Semgrep SAST Scan Report for All Repositories with tag portfolio-B

Report Generated at 2024-02-22 18:18

SAST Scan Summary

Findings Summary

Project Name	High/Ignored	High/Fixed	High/Removed	High/Open	Medium/Ignored	Medium/Fixed	Medium/Removed	Medium/Open	Low/Ignored	Low/Fixed	Low/Removed	Low/Open
Semgrep-Demo/js-app	0	12	0	22	0	12	0	66	0	0	0	17
Semgrep-Demo/pro-engine-demo	0	3	0	5	0	18	0	18	0	0	0	11

Semgrep SAST Scan Report for Repository: Semgrep-Demo/js-app

Report Generated at 2024-02-22 18:18

SAST Scan Summary

Vulnerability Severity	Vulnerability Count
Findings- SAST High Severity	34
Findings- SAST Medium Severity	78
Findings- SAST Low Severity	17

Findings Summary- HIGH Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
express-sequelize-injection	Detected a sequelize statement that is tainted by user-input. This could lead to SQL injection if the variable is user-controlled and is not properly sanitized. In order to prevent SQL injection, it is recommended to use parameterized queries or prepared statements.	high	unresolved	refs/pull/27/merge	data/static/codefixes/dbSchemaChallenge_1.ts#L17
express-mongo-nosqli	Detected a `../data/mongodb` statement that comes from a `req` argument. This could lead to NoSQL injection if the variable is user-controlled and is not properly sanitized. Be sure to properly sanitize the data if you absolutely must pass request data into a mongo query.	high	unresolved	refs/heads/main	routes/dataExport.ts#L61

Finding Title	Finding Description & Remediation	severity	state	ref	location
express-mongo-nosqli	Detected a `../data/mongodb` statement that comes from a `req` argument. This could lead to NoSQL injection if the variable is user-controlled and is not properly sanitized. Be sure to properly sanitize the data if you absolutely must pass request data into a mongo query.	high	unresolved	refs/heads/main	routes/likeProductReviews.ts#L18
express-mongo-nosqli	Detected a `../data/mongodb` statement that comes from a `req` argument. This could lead to NoSQL injection if the variable is user-controlled and is not properly sanitized. Be sure to properly sanitize the data if you absolutely must pass request data into a mongo query.	high	unresolved	refs/heads/main	routes/likeProductReviews.ts#L25
express-mongo-nosqli	Detected a `../data/mongodb` statement that comes from a `req` argument. This could lead to NoSQL injection if the variable is user-controlled and is not properly sanitized. Be sure to properly sanitize the data if you absolutely must pass request data into a mongo query.	high	unresolved	refs/heads/main	routes/likeProductReviews.ts#L31

Finding Title	Finding Description & Remediation	severity	state	ref	location
express-mongo-nosqli	Detected a `../data/mongodb` statement that comes from a `req` argument. This could lead to NoSQL injection if the variable is user-controlled and is not properly sanitized. Be sure to properly sanitize the data if you absolutely must pass request data into a mongo query.	high	unresolved	refs/heads/main	routes/likeProductReviews.ts#L42
express-mongo-nosqli	Detected a `../data/mongodb` statement that comes from a `req` argument. This could lead to NoSQL injection if the variable is user-controlled and is not properly sanitized. Be sure to properly sanitize the data if you absolutely must pass request data into a mongo query.	high	unresolved	refs/heads/main	routes/showProductReviews.ts#L34
express-mongo-nosqli	Detected a `../data/mongodb` statement that comes from a `req` argument. This could lead to NoSQL injection if the variable is user-controlled and is not properly sanitized. Be sure to properly sanitize the data if you absolutely must pass request data into a mongo query.	high	unresolved	refs/heads/main	routes/trackOrder.ts#L18

Finding Title	Finding Description & Remediation	severity	state	ref	location
express-mongo-nosqli	Detected a `../data/mongodb` statement that comes from a `req` argument. This could lead to NoSQL injection if the variable is user-controlled and is not properly sanitized. Be sure to properly sanitize the data if you absolutely must pass request data into a mongo query.	high	unresolved	refs/heads/main	routes/updateProductReviews.ts#L18
detected-generic-secret	Generic Secret detected	high	unresolved	refs/heads/main	data/static/users.yml#L150
express-sequelize-injection	Detected a sequelize statement that is tainted by user-input. This could lead to SQL injection if the variable is user-controlled and is not properly sanitized. In order to prevent SQL injection, it is recommended to use parameterized queries or prepared statements.	high	unresolved	refs/pull/16/merge	data/static/codefixes/dbSchemaChallenge_1.ts#L29

Finding Title	Finding Description & Remediation	severity	state	ref	location
express-sequelize-injection	Detected a sequelize statement that is tainted by user-input. This could lead to SQL injection if the variable is user-controlled and is not properly sanitized. In order to prevent SQL injection, it is recommended to use parameterized queries or prepared statements.	high	unresolved	refs/pull/16/merge	data/static/codefixes/dbSchemaChallenge_1.ts#L5
detected-artifactory-password	Artifactory token detected	high	fixed	refs/heads/main	frontend/src/assets/public/images/Welcome_Banner.svg#L1
detected-artifactory-password	Artifactory token detected	high	fixed	refs/heads/main	frontend/src/assets/public/images/Welcome_Banner.svg#L1
express-mongo-nosqli	Detected a `../data/mongodb` statement that comes from a `req` argument. This could lead to NoSQL injection if the variable is user-controlled and is not properly sanitized. Be sure to properly sanitize the data if you absolutely must pass request data into a mongo query.	high	fixed	refs/heads/main	routes/dataExport.ts#L61

Finding Title	Finding Description & Remediation	severity	state	ref	location
express-mongo-nosqli	Detected a `../data/mongodb` statement that comes from a `req` argument. This could lead to NoSQL injection if the variable is user-controlled and is not properly sanitized. Be sure to properly sanitize the data if you absolutely must pass request data into a mongo query.	high	fixed	refs/heads/main	routes/likeProductReviews.ts#L18
express-mongo-nosqli	Detected a `../data/mongodb` statement that comes from a `req` argument. This could lead to NoSQL injection if the variable is user-controlled and is not properly sanitized. Be sure to properly sanitize the data if you absolutely must pass request data into a mongo query.	high	fixed	refs/heads/main	routes/likeProductReviews.ts#L25
express-mongo-nosqli	Detected a `../data/mongodb` statement that comes from a `req` argument. This could lead to NoSQL injection if the variable is user-controlled and is not properly sanitized. Be sure to properly sanitize the data if you absolutely must pass request data into a mongo query.	high	fixed	refs/heads/main	routes/likeProductReviews.ts#L31

