vulnix

CONFIDENTIAL · VERSION 1.0

Security assessment report

demo-edubridge.example, portal.demo-edubridge.example

Prepared for Demo workspace

ASSESSMENT WINDOW	FINDINGS	REPORT ID	GENERATED
4 Aug 2026, 17:01 to 18:05 UTC	15	demo-run-edubridge-01	20 Nov 2026

REPORT

About this report

Contents

01	Executive summary	3
02	Scope and approach	5
03	All findings	7
04	Findings	9
05	Remediation plan	32
06	Appendix	34

Document control

Prepared for	Demo workspace
Report ID	`demo-run-edubridge-01`
Version	1.0
Classification	Confidential
Prepared by	Vulnix autonomous agent
Generated	20 Nov 2026
Run in Vulnix	Open the run

How to read it

Finding IDs (VLX-01) are unique within this report and sorted by priority. Every ID and title is a link to the finding.

Severity follows the CVSS v3.1 bands. Every severity label links to the severity model.

Each finding states where it is, how to reproduce it and how to fix it.

Status is the state of the finding when the report was generated.

Confidentiality

This report describes weaknesses in Demo workspace's systems and is for its authorised recipients only. Share findings with the people who fix them. Do not publish the report.

01 SUMMARY

Executive summary

Vulnix confirmed 15 vulnerabilities on 2 targets.

4 critical, 4 high, 6 medium and 1 low.

Most are in authentication and sessions (6) and access control (5).

Every finding includes the steps to reproduce it.

Fix the 8 critical and high findings first. They are listed below in the order to take them.

What this report does not cover, page 6.

FINDINGS	CRITICAL AND HIGH	HIGHEST CVSS
15	8	9.8
15 open, 0 fixed	4 critical, 4 high	VLX-01, critical



4 Critical 4 High 6 Medium 1 Low 0 Info

Fix these first

Ordered by severity, then CVSS score.

ID	SEVERITY	FINDING	CVSS	PAGE
VLX-01	CRITICAL	Hardcoded Admin API Token and Universal OTP Backdoor on Staging Rails API `/api/v1/admin/* and /api/v1/admin/login`	9.8	9
VLX-02	CRITICAL	Missing Row Level Security on Supabase backend of portal.demo-edubridge.example (anon key = full database read/write/realtime access) `https://demo-ba84718a.supabase.co/rest/v1 (and /realtime/v1/websocket)`	9.8	11
VLX-03	CRITICAL	Account takeover via unauthenticated session-token minting in create-session edge function (portal.demo-edubridge.example) `/functions/v1/create-session`	9.1	13
VLX-04	CRITICAL	PayLink payment bypass: paid-group-PayLink-submit marks lead paid on any receipt image (portal.demo-edubridge.example) `/functions/v1/paid-group-PayLink-submit (+ /functions/v1/paid-group-create-lead)`	9.1	15
VLX-05	HIGH	Payment bypass: test-mark-paid edge function marks leads paid without payment (portal.demo-edubridge.example) `/functions/v1/test-mark-paid`	8.2	17

ID	SEVERITY	FINDING	CVSS	PAGE
VLX-06	HIGH	Exposed Grafana 11.1.0 with known critical vulnerability (CVE-2024-9264 SQL Expressions command injection) /api/health	7.5	19
VLX-07	HIGH	Price tampering: one-to-one-paid-create-lead persists client-supplied package_amount (0/negative accepted) (portal.demo-edubridge.example) /functions/v1/one-to-one-paid-create-lead	7.5	20
VLX-08	HIGH	Brute-forceable 4-digit OTP login with weak rate limiting and no lockout on api.demo-edubridge.example /api/v1/student/login (and /api/v1/student/registrations/verify)	7.4	22

Where the weaknesses cluster

GROUP	FINDINGS	HIGHEST	IDS
Authentication and sessions An attacker can guess, reuse or bypass credentials and sessions to act as a user.	6	CRITICAL	VLX-01, VLX-08, VLX-10, VLX-11, VLX-13, VLX-15
Access control A user can read or change data that belongs to someone else, or reach functions meant for staff.	5	CRITICAL	VLX-02, VLX-03, VLX-04, VLX-05, VLX-14
Known-vulnerable components Software in use has published vulnerabilities. Upgrade to a fixed release.	2	HIGH	VLX-06, VLX-09
Business logic and abuse The application's own rules can be bypassed or exhausted: payments, quotas, uploads, rate limits.	1	HIGH	VLX-07
Data exposure Records, credentials or internal details are readable by people who should not see them.	1	MEDIUM	VLX-12

OWASP Top 10:2025 mapping

CODE	CATEGORY	IDS	FINDINGS
A01	Broken Access Control	VLX-02, VLX-03, VLX-04, VLX-05, VLX-12, VLX-14	6
A03	Software Supply Chain Failures	VLX-06, VLX-09	2
A06	Insecure Design	VLX-07	1
A07	Authentication Failures	VLX-01, VLX-08, VLX-10, VLX-11, VLX-15	5

1 finding has no entry in the Top 10:2025 category lists and is not counted above.

02 METHOD

Scope and approach

Targets

TARGET	TYPE
demo-edubridge.example	Domain
portal.demo-edubridge.example	Domain

Assessment

Assessment type	External black-box test. No source code was provided.
Depth	Deep
Assessment window	4 Aug 2026, 17:01 to 18:05 UTC (1 h 4 min)
Request rate limit	15 requests per second
Safe mode	On
Excluded paths	None

How the testing ran

STAGE	WHAT IT DOES	AGENTS
Reconnaissance	Maps hosts, endpoints, technologies and entry points inside the scope.	2
Testing	Probes the mapped surface by weakness class: access control, authentication, injection, business logic.	7
Validation	Reproduces candidate findings against the live target to confirm them.	14
Reporting	Writes up each confirmed finding with its reproduction steps and fix.	15

Stages are read from agent names. The lead agent is not counted. Agents: 39, Tool calls: 1,278, Shell commands: 853.

What counts as a finding

Vulnix reports a weakness only after the agent has demonstrated it against the target. Each finding carries its own reproduction steps, the affected location and a fix.

Severity model

SEVERITY	CVSS V3.1	MEANING
CRITICAL	9.0 to 10.0	Exploitable now, with no special access, and the result is control of data or accounts at scale. Fix immediately.
HIGH	7.0 to 8.9	Exploitable with little effort and the result is serious: account takeover, exposure of sensitive records, or a bypass of a core control. Fix this sprint.
MEDIUM	4.0 to 6.9	Exploitable, but needs a precondition (a valid account, user interaction, a second weakness) or the result is bounded. Plan the fix.
LOW	0.1 to 3.9	Weakens a defence or leaks minor detail. Fix as part of normal hardening.
INFO	0.0	Not exploitable on its own. Recorded so it is not lost, usually a hardening gap or a condition that raises the cost of another finding.

