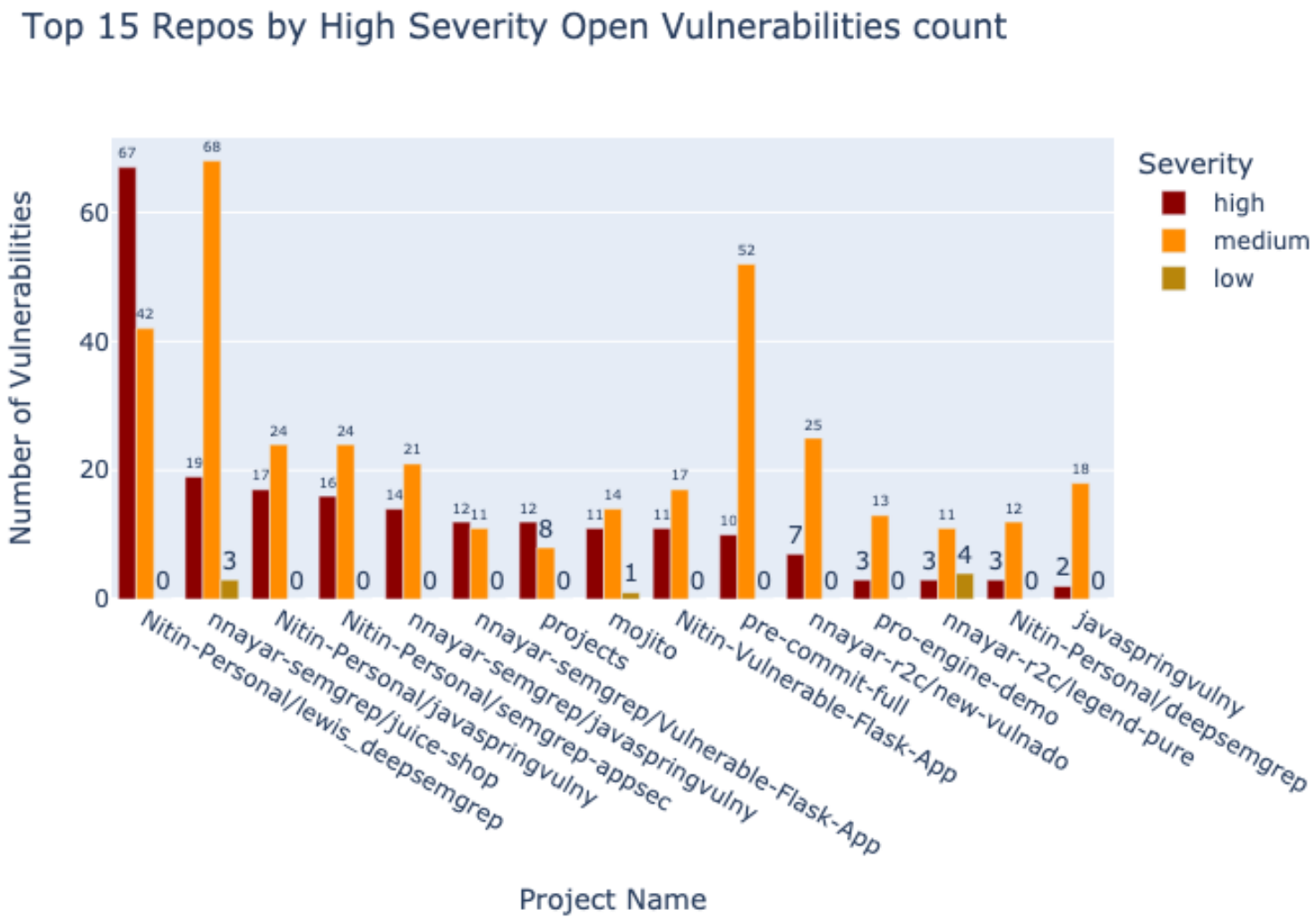
Semgrep SAST Scan Report for All Repositories with tag portfolio-B

