Violet — tryviolet.ai

CONFIDENTIAL

PENETRATION TESTING

Security Assessment Report

https://pentest-ground.com:81

Trial Scan

PREPARED FOR

Example Customer
customer@example.com
Example Org

ASSESSMENT DATE

April 28, 2026

PREPARED BY

Violet AI
tryviolet.ai

Table of Contents

1. Cover Page
2. **Executive Summary**
3. **Methodology**
4. **Scope Statement**
5. **Risk Overview**
6. **Observation Notes**
    - 6.1 Overview
    - 6.2 Authentication Testing
    - 6.3 Authorization Testing
    - 6.4 Data Validation & Injection
    - 6.5 Session Management
    - 6.6 Business Logic
7. **Findings Summary**
8. **Vulnerability Details**
    1. Unauthenticated Post Modification (Horizontal IDOR with Data Modification)
    2. Werkzeug Debugger Console PIN-Protected Exposure
    3. Insufficient Rate Limiting on Login Endpoint
    4. Insecure Session Cookie Configuration
    5. Missing Idempotency Key on POST /create (Replay Attack)
    6. Boolean-Based SQL Injection in Search Functionality
    7. Reflected XSS in Search Endpoint
    8. Unsanitized Reference URL Parameter Flows to HTTP Request Sink
    9. Stored XSS in Post Title (GET /post/{id})
    10. Potential Server-Side Request Forgery (SSRF) via Reference URL Field
    11. Missing Cache-Control Headers on Session Responses
    12. Missing CSRF Protection on State-Changing POST Endpoints
    13. Race Condition / TOCTOU on POST /{id}/edit (Code Analysis Only)
    14. Missing HSTS Header on HTTPS Endpoints
    15. Missing Rate Limiting on POST /create and POST /{id}/edit
    16. Missing Input Length Restrictions
9. **Remediation Roadmap & SLAs**

Executive Summary

Violet assessed the web application hosted at https://pentest-ground.com:81, a Flask-based security guard company website offering blog and post management capabilities behind a session-authenticated dashboard. Testing covered authentication controls, session management, cross-site scripting, SQL injection, server-side request forgery, authorization enforcement, and business logic integrity across all discovered endpoints. The assessment identified **2 critical, 7 high, and 4 medium severity vulnerabilities** alongside several informational hardening observations. The application demonstrates sound controls in authenticated route redirection and parameterized query handling on the login endpoint, while weaknesses were concentrated in unauthenticated access to content modification, input handling, and session security configuration.

The most severe finding is an exposed Werkzeug debugger console reachable from the public internet; if the console PIN is bypassed, an attacker achieves arbitrary Python code execution within the application context — equivalent to full server compromise. Separately, the post-editing endpoints carry no authentication or ownership checks, allowing any anonymous user to read and overwrite any content stored in the application, enabling defacement and data integrity violations without a valid account. A SQL injection in the search function enables full database exfiltration including user credentials, a stored cross-site scripting vulnerability allows JavaScript execution in any visitor's browser for session hijacking, and an unprotected server-side request forgery vector allows the application server to be weaponized as an internal-network proxy. Chained together, these weaknesses create a clear escalation path from anonymous internet access to persistent system control.

A full scan is recommended to verify exploitability of each finding, validate chained attack paths, and receive HTTP-level proof of impact with complete exploitation evidence.

10	4	2	24 hours
Urgent findings	**Other findings**	**Informational**	**Action required**
Critical + High	Medium + Low	Awareness only	For critical findings

Methodology

This assessment was conducted using the **OWASP Web Security Testing Guide (WSTG)** methodology. The testing approach was **black-box**, combining automated scanning tools with AI-driven analysis to identify security vulnerabilities. CVSS v3.1 scores and vector strings are AI-assessed and have not been independently verified by a human analyst.

Phases Executed

PHASE	DESCRIPTION	STATUS
Pre-Reconnaissance	External Scanning & Source Analysis	✓
Reconnaissance	Attack Surface Mapping	✓
Vulnerability Analysis	Automated Vulnerability Detection	✓
Exploitation	Vulnerability Exploitation & Validation	✓
Reporting	Report Generation	✓

Tools Available

The following external scanning tools were available during this assessment:

- nmap — Network discovery and port scanning
- whatweb — Web technology fingerprinting
- Playwright — Browser-based interaction testing
- testssl.sh — TLS/SSL configuration analysis

Assessment Window

START DATE
April 28, 2026 at 1:02 AM

COMPLETION DATE
April 28, 2026 at 1:35 AM

Scope Statement

The following defines the boundaries of this security assessment, including targets tested, access level, and any exclusions.

TARGET
`https://pentest-ground.com:81`

TESTING TYPE
Black-box

SOURCE CODE REVIEW
Not included

AUTHENTICATION
Agent-initiated authentication

ASSESSMENT START
April 28, 2026 at 1:02 AM

ASSESSMENT END
April 28, 2026 at 1:35 AM

Scope Exclusions

Full application scope — no exclusions configured.