Finding Title	Finding Description & Remediation	severity	state	ref	location
express-mongo-nosqli	Detected a `../data/mongodb` statement that comes from a `req` argument. This could lead to NoSQL injection if the variable is user-controlled and is not properly sanitized. Be sure to properly sanitize the data if you absolutely must pass request data into a mongo query.	high	fixed	refs/heads/main	routes/likeProductReviews.ts#L42
express-mongo-nosqli	Detected a `../data/mongodb` statement that comes from a `req` argument. This could lead to NoSQL injection if the variable is user-controlled and is not properly sanitized. Be sure to properly sanitize the data if you absolutely must pass request data into a mongo query.	high	fixed	refs/heads/main	routes/showProductReviews.ts#L34
express-mongo-nosqli	Detected a `../data/mongodb` statement that comes from a `req` argument. This could lead to NoSQL injection if the variable is user-controlled and is not properly sanitized. Be sure to properly sanitize the data if you absolutely must pass request data into a mongo query.	high	fixed	refs/heads/main	routes/trackOrder.ts#L18

Finding Title	Finding Description & Remediation	severity	state	ref	location
express-mongo-nosqli	Detected a `../data/mongodb` statement that comes from a `req` argument. This could lead to NoSQL injection if the variable is user-controlled and is not properly sanitized. Be sure to properly sanitize the data if you absolutely must pass request data into a mongo query.	high	fixed	refs/heads/main	routes/updateProductReviews.ts#L18
insecure-document-method	User controlled data in methods like `innerHTML`, `outerHTML` or `document.write` is an anti-pattern that can lead to XSS vulnerabilities	high	fixed	refs/heads/main	frontend/src/assets/private/three.js#L11375
express-sequelize-injection	Detected a sequelize statement that is tainted by user-input. This could lead to SQL injection if the variable is user-controlled and is not properly sanitized. In order to prevent SQL injection, it is recommended to use parameterized queries or prepared statements.	high	unresolved	refs/heads/main	data/static/codefixes/dbSchemaChallenge_1.ts#L5

Finding Title	**Finding Description & Remediation**	**severity**	**state**	**ref**	**location**
express-sequelize-injection	Detected a sequelize statement that is tainted by user-input. This could lead to SQL injection if the variable is user-controlled and is not properly sanitized. In order to prevent SQL injection, it is recommended to use parameterized queries or prepared statements.	high	unresolved	refs/heads/main	data/static/codefixes/dbSchemaChallenge_3.ts#L11
express-sequelize-injection	Detected a sequelize statement that is tainted by user-input. This could lead to SQL injection if the variable is user-controlled and is not properly sanitized. In order to prevent SQL injection, it is recommended to use parameterized queries or prepared statements.	high	unresolved	refs/heads/main	data/static/codefixes/unionSqlInjectionChallenge_1.ts#L6
express-sequelize-injection	Detected a sequelize statement that is tainted by user-input. This could lead to SQL injection if the variable is user-controlled and is not properly sanitized. In order to prevent SQL injection, it is recommended to use parameterized queries or prepared statements.	high	unresolved	refs/heads/main	data/static/codefixes/unionSqlInjectionChallenge_3.ts#L10

Finding Title	Finding Description & Remediation	severity	state	ref	location
express-sequelize-injection	Detected a sequelize statement that is tainted by user-input. This could lead to SQL injection if the variable is user-controlled and is not properly sanitized. In order to prevent SQL injection, it is recommended to use parameterized queries or prepared statements.	high	unresolved	refs/heads/main	routes/login.ts#L36
express-sequelize-injection	Detected a sequelize statement that is tainted by user-input. This could lead to SQL injection if the variable is user-controlled and is not properly sanitized. In order to prevent SQL injection, it is recommended to use parameterized queries or prepared statements.	high	unresolved	refs/heads/main	routes/search.ts#L23
detected-generic-secret	Generic Secret detected	high	fixed	refs/heads/main	data/static/users.yml#L150
detected-jwt-token	JWT token detected	high	unresolved	refs/heads/main	frontend/src/app/app.guard.spec.ts#L40
detected-jwt-token	JWT token detected	high	unresolved	refs/heads/main	frontend/src/app/last-login-ip/last-login-ip.component.spec.ts#L50
detected-jwt-token	JWT token detected	high	unresolved	refs/heads/main	frontend/src/app/last-login-ip/last-login-ip.component.spec.ts#L56

Finding Title	Finding Description & Remediation	severity	state	ref	location
insecure-document-method	User controlled data in methods like `innerHTML`, `outerHTML` or `document.write` is an anti-pattern that can lead to XSS vulnerabilities	high	unresolved	refs/heads/main	frontend/src/hacking-instructor/index.ts#L107

Findings Summary- MEDIUM Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
path-join-resolve-traversal	Detected possible user input going into a `path.join` or `path.resolve` function. This could possibly lead to a path traversal vulnerability, where the attacker can access arbitrary files stored in the file system. Instead, be sure to sanitize or validate user input first.	medium	unresolved	refs/heads/main	routes/vulnCodeSnippet.ts#L107
path-join-resolve-traversal	Detected possible user input going into a `path.join` or `path.resolve` function. This could possibly lead to a path traversal vulnerability, where the attacker can access arbitrary files stored in the file system. Instead, be sure to sanitize or validate user input first.	medium	unresolved	refs/heads/main	routes/fileServer.ts#L33
regexp-redos	Detected `req` argument enters calls to `RegExp`. This could lead to a Regular Expression Denial of Service (ReDoS) through catastrophic backtracking. If the input is attacker controllable, this vulnerability can lead to systems being non-responsive or may crash due to ReDoS. Where possible avoid calls to `RegExp` with user input, if required ensure user input is escaped or validated.	medium	unresolved	refs/heads/main	routes/vulnCodeSnippet.ts#L104
regexp-redos	Detected `req` argument enters calls to `RegExp`. This could lead to a Regular Expression Denial of Service (ReDoS) through catastrophic backtracking. If the input is attacker controllable, this vulnerability can lead to systems being non-responsive or may crash due to ReDoS. Where possible avoid calls to `RegExp` with user input, if required ensure user input is escaped or validated.	medium	unresolved	refs/heads/main	routes/vulnCodeSnippet.ts#L108

Finding Title	Finding Description & Remediation	severity	state	ref	location
regexp-redos	Detected `req` argument enters calls to `RegExp`. This could lead to a Regular Expression Denial of Service (ReDoS) through catastrophic backtracking. If the input is attacker controllable, this vulnerability can lead to systems being non-responsive or may crash due to ReDoS. Where possible avoid calls to `RegExp` with user input, if required ensure user input is escaped or validated.	medium	unresolved	refs/heads/main	routes/vulnCodeSnippet.ts#L123
regexp-redos	Detected `req` argument enters calls to `RegExp`. This could lead to a Regular Expression Denial of Service (ReDoS) through catastrophic backtracking. If the input is attacker controllable, this vulnerability can lead to systems being non-responsive or may crash due to ReDoS. Where possible avoid calls to `RegExp` with user input, if required ensure user input is escaped or validated.	medium	unresolved	refs/heads/main	routes/vulnCodeSnippet.ts#L125
express-check-directory-listing	Directory listing/indexing is enabled, which may lead to disclosure of sensitive directories and files. It is recommended to disable directory listing unless it is a public resource. If you need directory listing, ensure that sensitive files are inaccessible when querying the resource.	medium	unresolved	refs/heads/main	server.ts#L241
express-check-directory-listing	Directory listing/indexing is enabled, which may lead to disclosure of sensitive directories and files. It is recommended to disable directory listing unless it is a public resource. If you need directory listing, ensure that sensitive files are inaccessible when querying the resource.	medium	unresolved	refs/heads/main	server.ts#L246
express-check-directory-listing	Directory listing/indexing is enabled, which may lead to disclosure of sensitive directories and files. It is recommended to disable directory listing unless it is a public resource. If you need directory listing, ensure that sensitive files are inaccessible when querying the resource.	medium	unresolved	refs/heads/main	server.ts#L250
express-path-join-resolve-traversal	Possible writing outside of the destination, make sure that the target path is nested in the intended destination	medium	unresolved	refs/heads/main	routes/fileServer.ts#L33