Report Generated at 2024-02-28 09:28

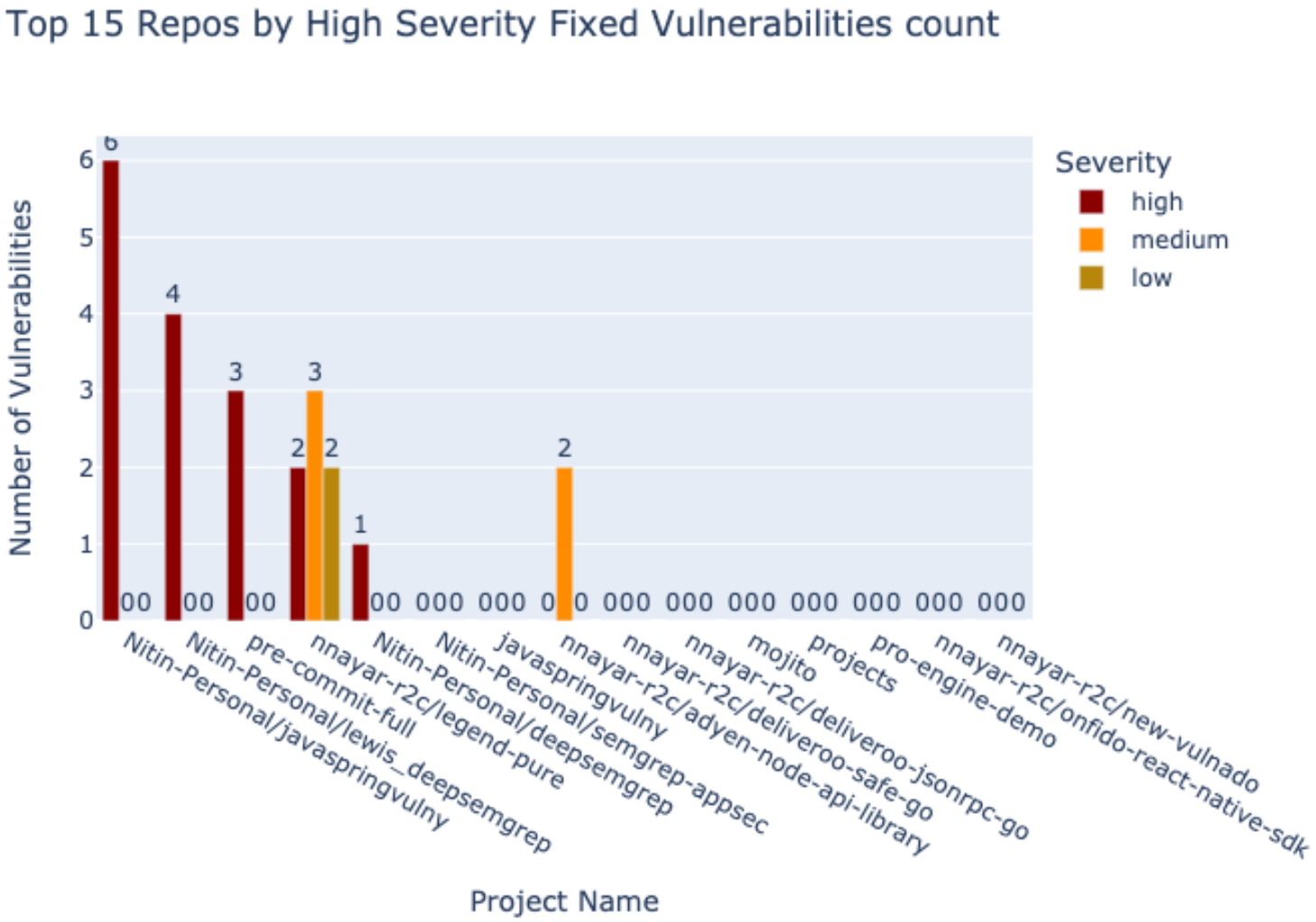
SAST Findings Summary

Project	Security Grade	Open-HIGH	Open-MEDIUM	Open-LOW	Fixed-HIGH	Fixed-MEDIUM	Fixed-LOW
Nitin-Personal/lewis_deepsemgrep	F	67	42	0	4	0	0
nnayar-semgrep/juice-shop	D	19	68	3	0	7	0
Nitin-Personal/javaspringvulny	D	17	24	0	6	0	0
Nitin-Personal/semgrep-appsec	D	16	24	0	0	0	0
nnayar-semgrep/javaspringvulny	D	14	21	0	0	10	0
nnayar-semgrep/Vulnerable-Flask-App	D	12	11	0	0	0	0
projects	D	12	8	0	0	0	0
mojito	D	11	14	1	0	0	0
Nitin-Vulnerable-Flask-App	D	11	17	0	0	0	0
pre-commit-full	D	10	52	0	3	0	0
nnayar-r2c/new-vulnado	C	7	25	0	0	0	0
pro-engine-demo	B	3	13	0	0	0	0
nnayar-r2c/legend-pure	B	3	11	4	2	3	2
Nitin-Personal/deepsemgrep	B	3	12	0	1	0	0
javaspringvulny	B	2	18	0	0	0	0
nnayar-r2c/finos-common-domain-model	B	1	4	0	0	0	0
nnayar-r2c/onfido-react-native-sdk	B	1	12	0	0	0	0
nnayar-r2c/onfido-java	B	0	16	0	0	0	0
nnayar-r2c/experian-nodejs	A	0	5	1	0	0	0
nnayar-r2c/Skyscanner-backpack-foundations	B	0	17	0	0	17	0
nnayar-r2c/deliveroo-jsonrpc-go	A	0	3	0	0	0	0
nnayar-r2c/deliveroo-safe-go	A	0	5	0	0	0	0
nnayar-r2c/adyen-node-api-library	B	0	23	0	0	2	0

Top 15 Projects with High Severity Open Vulnerability Count



Top 15 Projects with High Severity Fixed Vulnerability Count



Semgrep SAST Scan Report for Repository: Nitin-Personal/deepsemgrep

Report Generated at 2024-02-28 09:27

SAST Scan Summary

Vulnerability Severity	Vulnerability Count
Findings- SAST High Severity	4
Findings- SAST Medium Severity	12
Findings- SAST Low Severity	0

Findings Summary- HIGH Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
code-string-concat	Found data from an Express or Next web request flowing to `eval`. If this data is user-controllable this can lead to execution of arbitrary system commands in the context of your application process. Avoid `eval` whenever possible.	high	unresolved	main	lib/lib.js#L4
code-string-concat	Found data from an Express or Next web request flowing to `eval`. If this data is user-controllable this can lead to execution of arbitrary system commands in the context of your application process. Avoid `eval` whenever possible.	high	unresolved	main	test2.js#L6
code-string-concat	Found data from an Express or Next web request flowing to `eval`. If this data is user-controllable this can lead to execution of arbitrary system commands in the context of your application process. Avoid `eval` whenever possible.	high	unresolved	main	test2.js#L20
detected-github-token	GitHub Token detected	high	fixed	refs/heads/main	.github/workflows/scheduled_scan_all_repos.yml#L42

Findings Summary- MEDIUM Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
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/master	lib/lib.js#L4
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/master	lib/lib.js#L7
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/master	test2.js#L6
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/master	test2.js#L20
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	main	lib/lib.js#L4
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	main	lib/lib.js#L7
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	main	test2.js#L6
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	main	test2.js#L20
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	lib/lib.js#L4
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	lib/lib.js#L7