Assessment Limitations & Disclaimer

This assessment is a point-in-time evaluation of the target application's security posture as of the assessment end date. Findings are based on the scope, access, and timeframe defined in this scan's configuration and Violet Labs' Terms of Service. Findings are produced with AI assistance and have not been independently verified by a human analyst; false positives and false negatives are possible. Customers should validate each finding in their own environment before remediation and re-run an assessment after material code or infrastructure changes. The absence of identified vulnerabilities does not guarantee the absence of security weaknesses. Violet Labs makes no warranty regarding the completeness of testing coverage and this report is not a substitute for ongoing security controls, monitoring, or independent expert review.

This is a trial-tier assessment: exploitation testing was not performed, source code review may be limited, and the scope is intentionally narrower than a full scan.

Risk Overview

Risk Matrix

Findings are plotted by Impact (severity of damage if exploited) versus Likelihood (ease of exploitation and prerequisites required).

Impact ↓ / Likelihood →	High	Medium	Low
High	2	8	—
Medium	—	4	—
Low	—	—	2

+ 2 informational observations not plotted — no exploitable risk.

Observation Notes

The following sections describe the attack scenarios performed during this assessment, including techniques attempted, findings, and controls observed.

Overview

The target is a Flask-based website for a security guard company, presenting public marketing pages (about, services, clients) alongside a blog system where posts can be created, edited, and viewed by any visitor. An authenticated dashboard is accessible to registered users via a form-based login. Violet tested all discovered endpoints across six security domains: authentication and session management, authorization and access control, SQL and command injection, cross-site scripting, server-side request forgery, and business logic integrity. Coverage was comprehensive across the full public attack surface; limitations include the absence of source code review (requiring behavioral inference) and no exploitation being performed, as this is a trial scan.

Authentication Testing

The application uses form-based authentication at `POST /login` with credentials validated server-side and sessions managed through a signed Flask cookie. Violet tested authentication bypass against the Werkzeug debugger console at `/console`, brute-force and credential-stuffing resistance on the login endpoint, CSRF enforcement on state-changing forms, session cookie security flag configuration, transport-layer HTTPS downgrade protection via HSTS, and caching behavior for authentication responses.

CONTROLS VERIFIED

CONTROLS VERIFIED

- **Dashboard redirect enforcement:** `GET /dashboard` enforces a valid session requirement and issues a redirect to `/login` for unauthenticated requests, confirming the authentication guard is correctly wired to the protected route.
- **Anonymous endpoint design:** Endpoints including `/{id}/edit`, `/create`, and `/search` are intentionally accessible without authentication per the application's design, and this access model is uniformly applied without unintended privilege injection at these routes.

Violet identified significant weaknesses throughout the authentication posture. The `/console` Werkzeug debugger endpoint is publicly reachable without any credential requirement — only a guessable server-generated PIN separates an attacker from an interactive Python REPL with full application context access. The `POST /login` endpoint lacks all brute-force defenses: five consecutive failed login attempts returned `HTTP 200` with no rate-limit headers, no `HTTP 429` response, no account lockout, and no CAPTCHA challenge. Session cookies are issued without `HttpOnly`, `Secure`, or `SameSite` flags, rendering them accessible to injected JavaScript, potentially transmissible over unencrypted connections, and submittable from cross-origin requests. No `Strict-Transport-Security` header was present on any authenticated response, and authentication page responses carry no `Cache-Control: no-store` directive to prevent proxy or browser caching.

RECOMMENDATION

- Disable Flask debug mode in all production environments; set `DEBUG=False` in application configuration and restrict `/console` to `localhost` only, or remove the Werkzeug debugger entirely.
- Implement rate limiting and progressive lockout on `POST /login`: enforce `HTTP 429` after a configurable threshold of failed attempts per IP, with exponential backoff and optional CAPTCHA after repeated failures.

- Set `HttpOnly`, `Secure`, and `SameSite=Strict` on all session cookies (`SESSION_COOKIE_HTTPONLY=True`, `SESSION_COOKIE_SECURE=True`, `SESSION_COOKIE_SAMESITE='Strict'` in Flask config); emit `Strict-Transport-Security: max-age=31536000; includeSubDomains` on all HTTPS responses; add `Cache-Control: no-store` and `Pragma: no-cache` to all authentication and session-bearing response headers.

Authorization Testing

The application implements a two-tier access model: anonymous users access public and content-management endpoints, while authenticated users gain access to `/dashboard`. Violet tested horizontal access control by attempting to read and modify posts belonging to other users via `GET /{id}/edit` and `POST /{id}/edit` without authentication, and vertical access control by verifying the protection model on the dashboard, the Werkzeug console, and all discovered privilege-sensitive routes.

CONTROLS VERIFIED

CONTROLS VERIFIED

- **Dashboard session boundary:** `GET /dashboard` requires a valid session cookie and correctly redirects unauthenticated requests to `/login`, confirming a server-enforced vertical boundary between anonymous and authenticated access tiers.
- **Public endpoint design integrity:** `GET /post/{id}`, `POST /create`, and `GET /search` are publicly accessible by design, and Violet confirmed that no unintended privilege injection or role escalation is possible through these routes.
- **Login and logout flow:** `POST /login` and `GET /logout` operate correctly within the authenticated session lifecycle without exposing privilege escalation vectors.

