainId no se fija antes de firmar

Estado: CORREGIDO. Hallazgo original confirmado por lectura directa.

Ubicación original:

- src/hooks/useSwapFlow.ts:431-437 (sendTransaction sin chainId)
- src/hooks/useSwapFlow.ts:371-377 (writeContract approve sin chainId)
- src/hooks/useSwapFlow.ts:196-204 (useReadContract allowance sin chainId)
- src/components/pages/SwapInterface.tsx:469-475 (canSwap sin guard de cadena)
- src/components/pages/SwapInterface.tsx:247-253 + shouldAutoSwitchChain (auto-switch best-effort)

Descripción: en páginas de detalle de token (/tokens/[chain]/[symbol]), la cadena activa del swap se desacoplaba deliberadamente de la cadena a la que está conectada la wallet, para poder cotizar en la red del token aunque la wallet esté en otra. El auto-switch era best-effort (no se reintentaba si el usuario rechazaba el cambio de red), canSwap no comprobaba que la cadena de la wallet coincidiera con la del quote, y sendTransaction / writeContract no pasaban chainId, por lo que wagmi ejecutaba la transacción en la cadena actual de la wallet, fuera cual fuera.

3.1.1 Escenario de pérdida (original)

Usuario en /tokens/flare/flr con la wallet en Ethereum; rechaza (o ignora) el cambio de red y pulsa «Intercambiar»: el callData/to/value construidos para Flare se ejecutan sobre Ethereum. En un swap con token nativo, el value se envía a quote.to; si esa dirección no es el contrato esperado en Ethereum, los fondos nativos pueden perderse de forma irreversible. En el mejor caso revierte y solo se pierde el gas. El approve también corría en la cadena equivocada.

3.1.2 Corrección aplicada

- sendTransaction, writeContract y useReadContract reciben chainId: quoteData.chainId junto a to, data y value, de modo que wagmi lanza ChainMismatchError en lugar de ejecutar en la cadena incorrecta.
- canSwap incorpora el guard de cadena (chainId === quoteData.chainId); cuando la wallet está en una red distinta a la del quote, el botón de swap se bloquea y se muestra el CTA «Cambia a <red>».
- El flujo de auto-switch deja de ser la única barrera: la fijación explícita de chainId elimina la posibilidad de ejecución cross-chain aunque el usuario rechace el cambio de red.

3.2 [B] Alto: proxies públicos a APIs de pago sin autenticación ni rate-limiting (DoS financiero)

Estado: CORREGIDO. Ubicación original: las 18 rutas de src/app/api/**. No existía middleware de rate-limit ni librería de throttling.

Descripción: ninguna de las 18 rutas exigía autenticación ni aplicaba límite de tasa, y varias hacían de proxy a APIs metered de pago con claves del servidor: swap/quote (cache: 'no-store', cada petición facturada a OpenOcean, body sin validar); cmc/fear-greed-history (limit 1–500 variaba la URL upstream → 500 buckets de caché facturados); coingecko/markets, coingecko/chart y coingecko/coin/[id] (ids/days/id variables → cache-busting). La caché solo deduplicaba peticiones idénticas; los parámetros de cardinalidad libre la eludían, permitiendo agotar la cuota de las APIs de pago (coste económico / caída del servicio).

3.2.1 Corrección aplicada

- Rate-limiting por IP delante de todos los proxies de API, con prioridad en swap/quote y en las rutas con parámetros de cardinalidad libre.
- Cardinalidad de ids, limit y days acotada a valores conocidos, de forma que la caché (next: { revalidate }) vuelve a ser efectiva y no puede eludirse variando parámetros.
- Validación del body de swap/quote antes de la llamada upstream.

3.3 [C] Alto: rutas Zerion con parámetros sin validar ni codificar

Estado: CORREGIDO. Hallazgo original confirmado (dos revisiones independientes + lectura directa).

Ubicación original:

- src/app/api/zerion/[address]/portfolio/route.ts:22-23
- src/app/api/zerion/[address]/positions/route.ts:22-23
- src/app/api/zerion/[address]/transactions/route.ts:22-23
- src/app/api/zerion/[address]/charts/[period]/route.ts:22-23 (address y period)

Descripción: address (y period) se interpolaban crudos en la URL de fetch hacia ZERION_BASE/wallets/{address}/portfolio, sin encodeURIComponent, sin comprobación de formato EVM ni allowlist. Consecuencias: abuso de API de pago (enumerar direcciones distintas = llamadas facturadas ilimitadas) e inyección de parámetros / request-splitting (con %23/%3F/%26 codificados se podían añadir u overridear query params o alcanzar rutas no previstas de api.zerion.io con las credenciales del servidor). Radio de impacto acotado: el host era un prefijo estático, por lo que no constituía SSRF completo, y la clave no se filtraba.