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	test2.js#L6
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	test2.js#L20

Findings Summary- LOW Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location

Semgrep SAST Scan Report for Repository: Nitin-Personal/javaspringvulny

Report Generated at 2024-02-28 09:27

SAST Scan Summary

Vulnerability Severity	Vulnerability Count
Findings- SAST High Severity	23
Findings- SAST Medium Severity	24
Findings- SAST Low Severity	0

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	main	src/main/java/hawk/service/UserSearchService.java#L30
spring-tainted-path-traversal	The application builds a file path from potentially untrusted data, which can lead to a path traversal vulnerability. An attacker can manipulate the 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. In Java, you may also consider using a utility method such as `org.apache.commons.io.FilenameUtils.getName(...)` to only retrieve the file name from the path.	high	unresolved	main	src/main/java/hawk/service/tainted-1.java#L42
spring-tainted-path-traversal	The application builds a file path from potentially untrusted data, which can lead to a path traversal vulnerability. An attacker can manipulate the 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. In Java, you may also consider using a utility method such as `org.apache.commons.io.FilenameUtils.getName(...)` to only retrieve the file name from the path.	high	unresolved	main	src/main/java/hawk/service/tainted-1.java#L59
spring-tainted-path-traversal	The application builds a file path from potentially untrusted data, which can lead to a path traversal vulnerability. An attacker can manipulate the 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. In Java, you may also consider using a utility method such as `org.apache.commons.io.FilenameUtils.getName(...)` to only retrieve the file name from the path.	high	unresolved	main	src/main/java/hawk/service/tainted-1.java#L73
spring-tainted-path-traversal	The application builds a file path from potentially untrusted data, which can lead to a path traversal vulnerability. An attacker can manipulate the 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. In Java, you may also consider using a utility method such as `org.apache.commons.io.FilenameUtils.getName(...)` to only retrieve the file name from the path.	high	unresolved	main	src/main/java/hawk/service/tainted_file_path.java#L42
spring-tainted-path-traversal	The application builds a file path from potentially untrusted data, which can lead to a path traversal vulnerability. An attacker can manipulate the 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. In Java, you may also consider using a utility method such as `org.apache.commons.io.FilenameUtils.getName(...)` to only retrieve the file name from the path.	high	unresolved	main	src/main/java/hawk/service/tainted_file_path.java#L59
spring-tainted-path-traversal	The application builds a file path from potentially untrusted data, which can lead to a path traversal vulnerability. An attacker can manipulate the 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. In Java, you may also consider using a utility method such as `org.apache.commons.io.FilenameUtils.getName(...)` to only retrieve the file name from the path.	high	unresolved	main	src/main/java/hawk/service/tainted_file_path.java#L73