Violet identified a critical authorization gap on the content editing endpoints. Both `GET /{id}/edit` and `POST /{id}/edit` are accessible to unauthenticated users with no session check and no post-ownership validation. Any anonymous party can retrieve the full title and content of any post by requesting its edit form, and can overwrite the content of any post by submitting the form directly — no session cookie, no role, and no ownership proof is required. This was confirmed during testing by successfully modifying a live post's title and body without authentication and observing the changes reflected on the public post view.

RECOMMENDATION

- Require a valid authenticated session on both `GET /{id}/edit` and `POST /{id}/edit`; unauthenticated requests must be redirected to `/login` before the edit form is rendered or the submission processed.
- After authentication, enforce an ownership check: compare the requesting user's identity against the post's stored `owner_id` before granting read or write access to the edit form; return `HTTP 403 Forbidden` on mismatch.
- Audit all remaining content-modification routes for equivalent missing ownership guards, and enforce the same authentication-plus-ownership pattern on any future state-changing endpoints.

Data Validation & Injection

The application exposes several input surfaces: a search query field at `/search`, post `title` and `content` fields on the create and edit forms, and a URL-typed `reference` field on the post creation form. Violet tested SQL injection across all query parameters and form fields, command injection via the reference URL parameter, stored and reflected cross-site scripting across all output rendering contexts, and input length and format validation on all form fields.

CONTROLS VERIFIED

CONTROLS VERIFIED

- **Login username parameterization:** The `username` parameter at `POST /login` is bound to a parameterized SQL query; payloads including `admin' OR '1'='1` and `admin' AND SLEEP(5)` returned consistent error messages with no timing variation or database

error leakage, confirming prepared-statement protection.

- **Search result HTML escaping:** The `query` parameter at `GET /search` is rendered through Jinja2 auto-escaping; the test payload `<img src=x onerror="alert(1)">` was returned as fully HTML-encoded output, confirming safe rendering in the page-heading context.
- **Post content and edit form field escaping:** The `content` field rendered in `<p>` tags at `GET /post/{id}` and the `title` field rendered in `value=""` attributes at `GET /{id}/edit` are both correctly HTML-entity encoded, confirming safe handling in those specific rendering contexts.

Violet identified three critical input validation failures. First, the `query` parameter at `POST /search` is concatenated directly into a SQL `LIKE` clause without parameterization; boolean-based payloads (`' AND '1'='1` vs `' AND '1'='2`) produced demonstrably different result sets, confirming SQL injection with direct output reflection and UNION-based extraction capability, enabling full credential and data exfiltration. Second, post titles stored via `POST /create` or `POST /{id}/edit` are rendered inside an `<h2>` tag at `GET /post/{id}` without HTML encoding; a `<script>` payload injected into the title executed in the browser, confirming stored XSS that persists across all future visitors to the affected post. Third, the `reference` URL field at `POST /create` flows directly to an HTTP request sink with no URL validation or allowlisting, enabling an attacker to force the application server to make requests to arbitrary internal or external destinations.

RECOMMENDATION

- Replace the raw SQL string concatenation in the `/search` handler with a parameterized query using `?` or named parameter placeholders, consistent with the already-correct pattern used on `/login` .
- Apply HTML escaping to post titles in all rendering contexts, particularly the `<h2>` heading in the post view template; enable `autoescape=True` globally in Jinja2 configuration and audit all templates for unescaped `{{ variable }}` expressions.
- Validate the `reference` URL field against an explicit allowlist of permitted schemes (`https://`) and domains; reject any value pointing to private IP ranges, loopback addresses (`127.x.x.x` , `::1`), link-local ranges (`169.254.x.x`), or cloud metadata endpoints.

Session Management

Session state is managed through a Flask signed cookie issued at login. Violet analyzed session cookie attributes (`HttpOnly` , `Secure` , `SameSite`), HTTPS enforcement and HSTS policy, cache-control behavior on authentication responses, and the session lifecycle including creation, expiry, and invalidation on logout.

Violet verified few session management controls on this application. The sole transport-layer protection confirmed is that the application is served over HTTPS on port 81. However, no `Strict-Transport-Security` header is present to prevent protocol downgrade on a first visit, session cookies lack all three recommended security attributes — `HttpOnly` , `Secure` , and `SameSite` — and authentication responses carry no `Cache-Control: no-store` directive. The absence of `HttpOnly` is particularly consequential in combination with the confirmed stored XSS finding: an attacker who injects a JavaScript payload into a post title can exfiltrate the session cookie of any user who views that post. The absence of `Secure` means cookies could be transmitted over an unencrypted fallback connection, and the absence of `SameSite` removes the browser-enforced same-origin submission restriction.

RECOMMENDATION

- Configure Flask session cookie security flags: `SESSION_COOKIE_HTTPONLY=True` , `SESSION_COOKIE_SECURE=True` , and `SESSION_COOKIE_SAMESITE='Strict'` .

- Emit `Strict-Transport-Security: max-age=31536000; includeSubDomains` on all HTTPS responses, preferably at the Nginx reverse proxy layer so it applies uniformly.
- Add `Cache-Control: no-store, no-cache` and `Pragma: no-cache` to all responses that set or reference session cookies.

