Penetration Test Report

Demo Organization

Target included:

brokencrystals

June 8, 2026

Table of Contents

Executive Summary

Confidentiality Statement

This report is confidential and contains proprietary information of the customer and OX Security. Except as expressly permitted below, no reproduction, distribution, disclosure, or other use of this report, in whole or in part, may be made without the prior written consent of both the customer and OX Security. Notwithstanding the foregoing, the customer may use this report for its internal security, remediation, governance, audit, compliance, legal, insurance, diligence, and customer assurance purposes, and may disclose this report to its auditors, regulators, external legal and professional advisers, insurers, consultants, financing sources, actual or prospective investors or acquirers, and actual or prospective customers or business partners, in each case with a legitimate need to know and who are bound by confidentiality obligations or professional duties of confidentiality. The customer remains responsible for any authorized disclosure of this report by persons to whom it provides access. OX Security retains all right, title, and interest in and to its testing methodologies, tools, templates, know-how, and report format embodied in this report.

Conclusion

A total of 174 findings were identified across 4 assessed targets.

Assessment Overview

OX Security conducted a white-box agentic penetration test for “Demo Organization” on June 8, 2026. The table below summarizes the scope and outcome for each target in this assessment.

#	TARGET	TYPE	ENVIRONMENT	AUTH	VULNERABLE / SUB-TARGETS	FINDINGS BY SEVERITY
1	brokencrystals https://brokencrystals.com/	WEB	production	Simple Auth	33/87 37.9%	1C 9H 61M 84L
2	OX/brokencrystals https://brokencrystals.com	API	production	Simple Auth	3/1613 0.2%	9M 10L
3	Code Security / JavaScript / vulnerable / NodeGoat https://brokencrystals.com/	API	production	Simple Auth	3/1613 0.2%	1C 9H 61M 84L

Executive Summary

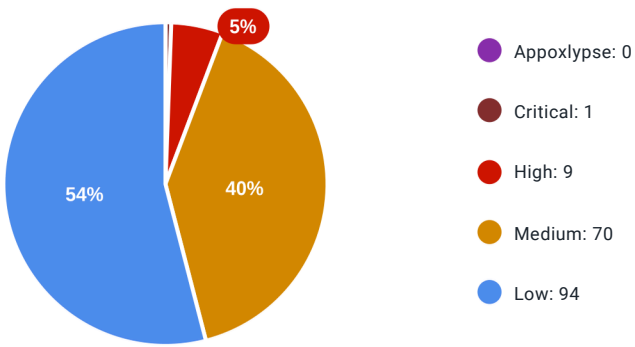
Assessment Overview

#	TARGET	TYPE	ENVIRONMENT	AUTH	VULNERABLE / SUB-TARGETS	FINDINGS BY SEVERITY
4	Code Security / Java / vulnerable / FaceBroke https://brokencrystals.com/	API	production	Simple Auth	-	none
Total across 4 targets					39/3313 1.2%	2C 18H 131M 178L

Testing Framework

The assessment was conducted in alignment with the OWASP Top 10 (2025) framework, covering all 10 vulnerability categories across all discovered endpoints.

Severity Distribution



Assessment Statistics

Findings Distribution by OWASP API Top 10

The following table maps identified findings to the OWASP API Top 10 framework using the same attack-category aggregation as the Agentic Pentester view.

OWASP API Category	ID	Findings	Severity Breakdown
Broken Object Level Authorization	API1	0	-
Broken Authentication	API2	0	-
Excessive Data Exposure Matched: Sensitive Data Disclosure, Information Disclosure	API3	9	1C/3H/4M/1L
Lack of Resources & Rate Limiting	API4	0	-
Broken Function Level Authorization	API5	0	-
Mass Assignment	API6	0	-
Security Misconfiguration Matched: Open Redirect, Cryptography & Transport, Clickjacking, MIME Sniffing	API7	6	2H/1M/3L
Injection Matched: Cross-Site Scripting (XSS)	API8	1	1L
Improper Assets Management	API9	0	-
Insufficient Logging & Monitoring	API10	0	-

Findings

This section presents all identified findings ordered by severity, from most critical to lowest. Each finding includes a full technical description and remediation recommendations.