Finding Title	Finding Description & Remediation	severity	state	ref	location
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	main	src/main/resources/keyStore.pem#L5
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	main	src/main/java/hawk/api/jwt/JwtLog4jController.java#L24
tainted-file-path	Detected user input controlling a file path. An attacker could control the location of this file, to include going backwards in the directory with '../'. To address this, ensure that user-controlled variables in file paths are sanitized. You may also consider using a utility method such as org.apache.commons.io.FilenameUtils.getName(...) to only retrieve the file name from the path.	high	unresolved	main	src/main/java/hawk/service/tainted-1.java#L42
tainted-file-path	Detected user input controlling a file path. An attacker could control the location of this file, to include going backwards in the directory with '../'. To address this, ensure that user-controlled variables in file paths are sanitized. You may also consider using a utility method such as org.apache.commons.io.FilenameUtils.getName(...) to only retrieve the file name from the path.	high	unresolved	main	src/main/java/hawk/service/tainted-1.java#L59
tainted-file-path	Detected user input controlling a file path. An attacker could control the location of this file, to include going backwards in the directory with '../'. To address this, ensure that user-controlled variables in file paths are sanitized. You may also consider using a utility method such as org.apache.commons.io.FilenameUtils.getName(...) to only retrieve the file name from the path.	high	unresolved	main	src/main/java/hawk/service/tainted-1.java#L73
tainted-file-path	Detected user input controlling a file path. An attacker could control the location of this file, to include going backwards in the directory with '../'. To address this, ensure that user-controlled variables in file paths are sanitized. You may also consider using a utility method such as org.apache.commons.io.FilenameUtils.getName(...) to only retrieve the file name from the path.	high	unresolved	main	src/main/java/hawk/service/tainted-1.java#L117
tainted-file-path	Detected user input controlling a file path. An attacker could control the location of this file, to include going backwards in the directory with '../'. To address this, ensure that user-controlled variables in file paths are sanitized. You may also consider using a utility method such as org.apache.commons.io.FilenameUtils.getName(...) to only retrieve the file name from the path.	high	unresolved	main	src/main/java/hawk/service/tainted_file_path.java#L42
tainted-file-path	Detected user input controlling a file path. An attacker could control the location of this file, to include going backwards in the directory with '../'. To address this, ensure that user-controlled variables in file paths are sanitized. You may also consider using a utility method such as org.apache.commons.io.FilenameUtils.getName(...) to only retrieve the file name from the path.	high	unresolved	main	src/main/java/hawk/service/tainted_file_path.java#L59
tainted-file-path	Detected user input controlling a file path. An attacker could control the location of this file, to include going backwards in the directory with '../'. To address this, ensure that user-controlled variables in file paths are sanitized. You may also consider using a utility method such as org.apache.commons.io.FilenameUtils.getName(...) to only retrieve the file name from the path.	high	unresolved	main	src/main/java/hawk/service/tainted_file_path.java#L73
tainted-file-path	Detected user input controlling a file path. An attacker could control the location of this file, to include going backwards in the directory with '../'. To address this, ensure that user-controlled variables in file paths are sanitized. You may also consider using a utility method such as org.apache.commons.io.FilenameUtils.getName(...) to only retrieve the file name from the path.	high	unresolved	main	src/main/java/hawk/service/tainted_file_path.java#L117
spring-tainted-path-traversal	The application builds a file path from potentially untrusted data, which can lead to a path traversal vulnerability. An attacker can manipulate the 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. In Java, you may also consider using a utility method such as `org.apache.commons.io.FilenameUtils.getName(...)` to only retrieve the file name from the path.	high	fixed	main	src/main/java/hawk/service/tainted-1.java#L42
spring-tainted-path-traversal	The application builds a file path from potentially untrusted data, which can lead to a path traversal vulnerability. An attacker can manipulate the 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. In Java, you may also consider using a utility method such as `org.apache.commons.io.FilenameUtils.getName(...)` to only retrieve the file name from the path.	high	fixed	main	src/main/java/hawk/service/tainted-1.java#L59

