High-Level Security Risk Analysis

Risk to consider based on Midpoint nature:

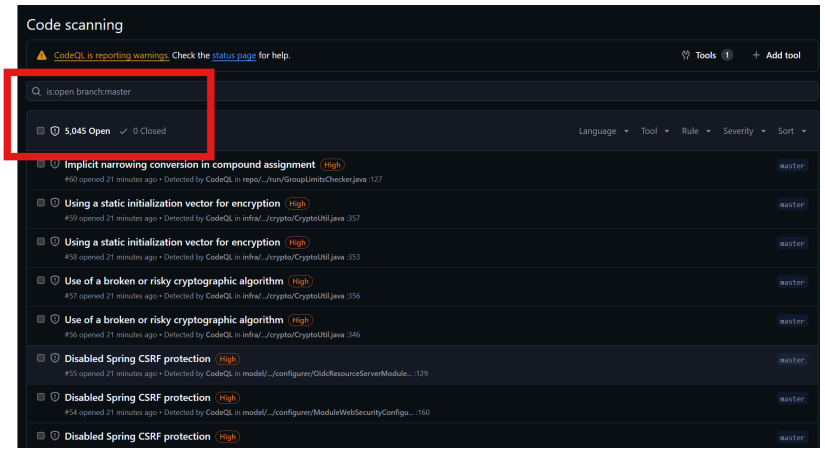
- **role and authorization misconfiguration**. Incorrect role definitions, inheritance, or overly broad permissions may allow users to access sensitive data or lead to potential privilege escalation.
- **insecure default configurations**.Use of default roles may lead to unintended consequences.
- Website UI presents risks.SAST analysis shows weaknesses, which may lead to cross-site scripting. Most consistent weakness is missing security headers. These UI weaknesses could be exploited to access data or hijack sessions if similar issues reappear.
- **Authentication and session management** in basic installation has no advanced authentication mechanisms by default and depends on integration with external identity providers for strong authentication and MFA. Password-only deployments and long-lived sessions increase the impact of credential compromise.
- **connector-based interactions with external systems** introduce risk. Authenticated users can trigger provisioning actions that depend on secure connector configuration.This could expose credentials or enable man-in-the-middle attacks.

Remediation: Analyze architecture, prepare Threat models and propose remediation steps

Part 1 - SAST Github CodeQL

Static Application Security Testing (SAST) was conducted using GitHub CodeQL across the Java and JavaScript components of the repository. Analysis found **175** real threats, from a total of 5,045 static analysis alerts.

Please see the full report in enclosed documents and on GitHub Advanced Security



"The CodeQL analysis generated more than 5,000(total 5,045) findings. The total number of findings exceeds this limit of this tool. Most of them are false ."

The absence of Critical findings indicates no immediately exploitable, systemic design flaws, while the volume and nature of High-severity findings point to **systematic weaknesses in logging practices, trust boundary enforcement, and request validation**, rather than isolated defects.

SAST - **Key Risk Themes (Attacker-Centric Interpretation)**

Rather than isolated bugs, the findings cluster into **clear, repeatable risk themes** relevant to authenticated users and administrators.

SAST - CWE Pareto Analysis (Top Security Weaknesses)

Rank	CWE ID	Weakness Name	Alerts	Typical Impact	Why It Matters Here
1	CWE-532	Insertion of Sensitive Information into Log Files	84	Information disclosure	Identity systems process credentials, tokens, and configuration data; logging these increases insider and post-compromise risk
2	CWE-117	Improper Output Neutralization for Logs	33	Log forgery, audit evasion	Undermines audit integrity and incident response, critical for IAM and governance platforms
3	CWE-352	Cross-Site Request Forgery (CSRF)	27	Unauthorized state change	Enables attackers to trigger privileged actions in authenticated user contexts
4	CWE-290	Authentication Bypass by Spoofing	8	Authentication bypass	Directly threatens trust boundaries and identity assurance
5	CWE-807	Reliance on Untrusted Inputs in Security Decisions	8	Authorization bypass	User-controlled inputs influencing security logic weaken access control guarantees
6	CWE-327	Use of Broken or Risky Cryptographic Algorithms	5	Cryptographic compromise	Weak crypto undermines confidentiality and integrity guarantees
7	CWE-022	Directory Traversal	5	Unauthorized file access	Risks exposure of configuration, secrets, or internal data

A. Sensitive Data Exposure via Logging (CWE-532, CWE-117)

The largest finding cluster relates to logging practices. Sensitive runtime information and user-influenced data are written to logs without sufficient control.

B. Trust Boundary Violations in Request Handling (CWE-352, CWE-807)

state-changing operations and security-relevant decisions rely on inputs that are not sufficiently protected or validated.

C. Authentication and Authorization Weaknesses (CWE-290)

A smaller but high-impact cluster highlights paths where authentication or identity checks can be bypassed through spoofed or user-controlled inputs. While not pervasive, these findings represent **high-leverage attack opportunities** if chained with other weaknesses.

D. Cryptographic and File System Safety Gaps (CWE-327, CWE-022)

Use of weak cryptographic algorithms and insufficient path restriction introduces risks to confidentiality and integrity, particularly for configuration files, keys, or exported data.