Sensitive Data Disclosure • JSON Configuration • No Auth Required • Admin Credentials Exposed

Property	Value
Severity	Critical
OWASP Category	Agentic Pentester
Affected Endpoints	https://brokencrystals.com/config.json
Target	*brokencrystals.com
API Route	GET /config.json

Description:

JSON configuration files (e.g., config.json) often contain sensitive operational data, API keys, or cloud credentials. Exposing these files provides a map of internal system architecture to attackers.
Impact
- Full System Compromise: Exposed cloud provider keys allow attackers to take over infrastructure.
- Information Disclosure: Reveals sensitive internal business rules and third-party integration secrets.
Example Payload
GET /config.json GET /settings.json

Sensitive Data Disclosure • JSON Configuration • No Auth Required • Admin Credentials Exposed

Attack Recap:

We requested https://brokencrystals.com and the server returned a JSON configuration file. Observing application/json confirms the file is publicly accessible. Configuration JSON files frequently contain database connection strings, API keys for payment processors and cloud services, internal service URLs, SMTP credentials, and OAuth client secrets. A single exposed config file can give an attacker direct access to every external service the application integrates with.

Remediation Recommendations:

1. Block access to all JSON configuration files. Move these files outside the web root, enforce least-privilege access, and rotate any credentials found within.

   Secure Implementation
   - Store operational configuration files in directories that are not served by the web engine.
   - Return generic 403/404 responses for any fuzzing attempts on common config names.
   - Rotate cloud keys or API tokens discovered in exposed configuration artifacts.

   Best Practices
   - Never name sensitive files using common patterns like config.json or db.json in the web root.
   - Never hardcode credentials directly in JSON files; use placeholder variables instead.
   - Configure your firewall to block specific JSON path patterns.
   - Migrate JSON-based secrets to a dedicated secret management service.

Open Redirect • No Auth Required • Client Side Execution

Property	Value
Severity	High
OWASP Category	Agentic Pentester
Affected Endpoints	https://brokencrystals.com/api/file?path=http://google.com&type=http://google.com
Target	*brokencrystals.com
API Route	GET /api/file?path=http://google.com&type=http://google.com

Description:

An external redirect occurs when an application accepts user-controlled input and uses it as a destination for an HTTP redirect. Without validation, an attacker can craft links that look internal but redirect victims to malicious sites.
Impact
- Phishing: Using your domain's reputation to trick users into visiting malware-hosting sites.
- Bypassing Filters: Some tools trust links from known domains; open redirects allow malicious links to bypass them.
- Token Theft: Sensitive tokens in the URL might be leaked to an attacker's server during redirects.
Example Payload
https://your-app.com/login?redirect_url=https://malicious-phishing-site.com/login

Attack Recap:

We sent a request to https://brokencrystals.com/api/file with a redirect payload http://google.com. We observed the server responding with a 'Location' header pointing to the external domain we specified. This confirms an Open Redirect vulnerability, which can be used to facilitate phishing attacks.

Open Redirect • No Auth Required • Client Side Execution

Remediation Recommendations:

1. Prevent open redirects by validating user-provided destinations against a hardcoded whitelist of allowed internal paths.

   Secure Implementation
   - Maintain a static list of permitted internal paths and verify that the requested redirect matches one of these entries.
   - If the requested redirect is invalid or absolute, force the user to a safe default page like /home.
   Node.js Example:

```
const allowed = ['/dashboard', '/profile']; if (allowed.includes(req.query.redirect)) { res.redirect(req.query.redirect); } else { res.redirect('/home'); }
```

   Best Practices
   - Never allow absolute URLs (starting with http://) from user input.
   - Never use simple "contains" checks that can be bypassed.
   - Inform users when they are being redirected to an external site.
   - Check that the redirect request originated from your own site.

Open Redirect • Unvalidated URL Redirect • No Auth Required

Property	Value
Severity	High
OWASP Category	Agentic Pentester
Affected Endpoints	https://brokencrystals.com/api/goto?url=https://google.com/, https://brokencrystals.com/api/goto?url=https://google.com?x=https://blockchain.info
Target	*brokencrystals.com
API Route	/api/goto?url=https://google.com/

Description:

Open Redirect occurs when an application redirects users to a URL specified in a parameter without validation. Attackers exploit this to conduct phishing attacks by redirecting users from a trusted site to a malicious one.
Impact
- Phishing: Redirecting users to identical fake sites to steal credentials.
- Malware Distribution: Sending users to sites that automatically download malware.
- Trust Exploitation: Users trust links from your domain, making phishing more effective.
Example Payload
https://trusted-site.com/redirect?url=https://evil.com

Attack Recap:

We requested https://brokencrystals.com/api/goto and identified a redirect parameter in the URL. The parameter was tested with an external URL to check if the server redirects to attacker-controlled destinations. Observing OPEN REDIRECT | PARAM: url confirms the server follows the redirect without proper validation. Attackers use open redirects to craft convincing phishing URLs that appear to originate from the trusted domain.

