Web Application Security Assessment

Sample Report — Company names and sensitive details redacted

Target	app.example-saas.com
Scope	Main app + all subdomains + APIs
Assessment	Standard Web Application Pentest
Date	May 2026
Prepared by	SecureAudit · info@appsecaudit.io
Version	1.0 — Final

Executive Summary

SecureAudit conducted a web application security assessment of **app.example-saas.com** between May 19–23, 2026. The assessment covered authentication, API endpoints, cloud storage, JavaScript bundles, CORS policy, and all discovered subdomains.

A total of **6 findings** were identified: 1 High, 2 Medium, 2 Low, and 1 Informational. The most critical issue — production API keys exposed in the JavaScript bundle — requires immediate remediation as the keys were verified to be active.

Severity	Count	Status
CRITICAL	0	—
HIGH	1	Immediate action required

MEDIUM	2	Fix within 30 days
LOW	2	Fix within 90 days
INFO	1	Informational

Remediation Summary

#	Finding	Severity	Priority
1	Production API keys in JS bundle	HIGH	Immediate
2	CORS misconfiguration — reflects any origin	MEDIUM	30 days
3	Unauthenticated write access to S3 bucket	MEDIUM	30 days
4	Verbose server version disclosure	LOW	90 days
5	Missing SRI on third-party scripts	LOW	90 days
6	Subdomain enumeration (informational)	INFO	Informational

Detailed Findings

Finding 1 HIGH

Production API Keys Exposed in JavaScript Bundle

CVSS	7.5	URL	https://app.example-saas.com/static/main.chunk.js

Description

The React SPA bundles hardcoded production API keys directly into the compiled JavaScript file served to all unauthenticated visitors. Keys for the internal feature-flag service (LaunchDarkly SDK key) and a third-party routing API were identified by scanning the minified bundle.

Impact

Any visitor can extract valid API keys. The LaunchDarkly key allows an attacker to read and modify feature flags for all users, potentially enabling hidden features or disabling security controls. The routing API key incurs direct financial cost per request.

Proof of Concept

```
# Key extracted from bundle (key rotated before report delivery): grep -oP 'sdk-[a-f0-9\-]{36}' main.chunk.js # Returns: sdk-a3f1c2d4-[REDACTED] # Verified active: curl -H "Authorization: sdk-[REDACTED]" \ https://app.launchdarkly.com/api/v2/flags/production # HTTP 200 — returns all feature flags
```

Remediation

Move all API keys to server-side environment variables. The frontend should request feature flags via your own backend endpoint which holds the SDK key. Rotate all exposed keys immediately. Add a secrets scanner (truffleHog, gitleaks) to your CI pipeline to prevent future exposure.

Finding 2 **MEDIUM**

CORS Misconfiguration — Management API Reflects Any Origin

CVSS	5.4	URL	https://api.example-saas.com/v2/

Description

The management API reflects arbitrary Origin headers in the Access-Control-Allow-Origin response header and includes Access-Control-Allow-Credentials: true. This combination allows cross-origin requests from any attacker-controlled domain on behalf of an authenticated user.

Impact

An attacker who tricks an authenticated user into visiting a malicious page can make authenticated API calls to the management API — reading account data, modifying settings, or performing actions as the victim. This is a browser-based attack requiring user interaction.

Proof of Concept

```
curl -s -H "Origin: https://attacker.com" \ -H "Cookie: session=[victim_session]" \
https://api.example-saas.com/v2/account \ -I | grep -i "access-control" # Response: #
Access-Control-Allow-Origin: https://attacker.com # Access-Control-Allow-Credentials: true
```

Remediation

Maintain an explicit allowlist of trusted origins. Never reflect arbitrary Origin values. If credentials are required, each allowed origin must be listed explicitly — wildcards cannot be used with credentials. Review all /v2/ endpoints for the same pattern.