Limits of this report

This report lists what the agent confirmed. An area with no finding means none was confirmed there, not that none exists.

Only the targets in scope were tested. Hosts, services and third-party systems outside the scope were not touched.

Findings describe the targets as they were during the assessment window. Later changes are not reflected.

The agent tests what it can reach over the network (and, for a source scan, the repository). Physical access and social engineering are out of scope.

This is a technical assessment. It is not a compliance attestation or a certification.

03 FINDINGS

All findings

ID	SEVERITY	FINDING	CVSS	CWE	STATUS	PAGE
VLX-01	CRITICAL	Hardcoded Admin API Token and Universal OTP Backdoor on Staging Rails API `/api/v1/admin/* and /api/v1/admin/login`	9.8	CWE-798	Open	9
VLX-02	CRITICAL	Missing Row Level Security on Supabase backend of portal.demo-edubridge.example (anon key = full database read/write/realtime access) `https://demo-ba84718a.supabase.co/rest/v1 (and /realtime/v1/websocket)`	9.8	CWE-862	Open	11
VLX-03	CRITICAL	Account takeover via unauthenticated session-token minting in create-session edge function (portal.demo-edubridge.example) `/functions/v1/create-session`	9.1	CWE-639	Open	13
VLX-04	CRITICAL	PayLink payment bypass: paid-group-PayLink-submit marks lead paid on any receipt image (portal.demo-edubridge.example) `/functions/v1/paid-group-PayLink-submit (+ /functions/v1/paid-group-create-lead)`	9.1	CWE-863	Open	15
VLX-05	HIGH	Payment bypass: test-mark-paid edge function marks leads paid without payment (portal.demo-edubridge.example) `/functions/v1/test-mark-paid`	8.2	CWE-863	Open	17
VLX-06	HIGH	Exposed Grafana 11.1.0 with known critical vulnerability (CVE-2024-9264 SQL Expressions command injection) `/api/health`	7.5	CWE-78	Open	19
VLX-07	HIGH	Price tampering: one-to-one-paid-create-lead persists client-supplied package_amount (0/negative accepted) (portal.demo-edubridge.example) `/functions/v1/one-to-one-paid-create-lead`	7.5	CWE-602	Open	20
VLX-08	HIGH	Brute-forceable 4-digit OTP login with weak rate limiting and no lockout on api.demo-edubridge.example `/api/v1/student/login (and /api/v1/student/registrations/verify)`	7.4	CWE-307	Open	22
VLX-09	MEDIUM	Exposed Portainer CE 2.39.1 with known unpatched vulnerabilities (CVE-2026-44881, CVE-2026-44883) `/api/status`	6.5	CWE-59	Open	24
VLX-10	MEDIUM	Missing authentication on POST /drive/sync (Tawthiq API) `/drive/sync`	6.5	CWE-306	Open	25

ID	SEVERITY	FINDING	CVSS	CWE	STATUS	PAGE
VLX-11	MEDIUM	WhatsApp OTP verification lacks lockout and backoff. 4-digit OTP brute force with user enumeration (portal.demo-edubridge.example) `/functions/v1/send-whatsapp-otp and /functions/v1/verify-otp-v2`	6.5	CWE-307	Open	26
VLX-12	MEDIUM	Unauthenticated Grafana /metrics endpoint exposes internal monitoring data `/metrics`	5.3	CWE-668	Open	28
VLX-13	MEDIUM	Unauthenticated Student Account Enumeration via send_verification Endpoint `/api/v1/student/registrations/send_verification`	5.3	CWE-204	Open	29
VLX-14	MEDIUM	Object-level authorization bypass (read) on /api/v1/student/projects/{id} exposes internal review metadata of other students' projects `/api/v1/student/projects/{id}`	4.3	CWE-639	Open	30
VLX-15	LOW	Session tokens never expire and survive logout on app.demo-edubridge.example (client-side only logout) `/api/v1/student/login (and /api/v1/student/account/memberships)`	3.1	CWE-613	Open	31

All findings VLX-02 → Open in Vulnix

CRITICAL VLX-01

Hardcoded Admin API Token and Universal OTP Backdoor on Staging Rails API

AFFECTED LOCATION `/api/v1/admin/*` and `/api/v1/admin/login`		
CVSS V3.1 9.8 Critical	STATUS Open	FOUND 4 Aug 2026
WEAKNESS CWE-798 · Use of Hard-coded Credentials		
OWASP TOP 10:2025 A07 Authentication Failures		GROUP Authentication and sessions

Description

The staging Rails API at `api-staging.demo-edubridge.example` authenticates a full administrator session using a hardcoded bearer token and a fixed one-time password that are shipped inside the public client-side bundle of the production application. An unauthenticated attacker can recover these values from the client bundle and obtain complete administrative control of the staging API, including access to student/tutor/session PII, one-click impersonation of any user via auto-login URLs, and bulk CSV export.

Reproduction steps

1. Retrieve the shipped client bundle from the production application host (`app.demo-edubridge.example`) and extract the hardcoded administrator token `q3aY…[redacted]` and the admin-login OTP `[redacted]` from the embedded QuickLogin tooling.
2. Send `GET /api/v1/admin/profile` to the staging API with header `Authorization: q3aY…[redacted]` and observe HTTP 200 returning the administrator profile.
3. Send `POST /api/v1/admin/login` to the staging API with JSON body `{"email":"admin@service-5684d0de.demo-edubridge.example","otp":"[redacted]"}` and observe HTTP 200 returning a valid admin session token.
4. Using either credential, list a small number of records Demo User 1e5e3ce2 via `/api/v1/admin/students?per_page=2`, `/api/v1/admin/tutors?per_page=2`, and `/api/v1/admin/sessions?per_page=2`, each returning HTTP 200.
5. Call `/api/v1/admin/students/3108/generate_auto_login_url` and use the returned token as `Authorization` on `/api/v1/student/profile` to demonstrate impersonation of another user.
6. Optionally call `/api/v1/admin/students/export` and `/api/v1/admin/sessions/export` to confirm bulk export capability. All steps target the staging environment only.

Remediation

Remove the QuickLogin developer tooling and all hardcoded secrets from production client bundles before release, and add automated secret scanning to the build pipeline so credential material cannot ship in front-end artifacts. Immediately revoke the hardcoded administrator token and invalidate the fixed-OTP login path on staging, rotating all credentials that were issued through them and forcing re-authentication of affected sessions. Replace the fixed OTP with a proper server-side authentication flow that uses per-user, single-use, expiring secrets and requires the normal credential validation path, with no static fallback values. Restrict impersonation endpoints such as `generate_auto_login_url` so they require explicit, authorized administrative actions, are scoped to non-production environments (or removed entirely), and are covered by audit logging and rate limiting. Ensure test credentials are never embedded in client code and are provisioned through environment-specific configuration instead, with least-privilege roles assigned to any test accounts.