Business Logic

The application's primary stateful workflows are post creation (`POST /create`), post editing (`POST /{id}/edit`), and user login (`POST /login`). Violet tested replay attack resistance on state-changing endpoints, rate limiting on content modification routes, concurrent edit race conditions on shared post objects, input length validation, and resource quota enforcement.

Violet verified few business logic controls on this application. Sequential numeric post IDs on `GET /post/{id}` are confirmed as an intentional design choice for a public blog where post discovery is expected, rather than an exploitable access control gap.

Violet identified two business logic weaknesses. The `POST /create` endpoint accepts and processes identical requests without any idempotency key, submission nonce, or duplicate detection: submitting the same payload five consecutive times each returned `HTTP 200` and created five distinct database records with separate sequential IDs, enabling resource exhaustion and unconstrained content spam. Concurrent editing of the same post via `POST /{id}/edit` also returned `HTTP 200` for all parallel requests without conflict detection, optimistic locking, or version verification, suggesting a TOCTOU race condition where last-writer silently wins and intermediate updates are irreversibly discarded.

RECOMMENDATION

- Require a unique idempotency key (hidden form field or `Idempotency-Key` request header) on `POST /create` and `POST /{id}/edit` ; cache processed keys and return `HTTP 409 Conflict` on duplicate submissions within a configurable window (e.g., 24 hours).
- Implement per-IP and per-user rate limiting on all state-changing endpoints with `HTTP 429` responses; apply a per-user quota for post creation to prevent content spam and storage exhaustion.
- Add optimistic concurrency control to the edit handler using a `version` column or ETag; reject updates where the submitted version does not match the current database record, and return `HTTP 409 Conflict` to signal the conflict to the client.

Findings Summary

#	FINDING	CATEGORY	SEVERITY	CVSS	IMPACT	LIKELIHOOD	CONFIDENCE	STATUS
1	Unauthenticated Post Modification (Horizontal IDOR with Data Modification)	Authz	CRITICAL	9.1	HIGH	HIGH	CONFIRMED	Open
2	Werkzeug Debugger Console PIN-Protected Exposure	Auth	CRITICAL	9.1	HIGH	HIGH	CONFIRMED	Open
3	Insufficient Rate Limiting on Login Endpoint	Auth	HIGH	8.2	HIGH	MEDIUM	CONFIRMED	Open
4	Insecure Session Cookie Configuration	Auth	HIGH	8.2	HIGH	MEDIUM	CONFIRMED	Open
5	Missing Idempotency Key on POST /create (Replay Attack)	Business-Logic	HIGH	5.3	HIGH	MEDIUM	CONFIRMED	Open
6	Boolean-Based SQL Injection in Search Functionality	Injection	HIGH	8.1	HIGH	MEDIUM	CONFIRMED	Open
7	Reflected XSS in Search Endpoint	Xss	HIGH	8.2	HIGH	MEDIUM	UNCONFIRMED	Open
8	Unsanitized Reference URL Parameter Flows to HTTP Request Sink	Ssrf	HIGH	7.7	HIGH	MEDIUM	CONFIRMED	Open
9	Stored XSS in Post Title (GET /post/{id})	Xss	HIGH	8.2	HIGH	MEDIUM	CONFIRMED	Open
10	Potential Server-Side Request Forgery (SSRF) via	Ssrf	HIGH	7.7	HIGH	MEDIUM	UNCONFIRMED	Open

#	FINDING	CATEGORY	SEVERITY	CVSS	IMPACT	LIKELIHOOD	CONFIDENCE	STATUS
	Reference URL Field							
11	Missing Cache-Control Headers on Session Responses	Auth	MEDIUM	5.4	MEDIUM	MEDIUM	CONFIRMED	Open
12	Missing CSRF Protection on State-Changing POST Endpoints	Auth	MEDIUM	5.4	MEDIUM	MEDIUM	CONFIRMED	Open
13	Race Condition / TOCTOU on POST /{id}/edit (Code Analysis Only)	Business-Logic	MEDIUM	3.1	MEDIUM	MEDIUM	CONFIRMED	Open
14	Missing HSTS Header on HTTPS Endpoints	Auth	MEDIUM	5.4	MEDIUM	MEDIUM	CONFIRMED	Open
15	Missing Rate Limiting on POST /create and POST /{id}/edit	Business-Logic	INFORMATIONAL	0.0	LOW	LOW	CONFIRMED	Open
16	Missing Input Length Restrictions	Business-Logic	INFORMATIONAL	0.0	LOW	LOW	CONFIRMED	Open

Vulnerability Details

FINDING #1 · AUTHZ · CWE-639

Unauthenticated Post Modification (Horizontal IDOR with Data Modification)

CRITICAL

CVSS 9.1

Impact: **HIGH** | Likelihood: **HIGH** | Status: **Open** | `CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N`

DESCRIPTION

The `POST /{id}/edit` endpoint allows unauthenticated attackers to modify any post in the system without providing authentication or proving ownership. The endpoint accepts POST requests with arbitrary post IDs in the path parameter and persists modifications directly to the database with no authorization checks.