Finding 3 **MEDIUM**

Unauthenticated Write Access to S3 Log Bucket

CVSS	5.3	URL	https://logs-prod-example.s3.amazonaws.com/

Description

The S3 bucket used for application logs accepts anonymous PUT requests from the internet. The bucket ACL was misconfigured during a recent infrastructure migration, leaving it world-writable. Files written are also publicly readable via GET.

Impact

An attacker can upload arbitrary files (including malware or phishing pages) to a domain trusted by the application. If downstream systems process log files (e.g., a SIEM ingesting S3 events), a log injection or file parsing attack becomes possible.

Proof of Concept

```
# Unauthenticated upload succeeds: curl -s -X PUT
"https://logs-prod-example.s3.amazonaws.com/test.txt" \ --data "secureaudit-probe" -w
"%{http_code}" # Returns: 200 # File publicly readable: curl -s
"https://logs-prod-example.s3.amazonaws.com/test.txt" # Returns: secureaudit-probe
```

Remediation

Set the bucket ACL to private. Enable S3 Block Public Access at the account level. Use IAM roles with least-privilege write permissions for the application. Enable S3 server-side access logging and set up an alert for unexpected PutObject events from non-application principals.

Finding 4 **LOW**

Verbose Server Version Disclosure

CVSS	3.1	URL	https://app.example-saas.com/

Description

Multiple HTTP response headers disclose exact server software versions: Server: nginx/1.24.0, X-Powered-By: PHP/8.1.12, and X-AspNet-Version headers are returned on all responses. The nginx version is below the current stable release and has published CVEs.

Impact

Version disclosure reduces attacker effort when targeting known CVEs. The disclosed nginx version (1.24.0) is affected by CVE-2023-44487 (HTTP/2 rapid reset). While exploitation requires additional conditions, removing the header costs nothing.

Proof of Concept

```
curl -sI https://app.example-saas.com/ | grep -iE 'server:|x-powered|x-asp' # Server: nginx/1.24.0
# X-Powered-By: PHP/8.1.12
```

Remediation

Set 'server_tokens off' in nginx.conf. Remove X-Powered-By via php.ini (expose_php = Off) or at the application framework level. Update nginx to the current stable release (1.26+). These changes have zero functional impact.

Finding 5 **LOW**

Missing Subresource Integrity on Third-Party Scripts

CVSS	2.6	URL	https://app.example-saas.com/

Description

Three third-party JavaScript files are loaded without Subresource Integrity (SRI) attributes. If any of these CDN endpoints were compromised, malicious code would execute in the context of your application with no browser-side detection.

Impact

Low practical risk — CDN compromise is rare. However, adding SRI is a one-line change per script that completely mitigates supply-chain script injection for these specific assets.

Proof of Concept

```
# Current tag (no integrity attribute): # Should be:
```

Remediation

Generate SRI hashes for all third-party scripts using srihash.org or the openssl CLI. Add integrity and crossorigin attributes to each script tag. Automate hash generation in your build pipeline to keep them current across version updates.

Finding 6 **INFO**

Subdomain Enumeration — Attack Surface Identified

CVSS	0.0	URL	*.example-saas.com

Description

Subdomain enumeration identified 14 active subdomains. All are correctly configured and no takeover candidates were found. Documented for completeness as part of the attack surface map.

Impact

Informational. No exploitable condition identified. The discovered subdomains (staging, api, cdn, mail, docs, status, admin, dev, beta, app, dashboard, portal, webhook, metrics) represent the full internet-facing attack surface.

Proof of Concept

```
subfinder -d example-saas.com -silent | httpx -silent # 14 live subdomains discovered # 0 CNAME pointing to unclaimed services # 0 dangling DNS records
```

Remediation

No immediate action required. Ensure any subdomains hosting staging or dev environments require authentication. Remove DNS records for decommissioned services promptly. Consider running periodic subdomain enumeration as part of your internal security process.

End of report. All findings were communicated to the client within the agreed timeline. This sample has been redacted for public distribution — company names, URLs, and key values have been replaced with placeholders.

SecureAudit · info@appsecaudit.io · appsecaudit.io