All findings ← VLX-01 VLX-03 → Open in Vulnix

CRITICAL VLX-02

Missing Row Level Security on Supabase backend of portal.demo-edubridge.example (anon key = full database read/write/realtime access)

AFFECTED LOCATION		
https://demo-ba84718a.supabase.co/rest/v1 (and /realtime/v1/websocket)		
CVSS V3.1 9.8 Critical	STATUS Open	FOUND 4 Aug 2026
WEAKNESS CWE-862 · Missing Authorization		
OWASP TOP 10:2025 A01 Broken Access Control		GROUP Access control

Description

The Supabase project backing **portal.demo-edubridge.example** (project ref `demo-ba84718a`) has Row Level Security (RLS) missing or misconfigured for the `anon` role on nearly all tables. The publicly embedded anon key (role=`anon`, not `service_role`) found in the client-side JavaScript bundle lets any unauthenticated internet user read, write, update, delete, and live-subscribe to sensitive database tables through the PostgREST and Realtime APIs, including ~20,000 student records with plaintext passwords, 3,210 plaintext login tokens, and 2,966 paying-customer payment records.

Reproduction steps

1. Load the portal.demo-edubridge.example SPA and extract the Supabase anon key from a client JS bundle (an `apikey` value with JWT shape), it is embedded client-side by design.
2. Decode the JWT and confirm the `role` claim is `anon` (project ref `demo-ba84718a`), proving the key is the public anon key, not `service_role`.
3. Unauthenticated read: send `GET https://demo-ba84718a.supabase.co/rest/v1/student_info_trial?select=*&limit=1` with headers `apikey: <anon>` and `Authorization: Bearer <anon>`. Observe HTTP 206 with `Content-Range: 0-0/19948` and a full row including `name`, `email`, `phone`, and plaintext `password` columns.
4. Repeat the same request for `access_tokens_trial` (3,210 rows), `paid_group_students` (2,966 rows), `users_trial` (91 rows), and the other readable tables; all return data.
5. Unauthenticated write cycle: `POST /rest/v1/prereg_clicks` with a clearly-marked test row and header `Prefer: return=representation` (HTTP 201); `PATCH /rest/v1/prereg_clicks?id=eq.<id>` (HTTP 204); `DELETE /rest/v1/prereg_clicks?id=eq.<id>` (HTTP 204); then verify the row is gone (0 rows remaining).
6. Realtime: connect to `wss://demo-ba84718a.supabase.co/realtime/v1/websocket?apikey=<anon>&vsn=1.0.0`, send a `phx_join` for topic `realtime:public:sessions_trial` with `postgres_changes=[{"event":"*","schema":"public","table":"sessions_trial"}]` and `access_token: <anon>`, and observe the system message "Subscribed to PostgreSQL".

Remediation

1. Enable Row Level Security on every table in the database and verify it is on for existing tables (including `student_info_trial`, `users_trial`, `access_tokens_trial`, `paid_group_students`, `sessions_trial`, `booked_sessions_trial`, `resume_tokens`, `vacation_requests`, `trial_account_family`, `system_settings`, `prereg_clicks`).
2. Replace any permissive `anon`/`authenticated` "allow all" policies with least-privilege policies that scope access to the caller's identity via `auth.uid()` and tenant/ownership columns; the `anon` role should have no access to sensitive tables.
3. Revoke `INSERT`, `UPDATE`, and `DELETE` for the `anon` role on all sensitive tables; require a properly authenticated session for any write operation and validate ownership server-side before applying changes.
4. Remove plaintext password storage entirely; migrate to salted, adaptive hashes (bcrypt/argon2) and rotate the currently exposed passwords and login/session tokens stored in `access_tokens_trial`.
5. Restrict Realtime publication to only the tables that legitimately need live sync, and enforce RLS-aware subscriptions so `postgres_changes` streams respect per-row authorization.
6. Rotate the anon key and any secrets referenced in payment records after applying the policy fixes, and re-audit the Supabase security posture (RLS status, policies, grants, publication membership) before redeployment.

All findings ← VLX-02 VLX-04 → Open in Vulnix

CRITICAL VLX-03

Account takeover via unauthenticated session-token minting in create-session edge function (portal.demo-edubridge.example)

AFFECTED LOCATION `/functions/v1/create-session`		
CVSS V3.1 9.1 Critical	STATUS Open	FOUND 4 Aug 2026
WEAKNESS CWE-639 · Authorization Bypass Through User-Controlled Key		
OWASP TOP 10:2025 A01 Broken Access Control		GROUP Access control

Description

The `create-session` Supabase edge function behind the edubridge tutoring app (`portal.demo-edubridge.example`) mints a valid, ~1-year session token for **any** user id supplied in the request body, using only the public anonymous API key embedded in the client-side JavaScript. No authentication or ownership check verifies that the caller is the user they claim to be. An unauthenticated attacker can mint a session for any real user, including administrators, and then access that user's account and admin-only functionality, achieving full account takeover.

Reproduction steps

1. Read an active admin id: `GET /rest/v1/users_trial?select=id,name,role,is_active&role=eq.admin&is_active=eq.true&limit=1` with the public anon key (no user session). The response contains an admin id, e.g. `394fac8e-82e7-4a4c-be61-51fcd9f6ad88` ("Demo User 01a0eeda").
2. Mint a session for that admin: `POST /functions/v1/create-session` with body `{"userId":"394fac8e-82e7-4a4c-be61-51fcd9f6ad88","userType":"staff","role":"admin"}` and only the anon key. The response is HTTP 200 with a `sessionToken` and `expiresAt` ~1 year out.
3. Validate the minted token: `POST /functions/v1/validate-session-v2` with body `{"sessionToken":"<token>"}`. The response is HTTP 200 `{"valid":true,"user":{...}}` containing the victim admin's full profile (name, email, phone, role).
4. Exercise admin functionality with the minted token: `POST /functions/v1/admin-waiting-list` with body `{"sessionToken":"<token>","action":"list"}` returns HTTP 200 with 257 waiting-list rows (names, phones, emails of prospective students). This is a Demo User 1e5e3ce2 action; no mutation is performed.
5. Prove minting is user-id-agnostic: `POST /functions/v1/create-session` with body `{"userId":"00000000-0000-4000-8000-0000000000fe","userType":"student","role":"admin"}` still returns HTTP 200 with a token; validating that token returns HTTP 404 `{"valid":false,"error":"User not found"}`, no ownership check exists.

6. (Aggravating vector) Read the session store: `GET /rest/v1/auth_sessions?select=session_token,user_id,role,expires_at&limit=3` with the anon key returns HTTP 200 with plaintext session tokens of real users, allowing session hijack without minting.