PROOF OF CONCEPT

Step-by-step exploitation — available on full scans.

EVIDENCE

Live attack output — available on full scans.

IMPACT

An attacker can modify any post's title and content by sending a POST request to /{id}/edit with modified form data. This enables unauthorized data manipulation affecting all users' published content. The impact extends to data integrity violations where public-facing content on the website can be altered by external attackers, potentially damaging the organization's reputation and user trust.

LIKELIHOOD

No special prerequisites required. The vulnerability is exploitable directly via the public HTTPS endpoint at https://pentest-ground.com:81/{id}/edit.

RECOMMENDATION

Remediation guidance — available on full scans.

References

A01:2021-Broken Access Control

CWE-284: Improper Access Control

CWE-639 — MITRE CWE

FINDING #2 · AUTH · CWE-215

Werkzeug Debugger Console PIN-Protected Exposure

CRITICAL

CVSS 9.1

Impact: **HIGH** | Likelihood: **HIGH** | Status: **Open** | `CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N`

DESCRIPTION

The Flask application exposes an interactive Werkzeug debugger console at the `/console` endpoint on the public internet. While the console is protected by a PIN, this mechanism is inherently weak as the PIN is printed to the server's standard output during application startup and can be guessed or discovered. This exposes the application to remote code execution if the PIN is bypassed.

PROOF OF CONCEPT

Step-by-step exploitation — available on full scans.

EVIDENCE

Live attack output — available on full scans.

IMPACT

An attacker gaining access to the Werkzeug console can execute arbitrary Python code within the application context, leading to complete system compromise including database access, file system access, and privilege escalation.

LIKELIHOOD

- PIN discovery via: server logs, environment variable exposure, or brute-force attempts
- Access to the `/console` endpoint (publicly accessible)

RECOMMENDATION

Remediation guidance — available on full scans.

References

A07:2021-Identification and Authentication Failures

CWE-287: Improper Authentication

CWE-215 — MITRE CWE

FINDING #3 · AUTH · CWE-307

Insufficient Rate Limiting on Login Endpoint

HIGH

CVSS 8.2

Impact: **HIGH** | Likelihood: **MEDIUM** | Status: **Open** | `CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N`

DESCRIPTION

The `/login` endpoint lacks any rate limiting or account lockout mechanism. Unauthenticated users can submit unlimited login attempts without throttling, backoff, or CAPTCHA challenges, making the endpoint susceptible to brute-force and credential-stuffing attacks.

PROOF OF CONCEPT

Step-by-step exploitation — available on full scans.

EVIDENCE

Live attack output — available on full scans.

IMPACT

An attacker can attempt unlimited password guesses against user accounts without triggering any defensive mechanism, enabling account compromise through automated credential attacks.

LIKELIHOOD

- Knowledge of valid usernames (or enumerate via other vectors)
- Network access to the `/login` endpoint (publicly accessible)

RECOMMENDATION

Remediation guidance — available on full scans.

References

A07:2021-Identification and Authentication Failures

CWE-287: Improper Authentication

CWE-307 — MITRE CWE

FINDING #4 · AUTH · CWE-614

Insecure Session Cookie Configuration

HIGH

CVSS 8.2

Impact: **HIGH** | Likelihood: **MEDIUM** | Status: **Open** | `CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N`

DESCRIPTION

Session cookies are set without critical security flags. The `session` cookie lacks the `HttpOnly` flag (allowing JavaScript access), the `Secure` flag (allowing transmission over HTTP), and the `SameSite` attribute (allowing cross-site submission). This configuration exposes sessions to XSS-based hijacking, man-in-the-middle attacks, and CSRF exploitation.

PROOF OF CONCEPT

Step-by-step exploitation — available on full scans.

EVIDENCE

Live attack output — available on full scans.

IMPACT

Attackers can steal session cookies via XSS attacks, replay sessions over unencrypted connections, or perform CSRF attacks using the victim's session.

LIKELIHOOD

- For HttpOnly bypass: Reflected or Stored XSS vulnerability (e.g., /search endpoint reflects user input)
- For Secure flag bypass: Network access to intercept HTTP traffic or HTTP fallback routes
- For CSRF: Victim authentication and visit to attacker-controlled page

RECOMMENDATION

Remediation guidance — available on full scans.

References

A07:2021-Identification and Authentication Failures

CWE-287: Improper Authentication

CWE-614 — MITRE CWE

FINDING #5 · BUSINESS-LOGIC · CWE-425

Missing Idempotency Key on POST /create (Replay Attack)

HIGH

CVSS 5.3

Impact: **HIGH** | Likelihood: **MEDIUM** | Status: **Open** | `CVSS:3.1/AV:N/AC:H/PR:L/UI:N/S:U/C:N/I:H/A:N`

DESCRIPTION

The `POST /create` endpoint lacks an idempotency key mechanism or duplicate detection. Submitting identical POST requests multiple times results in the creation of multiple distinct posts with different database IDs, each assigned a new sequential ID. This allows attackers to create duplicate content, exhaust disk/database resources, or spam the application.