3.3.1 Corrección aplicada

- address validado con la expresión ^0x[a-fA-F0-9]{40}$ (equivalente a isAddress de viem) antes de cualquier interpolación en las 4 rutas.
- period validado contra allowlist de valores admitidos en la ruta charts/[period].
- Aplicación de encodeURIComponent en la construcción de la URL upstream, alineando estas rutas con el patrón ya correcto de oo-tokens/[chain], coingecko/coin/[id] y prices/[pair].

3.4 [D] Alto: dependencias vulnerables (4 high, 11 moderate, 1 low)

Estado: CORREGIDO. Hallazgo original confirmado con pnpm audit --prod: 16 vulnerabilidades, casi todas transitivas del stack de wallet (@reown/appkit → wagmi → porto → hono/ws).

Sev	Paquete	Vulnerabilidad	Fix requerido	Estado
High	ws <8.21 / <7.5.11	Memory-exhaustion DoS	≥8.21	Corregido
High	form-data <4.0.6	CRLF injection	≥4.0.6	Corregido
High	hono <4.12.25	CORS refleja cualquier Origin con	≥4.12.25	Corregido

		credenciales		
Mod	postcss <8.5.10	XSS en stringify (build-time)	≥8.5.10	Corregido
Mod	ws <8.20.1	Divulgación de memoria no inicializada	≥8.20.1	Corregido
Mod	uuid <11.1.1	Falta de comprobación de límites de buffer	≥11.1.1	Corregido
Mod ×8	hono <4.12.25	Path traversal, cookie/JWT/body-limit (adaptadores AWS no usados aquí)	≥4.12.25	Corregido
Low	@babel/core ≤7.29.0	Lectura arbitraria de archivo vía sourceMappingURL	≥7.29.1	Corregido

Tabla 4. Vulnerabilidades reportadas por pnpm audit --prod en la auditoría inicial y su estado.

Nota de aplicabilidad (original): la mayoría de las de hono correspondían a features de adaptadores (AWS Lambda, serve-static) que este proyecto no usa, por lo que su impacto real era bajo. ws (DoS) y form-data (CRLF) eran las de mayor relevancia práctica.

3.4.1 Corrección aplicada

- Actualización del subárbol de @reown/appkit / wagmi.
- Uso de pnpm.overrides en package.json para forzar ws, form-data, hono y postcss a las versiones parcheadas indicadas en la Tabla 4 en aquellas transitivas no resueltas por la actualización.

3.5 [E] Medio: comisión de referral manipulable desde el cliente

Estado: CORREGIDO. Ubicación original: src/app/api/swap/quote/route.ts:52,59-70,99-110; src/lib/chainFeeEngine.ts:18-49; src/lib/feeEngine.ts:79-156.

Descripción: el referrerFee se derivaba de valores enviados por el cliente en el body (amountUSD, fromSymbol, toSymbol, fromAmountHuman). Dos manipulaciones: (1) gaming del tramo por amountUSD — selectTier elegía la tasa según el tamaño en USD declarado, de modo que un swap real de ~$50 podía enviar amountUSD: 100000 y pagar 0.01% en vez de 0.4%; (2) desacople símbolo↔dirección — la categoría de fee usaba fromSymbol/toSymbol mientras el swap real usaba las direcciones fromToken/toToken, permitiendo declarar fromSymbol:"USDC" para obtener la tarifa de stablecoin. Impacto: solo ingresos de la plataforma (no fondos del usuario).

3.5.1 Corrección aplicada

- amountUSD se calcula en el servidor a partir de amount/decimales × precio de confianza; el valor enviado por el cliente ya no interviene en la selección de tramo.
- La categoría de fee se deriva de las direcciones de token resueltas (fromToken/toToken), nunca de los símbolos enviados por el cliente.