Remediation

Enforce server-side authorization in `create-session`: the edge function must only issue a session token for the authenticated caller's own verified identity. The `userId`, `userType`, and `role` fields must never be accepted from unauthenticated client input, the function should derive the identity from a server-verified context (e.g., a Supabase Auth session, or a short-lived server-issued code bound to the phone number that was OTP-verified moments earlier) and reject requests that do not carry one. Role must be read from the server-side user record, not from the request body. Additionally: enable Row-Level Security on `auth_sessions`, `access_tokens_trial`, `users_trial`, and all PII tables so the anon role cannot read or write them; shorten session-token lifetime; and implement server-side token revocation (e.g., invalidate on logout and on password/phone change) so stolen tokens cannot remain valid for a year. Finally, audit all edge functions that accept a `sessionToken` to ensure they validate it against a server-side session store rather than trusting a client-asserted identity.

All findings ← VLX-03 VLX-05 → Open in Vulnix

CRITICAL VLX-04

PayLink payment bypass: paid-group-PayLink-submit marks lead paid on any receipt image (portal.demo-edubridge.example)

AFFECTED LOCATION `/functions/v1/paid-group-PayLink-submit (+ /functions/v1/paid-group-create-lead)`		
CVSS V3.1 9.1 Critical	STATUS Open	FOUND 4 Aug 2026
WEAKNESS CWE-863 · Incorrect Authorization		
OWASP TOP 10:2025 A01 Broken Access Control		GROUP Access control

Description

The group-classes PayLink payment flow on portal.demo-edubridge.example verifies payment by having the customer upload a receipt image. The Supabase edge function `paid-group-PayLink-submit` accepts any image file (no receipt/transfer verification) and immediately marks the lead as paid, sets `paid_at`, and mints a 1-year access token that unlocks the paid-tier dashboard. Any unauthenticated attacker can create a lead with fake data using the public anon key, upload a programmatically generated 1×1 transparent PNG as the "receipt", and obtain permanent paid-tier access and a valid access token without paying.

Reproduction steps

1. Generate a minimal valid 1×1 transparent RGBA PNG programmatically (PNG signature + IHDR + IDAT + IEND) and base64-encode it, this is not a bank receipt.
2. POST `/functions/v1/paid-group-create-lead` with the public anon key and fake data (test name/email/phone, `utm_source=Demo User 341cb5fb`). Note the returned `id` and `submit_token`.
3. POST `/functions/v1/paid-group-PayLink-submit` with that lead id, the returned `submit_token`, the base64 PNG as `file_base64`, `content_type=image/png`, `switches_count=0`.
4. Observe the HTTP 200 response: `status:"paid"`, `payment_status:"paid"`, `paid_at` set, and a 1-year `access_token`.
5. (Optional) POST `/functions/v1/paid-group-me` with the minted `access_token` to confirm it resolves to the paid student object.
6. Confirm persistence via `GET /rest/v1/paid_group_students?select=...&id=eq.<lead_id>` with the anon key.
7. Cleanup: attempt `DELETE /rest/v1/paid_group_students?id=eq.<lead_id>` (note: anon-role RLS blocks deletes on this table, so the fake test row persists and requires service-role/admin purge; no real data is used).

Remediation

Do not treat receipt upload as payment confirmation. Verify PayLink transfers server-side against the actual payment provider/bank (amount, reference, destination account, and status) before any state change, or require explicit manual admin approval of the submitted receipt before flipping the lead to paid. Issue the access token (and any paid-tier entitlement) only after payment is positively verified or manually approved, never on upload. Additionally, remove the client-issued submit_token from the trust boundary by tying token issuance and redemption to a server-side authenticated session, and consider enforcing authentication/rate-limiting on lead creation so anonymous callers cannot fabricate leads at scale. Backfill a review queue for the already-flagged pending receipts and treat `PayLink_review_status` as a hard gate on access rather than a post-hoc audit flag.

All findings ← VLX-04 VLX-06 → Open in Vulnix

HIGH VLX-05

Payment bypass: test-mark-paid edge function marks leads paid without payment (portal.demo-edubridge.example)

AFFECTED LOCATION /functions/v1/test-mark-paid		
CVSS V3.1 8.2 High	STATUS Open	FOUND 4 Aug 2026
WEAKNESS CWE-863 · Incorrect Authorization		
OWASP TOP 10:2025 A01 Broken Access Control		GROUP Access control

Description

The Supabase edge function `test-mark-paid` on portal.demo-edubridge.example lets any unauthenticated caller mark a lead as paid and enroll in the paid group course without any payment. Authorization is derived from an attacker-controlled lead email matched against a client-side allowlist, and no payment verification with the payment gateway occurs anywhere in the flow.

Reproduction steps

1. Send `POST /functions/v1/paid-group-create-lead` with fabricated lead data and the allowlisted email `user-8ce67f405541@demo-edubridge.example`; capture the returned `id` and `submit_token`.
2. Send `POST /functions/v1/test-mark-paid` with the returned `id` and `"product": "group"`.
3. Observe HTTP 200 with fabricated `transaction_id`, `stripe_session_id`, and `event_id`.
4. Query the lead row via `GET /rest/v1/paid_group_students?id=eq.<id>` and confirm `payment_status=paid`, `paid_at`, and `stripe_payment_intent` were written.
5. Call `POST /functions/v1/paid-group-verify-payment` with `{"lead_id":<id>,"canceled":false}` and `POST /functions/v1/paid-group-save-slots` with a preferred slot to confirm full enrollment without payment (`status=scheduled`).
6. Attempt cleanup with `DELETE /rest/v1/paid_group_students?id=eq.<id>`; the anonymous role cannot delete the row (RLS), so the fabricated record persists.

Remediation

Disable or remove the `test-mark-paid` edge function in production; test scaffolding must not be deployable against the live payment flow. Enforce server-side authorization on every payment-mutating function: require an authenticated admin session (service-role token or admin JWT) and verify the caller's identity server-side, never deriving authorization from a client-supplied email or from the email stored on a caller-created lead record. Remove the client-side allowlist bypass path from the production bundle. Derive `payment_status`

exclusively from verified gateway events such as Stripe `checkout.session.completed` webhooks, never from a client-invoked function. Add row-level-security policies that prevent anonymous users from creating or mutating paid-student records, and require verified contact details at lead creation.

All findings ← VLX-05 VLX-07 → Open in Vulnix

HIGH VLX-06

Exposed Grafana 11.1.0 with known critical vulnerability (CVE-2024-9264 SQL Expressions command injection)

AFFECTED LOCATION `/api/health`		
CVSS V3.1 7.5 High	STATUS Open	FOUND 4 Aug 2026
WEAKNESS CWE-78 · OS Command Injection		CVE CVE-2024-9264
OWASP TOP 10:2025 A03 Software Supply Chain Failures		GROUP Known-vulnerable components

Description

The Grafana instance at `https://grafana.apps.demo-edubridge.example` is internet-exposed and runs version **11.1.0**, which is affected by **CVE-2024-9264** (CRITICAL, CVSS 9.9), a command injection and local file inclusion vulnerability in the experimental SQL Expressions (duckdb) feature. The version is also affected by additional medium-severity CVEs (CVE-2024-6322, CVE-2024-11741). The exact version is disclosed unauthenticated via `/api/health` and `/metrics`.

