

PENETRATION TEST REPORT

Veridian Commerce checkout and merchant portal

Authorised grey-box assessment of the web application and its supporting API.

ENGAGEMENT	CL-2026-0147
CLIENT	Veridian Commerce Private Limited
TEST WINDOW	06 Oct 2026 – 15 Oct 2026
RETEST	29 Oct 2026
DOCUMENT VERSION	1.2 – post-retest, supersedes 1.0 and 1.1
TESTER OF RECORD	Named in the signed copy issued to the client
DISTRIBUTION	Client named recipients only. Not for onward distribution without written consent.

SPECIMEN DOCUMENT. Veridian Commerce Private Limited does not exist. Every host, finding, request and timestamp in this document is invented for illustration. No client data appears here and no real engagement is described. We publish this because asking a firm for a sample report usually produces a sales deck with one blurred page in it, and you deserve to see the shape of the deliverable before you buy it.

SECTION 01

Scope

Everything below was agreed in writing before any traffic was sent. Nothing outside this list was tested.

IN SCOPE	checkout.veridian.example · portal.veridian.example · api.veridian.example/v2 · merchant onboarding flow
OUT OF SCOPE	Payment processor integration (third party, no authorisation) · corporate email and staff devices · physical premises · any denial of service technique
ENVIRONMENT	Pre-production, seeded with synthetic records, schema matching production at commit 4e1c9ab
APPROACH	Grey-box. Source access granted for the API service. Six accounts issued, one per role.
ROLES ISSUED	shopper · merchant_staff · merchant_admin · support · ops · admin
STANDARDS	OWASP WSTG v4.2 · OWASP API Security Top 10 (2023) · OWASP ASVS 4.0 L2 for code review
EFFORT	8 testing days, 2 reporting days, 1 retest day

SECTION 02

Limitations

The payment processor integration was excluded, so no conclusion is offered about card data handling beyond the point where the application hands off. The test ran against pre-production; three findings below are marked as requiring confirmation against production configuration, because the environments differ in TLS termination and WAF policy.

No denial of service testing was performed. Where the report describes a resource consumption weakness, that assessment is based on reading the code path, not on demonstrating exhaustion.

SECTION 03

Executive summary

One critical finding. The checkout API returns any customer's invoice to any authenticated shopper.

The application is in reasonable shape structurally. Authentication is sound, the session implementation is conventional and correct, and the team has clearly thought about injection. The problems are concentrated in authorisation: the API trusts identifiers supplied by the client in three places, and in one of them the consequence is total exposure of the customer billing table.

A shopper account created through normal public signup can retrieve the invoices of every other customer by changing one integer in a request. That includes name, delivery address, order contents, partial card identifiers and GST registration numbers for business buyers. We stopped at ten records, which was enough to establish the pattern, and did not enumerate further.

1	2
CRITICAL	HIGH
2	1
MEDIUM	LOW
1	
INFORMATIONAL	

Reported early. CL-0147-01 was reported by telephone at 14:20 on 08 Oct 2026, two days into testing, with written detail following within the hour. We do not hold critical findings back for the report.

Findings register

ID	FINDING	SEVERITY	CVSS	OURS	RETEST
CL-0147-01	Invoice endpoint returns records belonging to any customer	CRITICAL	6.5	9.1	CLOSED
CL-0147-02	Role escalation through mass assignment on profile update	HIGH	8.1	8.1	CLOSED
CL-0147-03	Stored script injection in merchant display name reaches support console	HIGH	7.4	7.4	CLOSED
CL-0147-04	Password reset token remains valid after use	MEDIUM	5.3	5.3	CLOSED
CL-0147-05	No rate limit on OTP verification attempts	MEDIUM	5.3	6.8	OPEN
CL-0147-06	Session cookie issued without SameSite attribute	LOW	3.1	3.1	ACCEPTED
CL-0147-07	Framework version disclosed in error responses	INFO	—	—	ACCEPTED

CVSS column is the 3.1 base score. "Ours" is the rating after applying your environment. Where the two differ, the finding explains why.