Finding Title	Finding Description & Remediation	severity	state	ref	location
detect-non-literal-regexp	RegExp() called with a `req` function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.	medium	unresolved	refs/heads/main	routes/vulnCodeSnippet.ts#L104
detect-non-literal-regexp	RegExp() called with a `req` function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.	medium	unresolved	refs/heads/main	routes/vulnCodeSnippet.ts#L123
detect-non-literal-regexp	RegExp() called with a `req` function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.	medium	unresolved	refs/heads/main	routes/vulnCodeSnippet.ts#L125

Finding Title	Finding Description & Remediation	severity	state	ref	location
path-join-resolve-traversal	Detected possible user input going into a \`path.join\` or \`path.resolve\` function. This could possibly lead to a path traversal vulnerability, where the attacker can access arbitrary files stored in the file system. Instead, be sure to sanitize or validate user input first.	medium	unresolved	refs/heads/main	routes/vulnCodeSnippet.ts#L33
express-check-directory-listing	Directory listing/indexing is enabled, which may lead to disclosure of sensitive directories and files. It is recommended to disable directory listing unless it is a public resource. If you need directory listing, ensure that sensitive files are inaccessible when querying the resource.	medium	fixed	refs/heads/main	server.ts#L241
express-check-directory-listing	Directory listing/indexing is enabled, which may lead to disclosure of sensitive directories and files. It is recommended to disable directory listing unless it is a public resource. If you need directory listing, ensure that sensitive files are inaccessible when querying the resource.	medium	fixed	refs/heads/main	server.ts#L246
express-check-directory-listing	Directory listing/indexing is enabled, which may lead to disclosure of sensitive directories and files. It is recommended to disable directory listing unless it is a public resource. If you need directory listing, ensure that sensitive files are inaccessible when querying the resource.	medium	fixed	refs/heads/main	server.ts#L250
path-join-resolve-traversal	Detected possible user input going into a \`path.join\` or \`path.resolve\` function. This could possibly lead to a path traversal vulnerability, where the attacker can access arbitrary files stored in the file system. Instead, be sure to sanitize or validate user input first.	medium	fixed	refs/heads/main	routes/fileServer.ts#L33
path-join-resolve-traversal	Detected possible user input going into a \`path.join\` or \`path.resolve\` function. This could possibly lead to a path traversal vulnerability, where the attacker can access arbitrary files stored in the file system. Instead, be sure to sanitize or validate user input first.	medium	fixed	refs/heads/main	routes/vulnCodeSnippet.ts#L33

Finding Title	Finding Description & Remediation	severity	state	ref	location
path-join-resolve-traversal	Detected possible user input going into a `path.join` or `path.resolve` function. This could possibly lead to a path traversal vulnerability, where the attacker can access arbitrary files stored in the file system. Instead, be sure to sanitize or validate user input first.	medium	fixed	refs/heads/main	routes/vulnCodeSnippet.ts#L107
express-res-sendfile	The application processes user-input, this is passed to res.sendFile which can allow an attacker to arbitrarily read files on the system through path traversal. It is recommended to perform input validation in addition to canonicalizing the path. This allows you to validate the path against the intended directory it should be accessing.	medium	unresolved	refs/heads/main	routes/fileServer.ts#L33
jssha-sha1	The SHA1 hashing algorithm is considered to be weak. If this is used in any sensitive operation such as password hashing, or is used to ensure data integrity (collision sensitive) then you should use a stronger hashing algorithm. For passwords, consider using `Argon2id`, `scrypt`, or `bcrypt`. For data integrity, consider using `SHA-256`.	medium	unresolved	refs/heads/main	lib/utils.ts#L97
express-res-sendfile	The application processes user-input, this is passed to res.sendFile which can allow an attacker to arbitrarily read files on the system through path traversal. It is recommended to perform input validation in addition to canonicalizing the path. This allows you to validate the path against the intended directory it should be accessing.	medium	fixed	refs/heads/main	routes/fileServer.ts#L33
no-new-privileges	Service 'app' allows for privilege escalation via setuid or setgid binaries. Add 'no-new-privileges:true' in 'security_opt' to prevent this.	medium	unresolved	refs/heads/main	docker-compose.test.yml#L7
unknown-value-with-script-tag	Cannot determine what 'subs' is and it is used with a '<script>' tag. This could be susceptible to cross-site scripting (XSS). Ensure 'subs' is not externally controlled, or sanitize this data.	medium	unresolved	refs/heads/main	routes/videoHandler.ts#L57

Finding Title	Finding Description & Remediation	severity	state	ref	location
path-join-resolve-traversal	Detected possible user input going into a \`path.join\` or \`path.resolve\` function. This could possibly lead to a path traversal vulnerability, where the attacker can access arbitrary files stored in the file system. Instead, be sure to sanitize or validate user input first.	medium	unresolved	refs/heads/main	routes/fileUpload.ts#L29
path-join-resolve-traversal	Detected possible user input going into a \`path.join\` or \`path.resolve\` function. This could possibly lead to a path traversal vulnerability, where the attacker can access arbitrary files stored in the file system. Instead, be sure to sanitize or validate user input first.	medium	unresolved	refs/heads/main	routes/keyServer.ts#L14
path-join-resolve-traversal	Detected possible user input going into a \`path.join\` or \`path.resolve\` function. This could possibly lead to a path traversal vulnerability, where the attacker can access arbitrary files stored in the file system. Instead, be sure to sanitize or validate user input first.	medium	unresolved	refs/heads/main	routes/logfileServer.ts#L14
path-join-resolve-traversal	Detected possible user input going into a \`path.join\` or \`path.resolve\` function. This could possibly lead to a path traversal vulnerability, where the attacker can access arbitrary files stored in the file system. Instead, be sure to sanitize or validate user input first.	medium	unresolved	refs/heads/main	routes/quarantineServer.ts#L14
vm-runincontext-context-injection	Make sure that unverified user data can not reach vm.runInContext.	medium	unresolved	refs/heads/main	routes/b2bOrder.ts#L22
vm-runincontext-context-injection	Make sure that unverified user data can not reach vm.runInContext.	medium	unresolved	refs/heads/main	routes/fileUpload.ts#L80