Reproduction steps

1. Send an unauthenticated `GET` request to `https://grafana.apps.demo-edubridge.example/api/health`.
2. Observe `HTTP 200` and the JSON body containing `"version":"11.1.0"`.
3. Send an unauthenticated `GET` request to `https://grafana.apps.demo-edubridge.example/metrics`.
4. Observe `HTTP 200` and the `grafana_build_info` metric confirming `version="11.1.0"`.
5. Compare `11.1.0` against the advisory ranges: it falls in the affected range of CVE-2024-9264 (`>= 11.1.0, <= 11.1.7`), CVE-2024-6322 (`=11.1.0`), and CVE-2024-11741 (`>= 11.1.0, < 11.1.11`).

Remediation

1. Upgrade Grafana to **11.1.7+security-01** or later; **11.1.11+** is recommended to also cover CVE-2024-11741 and other fixes.
2. Disable the experimental SQL Expressions feature unless it is strictly required for the deployment.
3. Restrict network access to the Grafana instance (e.g., VPN/IP allowlist) so the Demo User 12f76e88 and monitoring data are not reachable from the public internet.
4. Enforce strong administrator credentials and enable multi-factor authentication; audit current users for stale Viewer/Editor accounts.
5. Monitor Grafana advisory feeds and apply security patches promptly after release.

All findings ← VLX-06 VLX-08 → Open in Vulnix

HIGH VLX-07

Price tampering: one-to-one-paid-create-lead persists client-supplied package_amount (0/negative accepted) (portal.demo-edubridge.example)

AFFECTED LOCATION /functions/v1/one-to-one-paid-create-lead		
CVSS V3.1 7.5 High	STATUS Open	FOUND 4 Aug 2026
WEAKNESS CWE-602 · Client-Side Enforcement of Server-Side Security		
OWASP TOP 10:2025 A06 Insecure Design		GROUP Business logic and abuse

Description

The `one-to-one-paid-create-lead` Supabase edge function behind portal.demo-edubridge.example accepts `package_tier`, `package_currency`, and `package_amount` directly from the client and persists them verbatim with no server-side price derivation or validation. An unauthenticated caller can set the price of a paid 1:1 tutoring package to any value, including `0` or a negative amount, instead of the authoritative tier price (advanced = 599 USD). The stored amount is subsequently returned by the payment-verification function for analytics and is passed into the downstream checkout/signup flow, so a tampered price can result in a paid package being purchased at $1, $0, or less.

Reproduction steps

1. Extract the public Supabase anon key (role=anon) from the client-side JavaScript bundle served by portal.demo-edubridge.example.
2. POST a JSON body to `https://demo-ba84718a.supabase.co/functions/v1/one-to-one-paid-create-lead` with a fake test lead (`name`, `email`, `age`, `phone`, `attribution`), `package_tier: "advanced"`, `package_currency: "USD"`, and `package_amount: 1` (the client-side pricing table lists the advanced tier at 599 USD). Use only the anon key for `apikey`/`Authorization` headers, no user session is required.
3. Observe the HTTP 200 response, which returns only `id`, `course_slug`, and `submit_token`, no server-recomputed amount.
4. Read the stored row back with `GET /rest/v1/paid_group_students?id=eq.<id>&select=package_tier,package_currency,package_amount,payment_status,source` and confirm `package_amount` is `1` with `package_tier: advanced`.
5. Repeat step 2 with `package_amount: 0` and `package_amount: -50`; both are accepted (HTTP 200) and stored verbatim.
6. No real payment is initiated anywhere in this PoC; all leads use fake test data.

Remediation

Derive the price entirely server-side: maintain an authoritative server-side pricing table keyed by `package_tier` and `package_currency` (mirroring the client `oneToOnePaidPricing` data) and compute `package_amount` from it inside the edge function, ignoring any client-supplied amount. Validate the submitted tier/currency against the table and reject unknown or mismatched combinations. If client-supplied pricing fields must exist for the request shape, drop them from the persisted payload and never store or propagate a client-controlled amount; ensure `one-to-one-paid-verify-payment`, analytics, and the downstream checkout flow all read the server-derived amount rather than the lead row's stored value. Additionally, enforce `package_amount > 0` as a defense-in-depth check and add server-side authorization/rate controls so lead creation and price fields cannot be abused by unauthenticated callers.

All findings ← VLX-07 VLX-09 → Open in Vulnix

HIGH VLX-08

Brute-forceable 4-digit OTP login with weak rate limiting and no lockout on api.demo-edubridge.example

AFFECTED LOCATION `/api/v1/student/login (and /api/v1/student/registrations/verify)`		
CVSS V3.1 7.4 High	STATUS Open	FOUND 4 Aug 2026
WEAKNESS CWE-307 · Excessive Authentication Attempts		
OWASP TOP 10:2025 A07 Authentication Failures		GROUP Authentication and sessions

Description

The student OTP login flow accepts a 4-digit one-time password as the account password and enforces only a short per-minute rolling throttle (10 login / 5 verify attempts) that resets after ~60-70 seconds with no permanent lockout. Because the OTP stays valid and reusable across windows, can be re-issued through an unauthenticated, unthrottled endpoint, and lives in a 10,000-code space, an attacker who knows a victim's email can brute-force the code and obtain a full session token, taking over the account.

Reproduction steps

1. Create a disposable inbox at `https://api.mail.tm` (`POST /accounts` with a random `@web-library.net` address and password; obtain a bearer token via `POST /token`).
2. Trigger OTP generation for that address with `POST /api/v1/student/registrations/send_verification` (`{"email":..., "is_login":true}`), no authentication required.
3. Poll the inbox (`GET https://api.mail.tm/messages`) and extract the 4-digit verification code from the "edubridge Verification" message.
4. Submit 10 wrong 4-digit codes to `POST /api/v1/student/login`, each returns `401`; the 11th request returns `429 Retry later`.
5. Wait ~65 seconds, then submit more wrong codes, they are accepted again (`401`), proving the window resets with no lockout.
6. Wait another ~65 seconds, then submit the correct OTP, the request returns `200` with a session token for the account, demonstrating the OTP is still valid and the account is fully compromised.

Remediation

1. Enforce a small fixed failure budget per account with a permanent lockout or exponential backoff after 3-5 failed attempts (per NIST SP 800-63B / OWASP Authentication Cheat Sheet), instead of a resettable per-minute window.
2. Make OTPs single-use and short-lived: consume the code on first use (success or failure) and expire it after a short TTL (e.g., 5 minutes).

3. Increase OTP entropy to at least 6 digits and implement rate limiting and time-based expiration server-side.
4. Require authentication and apply rate limiting to `POST /api/v1/student/registrations/send_verification` (including `is_login:true`), and rate-limit per email/phone and per IP to prevent unlimited OTP generation.
5. Combine the login and verify throttling into one shared counter per account so counters cannot be used independently, and apply the same budget to both endpoints.
6. Consider CAPTCHA or device/behavioral risk signals for repeated login attempts from the same source.