PROOF OF CONCEPT

Step-by-step exploitation — available on full scans.

EVIDENCE

Live attack output — available on full scans.

IMPACT

An attacker can craft a single POST request to create a blog post and then replay it multiple times—either manually or via automated tools—to create unlimited duplicate posts. This leads to:

LIKELIHOOD

No special prerequisites required. Any user can submit a POST request to /create without authentication.

RECOMMENDATION

Remediation guidance — available on full scans.

References

OWASP Top Ten

CWE-425 — MITRE CWE

FINDING #6 · INJECTION · CWE-89

Boolean-Based SQL Injection in Search Functionality

HIGH

CVSS 8.1

Impact: **HIGH** | Likelihood: **MEDIUM** | Status: **Open** | `CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N`

DESCRIPTION

Severity: High
CWE: CWE-89
Summary: The `POST /search` endpoint accepts a `query` parameter that is concatenated directly into a SQL LIKE clause without proper parameterization. An attacker can inject SQL logic to manipulate the query behavior and extract data or bypass authentication logic. The vulnerability is externally exploitable without authentication.

PROOF OF CONCEPT

Step-by-step exploitation — available on full scans.

EVIDENCE

Live attack output — available on full scans.

IMPACT

An attacker can use boolean-based SQL injection to systematically extract all data from the posts table (titles, content, references, metadata) by crafting payloads that alter the WHERE clause logic. The same technique can be extended to extract data from other tables (users, credentials) if accessible through JOIN operations. This enables unauthorized data disclosure including user credentials if stored in the database.

LIKELIHOOD

No special prerequisites required. The endpoint is publicly accessible and requires no authentication.

RECOMMENDATION

Remediation guidance — available on full scans.

References

A03:2021-Injection

CWE-89: SQL Injection

CWE-89 — MITRE CWE

FINDING #7 · XSS · CWE-79

Reflected XSS in Search Endpoint

HIGH

CVSS 8.2

Impact: **HIGH** | Likelihood: **MEDIUM** | UNCONFIRMED | Status: **Open**

`CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:H/A:N`

DESCRIPTION

The `/search` endpoint reflects user-controlled input directly into the HTML response without sanitization or encoding. The `query` parameter is displayed in a page heading with the format "You searched for: {query}". This allows reflected cross-site scripting (XSS) attacks.

PROOF OF CONCEPT

Step-by-step exploitation — available on full scans.

EVIDENCE

Live attack output — available on full scans.

LIKELIHOOD

Exploitation requires some prerequisites such as valid credentials, specific application state, or knowledge of internal endpoints.

RECOMMENDATION

Remediation guidance — available on full scans.

References

A03:2021-Injection (XSS)

CWE-79: Cross-Site Scripting

CWE-79 — MITRE CWE

FINDING #8 · SSRF · CWE-918

Unsanitized Reference URL Parameter Flows to HTTP Request Sink

HIGH

CVSS 7.7

Impact: **HIGH** | Likelihood: **MEDIUM** | Status: **Open** | `CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:N/A:N`

DESCRIPTION

The `POST /create` endpoint accepts a user-supplied "reference" parameter that is intended to contain a URL. The reconnaissance phase identified that this parameter flows directly to the application's http_request sink with no validation, sanitization, or allowlisting applied. This allows attackers to control the destination and content of server-side HTTP requests initiated by the application.

PROOF OF CONCEPT

Step-by-step exploitation — available on full scans.

EVIDENCE

Live attack output — available on full scans.

IMPACT

An attacker can leverage this SSRF vulnerability to:

- Request arbitrary URLs (including internal services, cloud metadata endpoints, or external attacker-controlled servers) using the application server as a proxy
- Bypass network-level access controls and firewalls that might restrict direct access from the attacker's network
- Perform internal network reconnaissance and service discovery
- Access cloud metadata services (AWS IMDSv2/v1, GCP metadata, Azure endpoints) to retrieve credentials and configuration
- Interact with internal APIs and services not exposed to the internet
- Perform server-side cache poisoning or SSRF-based DOS attacks

LIKELIHOOD

- No special prerequisites required. The POST /create endpoint is publicly accessible and requires no authentication.

RECOMMENDATION

Remediation guidance — available on full scans.

References

A10:2021-Server-Side Request Forgery

CWE-918: SSRF

CWE-918 — MITRE CWE

FINDING #9 · XSS · CWE-79

Stored XSS in Post Title (GET /post/{id})

HIGH

CVSS 8.2

Impact: **HIGH** | Likelihood: **MEDIUM** | Status: **Open** | `CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:H/A:N`

DESCRIPTION

The Flask application stores user-supplied post titles in the database and renders them in the page heading without HTML encoding. When a malicious post title containing JavaScript (e.g., `<script>alert(1)</script>`) is retrieved from the database and rendered in an `<h2>` tag, the script executes in the attacker's context. Any unauthenticated user can create or modify posts via the unauthenticated `/1/edit` endpoint, making this vulnerability externally exploitable.

PROOF OF CONCEPT

Step-by-step exploitation — available on full scans.

EVIDENCE

Live attack output — available on full scans.

