TECHNICAL DUE DILIGENCE REPORT

Audit of Northgate Payments API (demonstration codebase)

Reference **SAMPLE-NORTHGATE-PAYMENTS** · 2026-11-23

1. Summary

16 findings in product code (9 high, 4 medium, 3 low risk). Estimated cost to fix: $18,550 – $43,400, as engineering hours at $175/hour. What each estimate covers is listed with the finding. 2 further observations in test, documentation or example code are listed separately and not priced.

RISK	ESTIMATED COST TO FIX
9 high 4 medium 3 low	**$18,550 – $43,400**
How serious each finding could be. Separate from the money.	Engineering hours at $175/hour. Not a valuation.

QUADRANT	FINDINGS	COST TO FIX
Architecture & code	4	$4,025 – $9,450
Security & compliance	8	$7,000 – $17,850
Infrastructure & cloud	3	$525 – $2,100
Team & delivery	1	$7,000 – $14,000

How to read the figures. The cost to fix is the engineering work to remove each finding: published hours for each kind of finding at $175 an hour, shown as a range. It is not a valuation or a price adjustment, and it does not include the cost of an incident; the risk rating describes that. The hours for every kind of finding are published at veridence.agiphi.com/methodology.

2. Findings

Ordered by risk, then by cost. A finding that occurs in several places is listed once with every place.

FINDING	WHY IT MATTERS AND WHAT FIXING INVOLVES	COST TO FIX
HIGH **Restrictive copyleft licensing (GPL-3.0)** third_party/reportgen/package.json	Copyleft code inside the product can oblige the company to publish its own source code. **To fix:** Establish where the licensed code is used, then remove or replace it, or confirm the obligations are met. Legal review is not included. *Evidence (third_party/reportgen/package.json): The manifest's license field declares GPL-3.0.*	**$2,800 – $7,000** 16–40 hours
HIGH **Copyleft dependency: mysqlclient 2.2.4 (GPL-2.0)** requirements.txt	A copyleft production dependency can oblige the company to publish its own source code. **To fix:** Replace the package with a permissively licensed alternative, or obtain a commercial licence. Legal review is not included. *Evidence (requirements.txt): PYPI package mysqlclient 2.2.4 is licensed GPL-2.0 according to the package registry. It is a production dependency.*	**$1,400 – $4,200** 8–24 hours
HIGH **AWS access key ID in the code** app/settings.py	A recognisable credential in the code can be used by anyone who reads it. **To fix:** Revoke and rotate the credential, move it to a secrets store, remove it from the code and its history, and check access logs for misuse. *Evidence (app/settings.py): AWS_ACCESS_KEY_ID = "AKIA••••"*	**$700 – $1,400** 4–8 hours

FINDING	WHY IT MATTERS AND WHAT FIXING INVOLVES	COST TO FIX
HIGH **Hardcoded secret / API credential** app/settings.py	Anyone who can read the code, or its history, can use the credential. **To fix:** Revoke and rotate the credential, move it to a secrets store, remove it from the code and its history, and check access logs for misuse. *Evidence (app/settings.py): line 8: payment_gateway_secret = "ngp_••••"*	**$700 – $1,400** 4–8 hours
HIGH **Vulnerable dependency: pyyaml 5.3** requirements.txt	A published high-severity vulnerability is a known way into the system until the package is upgraded. **To fix:** Upgrade the package to a fixed version and re-test the parts of the system that use it. *Evidence (requirements.txt): 2 high-severity advisories for PYPI package pyyaml 5.3: CVE-2020-1747 (CVSS 9.8): Improper Input Validation in PyYAML CVE-2020-14343 (CVSS 9.8): Improper Input Validation…*	**$350 – $1,400** 2–8 hours
HIGH **Access policy allows every action ('*')** infra/iam-policy.json	A wildcard policy lets a compromised service do anything the account can do. **To fix:** Replace the wildcard with the specific actions and resources the service needs, and test the deployment. *Evidence (infra/iam-policy.json): "Action": "*",*	**$350 – $1,050** 2–6 hours
HIGH **Command run through the system shell** app/reports.py	Crafted input can extend the command to run anything the service account may run. **To fix:** Pass arguments as a list without the shell, or validate the input against an allow-list, and add tests. *Evidence (app/reports.py): line 67: subprocess.run(f"tar czf /var/archive/{directory}.tgz {directory}", shell=True)*	**$350 – $700** 2–4 hours
HIGH **Network ingress open to the whole internet (0.0.0.0/0)** infra/main.tf	Infrastructure settings that expose a service, run it with excess privilege or store data unencrypted widen the damage of any breach. **To fix:** Change the configuration, apply it to a staging environment and confirm the service still works. *Evidence (infra/main.tf): cidr_blocks = ["0.0.0.0/0"]*	**$175 – $700** 1–4 hours
HIGH **SQL query built from variables (injection risk)** app/server.py	Crafted input can change the statement that runs, exposing or destroying data. **To fix:** Replace string-built queries with parameterised queries and add tests with hostile input. *Evidence (app/server.py): line 8: cursor.execute(f"SELECT id, name, email FROM customers WHERE id = {customer_id}")*	**$350 – $700** 2–4 hours
MEDIUM **Key-person concentration**	When one person writes most of the code, their departure takes the working knowledge of the system with them. **To fix:** Document the system and pair the principal contributor with others to spread working knowledge. A retention arrangement is a commercial matter and is not included. *Evidence (repository): m.okafor made 80% of the 40 commits in the 12 months to 2026-11-24. Bus factor 1. Commit entropy 0.884 bits across 3 contributors.*	**$7,000 – $14,000** 40–80 hours
MEDIUM **High-complexity functions** app/reports.py	Functions with many branches are hard to test and change safely, which slows delivery and raises the risk of regressions. **To fix:** Split the functions into smaller ones and add tests: 0.5 to 1 hour for each point of complexity above 15. *Evidence (app/reports.py): statement_line() complexity 43 line 5: def statement_line(entry, currency, locale, include_tax): 1 function above complexity 15, 28 points over in total.*	**$2,450 – $4,900** 14–28 hours