3.6 [F] Medio: confianza en to/callData de OpenOcean sin allowlist de router (defensa en profundidad)

Estado: CORREGIDO. Ubicación original: src/hooks/useSwapFlow.ts:431-437,371-377; src/lib/api.ts:41.

Descripción: el destino (to), el callData y el value provenían de la respuesta de OpenOcean y se usaban verbatim, sin allowlist propia de direcciones de router por cadena; el minOutAmount devuelto por la ruta estaba documentado como «diagnostic only» y no se verificaba en el cliente. No era «confianza ciega»: existían salvaguardas reales (slippage protegido on-chain, approval de importe exacto, re-cotización al enviar, fetch server-side por HTTPS y confirmación en la wallet). El riesgo residual era que, ante una respuesta de OpenOcean comprometida, un to+callData maliciosos pudieran drenar hasta inAmount.

3.6.1 Corrección aplicada

- Allowlist por cadena de las direcciones de router de OpenOcean; se afirma quote.to ∈ allowlist antes de aprobar y enviar.
- Verificación en cliente de minOut derivado de toAmount × (1 − slippage) como capa adicional de defensa.

3.7 [G] Medio: ausencia de cabeceras de seguridad

Estado: CORREGIDO. Ubicación original: next.config.ts:1-18 (solo definía images.remotePatterns; sin async headers()).

Descripción: no se definían CSP, HSTS, X-Frame-Options, X-Content-Type-Options, Referrer-Policy ni Permissions-Policy. Sin CSP no había defensa en profundidad frente a XSS (relevante por la inyección de Google Tag Manager en layout.tsx:42-53) y la app era framable (clickjacking sobre el flujo de swap).

3.7.1 Corrección aplicada

- Definición de async headers() en next.config.ts con CSP, HSTS, X-Frame-Options, X-Content-Type-Options, Referrer-Policy y Permissions-Policy.
- CSP elaborada contemplando los orígenes de wallet / WalletConnect / GTM / analytics para no romper la conexión de wallet ni la analítica.

3.8 [H] Medio/Bajo: reenvío de cuerpos de error del upstream al cliente

Estado: CORREGIDO. Ubicación original: cmc/fear-greed-history/route.ts:36, cmc/market-history/route.ts:33, las 4 rutas Zerion (return NextResponse.json(body, { status: res.status })) y swap/quote/route.ts:118-124.

Descripción: estas rutas devolvían el body/status del upstream tal cual, filtrando códigos y mensajes internos del proveedor (CMC/Zerion/OpenOcean). No se filtraba ninguna clave, pero sí detalle interno del proveedor. Las rutas transformadas (cmc/market, altcoin-season, coingecko parseadas) sí normalizaban.

3.8.1 Corrección aplicada

- Todas las respuestas de error de las rutas afectadas se normalizan a mensajes genéricos, replicando el patrón ya correcto de las rutas transformadas.

3.9 Hallazgos Bajo / Info

ID	Sev	Hallazgo original	Ubicación	Corrección aplicada	Estado
I	Bajo	validUntil fabricado (Date.now()/1000 + 300) y no aplicado; no correspondía a ningún deadline on-chain	swap/quote/route.ts:170; SwapInterface.tsx:836-839	validUntil eliminado / alineado con la re-cotización real (getFreshQuote)	Corregido
J	Bajo	Precompilado nativo de Polygon (0x…1010) no normalizado al sentinel 0xEeee… de OpenOcean → posible mispricing en POL nativo	ooNativeAddress.ts:15-21; constants.ts:340	Normalización del precompilado al sentinel de OpenOcean	Corregido
K	Bajo	toWei tragaba errores de parseo y reenviaba el string humano como WEI; «Max» usaba aritmética	useSwapQuote.ts:31-37; SwapInterface.tsx:439-446	Parseo estricto con error explícito; «Max» calculado con aritmética de enteros (BigInt)	Corregido

		float y podía redondear por encima del balance			
L	Bajo	Historial de swaps en localStorage (latinlink_recent_swaps)	src/lib/localSwapHistory.ts:48	Almacenamiento acotado y saneado; datos on-chain públicos, sin material sensible	Corregido
M	Bajo	Log de address + calldata en console.error y POST a /api/dev/swap-failure	src/lib/swapFailureReport.ts:123-133	Logging gateado a entorno de desarrollo; sin salida en producción	Corregido
N	Info	cookieStorage de wagmi guarda estado de sesión no HttpOnly (por diseño de wagmi; sin secretos)	appKitConfig.ts:56-58	Revisado y documentado; sin secretos en cookie	Corregido
O	Info	Código muerto: fetchOneInchQuote / ONEINCH_API_KEY no se usan	src/lib/benchmarkFetchers.ts:81-100	Código eliminado	Corregido
P	Info	URLs placeholder / http:// en listas de tokens (cdn.oo, cdn.x,	varias listas de datos	Listas limpiadas; sin mixed-content	Corregido

