SECURITY AUDIT REPORT

[company] — redacted portfolio version

Joshua Wetzel
225-305-9321
josh@jawetzel.com

Non-Intrusive Public Information Review
Early 2026

Redaction notice

This is a redacted portfolio version of an original security audit deliverable. To protect the identity of the audited organization, the following redactions have been applied:

- Identifiers replaced with placeholders: `[company]`, `[domain]`, `[hostname]`, `[customer]`, `[staff]`, `[system]`, `[address]`.
- Admin URL paths → `[dashboard-N]` (numbered 1–14). AJAX endpoint names → `[endpoint-N]`.
- Industry-specific product categories generalized to "widgets."
- Live API keys and credentials cut entirely.
- Dollar amounts, counts, and scale figures swapped for fakes of the same order of magnitude. A "~$180K past-due" in this report corresponds to a real past-due of similar scale; exact figures are not preserved.
- Specific dates generalized to "Early 2026."

Every finding is real and was observed on the live site.

Table of Contents

1. Executive Summary
2. Internal Pages Exposed — Detail
3. Data Leaks — Detail
4. What We Looked At

1. Executive Summary

Date: Early 2026 · **Type:** Non-intrusive public information review

What was exposed

Using nothing but a web browser — no credentials, no insider knowledge — the following was publicly accessible on [company]'s website:

- **Warehouse inventory layouts** showing where widgets, consumable widgets, and hazardous widgets were stored, with real-time headcount of who was on each floor.
- **60+ employee names with direct phone extensions**, which is everything a social-engineering attacker needs to impersonate staff.
- **A list of customers, their contacts, and their financial information**, served to the public both through the web app and through a public file-storage bucket.
- **Live inventory counts for ~45,000 products**, enough for a competitor to track stock levels in real time.
- **~8,500 invoices, ~2,900 payment records, and the wholesale cost of every product in the catalog.**

All of it reachable with no login and no authentication check.

How it happened

Three patterns accounted for most of the findings:

1. **Internal dashboards reachable without a login.** Pages built for employees carried no authentication check, and search engines had indexed some of them, so they surfaced in public results.
2. **Customer documents in public storage with guessable URLs.** Statement PDFs were generated on demand and parked on a public bucket with the customer ID in the filename, so one URL gave away the pattern for every other customer's file.
3. **Server responses that contained the data the UI hides.** The screen showed "—" and "hidden," while the underlying response shipped to the browser in full.

2. Internal Pages Exposed — Detail

Each of the pages below was reachable by anyone on the public internet, with no login and no authentication check. Data on each page loaded automatically the moment the page opened.

Operations dashboard (HIGH)

Live operational data for the entire warehouse, refreshed in real time.

Metric	What was showing
Open order lines	~3,840
Open invoices	~1,720
Invoiced lines — today	~10,450
Invoiced lines — yesterday	~13,210
Packages shipped (7 days)	~54,800
Same-day shipping rate	~77%

Also visible: real invoice numbers, a hot-replenishment inventory table (~95 rows), and an endpoint that generates invoice PDFs to public storage.

Accounts receivable dashboard (HIGH)

Full dealer receivables visible in the page data. White text on a white background hid the data from a casual glance, but all of it was present in the underlying page and visible to anyone who right-clicked and chose "inspect."

Dealer accounts — 30+ days past due:

Customer	Salesperson	AR rep	Past due	Last payment
[customer]	[staff]	[staff]	**~$178,000**	2024
[customer]	[staff]	[staff]	**~$167,000**	2020
[customer]	[staff]	[staff]	~$32,000	2024

Collections + public file storage (HIGH)

From this page, clicking a group, then a customer, then "view statement" causes the site to generate a PDF of that customer's full financial statement and drop it on a **public** file-storage bucket. The URL uses the customer's account number in the filename, so stepping through numbers retrieves every customer's statement.

Per customer, the page revealed:

- Business name and account number
- Full physical address
- Email addresses
- Payment history and dates
- Receivables aging (current, 30-, 60-, 90-day balances)
- Assigned salesperson and AR representative

Web payment portal (HIGH)

Opening the page automatically loaded ~2,900 customer payment records (roughly 1 megabyte of data). Per record: customer name, payment amount, timestamp, transaction identifier.

Payment receipt URLs also referenced an internal server name — useful information for an attacker mapping the network.

Accounts payable invoices (HIGH)

Opening the page automatically loaded:

- ~410 **vendor records** — the complete list of companies [company] buys from
- ~8,500 **invoice headers** — purchase order numbers, vendor names, invoice dates, status

For a competitor, this is the complete purchasing relationship map: who the vendors are, how often orders go in, and with what cadence.