IMPACT

An attacker can inject arbitrary JavaScript that executes in the context of every user viewing the affected post. This enables session token theft (via `document.cookie`), account takeover, phishing attacks, and defacement. The session cookie lacks the HttpOnly flag, making token exfiltration straightforward.

LIKELIHOOD

No special prerequisites required. The `/create` and `/{id}/edit` endpoints accept user input without authentication.

RECOMMENDATION

Remediation guidance — available on full scans.

References

A03:2021-Injection (XSS)

CWE-79: Cross-Site Scripting

CWE-79 — MITRE CWE

FINDING #10 · SSRF · CWE-918

Potential Server-Side Request Forgery (SSRF) via Reference URL Field

HIGH

CVSS 7.7

Impact: **HIGH** | Likelihood: **MEDIUM** | **UNCONFIRMED** | Status: **Open**

`CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:N/A:N`

DESCRIPTION

The `/create` endpoint accepts a user-controlled `reference` URL parameter. If the server fetches or validates this URL, an attacker can supply URLs pointing to internal resources, cloud metadata endpoints, or sensitive services, potentially causing Server-Side Request Forgery (SSRF).

PROOF OF CONCEPT

Step-by-step exploitation — available on full scans.

EVIDENCE

Live attack output — available on full scans.

LIKELIHOOD

Exploitation requires some prerequisites such as valid credentials, specific application state, or knowledge of internal endpoints.

RECOMMENDATION

Remediation guidance — available on full scans.

References

A10:2021-Server-Side Request Forgery

CWE-918: SSRF

CWE-918 — MITRE CWE

FINDING #11 · AUTH · CWE-524

Missing Cache-Control Headers on Session Responses

MEDIUM

CVSS 5.4

Impact: **MEDIUM** | Likelihood: **MEDIUM** | Status: **Open** | `CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:N`

DESCRIPTION

Authentication responses do not include `Cache-Control: no-store` or `Pragma: no-cache` headers. This allows browsers, proxies, and other intermediaries to cache sensitive authentication responses, potentially leaking session information or authenticated content to other users of the same system.

PROOF OF CONCEPT

Step-by-step exploitation — available on full scans.

EVIDENCE

Live attack output — available on full scans.

IMPACT

On shared computers or networks, attackers can recover cached authentication responses containing session tokens or sensitive user data from browser/proxy caches.

LIKELIHOOD

- Shared computer environment or network with HTTP caching
- Access to cache storage (browser cache, proxy cache, intermediary storage)

RECOMMENDATION

Remediation guidance — available on full scans.

References

A07:2021-Identification and Authentication Failures

CWE-287: Improper Authentication

CWE-524 — MITRE CWE

FINDING #12 · AUTH · CWE-352

Missing CSRF Protection on State-Changing POST Endpoints

MEDIUM

CVSS 5.4

Impact: **MEDIUM** | Likelihood: **MEDIUM** | Status: **Open** | `CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:N`

DESCRIPTION

The `/create` and `/{id}/edit` endpoints accept POST requests without validating CSRF tokens. These endpoints modify application state (creating/editing posts) without any anti-CSRF mechanisms such as synchronizer tokens, double-submit cookies, or SameSite cookie attributes.

PROOF OF CONCEPT

Step-by-step exploitation — available on full scans.

EVIDENCE

Live attack output — available on full scans.

IMPACT

An attacker can craft malicious HTML pages that, when visited by authenticated users, automatically submit state-changing requests (POST) on behalf of the user. This enables unauthorized post creation, modification, or deletion.

LIKELIHOOD

- Victim must be authenticated (for edit operations; create is unauthenticated)
- Victim must visit an attacker-controlled page while authenticated

RECOMMENDATION

Remediation guidance — available on full scans.

References

A07:2021-Identification and Authentication Failures

CWE-287: Improper Authentication

CWE-352 — MITRE CWE

FINDING #13 · BUSINESS-LOGIC · CWE-367

Race Condition / TOCTOU on POST /{id}/edit (Code Analysis Only)

MEDIUM

CVSS 3.1

Impact: **MEDIUM** | Likelihood: **MEDIUM** | Status: **Open** | `CVSS:3.1/AV:N/AC:H/PR:L/UI:N/S:U/C:N/I:L/A:N`

DESCRIPTION

The `POST /{id}/edit` endpoint may be vulnerable to a time-of-check-time-of-use (TOCTOU) race condition if the backend performs separate fetch and update operations without atomic locking. When multiple concurrent requests edit the same post simultaneously, there is a window where one thread reads the post state, another thread updates it, and the first thread overwrites the second thread's changes. This results in lost updates.

PROOF OF CONCEPT

Step-by-step exploitation — available on full scans.

EVIDENCE

Live attack output — available on full scans.

IMPACT

An attacker sending concurrent edit requests to the same post ID may:

LIKELIHOOD

- Attacker must be able to send concurrent HTTP requests to the same endpoint
- Knowledge of an existing post ID (easily enumerated via numeric IDs)

RECOMMENDATION

Remediation guidance — available on full scans.