Finding Title	Finding Description & Remediation	severity	state	ref	location
spring-tainted-path-traversal	The application builds a file path from potentially untrusted data, which can lead to a path traversal vulnerability. An attacker can manipulate the 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. In Java, you may also consider using a utility method such as `org.apache.commons.io.FilenameUtils.getName(...)` to only retrieve the file name from the path.	high	fixed	main	src/main/java/hawk/service/tainted-1.java#L73
spring-tainted-path-traversal	The application builds a file path from potentially untrusted data, which can lead to a path traversal vulnerability. An attacker can manipulate the 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. In Java, you may also consider using a utility method such as `org.apache.commons.io.FilenameUtils.getName(...)` to only retrieve the file name from the path.	high	fixed	main	src/main/java/hawk/service/tainted_file_path.java#L42
spring-tainted-path-traversal	The application builds a file path from potentially untrusted data, which can lead to a path traversal vulnerability. An attacker can manipulate the 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. In Java, you may also consider using a utility method such as `org.apache.commons.io.FilenameUtils.getName(...)` to only retrieve the file name from the path.	high	fixed	main	src/main/java/hawk/service/tainted_file_path.java#L59
spring-tainted-path-traversal	The application builds a file path from potentially untrusted data, which can lead to a path traversal vulnerability. An attacker can manipulate the 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. In Java, you may also consider using a utility method such as `org.apache.commons.io.FilenameUtils.getName(...)` to only retrieve the file name from the path.	high	fixed	main	src/main/java/hawk/service/tainted_file_path.java#L73

Findings Summary- MEDIUM Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	main	src/main/java/hawk/service/tainted-1.java#L86
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	main	src/main/java/hawk/service/tainted-1.java#L110
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	main	src/main/java/hawk/service/tainted_file_path.java#L86
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	main	src/main/java/hawk/service/tainted_file_path.java#L110
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	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	main	src/main/java/hawk/controller/LoginController.java#L57
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	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	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	main	src/main/java/hawk/controller/LoginController.java#L57
spring-csrf-disabled	CSRF protection is disabled for this configuration. This is a security risk.	medium	unresolved	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	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	main	src/main/java/hawk/MultiHttpSecurityConfig.java#L110
spring-csrf-disabled	CSRF protection is disabled for this configuration. This is a security risk.	medium	unresolved	main	src/main/java/hawk/MultiHttpSecurityConfig.java#L181

Finding Title	Finding Description & Remediation	severity	state	ref	location
unrestricted-request-mapping	Detected a method annotated with 'RequestMapping' that does not specify the HTTP method. CSRF protections are not enabled for GET, HEAD, TRACE, or OPTIONS, and by default all HTTP methods are allowed when the HTTP method is not explicitly specified. This means that a method that performs state changes could be vulnerable to CSRF attacks. To mitigate, add the 'method' field and specify the HTTP method (such as 'RequestMethod.POST').	medium	unresolved	main	src/main/java/hawk/service/tainted-1.java#L36
unrestricted-request-mapping	Detected a method annotated with 'RequestMapping' that does not specify the HTTP method. CSRF protections are not enabled for GET, HEAD, TRACE, or OPTIONS, and by default all HTTP methods are allowed when the HTTP method is not explicitly specified. This means that a method that performs state changes could be vulnerable to CSRF attacks. To mitigate, add the 'method' field and specify the HTTP method (such as 'RequestMethod.POST').	medium	unresolved	main	src/main/java/hawk/service/tainted_file_path.java#L36
django-no-csrf-token	Manually-created forms in django templates should specify a csrf_token to prevent CSRF attacks	medium	unresolved	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	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	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	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	main	src/main/resources/templates/user-search.html#L14
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	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	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	main	docker-compose.yml#L3
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	main	docker-compose.yml#L12

Findings Summary- LOW Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location