Phone dashboard (MEDIUM)

Every employee on the phone system, by full name, with:

- Direct extension
- Real-time call availability (are they on a call right now?)
- Call volume metrics

Sales queue (MEDIUM)

Real-time sales call data for named employees, auto-refreshing every 7 seconds. Visible per person: completed calls, missed calls, average call duration. Also visible: a rolling sale-items table showing product descriptions and SKUs moving through the sales floor.

Organizational dashboard (MEDIUM)

The warehouse floor, stations, and live headcount.

Building layout shown:

- **Building A — Order processing:** picking, packing, quality-control stations
- **Building B — Accessory:** receiving, put-away, returns, and a locked-inventory cage
- **Building C — Regulated inventory:** stations handling hazardous and regulated stock
- **Universal:** general warehouse, management, cleaning, safety, maintenance

Real-time headcount: ~84 clocked in / ~62 clocked out at time of review.

The physical-security implication is that a would-be thief can look up where regulated inventory is stored and when the fewest people are on the floor, without any physical reconnaissance.

3. Data Leaks — Detail

Two separate code paths on [company]'s public site returned sensitive pricing and inventory data to any anonymous visitor, with no login required.

Wholesale cost + live inventory leaked through the product pages (HIGH)

When any visitor opened a product page, the server sent back the full record for that product — including wholesale cost, manufacturer-advertised price (MAP), and live inventory count. The visible page showed "COST: N/A" and "RETAIL: N/A" in the product details area, but the underlying data was present in full.

Nine products were sampled across different categories and [manufacturer]s. In every case, the response included:

Data point	What was exposed
Wholesale cost	Present
MAP (where enforced)	Present
MSRP (where set)	Present
Live inventory count	Actual count (not limited)
UPC code	Full
Manufacturer SKU	Full

The pattern held for product categories across the catalog (~45,000 products). Product IDs were sequential, meaning a simple script could walk through the entire catalog and collect wholesale cost + live inventory for every item in a single evening.

Business impact

- **Wholesale cost exposure.** Competitors could see [company]'s exact wholesale pricing across the entire catalog, which is among the most sensitive data in a distribution business.
- **Supply-chain visibility.** Live inventory counts let competitors track stockouts, anticipate re-ups, and price against shortages in real time.
- **Scripting-friendly.** Product IDs were sequential, so walking the catalog required no guesswork and only a few lines of code.

Same data leaked in every search, category, and department page (HIGH)

Every product search page, category page, department page, and [manufacturer] page embedded a JavaScript array in the page itself with ~23 data points per product — visible to any anonymous visitor.

Per product, the page revealed:

Field	Risk
Manufacturer-advertised price (MAP)	Competitor pricing intelligence
Sale price	Wholesale pricing intelligence
Three inventory quantity fields	Supply-chain intelligence
MAP-enforced flag	Pricing-restriction visibility
UPC codes	Catalog cross-referencing
Manufacturer SKUs	Direct [manufacturer] identification

Scale

Page type	Data exposed per page load
A single department page (widgets)	~5,140+ MAP prices
Any search result	MAP prices and inventory for the entire result set
Category group page	MAP prices, inventory up to ~1,620 units per item

4. What We Looked At

What Was Covered

Area	Coverage
HTTP Response Headers	Full
TLS/SSL Configuration	Full
Homepage & Page HTML Source	Full
JavaScript Variables & Logic	Full
Cookies & Attributes	Full
robots.txt / XML Sitemap	Full
Admin Endpoint Accessibility	Full (19 endpoints)
Product Search/Detail/Category/Department Pages	Full
AJAX Endpoint Discovery	Full (8 page types)
CORS Policy	Full
Admin Dashboard Data Exposure	Full (14 dashboards)
Normal User Flow Data Exposure	Full (11 page types)
Product Detail Data Exposure	Full (9 products)

Total unique pages/endpoints examined: 76+

What Was NOT Covered

Area	Reason
Authenticated session behavior	No login credentials available
API endpoint fuzzing/enumeration	Requires authorization
SQL injection / XSS testing	Requires authorization
File/directory brute-forcing	Requires authorization
Rate limiting / DoS resilience	Requires authorization
Subdomain enumeration	Out of scope

Area	Reason
Email security (SPF/DKIM/DMARC)	Out of scope
Mobile app / API analysis	Out of scope
Payment processing (PCI compliance)	Requires authorized assessment
Source code review	No access to server-side source code

Methodology

All findings were obtained through **passive browser-equivalent observation**:

- Visiting pages as an anonymous visitor would
- Reading source code of pages served to anonymous visitors
- Observing AJAX responses triggered by normal UI interaction (clicking published buttons)
- Reviewing HTTP headers, cookies, and TLS configuration

No credentials were used. No fuzzing, brute-forcing, injection, or enumeration was performed. No write operations were attempted — several endpoints *appeared* to support destructive actions (creating/deleting records, modifying customer data) but none were invoked.

The scoping was deliberate. A zero-knowledge review has a different legal and ethical posture than an authorized penetration test, so the report is limited to what could be observed from a browser.