FINDING	WHY IT MATTERS AND WHAT FIXING INVOLVES	COST TO FIX
MEDIUM **Container image runs as root (no USER set)** Dockerfile	Infrastructure settings that expose a service, run it with excess privilege or store data unencrypted widen the damage of any breach. **To fix:** Change the configuration, apply it to a staging environment and confirm the service still works. *Evidence (Dockerfile): FROM python:3.12-slim WORKDIR /srv*	**$175 – $700** 1–4 hours
MEDIUM **Storage volume encryption explicitly disabled** infra/main.tf	Infrastructure settings that expose a service, run it with excess privilege or store data unencrypted widen the damage of any breach. **To fix:** Change the configuration, apply it to a staging environment and confirm the service still works. *Evidence (infra/main.tf): encrypted = false*	**$175 – $700** 1–4 hours
LOW **Very large source file** app/billing/ledger.py	Very large files are slow to understand and risky to change. **To fix:** Split the file along its responsibilities and re-test. *Evidence (app/billing/ledger.py): File contains 1805 lines.*	**$1,400 – $4,200** 8–24 hours
LOW **Exception handler that discards the error** · 2 places app/reports.py, web/src/api.js	Errors that are caught and thrown away hide failures, so problems surface late and are hard to trace. **To fix:** Log or handle each error instead of discarding it. *Evidence (app/reports.py): line 75: except Exception:*	**$175 – $350** 1–2 hours

3. In test, documentation or example code

Listed for completeness and not priced: a placeholder key in a test fixture is not a flaw in the product.

FINDING	FILE	WHERE
Hardcoded secret / API credential	tests/test_server.py	test
SQL query built from variables (injection risk)	tests/test_server.py	test

4. Remediation plan: the first 100 days

Built only from the findings in this report: high-risk work first.

Days 1–30: high risk

- Isolate or replace the copyleft-licensed code in the repository.
- Replace or commercially license each copyleft production dependency.
- Revoke, rotate and remove the credentials found in the code.
- Upgrade the dependencies with high-severity advisories, and re-test.
- Narrow the wildcard access policies to what each service needs.
- Stop passing input through the system shell.
- Fix the infrastructure misconfigurations.
- Parameterise the SQL queries built from variables.

Days 31–60: medium risk

- Spread working knowledge of the system beyond the principal contributor.
- Split the most complex functions and add tests.

Days 61–100: low risk

- Split the very large source files along their responsibilities.
- Log or handle the errors that are currently discarded.

5. How this report was produced

Veridence read one exact commit of the repository in memory. Python, JavaScript and TypeScript were parsed into syntax trees and checked by the structure of the code; infrastructure files, access policies and other languages were checked by fixed text patterns. Licences were read from what the repository declares, and each pinned dependency's licence and published advisories were looked up in the public package registry. Twelve months of commit history give the team measures. The analysis is deterministic: there is no AI model, and the same commit always gives the same report. The code was discarded when the audit ended. Static analysis cannot show how the system behaves in production or under load, and finding nothing is not proof that a system is secure.