



CYRAVEX SECURITY

OFFENSIVE INSIGHT. DEFENSIVE STRENGTH.

SAMPLE DELIVERABLE / WEB + API PENETRATION TEST

Web & API Penetration Test Report

Illustrative client-facing report demonstrating CyraVex Security reporting quality, risk communication, technical evidence structure, and remediation guidance.

CLASSIFICATION

PUBLIC SAMPLE

No real client data

ASSESSMENT

WEB + API

Authorized test model

VERSION

1.0

September 2026

Fictional target only. Example hostnames use reserved test domains. No production system was tested.



00 / SAMPLE NOTICE

About this illustrative report

This document is a public sample created to demonstrate how Cyravex Security can communicate penetration-test results. It is not evidence of a real client engagement, and every target, hostname, IP address, user, finding, and outcome is fictional.

DOCUMENT CONTROL

PREPARED FOR	Illustrative Client
TARGET	Example SaaS Platform
PRIMARY HOST	portal.example.test
VERSION	1.0 / Public Sample
CLASSIFICATION	Sanitized / Illustrative

IMPORTANT LIMITATIONS

- Findings are examples designed to show report structure and depth.
- No assurance or guarantee of security is implied by this sample.
- A real report is tailored to the authorized scope, evidence, business context, and agreed rules of engagement.
- Real evidence is handled according to client-approved retention and access rules.

REPORTING PRINCIPLE

Findings should help engineering and leadership teams understand what happened, why it matters, and what to do next - not simply list scanner output.



Report structure

01	Executive summary	04
02	Risk at a glance	05
03	Assessment overview and scope	06
04	Rules, assumptions and constraints	07
05	Methodology	08
06	Severity framework	09
07	Findings summary	10
08	Detailed findings	11-17
09	Positive observations	18
10	Attack-path narrative	19
11	Remediation roadmap	20
12	Retest and validation	21
13	Evidence handling	22
14	Standards mapping	23
15	Conclusion	24



01 / EXECUTIVE SUMMARY

Security posture requires prioritized remediation

The illustrative assessment identified several weaknesses that could allow an authenticated or remote attacker to access data outside their intended authorization boundary, influence backend behavior, or increase the likelihood of account abuse. The sample risk posture is rated HIGH because multiple high-severity issues affect core application and API trust boundaries.

OVERALL RISK

HIGH

Priority remediation

FINDINGS

8

3 high / 3 medium

HIGH SEVERITY

3

Authorization / injection / SSRF

RETEST

RECOMMENDED

Validate closure

KEY RISK THEMES

- Authorization: object-level access decisions must be enforced server-side for every sensitive resource.
- Input and query safety: untrusted parameters should never reach database or internal network sinks without strong controls.
- Abuse resistance: authentication and recovery workflows need rate limits, lockouts, monitoring, and secure defaults.
- Browser trust boundaries: CORS, output encoding, and response headers should reduce opportunities for client-side abuse.

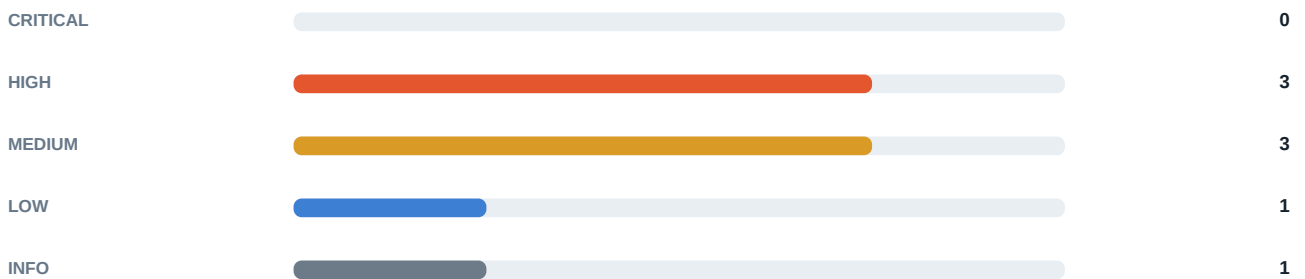
LEADERSHIP PRIORITY

Fix the three high-severity findings first, then close medium-severity abuse and browser-control gaps. Schedule a focused retest before treating the risks as closed.



Findings distribution and risk themes

SEVERITY DISTRIBUTION



Identity & Access

BOLA/IDOR demonstrates missing object-level authorization checks.