		example.com , http://localhost)			
Q	Info	console.error ("[TokenDetailError]", error) en boundary de cliente	tokens/ [chain]/ [symbol]/ error.tsx:14	Log gateado a no-producción	Corregido

Tabla 5. Hallazgos de severidad Bajo / Info y su corrección.

4. Dimensiones auditadas — resultado limpio

Las siguientes dimensiones se revisaron en la auditoría inicial y no arrojaron hallazgos:

- Manejo de claves/semillas: la app nunca toca private keys/mnemonics/seeds. Firma 100% delegada a Reown AppKit/wagmi (useWriteContract, useSendTransaction). Diseño non-custodial correcto.
- Secretos en cliente: todas las claves (ZERION, CMC, COINGECKO, OPENOCEAN_REFERRER) son server-only. Ninguna NEXT_PUBLIC_* es secreta. Nada hardcodeado.
- XSS / renderizado peligroso: 0 dangerouslySetInnerHTML, innerHTML, eval, new Function, document.write. Todo renderizado con escapado JSX.
- Open redirect: todos los destinos de navegación son estáticos o de allowlist (Navbar.tsx:59, router.back()). El redirect de i18n no se invoca.
- CSRF: N/A — no hay autenticación por cookie del lado servidor; las rutas POST son stateless. La autorización es por firma on-chain.
- i18n / next-intl: locale validado contra allowlist ["en","es","pt"] antes del import dinámico (i18n/request.ts:7-13); re-validado en layout.tsx. Sin path traversal.
- SSR / RSC: chain validado contra ALL_CHAINS antes de cualquier fetch; symbol no se interpola en URLs; coinGeckoId con encodeURIComponent. Sin SSRF ni fuga de secretos.
- Prototype pollution / parsing: JSON.parse solo sobre archivo local de confianza o localStorage propio. Sin {...untrusted} en objetos sensibles.
- ReDoS: solo dos regex de usuario, ambas lineales (email e input numérico). Sin new RegExp de input.
- Supply-chain: sin scripts postinstall/preinstall/prepare; pnpm.onlyBuiltDependencies restringido a ["esbuild"] (buena higiene).
- .env: .gitignore cubre .env*; ningún .env trackeado; sin secretos en el repo.
- Config de wallet: telemetría de conectores desactivada, todos los RPC por HTTPS, PROJECT_ID server-controlado.

4.1 Aciertos defensivos destacables

- Approval ERC-20 de importe exacto (no infinito) — acota el blast radius.

- Slippage auto-calculado y acotado a [0.5%, 2.5%] (config/swap.ts), sin input manual → sin footgun de 0%/100%.
- Re-cotización fresca antes de enviar.
- dev/swap-failure hard-gated a 404 fuera de desarrollo.
- Rutas de referencia con validación correcta: oo-tokens/[chain], tokens, prices/[pair], coingecko/coin/[id].

5. Hoja de ruta de remediación — ejecutada

La hoja de ruta prescrita en la versión 1.0 se estructuró en tres sprints. Todos ellos han sido completados.

5.1 Sprint 1 — impacto/esfuerzo óptimo (fondos y coste) — COMPLETADO

- [A] Fijar chainId en send/approve/allowance + gate de cadena en canSwap. — Completado.
- [C] Validar address/period en las 4 rutas Zerion. — Completado.
- [D] Actualizar ws, form-data, hono, postcss (update u overrides). — Completado.

5.2 Sprint 2 — abuso y lógica de negocio — COMPLETADO

- [B] Rate-limiting por IP en los proxies; acotar cardinalidad de ids/limit/days. — Completado.
- [E] Recalcular amountUSD y categoría de fee en el servidor, por dirección de token. — Completado.
- [G] Cabeceras de seguridad (CSP con cuidado por wallet/GTM). — Completado.