Semgrep SAST Scan Report for Repository: Nitin-Personal/lewis_deepsemgrep

Report Generated at 2024-02-28 09:27

SAST Scan Summary

Vulnerability Severity	Vulnerability Count
Findings- SAST High Severity	71
Findings- SAST Medium Severity	42
Findings- SAST Low Severity	0

Findings Summary- HIGH Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
tainted-cmd-from-http-request-deepsemgrep	Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent this vulnerability, avoid executing OS commands with user input. If the execution of OS commands is unavoidable, validate and sanitize the user input, and use safe methods for executing the commands. For more information, see [Java command injection prevention](https://semgrep.dev/docs/cheat-sheets/java-command-injection/).	high	unresolved	refs/heads/master	tainted.java#L67
tainted-cmd-from-http-request-deepsemgrep	Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent this vulnerability, avoid executing OS commands with user input. If the execution of OS commands is unavoidable, validate and sanitize the user input, and use safe methods for executing the commands. For more information, see [Java command injection prevention](https://semgrep.dev/docs/cheat-sheets/java-command-injection/).	high	unresolved	refs/heads/master	tainted.java#L176
tainted-cmd-from-http-request-deepsemgrep	Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent this vulnerability, avoid executing OS commands with user input. If the execution of OS commands is unavoidable, validate and sanitize the user input, and use safe methods for executing the commands. For more information, see [Java command injection prevention](https://semgrep.dev/docs/cheat-sheets/java-command-injection/).	high	unresolved	refs/heads/master	tainted.java#L262
tainted-code-injection-from-http-request-deepsemgrep	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	refs/heads/master	tainted-code-injection.java#L67
tainted-code-injection-from-http-request	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	refs/heads/master	tainted-code-injection.java#L67
tainted-cmd-from-http-request-deepsemgrep	Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent this vulnerability, avoid executing OS commands with user input. If the execution of OS commands is unavoidable, validate and sanitize the user input, and use safe methods for executing the commands. For more information, see [Java command injection prevention](https://semgrep.dev/docs/cheat-sheets/java-command-injection/).	high	unresolved	main	tainted.java#L67
tainted-cmd-from-http-request-deepsemgrep	Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent this vulnerability, avoid executing OS commands with user input. If the execution of OS commands is unavoidable, validate and sanitize the user input, and use safe methods for executing the commands. For more information, see [Java command injection prevention](https://semgrep.dev/docs/cheat-sheets/java-command-injection/).	high	unresolved	main	tainted.java#L176
tainted-cmd-from-http-request-deepsemgrep	Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent this vulnerability, avoid executing OS commands with user input. If the execution of OS commands is unavoidable, validate and sanitize the user input, and use safe methods for executing the commands. For more information, see [Java command injection prevention](https://semgrep.dev/docs/cheat-sheets/java-command-injection/).	high	unresolved	main	tainted.java#L262