Finding Title	Finding Description & Remediation	severity	state	ref	location
express-fs-filename	The application builds a file path from potentially untrusted data, which can lead to a path traversal vulnerability. An attacker can manipulate the file path which the application uses to access files. If the application does not validate user input and sanitize file paths, sensitive files such as configuration or user data can be accessed, potentially creating or overwriting files. To prevent this vulnerability, validate and sanitize any input that is used to create references to file paths. Also, enforce strict file access controls. For example, choose privileges allowing public-facing applications to access only the required files.	medium	unresolved	refs/heads/main	routes/profileImageFileUpload.ts#L28
express-fs-filename	The application builds a file path from potentially untrusted data, which can lead to a path traversal vulnerability. An attacker can manipulate the file path which the application uses to access files. If the application does not validate user input and sanitize file paths, sensitive files such as configuration or user data can be accessed, potentially creating or overwriting files. To prevent this vulnerability, validate and sanitize any input that is used to create references to file paths. Also, enforce strict file access controls. For example, choose privileges allowing public-facing applications to access only the required files.	medium	unresolved	refs/heads/main	routes/profileImageUrlUpload.ts#L31

Finding Title	Finding Description & Remediation	severity	state	ref	location
express-fs-filename	The application builds a file path from potentially untrusted data, which can lead to a path traversal vulnerability. An attacker can manipulate the file path which the application uses to access files. If the application does not validate user input and sanitize file paths, sensitive files such as configuration or user data can be accessed, potentially creating or overwriting files. To prevent this vulnerability, validate and sanitize any input that is used to create references to file paths. Also, enforce strict file access controls. For example, choose privileges allowing public-facing applications to access only the required files.	medium	unresolved	refs/heads/main	routes/vulnCodeFixes.ts#L81
express-fs-filename	The application builds a file path from potentially untrusted data, which can lead to a path traversal vulnerability. An attacker can manipulate the file path which the application uses to access files. If the application does not validate user input and sanitize file paths, sensitive files such as configuration or user data can be accessed, potentially creating or overwriting files. To prevent this vulnerability, validate and sanitize any input that is used to create references to file paths. Also, enforce strict file access controls. For example, choose privileges allowing public-facing applications to access only the required files.	medium	unresolved	refs/heads/main	routes/vulnCodeFixes.ts#L82
express-jwt-not-revoked	No token revoking configured for \`express-jwt\`. A leaked token could still be used and unable to be revoked. Consider using function as the \`isRevoked\` option.	medium	unresolved	refs/heads/main	lib/insecurity.ts#L53
express-jwt-not-revoked	No token revoking configured for \`express-jwt\`. A leaked token could still be used and unable to be revoked. Consider using function as the \`isRevoked\` option.	medium	unresolved	refs/heads/main	lib/insecurity.ts#L54

Finding Title	Finding Description & Remediation	severity	state	ref	location
express-path-join-resolve-traversal	Possible writing outside of the destination, make sure that the target path is nested in the intended destination	medium	unresolved	refs/heads/main	routes/dataErasure.ts#L69
express-path-join-resolve-traversal	Possible writing outside of the destination, make sure that the target path is nested in the intended destination	medium	unresolved	refs/heads/main	routes/keyServer.ts#L14
express-path-join-resolve-traversal	Possible writing outside of the destination, make sure that the target path is nested in the intended destination	medium	unresolved	refs/heads/main	routes/logfileServer.ts#L14
express-path-join-resolve-traversal	Possible writing outside of the destination, make sure that the target path is nested in the intended destination	medium	unresolved	refs/heads/main	routes/quarantineServer.ts#L14
express-res-sendfile	The application processes user-input, this is passed to res.sendFile which can allow an attacker to arbitrarily read files on the system through path traversal. It is recommended to perform input validation in addition to canonicalizing the path. This allows you to validate the path against the intended directory it should be accessing.	medium	unresolved	refs/heads/main	routes/keyServer.ts#L14
express-res-sendfile	The application processes user-input, this is passed to res.sendFile which can allow an attacker to arbitrarily read files on the system through path traversal. It is recommended to perform input validation in addition to canonicalizing the path. This allows you to validate the path against the intended directory it should be accessing.	medium	unresolved	refs/heads/main	routes/logfileServer.ts#L14
express-res-sendfile	The application processes user-input, this is passed to res.sendFile which can allow an attacker to arbitrarily read files on the system through path traversal. It is recommended to perform input validation in addition to canonicalizing the path. This allows you to validate the path against the intended directory it should be accessing.	medium	unresolved	refs/heads/main	routes/quarantineServer.ts#L14

Finding Title	Finding Description & Remediation	severity	state	ref	location
express-ssrf	The following request request.get() was found to be crafted from user-input `req` which can lead to Server-Side Request Forgery (SSRF) vulnerabilities. It is recommended where possible to not allow user-input to craft the base request, but to be treated as part of the path or query parameter. When user-input is necessary to craft the request, it is recommeneded to follow OWASP best practices to prevent abuse.	medium	unresolved	refs/heads/main	routes/profileImageUrlUpload.ts#L23
express-insecure-template-usage	User data from `req` is being compiled into the template, which can lead to a Server Side Template Injection (SSTI) vulnerability.	medium	unresolved	refs/heads/main	routes/userProfile.ts#L56
session-fixation	Detected `req` argument which enters `res.cookie`, this can lead to session fixation vulnerabilities if an attacker can control the cookie value. This vulnerability can lead to unauthorized access to accounts, and in some esoteric cases, Cross-Site-Scripting (XSS). Users should not be able to influence cookies directly, for session cookies, they should be generated securely using an approved session management library. If the cookie does need to be set by a user, consider using an allow-list based approach to restrict the cookies which can be set.	medium	unresolved	refs/heads/main	lib/insecurity.ts#L211
angular-route-bypass-security-trust	Untrusted input could be used to tamper with a web page rendering, which can lead to a Cross-site scripting (XSS) vulnerability. XSS vulnerabilities occur when untrusted input executes malicious JavaScript code, leading to issues such as account compromise and sensitive information leakage. Validate the user input, perform contextual output encoding, or sanitize the input. A popular library used to prevent XSS is DOMPurify. You can also use libraries and frameworks such as Angular, Vue, and React, which offer secure defaults when rendering input.	medium	unresolved	refs/heads/main	frontend/src/app/search-result/search-result.component.ts#L152

Finding Title	Finding Description & Remediation	severity	state	ref	location
open-redirect-deepsemgrep	The application builds a URL using user-controlled input which can lead to an open redirect vulnerability. An attacker can manipulate the URL and redirect users to an arbitrary domain. Open redirect vulnerabilities can lead to issues such as Cross-site scripting (XSS) or redirecting to a malicious domain for activities such as phishing to capture users' credentials. To prevent this vulnerability perform strict input validation of the domain against an allowlist of approved domains. Notify a user in your application that they are leaving the website. Display a domain where they are redirected to the user. A user can then either accept or deny the redirect to an untrusted site.	medium	unresolved	refs/heads/main	routes/redirect.ts#L19
ssrf-deepsemgrep	Untrusted input might be used to build an HTTP request, which can lead to a Server-side request forgery (SSRF) vulnerability. SSRF allows an attacker to send crafted requests from the server side to other internal or external systems. SSRF can lead to unauthorized access to sensitive data and, in some cases, allow the attacker to control applications or systems that trust the vulnerable service. To prevent this vulnerability, avoid allowing user input to craft the base request. Instead, treat it as part of the path or query parameter and encode it appropriately. When user input is necessary to prepare the HTTP request, perform strict input validation. Additionally, whenever possible, use allowlists to only interact with expected, trusted domains.	medium	unresolved	refs/heads/main	routes/profileImageUrlUpload.ts#L23
express-open-redirect	The application redirects to a URL specified by user-supplied input `query` that is not validated. This could redirect users to malicious locations. Consider using an allow-list approach to validate URLs, or warn users they are being redirected to a third-party website.	medium	unresolved	refs/heads/main	routes/redirect.ts#L19