SAST - Recommendations and Next Steps

1. **Prioritize High-severity remediation by theme**, starting with logging controls and request validation, rather than addressing findings individually.
2. **Harden logging practices**: eliminate sensitive data from logs, enforce structured logging, and sanitize user-controlled log content.
3. **Strengthen request protection**: enforce CSRF protections consistently on all state-changing operations.
4. **Review trust decisions**: ensure authentication and authorization logic is never driven by user-controlled inputs.
5. **Modernize cryptography**: replace deprecated or risky algorithms with current, vetted standards.

SAST - Summary

> The findings indicate **mature detection of systemic risk patterns**, not a brittle or critically exposed system. The absence of Critical vulnerabilities, combined with clearly defined High-severity themes, provides a strong foundation for targeted remediation and measurable security improvement.

Dependabot - Dependency Vulnerability Assessment

The assessment identified **6 relevant dependency vulnerabilities** across backend and frontend components

Severity	Component	Vulnerability Description	Risk Context
Critical	Terracotta Quartz Scheduler	XML External Entity (XXE) Injection	Potential data exfiltration, SSRF, or denial-of-service if attacker-controlled XML is processed
High	node-tar	Race Condition via Unicode Ligature Collisions (macOS APFS)	File overwrite or corruption during archive extraction
High	node-tar	Arbitrary File Creation/Overwrite via Hardlink Path Traversal	Write access outside intended extraction directory
High	node-tar	Arbitrary File Overwrite & Symlink Poisoning	Privilege escalation or data corruption during file handling
Moderate	Bootstrap Multiselect	CSRF and Reflective XSS via Arbitrary POST Data	Client-side request forgery and reflected script execution

Risk Interpretation

- The **Critical XXE vulnerability** in the Quartz Scheduler is the most severe finding and is particularly relevant in environments where XML input may originate from external or semi-trusted sources.
- The **node-tar vulnerabilities** form a **cluster of related filesystem risks**, indicating elevated exposure during archive extraction or build-time operations, especially on developer workstations or CI systems.
- The **Bootstrap Multiselect vulnerability** affects frontend interaction integrity but does not directly compromise backend authorization logic.

No evidence suggests active exploitation within the repository; however, **known exploit primitives exist** for several of these vulnerabilities.

Github - Branch Protection Rules (Strongly Recommended)

- Require CodeQL checks to pass
- Require PR review
- Prevent direct pushes to `master`

Github - Secret Scanning

GitHub Secret Scanning identified no unresolved secrets or exposed credentials in the repository.

Part 2 - DAST:

This report presents the results of a security assessment performed using OWASP ZAP (Checkmarx edition) against the Midpoint running at `http://localhost:8080`.

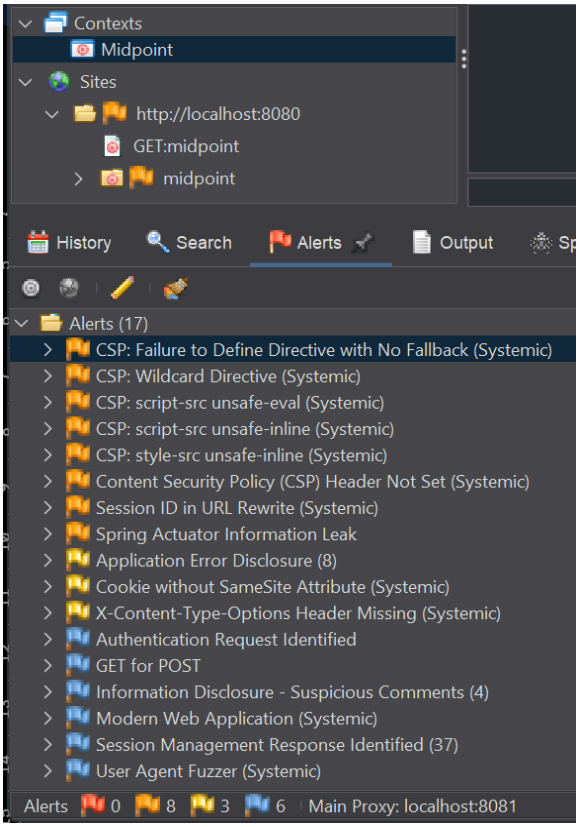
The scan identified **17 findings in total**, with **no High-risk vulnerabilities**, but several **Medium,** Low, and Informational issues.

The most significant findings are **Medium-risk vulnerabilities**, primarily related to **Content Security Policy (CSP) misconfigurations**, **session handling weaknesses**, and a **Spring Actuator information exposure**.

Additional Low-risk issues involve missing or weak security headers and minor information disclosures. Informational alerts highlight application behavior, session management patterns, and general technology fingerprinting.

Overall, while no critical security flaws were detected, the results indicate opportunities to **strengthen security posture**, particularly by improving HTTP security headers, hardening CSP configuration, and reducing unnecessary information exposure.

Please see the full ZAP report in enclosed documents.



Tools and Process.

Tools:

- GitHub
  - Dependabot
  - CodeQL
  - Secret Scanning
- ZAP - Zed Attack Proxy
- Chat GPT
- Docker

Process: methodology based on OWASP recomendations