Finding Title	Finding Description & Remediation	severity	state	ref	location
tainted-code-injection-from-http-request-deepsemgrep	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	main	tainted-code-injection.java#L67
tainted-code-injection-from-http-request	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	main	tainted-code-injection.java#L67
command-injection-process-builder	A formatted or concatenated string was detected as input to a ProcessBuilder call. This is dangerous if a variable is controlled by user input and could result in a command injection. Ensure your variables are not controlled by users or sufficiently sanitized.	high	unresolved	refs/heads/main	tainted.java#L67
command-injection-process-builder	A formatted or concatenated string was detected as input to a ProcessBuilder call. This is dangerous if a variable is controlled by user input and could result in a command injection. Ensure your variables are not controlled by users or sufficiently sanitized.	high	unresolved	refs/heads/main	tainted.java#L259
command-injection-process-builder	A formatted or concatenated string was detected as input to a ProcessBuilder call. This is dangerous if a variable is controlled by user input and could result in a command injection. Ensure your variables are not controlled by users or sufficiently sanitized.	high	unresolved	refs/heads/main	tainted.java#L312
tainted-cmd-from-http-request	Detected input from a HTTPServletRequest going into a 'ProcessBuilder' or 'exec' command. This could lead to command injection if variables passed into the exec commands are not properly sanitized. Instead, avoid using these OS commands with user-supplied input, or, if you must use these commands, use a whitelist of specific values.	high	unresolved	refs/heads/main	tainted.java#L63
tainted-cmd-from-http-request	Detected input from a HTTPServletRequest going into a 'ProcessBuilder' or 'exec' command. This could lead to command injection if variables passed into the exec commands are not properly sanitized. Instead, avoid using these OS commands with user-supplied input, or, if you must use these commands, use a whitelist of specific values.	high	unresolved	refs/heads/main	tainted.java#L115
tainted-cmd-from-http-request	Detected input from a HTTPServletRequest going into a 'ProcessBuilder' or 'exec' command. This could lead to command injection if variables passed into the exec commands are not properly sanitized. Instead, avoid using these OS commands with user-supplied input, or, if you must use these commands, use a whitelist of specific values.	high	unresolved	refs/heads/main	tainted.java#L176
tainted-cmd-from-http-request	Detected input from a HTTPServletRequest going into a 'ProcessBuilder' or 'exec' command. This could lead to command injection if variables passed into the exec commands are not properly sanitized. Instead, avoid using these OS commands with user-supplied input, or, if you must use these commands, use a whitelist of specific values.	high	unresolved	refs/heads/main	tainted.java#L257
tainted-cmd-from-http-request-deepsemgrep	Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent this vulnerability, avoid executing OS commands with user input. If the execution of OS commands is unavoidable, validate and sanitize the user input, and use safe methods for executing the commands. For more information, see [Java command injection prevention](https://semgrep.dev/docs/cheat-sheets/java-command-injection/).	high	unresolved	refs/heads/main	tainted.java#L115
tainted-cmd-from-http-request-deepsemgrep	Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent this vulnerability, avoid executing OS commands with user input. If the execution of OS commands is unavoidable, validate and sanitize the user input, and use safe methods for executing the commands. For more information, see [Java command injection prevention](https://semgrep.dev/docs/cheat-sheets/java-command-injection/).	high	fixed	refs/heads/main	tainted.java#L176

Finding Title	Finding Description & Remediation	severity	state	ref	location
tainted-cmd-from-http-request	Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent command injection, avoid executing OS commands with user input. If this is unavoidable, validate and sanitize the user input, and use safe methods for executing the commands. For more information, see [JavaScript command injection prevention](https://semgrep.dev/docs/cheat-sheets/java-command-injection/)	high	unresolved	refs/heads/main	tainted.java#L63
tainted-cmd-from-http-request	Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent command injection, avoid executing OS commands with user input. If this is unavoidable, validate and sanitize the user input, and use safe methods for executing the commands. For more information, see [JavaScript command injection prevention](https://semgrep.dev/docs/cheat-sheets/java-command-injection/)	high	unresolved	refs/heads/main	tainted.java#L115
tainted-cmd-from-http-request	Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent command injection, avoid executing OS commands with user input. If this is unavoidable, validate and sanitize the user input, and use safe methods for executing the commands. For more information, see [JavaScript command injection prevention](https://semgrep.dev/docs/cheat-sheets/java-command-injection/)	high	unresolved	refs/heads/main	tainted.java#L176
tainted-cmd-from-http-request	Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent command injection, avoid executing OS commands with user input. If this is unavoidable, validate and sanitize the user input, and use safe methods for executing the commands. For more information, see [JavaScript command injection prevention](https://semgrep.dev/docs/cheat-sheets/java-command-injection/)	high	unresolved	refs/heads/main	tainted.java#L257
tainted-code-injection-from-http-request-deepsemgrep	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	refs/heads/main	tainted-code-injection.java#L42
tainted-code-injection-from-http-request-deepsemgrep	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	refs/heads/main	tainted-code-injection.java#L52
tainted-code-injection-from-http-request-deepsemgrep	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	refs/heads/main	tainted-code-injection.java#L80
tainted-code-injection-from-http-request-deepsemgrep	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	refs/heads/main	tainted-code-injection.java#L96
tainted-code-injection-from-http-request-deepsemgrep	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	refs/heads/main	tainted-code-injection.java#L118