Finding Title	Finding Description & Remediation	severity	state	ref	location
node-sequelize-hardcoded-secret-argument	A hard-coded credential was detected. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module).	medium	unresolved	refs/pull/1/merge	models/index.ts#L31
eval-detected	Detected the use of eval(). eval() can be dangerous if used to evaluate dynamic content. If this content can be input from outside the program, this may be a code injection vulnerability. Ensure evaluated content is not definable by external sources.	medium	unresolved	refs/heads/main	routes/captcha.ts#L23
eval-detected	Detected the use of eval(). eval() can be dangerous if used to evaluate dynamic content. If this content can be input from outside the program, this may be a code injection vulnerability. Ensure evaluated content is not definable by external sources.	medium	unresolved	refs/heads/main	routes/userProfile.ts#L36
express-detect-notevil-usage	Detected usage of the \`notevil\` package, which is unmaintained and has vulnerabilities. Using any sort of \`eval()\` functionality can be very dangerous, but if you must, the \`eval\` package is an up to date alternative. Be sure that only trusted input reaches an \`eval()\` function.	medium	unresolved	refs/heads/main	routes/b2bOrder.ts#L22
express-libxml-vm-noent	Detected use of parseXml() function with the \`noent\` field set to \`true\`. This can lead to an XML External Entities (XXE) attack if untrusted data is passed into it.	medium	unresolved	refs/heads/main	routes/fileUpload.ts#L80
express-open-redirect	The application redirects to a URL specified by user-supplied input \`query\` that is not validated. This could redirect users to malicious locations. Consider using an allow-list approach to validate URLs, or warn users they are being redirected to a third-party website.	medium	fixed	refs/heads/main	routes/redirect.ts#L19

Finding Title	Finding Description & Remediation	severity	state	ref	location
template-explicit-unescape	Detected an explicit unescape in a Pug template, using either '!=' or '!{...}'. If external data can reach these locations, your application is exposed to a cross-site scripting (XSS) vulnerability. If you must do this, ensure no external data can reach this location.	medium	unresolved	refs/heads/main	views/promotionVideo.pug#L79
jwt-exposed-data	The object is passed strictly to jsonwebtoken.sign(...) Make sure that sensitive information is not exposed through JWT token payload.	medium	unresolved	refs/heads/main	lib/insecurity.ts#L55
hardcoded-jwt-secret	A hard-coded credential was detected. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module).	medium	fixed	refs/heads/main	lib/insecurity.ts#L55
detected-private-key	A hard-coded credential was detected. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module).	medium	fixed	refs/heads/main	lib/insecurity.ts#L22
detect-non-literal-regexp	RegExp() called with a `key` function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.	medium	unresolved	refs/heads/main	routes/vulnCodeSnippet.ts#L104

Finding Title	Finding Description & Remediation	severity	state	ref	location
detect-non-literal-regexp	RegExp() called with a `key` function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.	medium	unresolved	refs/heads/main	routes/vulnCodeSnippet.ts#L123
detect-non-literal-regexp	RegExp() called with a `key` function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.	medium	unresolved	refs/heads/main	routes/vulnCodeSnippet.ts#L125
hardcoded-hmac-key	Detected a hardcoded hmac key. Avoid hardcoding secrets and consider using an alternate option such as reading the secret from a config file or using an environment variable.	medium	unresolved	refs/heads/main	lib/insecurity.ts#L43
hardcoded-hmac-key	Detected a hardcoded hmac key. Avoid hardcoding secrets and consider using an alternate option such as reading the secret from a config file or using an environment variable.	medium	fixed	refs/heads/main	lib/insecurity.ts#L166
path-join-resolve-traversal	Detected possible user input going into a `path.join` or `path.resolve` function. This could possibly lead to a path traversal vulnerability, where the attacker can access arbitrary files stored in the file system. Instead, be sure to sanitize or validate user input first.	medium	unresolved	refs/heads/main	data/datacreator.ts#L41

Finding Title	Finding Description & Remediation	severity	state	ref	location
path-join-resolve-traversal	Detected possible user input going into a \`path.join\` or \`path.resolve\` function. This could possibly lead to a path traversal vulnerability, where the attacker can access arbitrary files stored in the file system. Instead, be sure to sanitize or validate user input first.	medium	unresolved	refs/heads/main	lib/startup/restoreOverwrittenFilesWithOriginals.ts#L30
path-join-resolve-traversal	Detected possible user input going into a \`path.join\` or \`path.resolve\` function. This could possibly lead to a path traversal vulnerability, where the attacker can access arbitrary files stored in the file system. Instead, be sure to sanitize or validate user input first.	medium	unresolved	refs/heads/main	lib/startup/validatePreconditions.ts#L95
path-join-resolve-traversal	Detected possible user input going into a \`path.join\` or \`path.resolve\` function. This could possibly lead to a path traversal vulnerability, where the attacker can access arbitrary files stored in the file system. Instead, be sure to sanitize or validate user input first.	medium	unresolved	refs/heads/main	routes/dataErasure.ts#L69
path-join-resolve-traversal	Detected possible user input going into a \`path.join\` or \`path.resolve\` function. This could possibly lead to a path traversal vulnerability, where the attacker can access arbitrary files stored in the file system. Instead, be sure to sanitize or validate user input first.	medium	unresolved	refs/heads/main	routes/fileUpload.ts#L39
path-join-resolve-traversal	Detected possible user input going into a \`path.join\` or \`path.resolve\` function. This could possibly lead to a path traversal vulnerability, where the attacker can access arbitrary files stored in the file system. Instead, be sure to sanitize or validate user input first.	medium	unresolved	refs/heads/main	routes/order.ts#L46
path-join-resolve-traversal	Detected possible user input going into a \`path.join\` or \`path.resolve\` function. This could possibly lead to a path traversal vulnerability, where the attacker can access arbitrary files stored in the file system. Instead, be sure to sanitize or validate user input first.	medium	unresolved	refs/heads/main	routes/vulnCodeSnippet.ts#L33