References

OWASP Top Ten

CWE-367 — MITRE CWE

FINDING #14 · AUTH · CWE-311

Missing HSTS Header on HTTPS Endpoints

MEDIUM

CVSS 5.4

Impact: **MEDIUM** | Likelihood: **MEDIUM** | Status: **Open** | `CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:N`

DESCRIPTION

The application serves authentication endpoints over HTTPS but does not include the Strict-Transport-Security (HSTS) header in responses. Without HSTS, clients can be downgraded to HTTP on the first visit, exposing credentials and session tokens to man-in-the-middle attacks.

PROOF OF CONCEPT

Step-by-step exploitation — available on full scans.

EVIDENCE

Live attack output — available on full scans.

IMPACT

An attacker performing network-level attacks (MITM) can downgrade the connection to HTTP, intercept authentication credentials and session cookies, and compromise user accounts.

LIKELIHOOD

- Network-level attack capability (MITM, ARP spoofing, DNS hijacking)
- Victim's first visit to the application (before HSTS preload list is consulted)

RECOMMENDATION

Remediation guidance — available on full scans.

References

A07:2021-Identification and Authentication Failures

CWE-287: Improper Authentication

CWE-311 — MITRE CWE

FINDING #15 · BUSINESS-LOGIC · CWE-770

Missing Rate Limiting on POST /create and POST /{id}/edit

INFORMATIONAL

CVSS 0.0

Impact: **LOW** Likelihood: **LOW** Status: **Open**

DESCRIPTION

The `POST /create` and `POST /{id}/edit` endpoints have no observable rate limiting. An attacker can submit unlimited requests in rapid succession without throttling or per-user request quotas. Combined with the missing idempotency key, this enables spam and resource exhaustion attacks.

PROOF OF CONCEPT

Step-by-step exploitation — available on full scans.

EVIDENCE

Live attack output — available on full scans.

IMPACT

An attacker can:

LIKELIHOOD

Exploitation requires significant prerequisites including privileged access, specific configurations, or chaining with other vulnerabilities.

RECOMMENDATION

Remediation guidance — available on full scans.

References

OWASP Top Ten

CWE-770 — MITRE CWE

FINDING #16 · BUSINESS-LOGIC · CWE-400

Missing Input Length Restrictions

INFORMATIONAL

CVSS 0.0

Impact: **LOW** | Likelihood: **LOW** | Status: **Open**

DESCRIPTION

The `POST /create` and `POST /{id}/edit` endpoints have no explicit maximum length restrictions on title and content fields. A 10,000 character title was accepted without truncation or error. This enables:

- Buffer overflows if backed by fixed-size buffers
- Memory exhaustion if very large payloads are accepted
- DoS via oversized input by submitting gigabytes of data

PROOF OF CONCEPT

Step-by-step exploitation — available on full scans.

EVIDENCE

Live attack output — available on full scans.

LIKELIHOOD

Exploitation requires significant prerequisites including privileged access, specific configurations, or chaining with other vulnerabilities.

RECOMMENDATION

Remediation guidance — available on full scans.

References

OWASP Top Ten

CWE-400 — MITRE CWE

Remediation Roadmap & SLAs

Recommended Remediation Timelines

The following timelines are recommended based on industry standards and the severity of each finding. Organizations should adjust these timelines based on their risk tolerance and operational constraints.

SEVERITY	RECOMMENDED TIMELINE	RATIONALE
CRITICAL	24 hours	Active exploitation risk; immediate patching required
HIGH	7 days	Significant risk; prioritize in current sprint
MEDIUM	30 days	Moderate risk; schedule in next release cycle
LOW	90 days	Minor risk; address during regular maintenance

Remediation Tracking

10 critical/high findings require remediation within 7 days. 16 total findings identified.

#	FINDING	SEVERITY	SLA DEADLINE	STATUS
1	Unauthenticated Post Modification (Horizontal IDOR with Data Modification)	CRITICAL	April 29, 2026	Open
2	Werkzeug Debugger Console PIN-Protected Exposure	CRITICAL	April 29, 2026	Open
3	Insufficient Rate Limiting on Login Endpoint	HIGH	May 5, 2026	Open
4	Insecure Session Cookie Configuration	HIGH	May 5, 2026	Open
5	Missing Idempotency Key on POST /create (Replay Attack)	HIGH	May 5, 2026	Open
6	Boolean-Based SQL Injection in Search Functionality	HIGH	May 5, 2026	Open
7	Reflected XSS in Search Endpoint	HIGH	May 5, 2026	Open
8	Unsanitized Reference URL Parameter Flows to HTTP Request Sink	HIGH	May 5, 2026	Open
9	Stored XSS in Post Title (GET /post/{id})	HIGH	May 5, 2026	Open
10	Potential Server-Side Request Forgery (SSRF) via Reference URL Field	HIGH	May 5, 2026	Open
11	Missing Cache-Control Headers on Session Responses	MEDIUM	May 28, 2026	Open
12	Missing CSRF Protection on State-Changing POST Endpoints	MEDIUM	May 28, 2026	Open
13	Race Condition / TOCTOU on POST /{id}/edit (Code Analysis Only)	MEDIUM	May 28, 2026	Open
14	Missing HSTS Header on HTTPS Endpoints	MEDIUM	May 28, 2026	Open