All findings ← VLX-08 VLX-10 → Open in Vulnix

MEDIUM VLX-09

Exposed Portainer CE 2.39.1 with known unpatched vulnerabilities (CVE-2026-44881, CVE-2026-44883)

AFFECTED LOCATION `/api/status`		
CVSS V3.1 6.5 Medium	STATUS Open	FOUND 4 Aug 2026
WEAKNESS CWE-59 · Link Following		CVE CVE-2026-44881
OWASP TOP 10:2025 A03 Software Supply Chain Failures		GROUP Known-vulnerable components

Description

The Docker management interface at `https://portainer.apps.demo-edubridge.example` exposes Portainer CE 2.39.1 to the public internet, and that version is affected by two known high-severity vulnerabilities fixed only in 2.39.2: an arbitrary file read via Git symlink injection in stack auto-update (CVE-2026-44881) and a JWT-in-URL-query token leakage issue (CVE-2026-44883). Version and instance details are disclosed to unauthenticated callers.

Reproduction steps

1. Send an unauthenticated `GET` request to `https://portainer.apps.demo-edubridge.example/api/status`.
2. Observe the HTTP 200 response containing `"Version":"2.39.1"`.
3. Compare the disclosed version against the advisory range for CVE-2026-44881 (GHSA-rpgq-m5fp-32wr): affected `>= 2.39.0, < 2.39.2`; `2.39.1` falls inside the affected range and is fixed only in `2.39.2`.
4. Confirm the admin account exists via unauthenticated `GET /api/users/admin/check` (HTTP 204), and that `/api/settings/public` is reachable unauthenticated.

Remediation

Upgrade Portainer CE to version 2.39.2 or later, which fixes CVE-2026-44881 and CVE-2026-44883. In parallel, restrict access to the dashboard to trusted networks via IP allowlisting or a VPN so the Docker management plane is no longer publicly reachable. Enforce strong, unique administrator credentials with multi-factor authentication, and audit existing accounts and access logs. Consider removing public exposure of management interfaces entirely and terminating the deployment behind an internal-only network.

All findings ← VLX-09 VLX-11 → Open in Vulnix

MEDIUM VLX-10

Missing authentication on POST /drive/sync (Tawthiq API)

AFFECTED LOCATION		
/drive/sync		
CVSS V3.1 6.5 Medium	STATUS Open	FOUND 4 Aug 2026
WEAKNESS CWE-306 · Missing Authentication for Critical Function		
OWASP TOP 10:2025 A07 Authentication Failures		GROUP Authentication and sessions

Description

The `POST /drive/sync` endpoint of the Tawthiq document-verification admin API executes its request validation and handler logic **without requiring the Bearer token** that protects every other endpoint. Unauthenticated callers can trigger Google Drive sync operations with an attacker-controlled `folder_id` and learn internal configuration state; today the endpoint only fails because the Drive service-account configuration is unset.

Reproduction steps

1. Send `POST /drive/sync` with no `Authorization` header and no body. Observe `HTTP 503` with detail `Google Drive sync is not configured. Set GDRIVE_SERVICE_ACCOUNT_JSON.` instead of a `401`.
2. Send `POST /drive/sync` with no auth and JSON body `{"folder_id":"anything"}`. Observe the same `503`, the handler executes with an attacker-controlled `folder_id`.
3. Send `POST /drive/sync` with no auth and body `notjson` plus `Content-Type: application/json`. Observe `HTTP 422` with `json_invalid`, proving request validation runs before any authentication check.
4. Send `POST /drive/sync` with `Authorization: Bearer invalid-test-token` and body `{}`. Observe `503`, invalid tokens are never rejected with `401`.
5. Control: send `POST /submissions/batch` with no auth (observe `401`) and `GET /submissions?limit=1` with no auth (observe `401` with `Www-Authenticate: Bearer`), confirming authentication is enforced on the other routes.

Remediation

Apply the same Bearer-token authentication dependency to `POST /drive/sync` that protects the other routes, so missing or invalid `Authorization` headers return `401` before body validation or handler execution. Where feasible, additionally restrict the endpoint to internal callers or to an allowlist of service identities, since Drive sync is an administrative operation. Independently, stop returning internal environment/configuration state in error responses: the unauthenticated `503` currently discloses the exact environment-variable name (`GDRIVE_SERVICE_ACCOUNT_JSON`) and its configured/unconfigured status, replace it with a generic message (e.g. `Drive sync unavailable`) so anonymous callers cannot learn deployment internals.

All findings ← VLX-10 VLX-12 → Open in Vulnix

MEDIUM VLX-11

WhatsApp OTP verification lacks lockout and backoff. 4-digit OTP brute force with user enumeration (portal.demo-edubridge.example)

AFFECTED LOCATION		
/functions/v1/send-whatsapp-otp and /functions/v1/verify-otp-v2		
CVSS V3.1 6.5 Medium	STATUS Open	FOUND 4 Aug 2026
WEAKNESS CWE-307 · Excessive Authentication Attempts		
OWASP TOP 10:2025 A07 Authentication Failures		GROUP Authentication and sessions

Description

The WhatsApp-OTP login flow on portal.demo-edubridge.example (Supabase backend) allows a remote, unauthenticated attacker to brute-force the 4-digit login OTP. The verify endpoint enforces only a per-OTP 5-attempt budget with no permanent lockout and no exponential backoff, and the send endpoint reveals which phone numbers are registered (user enumeration). A determined attacker can iterate the full 10,000-code space over ~2,000 send+verify batches (~25-33 hours, thousands of WhatsApp messages to the victim), enabling account takeover including admin accounts whose phone numbers are readable with the public anon key.

Reproduction steps

1. Send an OTP to a registered student phone (`send-whatsapp-otp` with `{"identifier":"+1-202-555-0114","userType":"student"}`) and observe HTTP 200 `{"success":true,...,"expiresIn":600}`.
2. Submit a wrong code (`verify-otp-v2` with `otpCode:"0000"`) and observe HTTP 400 with `attemptsRemaining:4`.
3. Submit another wrong code (`otpCode:"1111"`) and observe HTTP 400 with `attemptsRemaining:3`, the per-OTP budget decrements on each guess.
4. Immediately re-send the OTP to the same phone and observe HTTP 429 (send-side throttle exists, bounding the brute-force rate).
5. Wait out the cooldown window (~60s), send again, and submit a third wrong code, observe the `attemptsRemaining` counter (reset to 4 if a fresh OTP was minted, demonstrating that the 5-attempt budget is re-armed by re-sends; no permanent lockout in either case).
6. Send an OTP to a synthetic, unregistered phone and observe HTTP 404, proving the user-enumeration oracle (no WhatsApp message is dispatched for the synthetic number).
7. Optionally call `check-user-status` for a registered vs unregistered phone to confirm the differential enumeration response.