Finding Title	Finding Description & Remediation	severity	state	ref	location
prototype-pollution-loop	Possibility of prototype polluting function detected. By adding or modifying attributes of an object prototype, it is possible to create attributes that exist on every object, or replace critical attributes with malicious ones. This can be problematic if the software depends on existence or non-existence of certain attributes, or uses pre-defined attributes of object prototype (such as hasOwnProperty, toString or valueOf). Possible mitigations might be: freezing the object prototype, using an object without prototypes (via Object.create(null)), blocking modifications of attributes that resolve to object prototype, using Map instead of object.	medium	unresolved	refs/heads/main	frontend/src/hacking-instructor/helpers/helpers.ts#L36
unknown-value-with-script-tag	Cannot determine what 'subs' is and it is used with a '<script>' tag. This could be susceptible to cross-site scripting (XSS). Ensure 'subs' is not externally controlled, or sanitize this data.	medium	unresolved	refs/heads/main	routes/videoHandler.ts#L69
node-sequelize-hardcoded-secret-argument	A secret is hard-coded in the application. Secrets stored in source code, such as credentials, identifiers, and other types of sensitive data, can be leaked and used by internal or external malicious actors. Use environment variables to securely provide credentials and other secrets or retrieve them from a secure vault or Hardware Security Module (HSM).	medium	unresolved	refs/heads/main	models/index.ts#L31
no-new-privileges	Service 'app' allows for privilege escalation via setuid or setgid binaries. Add 'no-new-privileges:true' in 'security_opt' to prevent this.	medium	fixed	refs/heads/main	docker-compose.test.yml#L7
writable-filesystem-service	Service 'app' is running with a writable root filesystem. This may allow malicious applications to download and run additional payloads, or modify container files. If an application inside a container has to save something temporarily consider using a tmpfs. Add 'read_only: true' to this service to prevent this.	medium	unresolved	refs/heads/main	docker-compose.test.yml#L7

Findings Summary- LOW Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
generic-api-key	A gitleaks generic-api-key was detected which attempts to identify hard-coded credentials. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module). This rule can introduce a lot of false positives, it is not recommended to be used in PR comments.	low	unresolved	refs/heads/main	data/static/users.yml#L88
generic-api-key	A gitleaks generic-api-key was detected which attempts to identify hard-coded credentials. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module). This rule can introduce a lot of false positives, it is not recommended to be used in PR comments.	low	unresolved	refs/heads/main	data/static/users.yml#L150
generic-api-key	A gitleaks generic-api-key was detected which attempts to identify hard-coded credentials. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module). This rule can introduce a lot of false positives, it is not recommended to be used in PR comments.	low	unresolved	refs/heads/main	frontend/src/app/oauth/oauth.component.spec.ts#L85

Finding Title	Finding Description & Remediation	severity	state	ref	location
generic-api-key	A gitleaks generic-api-key was detected which attempts to identify hard-coded credentials. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module). This rule can introduce a lot of false positives, it is not recommended to be used in PR comments.	low	unresolved	refs/heads/main	frontend/src/app/oauth/oauth.component.spec.ts#L85
generic-api-key	A gitleaks generic-api-key was detected which attempts to identify hard-coded credentials. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module). This rule can introduce a lot of false positives, it is not recommended to be used in PR comments.	low	unresolved	refs/heads/main	frontend/src/app/oauth/oauth.component.spec.ts#L92
generic-api-key	A gitleaks generic-api-key was detected which attempts to identify hard-coded credentials. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module). This rule can introduce a lot of false positives, it is not recommended to be used in PR comments.	low	unresolved	refs/heads/main	routes/login.ts#L66

Finding Title	Finding Description & Remediation	severity	state	ref	location
hashicorp-tf-password	A gitleaks hashicorp-tf-password was detected which attempts to identify hard-coded credentials. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module).	low	unresolved	refs/heads/main	frontend/src/app/change-password/change-password.component.html#L33
hashicorp-tf-password	A gitleaks hashicorp-tf-password was detected which attempts to identify hard-coded credentials. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module).	low	unresolved	refs/heads/main	frontend/src/app/change-password/change-password.component.html#L51
hashicorp-tf-password	A gitleaks hashicorp-tf-password was detected which attempts to identify hard-coded credentials. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module).	low	unresolved	refs/heads/main	frontend/src/app/forgot-password/forgot-password.component.html#L53
hashicorp-tf-password	A gitleaks hashicorp-tf-password was detected which attempts to identify hard-coded credentials. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module).	low	unresolved	refs/heads/main	frontend/src/app/forgot-password/forgot-password.component.html#L66

Finding Title	Finding Description & Remediation	severity	state	ref	location
hashicorp-tf-password	A gitleaks hashicorp-tf-password was detected which attempts to identify hard-coded credentials. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module).	low	unresolved	refs/heads/main	frontend/src/app/login/login.component.html#L27
hashicorp-tf-password	A gitleaks hashicorp-tf-password was detected which attempts to identify hard-coded credentials. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module).	low	unresolved	refs/heads/main	frontend/src/app/register/register.component.html#L25
jwt	A gitleaks jwt was detected which attempts to identify hard-coded credentials. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module).	low	unresolved	refs/heads/main	frontend/src/app/app.guard.spec.ts#L40
jwt	A gitleaks jwt was detected which attempts to identify hard-coded credentials. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module).	low	unresolved	refs/heads/main	frontend/src/app/last-login-ip/last-login-ip.component.spec.ts#L50

Finding Title	Finding Description & Remediation	severity	state	ref	location
detect-replaceall-sanitization	Detected a call to `replaceAll()` in an attempt to HTML escape the string `tableData[i].description`. Manually sanitizing input through a manually built list can be circumvented in many situations, and it's better to use a well known sanitization library such as `sanitize-html` or `DOMPurify`.	low	unresolved	refs/heads/main	data/static/codefixes/restfulXssChallenge_2.ts#L59
detect-replaceall-sanitization	Detected a call to `replaceAll()` in an attempt to HTML escape the string `tableData[i].description.replaceAll('<', '&lt;')`. Manually sanitizing input through a manually built list can be circumvented in many situations, and it's better to use a well known sanitization library such as `sanitize-html` or `DOMPurify`.	low	unresolved	refs/heads/main	data/static/codefixes/restfulXssChallenge_2.ts#L59
express-check-csurf-middleware-usage	A CSRF middleware was not detected in your express application. Ensure you are either using one such as `csurf` or `csrf` (see rule references) and/or you are properly doing CSRF validation in your routes with a token or cookies.	low	unresolved	refs/heads/main	server.ts#L91

Semgrep SAST Scan Report for Repository: Semgrep-Demo/pro-engine-demo

Report Generated at 2024-02-22 18:18

SAST Scan Summary

Vulnerability Severity	Vulnerability Count
Findings- SAST High Severity	8
Findings- SAST Medium Severity	36
Findings- SAST Low Severity	11

Findings Summary- HIGH Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
tainted-sql-string	User data flows into this manually-constructed SQL string. User data can be safely inserted into SQL strings using prepared statements or an object-relational mapper (ORM). Manually-constructed SQL strings is a possible indicator of SQL injection, which could let an attacker steal or manipulate data from the database. Instead, use prepared statements (`connection.PreparedStatement`) or a safe library.	high	unresolved	refs/heads/main	src/main/java/hawk/service/UserSearchService.java#L30
tainted-sql-string	User data flows into this manually-constructed SQL string. User data can be safely inserted into SQL strings using prepared statements or an object-relational mapper (ORM). Manually-constructed SQL strings is a possible indicator of SQL injection, which could let an attacker steal or manipulate data from the database. Instead, use prepared statements (`connection.PreparedStatement`) or a safe library.	high	fixed	refs/heads/main	src/main/java/hawk/service/UserSearchService.java#L30
crlf-injection-logs-deepsemgrep-javaorg-copy	When data from an untrusted source is put into a logger and not neutralized correctly, an attacker could forge log entries or include malicious content. Please use the Jsoup.clean() function to sanitize data.	high	unresolved	refs/heads/main	src/main/java/hawk/api/jwt/JwtLog4jController.java#L24