Server Trust

SQL injection and SSRF show unsafe transitions from user input to trusted backend resources.

Abuse & Browser Controls

Rate limiting, CORS, headers, and output encoding need stronger defaults.



03 / ASSESSMENT OVERVIEW

Objectives, authorized scope, and exclusions

This sample assumes a time-boxed, authorized black/gray-box test of an internet-facing SaaS application and its documented API. In a real engagement, scope and methods are confirmed in writing before testing begins.

IN SCOPE

- portal.example.test
- api.example.test/v1
- Two provided test accounts
- Authentication and authorization flows
- Business logic and API abuse cases
- Common web and API vulnerabilities

OUT OF SCOPE

- Denial-of-service or stress testing
- Social engineering and phishing
- Production customer accounts
- Third-party services not owned by the client
- Destructive exploitation
- Source-code review unless separately authorized

OBJECTIVE

Identify exploitable weaknesses, validate realistic impact safely, provide reproducible evidence, prioritize remediation, and give engineering teams clear validation criteria for closure.



04 / RULES & CONSTRAINTS

Safe testing boundaries

AUTHORIZATION

Written authorization, named assets, agreed test window, and approved contacts are mandatory.

SAFETY

No destructive actions, service disruption, persistence, or uncontrolled data extraction.

ESCALATION

Critical or actively exploitable issues are escalated through agreed emergency contacts.

EVIDENCE

Only the minimum evidence needed to demonstrate risk is collected and retained securely.

ASSUMPTIONS

- Test accounts represent ordinary application users unless otherwise noted.
- Reserved example domains and IPs are used throughout this sample.
- Risk ratings may change when business context or compensating controls are confirmed.
- A penetration test is a point-in-time assessment and cannot prove the absence of vulnerabilities.



05 / METHODOLOGY

Manual-led, tool-supported testing workflow

CyraVex combines structured methodology, manual abuse-case analysis, and targeted tooling. Automated scanners may help discover signals, but findings are manually validated before they are reported.

- 01 DISCOVERY**
Understand architecture, accounts, roles, APIs, and business flows.
- 02 MAPPING**
Map attack surface, endpoints, trust boundaries, and data flows.
- 03 TESTING**
Perform manual and tool-assisted testing against agreed scenarios.
- 04 VALIDATION**
Confirm exploitability, impact, root cause, and affected assets.
- 05 QA**
Peer-review severity, evidence, wording, and remediation quality.
- 06 READOUT**
Deliver report, walk through priorities, and support remediation planning.

REFERENCE FRAMEWORKS

OWASP Web Security Testing Guide • OWASP API Security Top 10 • OWASP ASVS concepts • CWE • CVSS • NIST SP 800-115 concepts



06 / SEVERITY FRAMEWORK

Risk ratings combine technical severity and business context

Cyravex does not treat a CVSS number as the entire risk decision. Severity considers exploitability, privileges, user interaction, data sensitivity, affected functionality, blast radius, compensating controls, and realistic business impact.

CRITICAL

Immediate or near-immediate compromise of critical systems/data with little constraint.

HIGH

Serious compromise, sensitive data exposure, or major trust-boundary failure.

MEDIUM

Meaningful security weakness requiring additional conditions or with more limited impact.

LOW

Limited direct impact but improves defensive depth or reduces attack surface.

INFORMATIONAL

Observation or hardening opportunity without direct demonstrated exploitability.

SEVERITY CAN CHANGE

If the client provides verified compensating controls, data classification, environment context, or exploit constraints, Cyravex may adjust the final business-risk rating.



07 / FINDINGS SUMMARY

Eight illustrative findings

F-01 Broken object-level authorization in invoice API

HIGH

CVSS 8.1 | CWE-639 | OWASP API1:2023

F-02 SQL injection in reporting filter

HIGH

CVSS 8.6 | CWE-89 | OWASP A03:2021

F-03 Server-side request forgery in webhook test feature

HIGH

CVSS 8.2 | CWE-918 | OWASP A10:2021

F-04 Stored cross-site scripting in support ticket metadata

MEDIUM

CVSS 6.1 | CWE-79 | OWASP A03:2021

F-05 Insufficient rate limiting on password-reset verification

MEDIUM

CVSS 6.5 | CWE-307 | OWASP A07:2021

F-06 Overly permissive CORS on authenticated API routes

MEDIUM

CVSS 5.8 | CWE-942 | OWASP A05:2021

F-07 Missing browser hardening headers

LOW

CVSS 3.1 | CWE-693 | OWASP A05:2021

F-08 Verbose framework and error disclosure

INFO

CVSS N/A | CWE-209 | OWASP A05:2021

Prioritization: F-01 through F-03 should be treated as the first remediation wave because each crosses a core authorization, data, or server-side trust boundary.