Open Redirect • Unvalidated URL Redirect • No Auth Required

Remediation Recommendations:

1. Validate all redirect URLs against a strict allowlist of permitted domains. Never allow redirects to arbitrary external URLs based on user input.

   Secure Implementation
   - Create a hardcoded list of internal paths or trusted partner domains.
   - Parse incoming URLs and verify the protocol (https) and hostname match the allowlist.
   - Instead of accepting full URLs, use tokens or IDs that map to safe destinations on the server.
   Node.js Example:

```
const ALLOWED_DOMAINS = ['www.myapp.com', 'app.myapp.com']; app.get('/redirect', (req, res) => { const target = new URL(req.query.url); if (ALLOWED_DOMAINS.includes(target.hostname) && target.protocol === 'https:') { res.redirect(target.href); } else { res.status(400).json({ error: 'Invalid destination' }); } });
```

   Best Practices
   - Never redirect to absolute URLs provided by the user without prior verification.
   - Never use regex-only validation; always use a robust URL parsing library.
   - Never allow redirects to javascript: or data: schemes.
   - Only allow relative URLs (starting with /) for most internal navigation.
   - Display a warning page before redirecting a user to an external domain.
   - Verify the Referer header to ensure the request originated from your own app.

Appendix

A. Methodology

OX Security's Agentic Pentester employs an AI-driven approach that combines static analysis with dynamic testing to achieve comprehensive vulnerability coverage. The methodology follows four phases:

- Target Discovery: Automated crawling and endpoint enumeration to map the full attack surface, including API endpoints, web pages, and configuration files.
- Static Analysis Matching: Source code analysis (when repositories are connected) to identify potential vulnerability patterns and correlate them with discovered endpoints.
- Payload Crafting & Execution: AI-generated payloads tailored to each identified attack vector, executed against live endpoints to confirm exploitability.
- Response Analysis & Validation: Automated analysis of server responses to confirm vulnerability presence, assess severity, and eliminate false positives - as well as to verify the absence of vulnerabilities, validate secure configurations, and confirm that defenses are working as expected.

This assessment was conducted as a white-box/ black-box test with full access to the application source code via the connected repository. The methodology aligns with OWASP Testing Guide v4.2, PTES (Penetration Testing Execution Standard), and NIST SP 800-115.

B. Severity Classification

Severity	Findings
Appoxlypse	The highest level of severity, reserved for critical risks that pose a severe threat.
Critical	High-impact, exploitable vulnerabilities.
High	Serious vulnerabilities requiring prompt attention.
Medium	Vulnerabilities with moderate risk.
Low	Low-risk findings.

C. Glossary

Term	Definition
CVSS	Common Vulnerability Scoring System — standardized severity rating methodology
CSP	Content Security Policy — HTTP header restricting resource loading
CSRF	Cross-Site Request Forgery — attack forcing authenticated users to perform unintended actions
IDOR	Insecure Direct Object Reference — accessing resources without authorization checks
LFI	Local File Inclusion — reading server files via path manipulation
RCE	Remote Code Execution — running arbitrary commands on a remote server
SSTI	Server-Side Template Injection — injecting code into template engines
WAF	Web Application Firewall — security layer filtering malicious HTTP traffic
White Box	White box penetration testing is security testing performed with full knowledge of the system and its code
XSS	Cross-Site Scripting — injecting malicious scripts into web pages
OWASP top 10	The OWASP Top 10 is a widely recognized standard outlining the most critical security risks to web applications
Sub Target	An individual endpoint discovered within a target application during the scanning phase

D. Contact Information

For questions about findings in this report or to request a retest:

Support Email: support@ox.security
Documentation: https://docs.ox.security
Retest Requests: Contact your OX Security account representative or initiate via the OX platform