Finding Title	Finding Description & Remediation	severity	state	ref	location
tainted-sql-string	User data flows into this manually-constructed SQL string. User data can be safely inserted into SQL strings using prepared statements or an object-relational mapper (ORM). Manually-constructed SQL strings is a possible indicator of SQL injection, which could let an attacker steal or manipulate data from the database. Instead, use prepared statements (`connection.PreparedStatement`) or a safe library.	high	fixed	refs/heads/main	src/main/java/hawk/service/UserSearchService.java#L30
crlf-injection-logs-deepsemgrep	When data from an untrusted source is put into a logger and not neutralized correctly, an attacker could forge log entries or include malicious content.	high	unresolved	refs/heads/main	src/main/java/hawk/api/jwt/JwtLog4jController.java#L24
detected-generic-secret	Generic Secret detected	high	fixed	refs/heads/main	src/main/java/hawk/api/jwt/JwtTokenProvider.java#L21
detected-private-key	Private Key detected. This is a sensitive credential and should not be hardcoded here. Instead, store this in a separate, private file.	high	unresolved	refs/heads/main	src/main/resources/keyStore.pem#L5
spring-actuator-fully-enabled	Spring Boot Actuator is fully enabled. This exposes sensitive endpoints such as /actuator/env, /actuator/logfile, /actuator/heapdump and others. Unless you have Spring Security enabled or another means to protect these endpoints, this functionality is available without authentication, causing a significant security risk.	high	unresolved	refs/heads/main	src/main/resources/application-postgresql.properties#L36

Findings Summary- MEDIUM Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
django-no-csrf-token	Manually-created forms in django templates should specify a csrf_token to prevent CSRF attacks	medium	unresolved	refs/heads/main	src/main/resources/templates/general.html#L30
django-no-csrf-token	Manually-created forms in django templates should specify a csrf_token to prevent CSRF attacks	medium	unresolved	refs/heads/main	src/main/resources/templates/login-form-multi.html#L15
django-no-csrf-token	Manually-created forms in django templates should specify a csrf_token to prevent CSRF attacks	medium	unresolved	refs/heads/main	src/main/resources/templates/login.html#L15
django-no-csrf-token	Manually-created forms in django templates should specify a csrf_token to prevent CSRF attacks	medium	unresolved	refs/heads/main	src/main/resources/templates/search.html#L14
django-no-csrf-token	Manually-created forms in django templates should specify a csrf_token to prevent CSRF attacks	medium	unresolved	refs/heads/main	src/main/resources/templates/user-search.html#L14
django-no-csrf-token	Manually-created forms in django templates should specify a csrf_token to prevent CSRF attacks	medium	fixed	refs/heads/main	src/main/resources/templates/general.html#L30
django-no-csrf-token	Manually-created forms in django templates should specify a csrf_token to prevent CSRF attacks	medium	fixed	refs/heads/main	src/main/resources/templates/login-form-multi.html#L15
django-no-csrf-token	Manually-created forms in django templates should specify a csrf_token to prevent CSRF attacks	medium	fixed	refs/heads/main	src/main/resources/templates/login.html#L15
django-no-csrf-token	Manually-created forms in django templates should specify a csrf_token to prevent CSRF attacks	medium	fixed	refs/heads/main	src/main/resources/templates/search.html#L14
django-no-csrf-token	Manually-created forms in django templates should specify a csrf_token to prevent CSRF attacks	medium	fixed	refs/heads/main	src/main/resources/templates/user-search.html#L14
django-no-csrf-token	Manually-created forms in django templates should specify a csrf_token to prevent CSRF attacks	medium	fixed	refs/heads/main	src/main/resources/templates/general.html#L30
django-no-csrf-token	Manually-created forms in django templates should specify a csrf_token to prevent CSRF attacks	medium	fixed	refs/heads/main	src/main/resources/templates/login-form-multi.html#L15
django-no-csrf-token	Manually-created forms in django templates should specify a csrf_token to prevent CSRF attacks	medium	fixed	refs/heads/main	src/main/resources/templates/login.html#L15
django-no-csrf-token	Manually-created forms in django templates should specify a csrf_token to prevent CSRF attacks	medium	fixed	refs/heads/main	src/main/resources/templates/search.html#L14
django-no-csrf-token	Manually-created forms in django templates should specify a csrf_token to prevent CSRF attacks	medium	fixed	refs/heads/main	src/main/resources/templates/user-search.html#L14
django-no-csrf-token	Manually-created forms in django templates should specify a csrf_token to prevent CSRF attacks	medium	fixed	refs/heads/main	src/main/resources/templates/general.html#L30

Finding Title	Finding Description & Remediation	severity	state	ref	location
django-no-csrf-token	Manually-created forms in django templates should specify a csrf_token to prevent CSRF attacks	medium	fixed	refs/heads/main	src/main/resources/templates/login-form-multi.html#L15
django-no-csrf-token	Manually-created forms in django templates should specify a csrf_token to prevent CSRF attacks	medium	fixed	refs/heads/main	src/main/resources/templates/login.html#L15
django-no-csrf-token	Manually-created forms in django templates should specify a csrf_token to prevent CSRF attacks	medium	fixed	refs/heads/main	src/main/resources/templates/search.html#L14
django-no-csrf-token	Manually-created forms in django templates should specify a csrf_token to prevent CSRF attacks	medium	fixed	refs/heads/main	src/main/resources/templates/user-search.html#L14
third-party-action-not-pinned-to-commit-sha	An action sourced from a third-party repository on GitHub is not pinned to a full length commit SHA. Pinning an action to a full length commit SHA is currently the only way to use an action as an immutable release. Pinning to a particular SHA helps mitigate the risk of a bad actor adding a backdoor to the action's repository, as they would need to generate a SHA-1 collision for a valid Git object payload.	medium	unresolved	refs/heads/main	.github/workflows/hawkscan.yml#L23
cookie-missing-httponly	A cookie was detected without setting the 'HttpOnly' flag. The 'HttpOnly' flag for cookies instructs the browser to forbid client-side scripts from reading the cookie. Set the 'HttpOnly' flag by calling 'cookie.setHttpOnly(true);'	medium	unresolved	refs/heads/main	src/main/java/hawk/controller/LoginController.java#L57
cookie-missing-samesite	The application does not appear to verify inbound requests which can lead to a Cross-site request forgery (CSRF) vulnerability. If the application uses cookie-based authentication, an attacker can trick users into sending authenticated HTTP requests without their knowledge from any arbitrary domain they visit. To prevent this vulnerability start by identifying if the framework or library leveraged has built-in features or offers plugins for CSRF protection. CSRF tokens should be unique and securely random. The \`Synchronizer Token\` or \`Double Submit Cookie\` patterns with defense-in-depth mechanisms such as the \`sameSite\` cookie flag can help prevent CSRF. For more information, see: [Cross-site request forgery prevention] (https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html)	medium	unresolved	refs/heads/main	src/main/java/hawk/controller/LoginController.java#L57
cookie-missing-secure-flag	A cookie was detected without setting the 'secure' flag. The 'secure' flag for cookies prevents the client from transmitting the cookie over insecure channels such as HTTP. Set the 'secure' flag by calling 'new Cookie("XLOGINID", cookieCode).setSecure(true);'	medium	unresolved	refs/heads/main	src/main/java/hawk/controller/LoginController.java#L57