F-01 / DETAILED FINDING

Broken object-level authorization in invoice API

HIGH

CVSS 8.1 | CWE-639 | OWASP API1:2023

ISSUE SUMMARY

A low-privilege user could request invoice objects belonging to another tenant by changing the object identifier in an authenticated API request. The server verified that the requester was logged in but did not consistently verify ownership of the requested object.

BUSINESS IMPACT

An attacker with a valid account could access other customers' invoice metadata and potentially sensitive billing information. In a multi-tenant SaaS platform, this violates tenant isolation and can create material confidentiality and contractual risk.

ILLUSTRATIVE REQUEST

```
GET /v1/invoices/84219 HTTP/1.1
Authorization: Bearer [test-token]
Host: api.example.test
```

ILLUSTRATIVE RESPONSE

```
HTTP/1.1 200 OK
Content-Type: application/json
{"invoice_id":84219,"tenant":"Tenant-B","total":"[redacted]"}
```

REMEDIATION

- Enforce object-level authorization on every request, not only at the UI layer.
- Derive tenant/user authorization context server-side and include it in database queries.
- Use centralized authorization middleware or policy checks to reduce inconsistent enforcement.
- Add automated negative tests that verify users cannot access objects owned by another tenant.



F-02 / DETAILED FINDING

SQL injection in reporting filter

HIGH

CVSS 8.6 | CWE-89 | OWASP A03:2021

ISSUE SUMMARY

The reporting endpoint passed a user-controlled filter value into a database query without safe parameterization. Manual validation demonstrated that query behavior changed based on crafted input. The exact test payload is intentionally omitted from this public sample.

BUSINESS IMPACT

Successful exploitation could allow unauthorized reading or modification of application data depending on database permissions and query context. The issue also increases the risk of service disruption or credential/material exposure if the database account is over-privileged.

ILLUSTRATIVE REQUEST

```
POST /v1/reports/search HTTP/1.1
Content-Type: application/json

{"filter":"[test payload omitted]"}
```

ILLUSTRATIVE RESPONSE

```
HTTP/1.1 500 Internal Server Error
{"error":"database query failed","trace_id":"[sample]"}
```

REMEDIATION

- Use parameterized queries / prepared statements for every user-controlled value.
- Remove dynamic SQL construction where possible and use allow-listed query options.
- Run the application with a least-privileged database account.
- Add SAST/DAST checks and targeted regression tests for injection-sensitive endpoints.



F-03 / DETAILED FINDING

Server-side request forgery in webhook test feature

HIGH

CVSS 8.2 | CWE-918 | OWASP A10:2021

ISSUE SUMMARY

The webhook-test feature accepted an arbitrary URL and caused the backend service to issue an outbound request. The control did not sufficiently restrict internal, loopback, link-local, or cloud-metadata destinations.

BUSINESS IMPACT

An attacker with access to the feature could use the application as a network pivot to reach services that are not directly exposed to the internet. Depending on cloud/network architecture, this can lead to internal service discovery, metadata access, or credential exposure.

ILLUSTRATIVE REQUEST

```
POST /v1/webhooks/test HTTP/1.1
Content-Type: application/json

{"url":"http://192.0.2.10/example"}
```

ILLUSTRATIVE RESPONSE

```
HTTP/1.1 200 OK
{"status":"request attempted","upstream":"[sample]"}
```

REMEDIATION

- Use an allow-list of approved webhook schemes, hostnames, and ports where feasible.
- Resolve and validate destinations server-side, blocking loopback, private, link-local, and metadata ranges before every request and redirect.
- Disable unnecessary redirects and revalidate each redirect target.
- Apply egress controls so the application cannot access sensitive internal networks or metadata services.



F-04 / DETAILED FINDING

Stored cross-site scripting in support ticket metadata

MEDIUM

CVSS 6.1 | CWE-79 | OWASP A03:2021

ISSUE SUMMARY

A support-ticket metadata field was stored and later rendered in an administrative interface without context-appropriate output encoding. A crafted value could therefore execute script in a privileged user's browser when the ticket was viewed.