Remediation

Enforce strict rate limits on both the send and verify endpoints, keyed per identifier AND per IP, with exponential backoff and a permanent lockout after a threshold of failed verifications for an identifier. Increase OTP entropy (e.g. 6+ digit codes or a TOTP/push-based flow) so the code space is not brute-forceable within the OTP validity window. Apply the same throttling to `check-user-status` and remove the differential user-enumeration oracles (uniform responses for registered and unregistered phones at send time, and for pending vs expired OTPs at verify time). Ensure the send cooldown is enforced per-phone as well as per-IP so distributed attackers cannot bypass it, and add alerting on repeated OTP requests for a single identifier.

All findings ← VLX-11 VLX-13 → Open in Vulnix

MEDIUM VLX-12

Unauthenticated Grafana /metrics endpoint exposes internal monitoring data

AFFECTED LOCATION `/metrics`		
CVSS V3.1 5.3 Medium	STATUS Open	FOUND 4 Aug 2026
WEAKNESS CWE-668 · Exposure of Resource to Wrong Sphere		
OWASP TOP 10:2025 A01 Broken Access Control		GROUP Data exposure

Description

The Grafana instance at https://grafana.apps.demo-edubridge.example exposes its Prometheus metrics endpoint (`/metrics`) without any authentication. An unauthenticated attacker can retrieve the full internal telemetry payload, including the exact Grafana version, build revision, alerting configuration, and internal HTTP handler performance data.

Reproduction steps

1. Send an unauthenticated GET request to `https://grafana.apps.demo-edubridge.example/metrics`.
2. Observe that the server responds with HTTP 200 and a body of 487,116 bytes of Prometheus text metrics.
3. Search the response for `grafana_build_info` and `grafana_alerting_alertmanager_receivers` to confirm the leak of version, revision, and alerting configuration data.

Remediation

Set `basic_auth_username` and `basic_auth_password` in the `[metrics]` section of `grafana.ini` (or set `enabled = false` if the metrics endpoint is not consumed by a monitoring stack), then restart Grafana. Alternatively, block `/metrics` at the reverse proxy (Traefik) using an IP allowlist so only the monitoring infrastructure can reach it. Monitor access logs for unauthorized `/metrics` requests and rotate the Grafana admin credentials if there is any sign of prior exposure.

All findings ← VLX-12 VLX-14 → Open in Vulnix

MEDIUM VLX-13

Unauthenticated Student Account Enumeration via send_verification Endpoint

AFFECTED LOCATION		
`/api/v1/student/registrations/send_verification`		
CVSS V3.1 5.3 Medium	STATUS Open	FOUND 4 Aug 2026
WEAKNESS CWE-204 · Observable Response Discrepancy		GROUP Authentication and sessions

Description

The unauthenticated student registration endpoint `POST /api/v1/student/registrations/send_verification` discloses whether a given Demo User 86327ef8 or phone number is already registered. Unknown identifiers return `200 {"status":"success"}`, while registered emails or phones return `422` with distinct messages (`Email already exist`, `Student with phone number already exists`), creating a remote enumeration oracle that requires no authentication and is not rate limited.

Reproduction steps

1. Generate a random, unregistered Demo User 86327ef8 and a random unregistered regional mobile phone number (`+20` prefix followed by 11 digits).
2. Send an unauthenticated `POST` request with a JSON body `{"email": "...", "phone": "..."}` to `/api/v1/student/registrations/send_verification`.
3. Observe the `200` success response for the unknown identifiers.
4. Replace one identifier with a registered Demo User 86327ef8 (or phone number) and re-send the request; observe the `422` response with the distinct `Email already exist` or `Student with phone number already exists` message.
5. Repeat across candidate identifiers to enumerate registered accounts; no rate limiting is enforced.

Remediation

Return a single uniform success response regardless of whether the submitted Demo User 86327ef8 or phone number already exists, and defer all duplicate-identifier validation until after the verification step completes (for example, after OTP confirmation) so the pre-authentication endpoint reveals nothing about account existence. Remove the distinct error strings (`Email already exist`, `Student with phone number already exists`) and use one generic message, ensuring identical status codes and response timing across all identifier states. Enforce rate limiting and throttling on `send_verification` per source IP and per identifier (email/phone), adding CAPTCHA after a small number of attempts to prevent automated enumeration. As defense in depth, keep all pre-authentication responses (including login) uniform so no endpoint acts as an account-existence oracle.

All findings ← VLX-13 VLX-15 → Open in Vulnix

MEDIUM VLX-14

Object-level authorization bypass (read) on /api/v1/student/projects/{id} exposes internal review metadata of other students' projects

AFFECTED LOCATION `/api/v1/student/projects/{id}`		
CVSS V3.1 4.3 Medium	STATUS Open	FOUND 4 Aug 2026
WEAKNESS CWE-639 · Authorization Bypass Through User-Controlled Key		
OWASP TOP 10:2025 A01 Broken Access Control		GROUP Access control

Description

The student-scoped projects endpoint of the Rails API returns any project referenced by its numeric ID regardless of ownership. Any authenticated student can read internal review metadata. `editable`, `is_reviewed`, `is_late_review`, `views_count`, `tutor_comment`, plus raw database columns of other students' published projects, data that the public project endpoint deliberately omits.

Reproduction steps

1. Register a student account via the public signup flow and capture the resulting API token (the value sent in the `Authorization` header).
2. Send `GET https://api.demo-edubridge.example/api/v1/student/projects/3` with the token in the `Authorization` header.
3. Observe HTTP 200 returning project "more python list operation methods", owned by a different student (id 8498), with internal fields `editable`, `is_reviewed`, `is_late_review`, `views_count`, and `tutor_comment`.
4. Send `GET https://api.demo-edubridge.example/api/v1/public/projects/3` without authentication and compare: the internal fields are absent and `other_projects[]` is sanitized.
5. Repeat with other foreign ids (e.g., 4, 22, 28, 50) to confirm generality, and with an out-of-range id (e.g., 999999) to confirm the endpoint 404s only on non-existent records.

Remediation

Scope the show action to the authenticated student's own projects: load the record through the current student's association (e.g., `current_student.projects.find(id)`) so that foreign ids fail with 404, matching the behavior already enforced on the update action. Alternatively, when a non-owned published project is requested, serve it through the public serializer used by `/api/v1/public/projects/{id}` so internal review fields are never serialized. Add an authorization policy check on the show action as defense-in-depth, and ensure the `other_projects[]` association never exposes raw database columns to student clients.