Finding Title	Finding Description & Remediation	severity	state	ref	location
no-new-privileges	Service 'db' allows for privilege escalation via setuid or setgid binaries. Add 'no-new-privileges:true' in 'security_opt' to prevent this.	medium	unresolved	refs/heads/main	docker-compose.yml#L3
no-new-privileges	Service 'javavulny' allows for privilege escalation via setuid or setgid binaries. Add 'no-new-privileges:true' in 'security_opt' to prevent this.	medium	unresolved	refs/heads/main	docker-compose.yml#L12
jsch-hardcoded-secret	A secret is hard-coded in the application. Secrets stored in source code, such as credentials, identifiers, and other types of sensitive data, can be leaked and used by internal or external malicious actors. Use environment variables to securely provide credentials and other secrets or retrieve them from a secure vault or Hardware Security Module (HSM).	medium	unresolved	refs/pull/3/merge	src/main/java/hawk/api/vuln1.java#L19
jsch-hardcoded-secret	A secret is hard-coded in the application. Secrets stored in source code, such as credentials, identifiers, and other types of sensitive data, can be leaked and used by internal or external malicious actors. Use environment variables to securely provide credentials and other secrets or retrieve them from a secure vault or Hardware Security Module (HSM).	medium	unresolved	refs/pull/3/merge	src/main/java/hawk/api/vuln1.java#L23
spring-csrf-disabled	CSRF protection is disabled for this configuration. This is a security risk.	medium	unresolved	refs/heads/main	src/main/java/hawk/MultiHttpSecurityConfig.java#L47
spring-csrf-disabled	CSRF protection is disabled for this configuration. This is a security risk.	medium	unresolved	refs/heads/main	src/main/java/hawk/MultiHttpSecurityConfig.java#L88
spring-csrf-disabled	CSRF protection is disabled for this configuration. This is a security risk.	medium	unresolved	refs/heads/main	src/main/java/hawk/MultiHttpSecurityConfig.java#L110
no-new-privileges	Service 'db' allows for privilege escalation via setuid or setgid binaries. Add 'no-new-privileges:true' in 'security_opt' to prevent this.	medium	fixed	refs/heads/main	docker-compose.yml#L3
no-new-privileges	Service 'javavulny' allows for privilege escalation via setuid or setgid binaries. Add 'no-new-privileges:true' in 'security_opt' to prevent this.	medium	fixed	refs/heads/main	docker-compose.yml#L12
writable-filesystem-service	Service 'db' is running with a writable root filesystem. This may allow malicious applications to download and run additional payloads, or modify container files. If an application inside a container has to save something temporarily consider using a tmpfs. Add 'read_only: true' to this service to prevent this.	medium	unresolved	refs/heads/main	docker-compose.yml#L3

Finding Title	Finding Description & Remediation	severity	state	ref	location
writable-filesystem-service	Service 'javavulny' is running with a writable root filesystem. This may allow malicious applications to download and run additional payloads, or modify container files. If an application inside a container has to save something temporarily consider using a tmpfs. Add 'read_only: true' to this service to prevent this.	medium	unresolved	refs/heads/main	docker-compose.yml#L12
third-party-action-not-pinned-to-commit-sha	An action sourced from a third-party repository on GitHub is not pinned to a full length commit SHA. Pinning an action to a full length commit SHA is currently the only way to use an action as an immutable release. Pinning to a particular SHA helps mitigate the risk of a bad actor adding a backdoor to the action's repository, as they would need to generate a SHA-1 collision for a valid Git object payload.	medium	fixed	refs/heads/main	.github/workflows/hawkscan.yml#L23

Findings Summary- LOW Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
hashicorp-tf-password	A gitleaks hashicorp-tf-password was detected which attempts to identify hard-coded credentials. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module).	low	unresolved	refs/heads/main	integration_tests/conf_files/stackhawk-jsv-json-token.yml#L20
hashicorp-tf-password	A gitleaks hashicorp-tf-password was detected which attempts to identify hard-coded credentials. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module).	low	unresolved	refs/heads/main	stackhawk-actions.yml#L22
hashicorp-tf-password	A gitleaks hashicorp-tf-password was detected which attempts to identify hard-coded credentials. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module).	low	unresolved	refs/heads/main	stackhawk.d/stackhawk-auth-form-cookie.yml#L14
hashicorp-tf-password	A gitleaks hashicorp-tf-password was detected which attempts to identify hard-coded credentials. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module).	low	unresolved	refs/heads/main	stackhawk.d/stackhawk-auth-json-token.yml#L14

Finding Title	Finding Description & Remediation	severity	state	ref	location
hashicorp-tf-password	A gitleaks hashicorp-tf-password was detected which attempts to identify hard-coded credentials. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module).	low	unresolved	refs/heads/main	stackhawk.d/stackhawk-custom-spider-curl.yml#L20
hashicorp-tf-password	A gitleaks hashicorp-tf-password was detected which attempts to identify hard-coded credentials. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module).	low	unresolved	refs/heads/main	stackhawk.d/stackhawk-custom-spider-newman.yml#L20
hashicorp-tf-password	A gitleaks hashicorp-tf-password was detected which attempts to identify hard-coded credentials. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module).	low	unresolved	refs/heads/main	stackhawk.d/stackhawk-jsv-form-cookie.yml#L22
hashicorp-tf-password	A gitleaks hashicorp-tf-password was detected which attempts to identify hard-coded credentials. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module).	low	unresolved	refs/heads/main	stackhawk.d/stackhawk-jsv-json-token.yml#L19

Finding Title	Finding Description & Remediation	severity	state	ref	location
hashicorp-tf-password	A gitleaks hashicorp-tf-password was detected which attempts to identify hard-coded credentials. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module).	low	unresolved	refs/heads/main	stackhawk.d/stackhawk-openapi.yml#L17
hashicorp-tf-password	A gitleaks hashicorp-tf-password was detected which attempts to identify hard-coded credentials. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module).	low	unresolved	refs/heads/main	stackhawk.yml#L20
private-key	A gitleaks private-key was detected which attempts to identify hard-coded credentials. It is not recommended to store credentials in source-code, as this risks secrets being leaked and used by either an internal or external malicious adversary. It is recommended to use environment variables to securely provide credentials or retrieve credentials from a secure vault or HSM (Hardware Security Module).	low	unresolved	refs/heads/main	src/main/resources/keyStore.pem#L5