CRITICAL

CL-0147-01 CVSS 6.5 → RATED 9.1

CLOSED 29 OCT

Invoice endpoint returns records belonging to any customer

AFFECTED

GET /v2/invoices/{invoice_id} on api.veridian.example

Handler: services/billing/handlers/invoice.go:118

REPRODUCTION

Authenticate as any shopper account created through public signup. Place one order to obtain a valid invoice identifier, then decrement it.

```
# 1. our own invoice, as shopper_a - expected
GET /v2/invoices/80419 HTTP/1.1
Host: api.veridian.example
Authorization: Bearer [REDACTED]

HTTP/1.1 200 OK
{"invoice_id":80419,"customer_id":5512,"name":"Test Shopper A", ... }

# 2. decrement the identifier - not our invoice
GET /v2/invoices/80418 HTTP/1.1
Host: api.veridian.example
Authorization: Bearer [REDACTED]

HTTP/1.1 200 OK
{"invoice_id":80418,"customer_id":4471,
 "name":"[REDACTED]","gstn":"[REDACTED]",
 "ship_to":{"line1":"[REDACTED]","city":"Pune"},
 "card_last4":"[REDACTED]","total":"14,990.00"}
```

Specimen capture. Personal fields are masked in this published copy; the client copy carries them in full.

CAUSE

The handler loads the invoice by primary key and returns it. There is no check that the invoice belongs to the authenticated customer. Authentication is verified by middleware, which is why this was not caught: the request is authenticated, it is simply not authorised.

```
// services/billing/handlers/invoice.go:118
inv, err := h.repo.GetInvoice(ctx, invoiceID)
if err != nil { return notFound(w) }
// no ownership check between load and render
return json.NewEncoder(w).Encode(inv)
```

IMPACT

Any person who can complete public signup can read the billing record of every customer in the platform. Identifiers are sequential, so enumeration is trivial and requires no special tooling. At the time of testing the invoice table held approximately 80,400 rows.

Exposed per record: customer name, full delivery address, order contents, order value, last four digits of the payment instrument, and for business buyers the GST registration number. For a consumer marketplace this is a reportable personal data breach, and the combination of name, address and purchase history is directly usable for targeted fraud against your customers.

SEVERITY

CVSS 3.1 base scores this 6.5 – `AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N` – because it requires privileges and affects confidentiality only. That score is calculated without knowing what the data is or how many records exist.

We rate it 9.1. The privilege required is a free account anyone can create in under a minute, which is not meaningfully a privilege. The identifiers are sequential, so the whole table is reachable. We disagree with the base score and have recorded both so your auditor sees the standard calculation and your engineering lead sees the number that should decide priority.

REMEDIATION

Scope the query to the authenticated principal rather than filtering after load. Change the repository call at `invoice.go:118` to take the customer identifier from the session context:

```
inv, err := h.repo.GetInvoiceForCustomer(ctx, invoiceID, auth.CustomerID(ctx))
if err != nil { return notFound(w) }
```

Return `404` rather than `403` so the endpoint does not confirm which identifiers exist.

The same pattern appears at `orders.go:74` and `shipments.go:91`; both are recorded as CL-0147-02 and should be fixed in the same change. Longer term, move ownership scoping into the repository layer so a handler cannot load a record it is not entitled to.

SECTION 05

Retest and closure

Retested 29 Oct 2026 against the same scope. Four closed, one open, two accepted.

CL-0147-01

CLOSED Ownership scoping moved into the repository layer. Re-ran the original enumeration across 200 sequential identifiers; all returned **404**. The sibling paths at **orders.go** and **shipments.go** were fixed in the same change and retested.

CL-0147-05

OPEN A rate limit was added at the gateway but applies per source address, and the OTP endpoint is reachable through a mobile client pool that shares egress addresses. We could still submit 300 attempts in 60 seconds from a single session. Our rating stays at 6.8. The client has scheduled a per-account limiter for the following sprint.

CL-0147-06

ACCEPTED Client accepts the risk pending a cookie rework already in progress. Recorded here rather than removed, so the decision is visible to whoever reads this next year.

What this document is not

It is not a certificate and it does not say the application is secure. It says what was tested, across which surfaces, to what depth, within a fixed window, and what was found. A retest confirms the listed findings were closed; it does not revisit the whole scope and it cannot speak to code shipped afterwards.

On accepted risk. Two findings above are marked accepted rather than closed. We leave them in the register permanently. A report that quietly deletes what the client chose not to fix is worth less than no report at all, because the next person to read it will believe the work was finished.

Document control

V1.0	17 Oct 2026 – issued after testing
V1.1	21 Oct 2026 – CL-0147-04 severity revised down after client evidence of compensating control
V1.2	29 Oct 2026 – retest results and closure statement added
EVIDENCE	Destroyed per the retention period in the signed scope, with written confirmation to the client