Overpass — Application Security Assessment

Vulnerability Class: Authentication Bypass (Broken Authentication)
Risk Rating: Critical (CVSS 3.1: 9.8 — AV:N / AC:L / PR:N / UI:N / S:U / C:H / I:H / A:H)

Scope / Authorization: Testing was conducted in a controlled, authorized lab environment. No production systems, real data, or external services were accessed.

1. Executive Summary

An assessment of the Overpass web application identified a critical authentication bypass caused by trusting client-supplied authentication cookies without server-side validation. An unauthenticated attacker can supply an arbitrary cookie value and be treated as a logged-in user, gaining access to sensitive resources and administrative functions without credentials. This flaw compromises the integrity of the authentication model and enables account takeover, data exposure, and full application compromise.

Severity: Critical
Business Risk: Loss of confidentiality, integrity, regulatory exposure, reputational damage.
Exploitability: Trivial — no credentials or specialized tools required.

2. Application Overview

Overpass is a self-hosted password-management application with a web UI and backend service layer.

Key Assets Protected: Stored credentials and user secrets; authenticated session state; administrative functionality; application configuration data.

Authentication Overview: Custom cookie-based authentication; token integrity not verified server-side; authorization decisions derived from client data. The system assumes tokens are valid instead of verifying them.

3. Threat Model

Attacker Profile: External, unauthenticated user with basic HTTP interception capability.

Objective: Access protected resources, escalate privileges, extract sensitive data.

Broken Assumptions: Client-provided cookies are trustworthy; tokens represent legitimate sessions; validation occurs elsewhere (it does not). Result: authentication integrity collapses.

4. Attack Hypotheses

1) Custom token logic might allow forgery. 2) The server may trust any token that appears present. 3) Session validation may not occur server■side. 4) Sensitive endpoints might be reachable without authentication.

5. Vulnerability Analysis

Description: The application performs authentication using a cookie that indicates session state. The server does not verify who issued the token, whether it is valid, or whether it belongs to the requesting user. Presence of the cookie alone triggers authenticated status.

Root Cause: Absence of cryptographic signing, missing server■side validation, authorization logic tied to attacker■controlled input.

Classification: OWASP Identification & Authentication Failures (A07:2021); ASVS V2 & V3.

6. Steps to Reproduce

Component: Cookie-based session handling

1) Navigate to the application as an unauthenticated user.
2) Intercept the request using a proxy or browser developer tools.
3) Add a cookie named **session** (or modify the existing one) with any non■empty value.
`Cookie: session=abc123`
4) Request a protected endpoint such as **/admin** or **/secrets**.

Observed: Access is granted.
Expected: Request is rejected and redirected to login.

7. Evidence Summary

Screenshots demonstrate access to administrative resources and protected content without valid login. Sensitive values were redacted.

8. Impact

This vulnerability enables account takeover, data exposure, privilege escalation, and potential lateral compromise of other systems via leaked credentials. In production environments, this commonly results in reportable breaches.

9. Remediation

Design Changes: Replace custom authentication with a vetted framework (OIDC, OAuth2, or secure session libraries) and treat client-supplied tokens as untrusted.

Technical Controls: Validate sessions server■side on every request; cryptographically sign tokens; bind sessions to identity and expiry; invalidate on logout; store secrets in environment variables or a dedicated secrets vault.

Standards Mapping: OWASP Top 10 (Identification & Authentication Failures); OWASP ASVS V2 & V3. This requires architectural redesign, not patching.

10. Detection & Hardening

Monitoring: Alert on admin access without preceding login; detect malformed or duplicate tokens across IPs; log rejected tokens. **Preventive Controls:** Restrict direct admin access, enforce rate limits, and ensure secure cookie flags (HttpOnly, Secure, SameSite).

11. Lessons Learned

Custom authentication mechanisms are inherently risky, and authentication controls must be validated during design. Any token accepted without cryptographic verification represents a systemic security failure.

Conclusion: Authentication integrity must be enforced server■side on every request.