All findings ← VLX-14 Open in Vulnix

LOW VLX-15

Session tokens never expire and survive logout on app.demo-edubridge.example (client-side only logout)

AFFECTED LOCATION `/api/v1/student/login (and /api/v1/student/account/memberships)`		
CVSS V3.1 3.1 Low	STATUS Open	FOUND 4 Aug 2026
WEAKNESS CWE-613 · Insufficient Session Expiration		
OWASP TOP 10:2025 A07 Authentication Failures		GROUP Authentication and sessions

Description

The user-3c01d32bbd88 portal issues opaque session tokens that never expire and are not invalidated when the user logs out: the SPA clears the token only from browser localStorage and never calls the API. Any session token that leaks (via XSS, shared devices, logs, or the API echoing the token inside account responses) grants indefinite access to the student account that the legitimate user's logout cannot terminate.

Reproduction steps

1. Fetch the 4-digit OTP for the test account from the mail.tm inbox.
2. Log in with `POST /api/v1/student/login` using the email and OTP; capture the returned session token and note that the login response contains no expiry field.
3. Call `GET /api/v1/student/profile` with the token, observe HTTP 200.
4. Call `GET /api/v1/student/account/memberships` with the token, observe HTTP 200 and a live session token in `data…[redacted]`.
5. Call `GET /api/v1/student/profile` with the exposed token, observe HTTP 200, proving the response-embedded token is a valid session credential.
6. Perform the SPA logout (clears localStorage only, issues no HTTP request), then reuse the original token on `GET /api/v1/student/profile`, observe HTTP 200, proving the session survives logout and has no expiry.

Remediation

1. Make the SPA invoke the server-side revocation endpoint (`DELETE /api/v1/student/logout` or an equivalent) on logout so the presented token is invalidated immediately.
2. Issue tokens with an explicit server-enforced expiry (short-lived) and validate expiry on every authenticated request; rotate the token on each login so previously issued tokens are invalidated.
3. Stop returning live session tokens in account-data responses such as `/api…[redacted]`.
4. Implement a server-side session registry supporting revocation on logout and password change, and store session tokens in HttpOnly, Secure cookies rather than localStorage to reduce XSS-based theft.

05 NEXT STEPS

Remediation plan

Now

Critical and high findings.

VLX-01	CRITICAL	Hardcoded Admin API Token and Universal OTP Backdoor on Staging Rails API	`/api/v1/admin/* and /api/v1/admin/login`
VLX-02	CRITICAL	Missing Row Level Security on Supabase backend of portal.demo-edubridge.example (anon key = full database read/write/realtime access)	`https://demo-ba84718a.supabase.co/rest/v1 (and /realtime/v1/websocket)`
VLX-03	CRITICAL	Account takeover via unauthenticated session-token minting in create-session edge function (portal.demo-edubridge.example)	`/functions/v1/create-session`
VLX-04	CRITICAL	PayLink payment bypass: paid-group-PayLink-submit marks lead paid on any receipt image (portal.demo-edubridge.example)	`/functions/v1/paid-group-PayLink-submit (+ /functions/v1/paid-group-create-lead)`
VLX-05	HIGH	Payment bypass: test-mark-paid edge function marks leads paid without payment (portal.demo-edubridge.example)	`/functions/v1/test-mark-paid`
VLX-06	HIGH	Exposed Grafana 11.1.0 with known critical vulnerability (CVE-2024-9264 SQL Expressions command injection)	`/api/health`
VLX-07	HIGH	Price tampering: one-to-one-paid-create-lead persists client-supplied package_amount (0/negative accepted) (portal.demo-edubridge.example)	`/functions/v1/one-to-one-paid-create-lead`
VLX-08	HIGH	Brute-forceable 4-digit OTP login with weak rate limiting and no lockout on api.demo-edubridge.example	`/api/v1/student/login (and /api/v1/student/registrations/verify)`

Next

Medium findings.

VLX-09	MEDIUM	Exposed Portainer CE 2.39.1 with known unpatched vulnerabilities (CVE-2026-44881, CVE-2026-44883)	`/api/status`
VLX-10	MEDIUM	Missing authentication on POST /drive/sync (Tawthiq API)	`/drive/sync`
VLX-11	MEDIUM	WhatsApp OTP verification lacks lockout and backoff. 4-digit OTP brute force with user enumeration (portal.demo-edubridge.example)	`/functions/v1/send-whatsapp-otp and /functions/v1/verify-otp-v2`

VLX-12	MEDIUM	Unauthenticated Grafana /metrics endpoint exposes internal monitoring data `/metrics`
VLX-13	MEDIUM	Unauthenticated Student Account Enumeration via send_verification Endpoint `/api/v1/student/registrations/send_verification`
VLX-14	MEDIUM	Object-level authorization bypass (read) on /api/v1/student/projects/{id} exposes internal review metadata of other students' projects `/api/v1/student/projects/{id}`

Later

Low and info findings.

VLX-15	LOW	Session tokens never expire and survive logout on app.demo-edubridge.example (client-side only logout) `/api/v1/student/login (and /api/v1/student/account/memberships)`

Confirm the fixes

Open a finding in Vulnix and run Validate-Fix. It replays that finding's exploit against the fixed build and records whether it still works. To check the whole scope, run it again and compare the two runs under Reports, Compare.

Open this run in Vulnix · Compare two runs

06 REFERENCE

Appendix

Weakness types found

CWE	NAME	IDS	FINDINGS
CWE-307	Excessive Authentication Attempts	VLX-08, VLX-11	2
CWE-639	Authorization Bypass Through User-Controlled Key	VLX-03, VLX-14	2
CWE-863	Incorrect Authorization	VLX-04, VLX-05	2
CWE-204	Observable Response Discrepancy	VLX-13	1
CWE-306	Missing Authentication for Critical Function	VLX-10	1
CWE-59	Link Following	VLX-09	1
CWE-602	Client-Side Enforcement of Server-Side Security	VLX-07	1
CWE-613	Insufficient Session Expiration	VLX-15	1
CWE-668	Exposure of Resource to Wrong Sphere	VLX-12	1
CWE-78	OS Command Injection	VLX-06	1
CWE-798	Use of Hard-coded Credentials	VLX-01	1
CWE-862	Missing Authorization	VLX-02	1

Report integrity

Report ID	demo-run-edubridge-01
Run	demo-run-edubridge-01
Content fingerprint	`11264935 5bf0d3c4 7539b89c a54df6d2`
Generated	20 Nov 2026 22:05 UTC
Secrets masked in evidence	6

The fingerprint is a SHA-256 digest of the findings' content. Two copies of this report with the same fingerprint describe the same findings.

Glossary

CVSS	Common Vulnerability Scoring System, version 3.1. A 0 to 10 score of a weakness's technical severity.
CWE	Common Weakness Enumeration. The public catalogue of weakness types, maintained by MITRE.
CVE	Common Vulnerabilities and Exposures. The identifier of one published vulnerability in a specific product.
OWASP Top 10	OWASP's ranking of the most common web application risks. This report uses the 2025 edition.
Proof of concept	The steps, requests or script that reproduce a finding against the target.
Validate-Fix	The Vulnix action that replays a finding's exploit after a fix, to confirm the fix works.



Questions about this report

support@vulnix.dev

docs.vulnix.dev