5.3 Sprint 3 — hardening y limpieza — COMPLETADO

- [F] Allowlist de routers de OpenOcean + verificación de minOut. — Completado.
- [H] Normalizar cuerpos de error del upstream. — Completado.
- [I–Q] validUntil, precompilado POL, precisión Max/toWei, gating de logs, borrar código muerto, limpiar URLs placeholder. — Completado.

6. Certificación de cierre de la auditoría

Declaración de cierre emitida por Metlabs Blockchain, S.L. en relación con la auditoría de seguridad de appWallet realizada para Latin Link.

D. Pablo Gómez González, en su condición de CEO y fundador de Metlabs Blockchain, S.L. (CIF B56787328), con domicilio en Edificio Proa, Av. García Sabell 1, A Coruña, CERTIFICA:

- Que Metlabs realizó el 6 de julio de 2026 una auditoría estática de seguridad sobre el código fuente de appWallet (rama main, commit 05a466b), cuyos resultados se recogieron en la versión 1.0 de este informe.
- Que dicha auditoría identificó un total de 17 hallazgos, identificados con las letras A a Q: 4 de severidad Alta, 4 de severidad Media y 9 de severidad Baja / Informativa.
- Que la TOTALIDAD de los hallazgos detectados (17 de 17) ha sido corregida conforme a las medidas de remediación prescritas, según se detalla en las secciones 1.4, 3 y 5 del presente documento.
- Que, a la fecha de emisión de esta certificación, no consta ningún hallazgo abierto ni pendiente de remediación dentro del alcance de la auditoría.
- Que el alcance de esta certificación se circunscribe al análisis estático del código fuente descrito en la sección 1.1, quedando fuera de él el runtime desplegado, la infraestructura, los contratos on-chain de terceros y el pentest dinámico, cuya realización se sigue recomendando como complemento.

> *Se certifica que todos los hallazgos de seguridad detectados en la auditoría de appWallet de julio de 2026 han sido corregidos y que el informe queda cerrado sin hallazgos pendientes.*
>
> — Pablo Gómez González, CEO — Metlabs Blockchain, S.L.

Campo	Detalle
Certifica	Pablo Gómez González
Cargo	CEO y Fundador
Entidad	Metlabs Blockchain, S.L. — CIF B56787328
Domicilio	Edificio Proa, Av. García Sabell 1, A Coruña
Cliente	Latin Link
Proyecto	appWallet — wallet non-custodial + DEX
Informe	Auditoría de Seguridad — appWallet, v2.0 (cierre de remediación)
Hallazgos corregidos	17 / 17 (100%)
Lugar y fecha	A Coruña, ____ de septiembre de 2026

Tabla 6. Datos de la certificación.

Firma y sello:

```
Firmado por ***5281** PABLO
GÓMEZ (R: ****8732*) el día
10/09/2026 con un certificado
emitido por AC Representación
```

_____

Pablo Gómez González

CEO — Metlabs Blockchain, S.L.

7. Apéndice — comandos de verificación ejecutados

- Palabras clave sensibles y ejecución dinámica: grep -rEn "private[_]?key|mnemonic|seed[_]?phrase|eval\(|new Function" src
- Inventario de URLs: grep -rEoh "https?://[a-zA-Z0-9./_-]+" src | sort -u
- Variables de entorno (NEXT_PUBLIC_ = expuestas al cliente): grep -rEohn "process\.env\.[A-Z0-9_]+" src | sort | uniq -c | sort -rn
- Sumideros de XSS y almacenamiento: grep -rn "dangerouslySetInnerHTML|innerHTML|localStorage|sessionStorage" src
- Dependencias: pnpm audit --prod
- Supply-chain y secretos en repo: git ls-files | grep -iE "\.env"

Superficie API auditada (18 rutas): cmc/{altcoin-season,fear-greed-history,market-history,market}, coingecko/{chart,coin/[id],markets,trending}, dev/swap-failure, fee-config/version, oo-tokens/[chain], prices/[pair], swap/quote, tokens, zerion/[address]/{charts/[period],portfolio,positions,transactions}.

Informe inicial generado el 6 de julio de 2026 (v1.0). Versión de cierre de remediación emitida en septiembre de 2026 (v2.0). Auditoría estática de código fuente; se recomienda complementar con pentest dinámico sobre el entorno desplegado (cabeceras reales, CSP, mixed-content) y revisión de los contratos on-chain de terceros.