BUSINESS IMPACT

Exploitation could allow actions in the context of a support/admin user, including reading page-accessible data or issuing same-origin requests. Impact depends on browser protections, session controls, and the privileges of the viewing user.

ILLUSTRATIVE REQUEST

```
POST /v1/tickets HTTP/1.1
Content-Type: application/json

{"subject":"[encoded test marker]"}
```

ILLUSTRATIVE RESPONSE

```
HTTP/1.1 201 Created
{"ticket_id":"T-10042","status":"created"}
```

REMEDIATION

- Apply context-aware output encoding at render time.
- Sanitize rich-text fields using a well-maintained allow-list sanitizer.
- Avoid rendering user-controlled HTML unless it is required.
- Use a restrictive Content-Security-Policy as defense in depth, not as the primary fix.



F-05 / DETAILED FINDING

Insufficient rate limiting on password-reset verification

MEDIUM

CVSS 6.5 | CWE-307 | OWASP A07:2021

ISSUE SUMMARY

The password-reset verification endpoint accepted repeated incorrect verification attempts without an effective per-account attempt threshold, cooldown, or progressive delay.

BUSINESS IMPACT

An attacker could automate attempts against short-lived verification codes, increasing the probability of unauthorized account recovery. Risk is higher for accounts with weak recovery controls or longer verification-code validity.

ILLUSTRATIVE REQUEST

```
POST /v1/auth/reset/verify HTTP/1.1
Content-Type: application/json

{"user":"sample@example.test","code":"000000"}
```

ILLUSTRATIVE RESPONSE

```
HTTP/1.1 400 Bad Request
{"error":"invalid code"}
[Repeated attempts continued without effective throttling]
```

REMEDIATION

- Apply per-account and per-source attempt limits with progressive delay or temporary lockout.
- Use sufficiently strong, short-lived, single-use verification values.
- Invalidate old codes when a new code is issued.
- Alert on abnormal recovery attempts and add device/risk context where appropriate.



F-06 / DETAILED FINDING

Overly permissive CORS on authenticated API routes

MEDIUM

CVSS 5.8 | CWE-942 | OWASP A05:2021

ISSUE SUMMARY

Authenticated API responses reflected untrusted Origin values and allowed credentialed cross-origin requests on routes that return user-specific data.

BUSINESS IMPACT

A malicious website visited by an authenticated user could potentially cause the browser to issue credentialed requests and expose readable responses to an attacker-controlled origin, depending on session and browser behavior.

ILLUSTRATIVE REQUEST

```
GET /v1/profile HTTP/1.1
Origin: https://attacker.example.test
Cookie: session=[sample]
```

