Delvant

Example report. Pellwick is a company we made up, and so are its systems and these findings.

Security scan report: Pellwick, November 2026

For	Pellwick. A company we made up. It sells invoicing software to small businesses.
Asset scanned	The customer web application and its API, at app.pellwick.example
How we tested	From the internet, as a visitor and as two customers, with two accounts that Pellwick opened for the scan and its written permission to test.
Not tested	Taking the service down with load, the staff side of the application behind its sign-in, Pellwick's office network, and tricking its staff.
Findings	1 High, 1 Medium, 1 Low

Summary

Of the 12 parts of the application we tested, 9 held up and 3 have a finding: 1 High, 1 Medium, 1 Low.

Fix finding 1 first. Until it is fixed, every customer's invoices are open to every other customer. The fix is the same check on the server in each place that takes an invoice number.

Finding 2 can wait for your next release. Finding 3 needs no release: take the staging site offline.

What we tested

Sign-in	We tried to guess passwords and to sign in without one. Wrong guesses are slowed down after five tries, and the error does not say whether an account exists.	No finding
Sessions	We checked how a session starts, ends and expires. The cookie is kept from scripts and sent only over HTTPS, and signing out ends the session on the server.	No finding
Password reset	We asked for reset links and tried to guess, alter and reuse them. A link that has been used still works.	**Finding 2: Medium**
File upload	We sent the logo upload files of the wrong kind, oversized files and files with misleading names. Each was refused, and uploaded files are served from a separate address.	No finding
Invoice API	As one test customer we asked for, and changed, the other's invoices by their numbers. The API allowed both.	**Finding 1: High**
Search	We put database and script syntax into the search box and into its API. All of it was treated as plain text: neither the database nor the page ran it.	No finding
Admin pages	We looked for staff pages and asked for them as a customer. Each answered "not found" without a staff session.	No finding
TLS	We checked the certificate, the protocol versions and the ciphers. Only TLS 1.2 and 1.3 are offered, and HTTP redirects to HTTPS.	No finding
Old staging site	We looked for forgotten copies of the application. One is online, at staging.pellwick.example.	**Finding 3: Low**
Response headers	We read the security headers on every kind of response. A content security policy, HSTS and a framing rule are set.	No finding
Libraries	We compared the JavaScript libraries the pages load with published advisories. None has a known flaw that can be reached here.	No finding

Subdomains	We listed Pellwick's subdomains from public records and checked each. Apart from the staging site, none is forgotten or points at a service someone else could claim.	No finding

How we rate severity

Each finding has a base score from 0 to 10 on CVSS 3.1, a public scale for security flaws. The score is worked out from the finding's vector, which records how the flaw is reached, what it takes to use it and what it gives an attacker. The score sets the word.

Critical	9.0 to 10.0
High	7.0 to 8.9
Medium	4.0 to 6.9
Low	0.1 to 3.9

The score says how serious a flaw of this kind is anywhere. Under Impact, each finding says what it means for this application.

Findings

High severity, 7.1 of 10

The invoice API shows other customers' invoices

CVSS 3.1	Base score 7.1, from the vector `CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:L/A:N` Reachable over the network · Low attack complexity · Low privileges required · No user interaction · High confidentiality impact · Low integrity impact
Where	`GET and PATCH /api/v1/invoices/{number}`

What we found

The API checks that a request comes from a signed-in customer, but not that the invoice belongs to that customer. Invoice numbers run in sequence, so another customer's invoice is one number away. The PDF download and the payment reminder take the same number and have the same gap.

Evidence

Signed in as the first test customer, account 2041, we asked for invoice 88213, which belongs to the second test customer, account 2042.

```
GET /api/v1/invoices/88213 HTTP/1.1
Host: app.pellwick.example
Cookie: session=[test customer one]

HTTP/1.1 200 OK
Content-Type: application/json

{
  "number": 88213,
  "account": 2042,
  "issued_by": "Test Customer Two",
  "billed_to": "Test Client, 14 Test Road, Pune 411001",
  "client_tax_number": "[the test client's]",
  "lines": "[what was sold, line by line]",
  "total": "18400.00",
  "currency": "INR",
  "status": "unpaid"
}
```

A PATCH request to the same address that set the status to "paid" was accepted too, and the invoice then showed as paid in the second customer's account. We read and changed only the invoices of our two test accounts, and set the invoice back afterwards.

Impact

A signed-in customer can read any other customer's invoices by changing one number in the request.

An invoice carries the name, address and tax number of a business and of its client, and what was sold. Anyone can open a trial account, so in practice anyone can read every invoice Pellwick holds, and can mark any of them paid or unpaid.

Fix	1. On every request for an invoice, check on the server that the invoice's account is the signed-in customer's, and answer "not found" when it is not. 2. Make the same check on the PDF download and the payment reminder, and wherever else a request carries an invoice number. 3. Add an automated test that asks for another account's invoice and expects "not found".

Medium severity, 6.8 of 10

Password reset links can be used more than once

CVSS 3.1	Base score 6.8, from the vector `CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:H/I:H/A:N` Reachable over the network · High attack complexity · No privileges required · User interaction required · High confidentiality impact · High integrity impact
Where	`POST /api/v1/password-reset/confirm`
What we found	A reset link carries a token, which should stop working once a password has been set with it. It keeps working until it expires, 24 hours after it was sent.

Evidence	We asked for a reset link for the first test account and set a new password with it. Forty minutes later we sent the same token again with another password.
	
	`POST /api/v1/password-reset/confirm HTTP/1.1`
	`Host: app.pellwick.example`
	`Content-Type: application/json`
	
	`{`
	`  "token": "[the token of the used link]",`
	`  "new_password": "[a second new password]"`
	`}`
	
	`HTTP/1.1 200 OK`
	
	The second password then signed us in. A day later the same token was refused.

Impact	Anyone who gets hold of an old reset email can set a new password and take over that account. Reset emails are forwarded, and are kept by mail gateways and backups. The attacker needs one that a customer has already used, within 24 hours of its sending: that is why this is Medium and not High.
Fix	1. Mark the token as used in the same step that sets the password, and refuse a used token. 2. Shorten the token's life from 24 hours to one hour.

Low severity, 3.7 of 10

An old staging site is still online

CVSS 3.1	Base score 3.7, from the vector `CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N` Reachable over the network · High attack complexity · No privileges required · No user interaction · Low confidentiality impact
Where	`staging.pellwick.example`
What we found	It is a copy of the application with its own sign-in page. No page links to it; we found it in the public record of certificates issued for Pellwick's domain. Its responses name the software it runs and its version.

Evidence	We asked the staging site and the live application for their sign-in pages.
	`GET /sign-in HTTP/1.1`
	`Host: staging.pellwick.example`
	
	`HTTP/1.1 200 OK`
	`Server: nginx/1.18.0`
	`X-Powered-By: Express`
	`X-App-Version: 2025.03.2`
	
	The live application is on version 2026.10.4, and its responses name neither its server nor its framework. Our test accounts do not exist on the staging site, so we did not test behind its sign-in.

Impact	It runs an out-of-date copy of the application, which anyone who finds its address can reach. Fixes made to the live application since March 2025 are missing from this copy, and it tells an attacker which software Pellwick runs: nginx 1.18.0 is an old release with published flaws, which we noted and did not try. If it holds a copy of real customer data, that data is only as safe as the old copy. By itself the site gives away little, which is why it scores the lowest of the three.
Fix	1. Take the staging site offline if nobody uses it. 2. If it is needed, put it behind a VPN or a list of allowed addresses, keep it on the live version, and fill it with made-up data.