ILLUSTRATIVE RESPONSE

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: https://attacker.example.test
Access-Control-Allow-Credentials: true
```

REMEDIATION

- Allow-list only trusted application origins and reject all others.
- Do not combine broad/dynamic origins with Access-Control-Allow-Credentials.
- Apply CORS policy centrally and test sensitive routes explicitly.
- Where possible, use same-site session controls and CSRF protections as complementary defenses.



F-07 + F-08 / DETAILED FINDINGS

Defense-in-depth and information-disclosure observations

LOW

F-07 Missing browser hardening headers

Several responses lacked one or more recommended browser hardening controls, such as a restrictive Content-Security-Policy and clear framing policy. These omissions do not automatically create an exploit but reduce defense in depth.

REMEDIATION

Define a tested CSP, use frame-ancestors to prevent unwanted framing, set X-Content-Type-Options, Referrer-Policy, and other headers appropriate to the application. Roll out CSP carefully to avoid breaking legitimate application behavior.

INFORMATIONAL

F-08 Verbose framework and error disclosure

Certain error responses disclosed framework/version information and internal exception context. The data did not directly enable compromise in this sample, but it can improve attacker reconnaissance and make targeted exploitation easier.

REMEDIATION

Return generic client-facing errors, keep detailed diagnostics in protected server-side logs, suppress unnecessary version banners, and verify that production builds disable debug behavior.



08 / POSITIVE OBSERVATIONS

Controls that reduced risk during the sample assessment

A useful penetration-test report should identify what is working as well as what needs remediation. Positive observations help preserve effective controls during change and provide context for prioritization.

● MFA AVAILABLE

Multi-factor authentication was available for administrative users in the illustrative environment.

● SECURE TRANSPORT

HTTPS was enforced for tested public application routes.

● COOKIE BASICS

Session cookies used Secure and HttpOnly attributes in the sample flow.

● ROLE SEPARATION

Distinct user/admin roles existed, even though some object-level authorization checks were inconsistent.

● LOGGING SIGNALS

Unique request/trace identifiers appeared in several application errors, supporting investigation workflows.

● DEPENDENCY HYGIENE

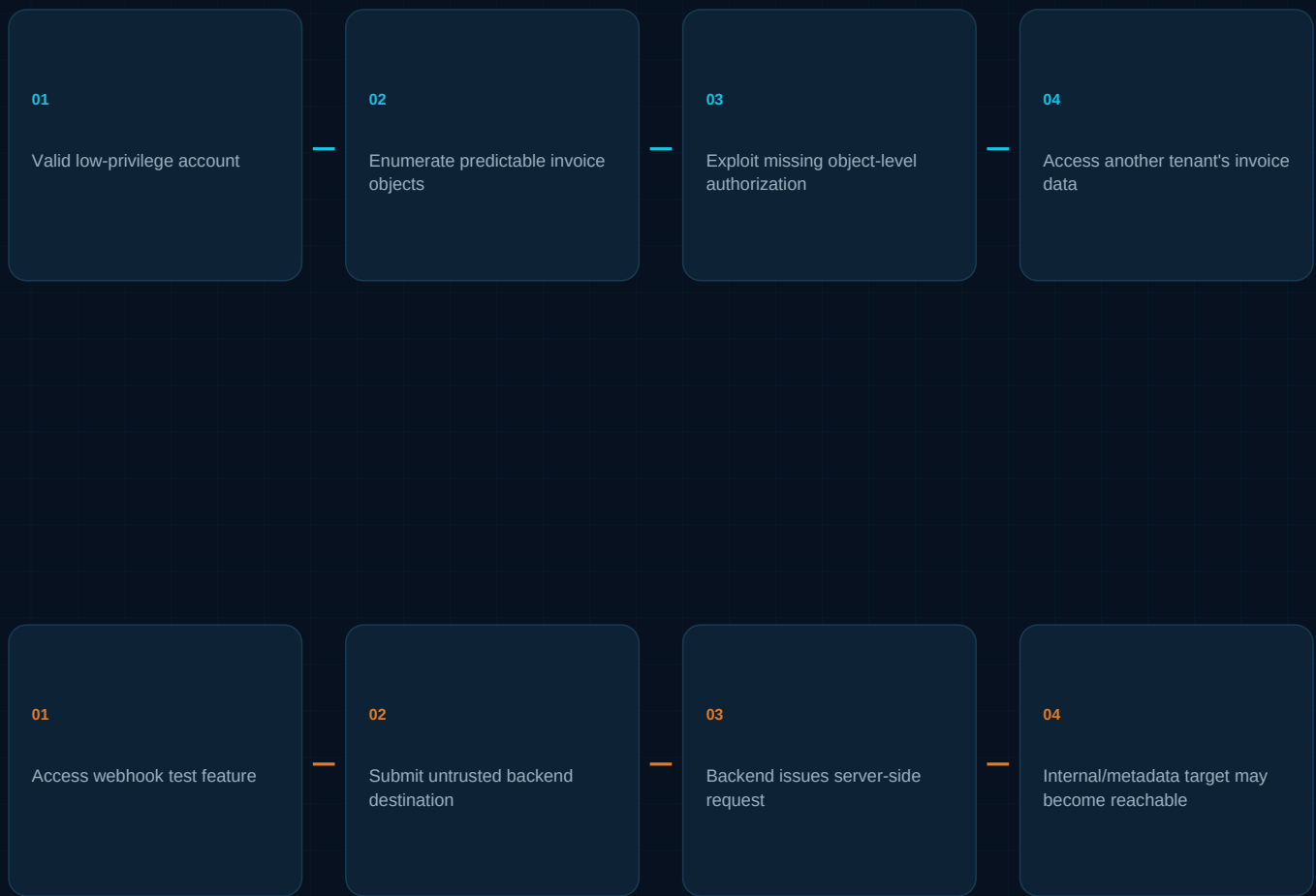
No obviously obsolete client-side framework was identified during the limited sample review.



09 / ATTACK-PATH NARRATIVE

How separate weaknesses can compound business risk

Individual findings should not always be viewed in isolation. The illustrative paths below show how a determined attacker could combine trust-boundary failures and abuse weaknesses to increase impact.



Risk principle: Fix trust-boundary failures at the root cause. WAF rules, UI restrictions, or obscurity should not be treated as substitutes for server-side authorization, safe query construction, or egress control.



Prioritize root causes, then validate closure

0-7 DAYS

F-01 Authorization

Implement object-level authorization centrally; review similar API resources.

0-7 DAYS

F-02 Injection

Parameterize queries, least-privilege DB account, regression tests.

0-14 DAYS

F-03 SSRF

Destination allow-list / IP validation / egress controls / redirect validation.

0-30 DAYS

F-04 to F-06

Output encoding, rate limiting, CORS policy, monitoring.

0-45 DAYS

F-07 to F-08

Browser hardening headers and production error hygiene.

AFTER FIXES

Focused retest

Validate each fix and test adjacent endpoints for regression/generalization.

Recommended ownership: assign a named engineering owner and target date for each finding. Track compensating controls explicitly rather than treating them as permanent fixes by default.



11 / RETEST & VALIDATION

Closure requires evidence, not status changes

A retest should confirm that the original exploit path no longer works and that the fix has not simply moved the weakness to an adjacent endpoint or role. CyraVex records a clear retest status and supporting evidence for each finding.

FIXED

Original issue is no longer reproducible under the agreed retest conditions.

PARTIALLY FIXED

Risk is reduced but the root cause or a related path remains.

NOT FIXED

Original condition remains reproducible or remediation is ineffective.

EXAMPLE RETEST RECORD

FINDING

F-01 - Broken object-level authorization

STATUS

FIXED (illustrative)

VALIDATION

Cross-tenant object requests returned 403/404 and tenant context was enforced server-side.

REGRESSION

Adjacent invoice/export endpoints tested with both supplied roles.

EVIDENCE

Sanitized request/response pair and tester notes retained with the engagement record.



Minimum necessary evidence, protected throughout the engagement

Security evidence can contain session identifiers, personal data, internal paths, stack traces, or business-sensitive records. Cyravex treats evidence as controlled engagement data rather than ordinary document content.

COLLECT MINIMALLY

Capture only what is needed to prove the issue and support remediation.

SANITIZE OUTPUTS

Redact tokens, credentials, PII, customer records, and unnecessary secrets from the report.

RESTRICT ACCESS

Limit engagement evidence to authorized personnel and approved storage locations.

RETENTION

Follow agreed retention/deletion periods; remove evidence when no longer required.

ESCALATE SENSITIVE DATA

If unexpected sensitive data is encountered, stop/limit collection and notify the client contact.

PUBLIC SAMPLES

Use fictional or fully sanitized data. Never repurpose client evidence without explicit permission.



13 / STANDARDS MAPPING

Illustrative mapping for remediation and governance teams

Finding	CWE	OWASP	Primary control theme
F-01 BOLA	CWE-639	API1:2023	Object-level authorization
F-02 SQLi	CWE-89	A03:2021	Parameterized queries / least privilege
F-03 SSRF	CWE-918	A10:2021	Destination validation / egress control
F-04 Stored XSS	CWE-79	A03:2021	Output encoding / sanitization
F-05 Rate limiting	CWE-307	A07:2021	Abuse resistance / authentication
F-06 CORS	CWE-942	A05:2021	Origin policy / browser trust
F-07 Headers	CWE-693	A05:2021	Security configuration
F-08 Error disclosure	CWE-209	A05:2021	Production error handling

REFERENCES

OWASP Web Security Testing Guide • OWASP API Security Top 10 (2023) • OWASP Top 10 (2021) • Common Weakness Enumeration (CWE) • Common Vulnerability Scoring System (CVSS) • NIST SP 800-115 concepts.

Framework mappings are supporting references, not certifications or compliance attestations.



From findings to measurable risk reduction

A penetration test is most valuable when the output is usable by both leadership and engineering. Cyravex reporting is designed to connect technical evidence to business impact, root cause, prioritized remediation, and objective retest criteria.

MANUAL-LED

Validate real attack paths rather than forwarding raw scanner output.

BUSINESS-AWARE

Explain why the issue matters in the client's operating context.

ACTIONABLE

Give engineering teams specific remediation and validation guidance.

VERIFIABLE

Retest fixes and record clear closure status.

REQUEST A SECURITY ASSESSMENT

Web + API Penetration Testing • AI / LLM Security Assessments • Application Security • Security Advisory

cyravex.com | asad@cyravex.com