Finding Title	Finding Description & Remediation	severity	state	ref	location
tainted-code-injection-from-http-request	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	refs/heads/main	tainted-code-injection.java#L42
tainted-code-injection-from-http-request	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	refs/heads/main	tainted-code-injection.java#L52
command-injection-process-builder	A formatted or concatenated string was detected as input to a ProcessBuilder call. This is dangerous if a variable is controlled by user input and could result in a command injection. Ensure your variables are not controlled by users or sufficiently sanitized.	high	unresolved	main	tainted.java#L67
command-injection-process-builder	A formatted or concatenated string was detected as input to a ProcessBuilder call. This is dangerous if a variable is controlled by user input and could result in a command injection. Ensure your variables are not controlled by users or sufficiently sanitized.	high	unresolved	main	tainted.java#L259
command-injection-process-builder	A formatted or concatenated string was detected as input to a ProcessBuilder call. This is dangerous if a variable is controlled by user input and could result in a command injection. Ensure your variables are not controlled by users or sufficiently sanitized.	high	unresolved	main	tainted.java#L312
tainted-cmd-from-http-request	Detected input from a HTTPServletRequest going into a 'ProcessBuilder' or 'exec' command. This could lead to command injection if variables passed into the exec commands are not properly sanitized. Instead, avoid using these OS commands with user-supplied input, or, if you must use these commands, use a whitelist of specific values.	high	unresolved	main	tainted.java#L63
tainted-cmd-from-http-request	Detected input from a HTTPServletRequest going into a 'ProcessBuilder' or 'exec' command. This could lead to command injection if variables passed into the exec commands are not properly sanitized. Instead, avoid using these OS commands with user-supplied input, or, if you must use these commands, use a whitelist of specific values.	high	unresolved	main	tainted.java#L115
tainted-cmd-from-http-request	Detected input from a HTTPServletRequest going into a 'ProcessBuilder' or 'exec' command. This could lead to command injection if variables passed into the exec commands are not properly sanitized. Instead, avoid using these OS commands with user-supplied input, or, if you must use these commands, use a whitelist of specific values.	high	unresolved	main	tainted.java#L176
tainted-cmd-from-http-request	Detected input from a HTTPServletRequest going into a 'ProcessBuilder' or 'exec' command. This could lead to command injection if variables passed into the exec commands are not properly sanitized. Instead, avoid using these OS commands with user-supplied input, or, if you must use these commands, use a whitelist of specific values.	high	unresolved	main	tainted.java#L257
tainted-cmd-from-http-request-deepsemgrep	Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent this vulnerability, avoid executing OS commands with user input. If the execution of OS commands is unavoidable, validate and sanitize the user input, and use safe methods for executing the commands. For more information, see [Java command injection prevention](https://semgrep.dev/docs/cheat-sheets/java-command-injection/).	high	unresolved	main	tainted.java#L115
tainted-cmd-from-http-request-deepsemgrep	Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent this vulnerability, avoid executing OS commands with user input. If the execution of OS commands is unavoidable, validate and sanitize the user input, and use safe methods for executing the commands. For more information, see [Java command injection prevention](https://semgrep.dev/docs/cheat-sheets/java-command-injection/).	high	fixed	main	tainted.java#L176

Finding Title	Finding Description & Remediation	severity	state	ref	location
tainted-cmd-from-http-request	Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent command injection, avoid executing OS commands with user input. If this is unavoidable, validate and sanitize the user input, and use safe methods for executing the commands. For more information, see [JavaScript command injection prevention](https://semgrep.dev/docs/cheat-sheets/java-command-injection/)	high	unresolved	main	tainted.java#L63
tainted-cmd-from-http-request	Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent command injection, avoid executing OS commands with user input. If this is unavoidable, validate and sanitize the user input, and use safe methods for executing the commands. For more information, see [JavaScript command injection prevention](https://semgrep.dev/docs/cheat-sheets/java-command-injection/)	high	unresolved	main	tainted.java#L115
tainted-cmd-from-http-request	Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent command injection, avoid executing OS commands with user input. If this is unavoidable, validate and sanitize the user input, and use safe methods for executing the commands. For more information, see [JavaScript command injection prevention](https://semgrep.dev/docs/cheat-sheets/java-command-injection/)	high	unresolved	main	tainted.java#L176
tainted-cmd-from-http-request	Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent command injection, avoid executing OS commands with user input. If this is unavoidable, validate and sanitize the user input, and use safe methods for executing the commands. For more information, see [JavaScript command injection prevention](https://semgrep.dev/docs/cheat-sheets/java-command-injection/)	high	unresolved	main	tainted.java#L257
tainted-code-injection-from-http-request-deepsemgrep	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	main	tainted-code-injection.java#L42
tainted-code-injection-from-http-request-deepsemgrep	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	main	tainted-code-injection.java#L52
tainted-code-injection-from-http-request-deepsemgrep	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	main	tainted-code-injection.java#L80
tainted-code-injection-from-http-request-deepsemgrep	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	main	tainted-code-injection.java#L96
tainted-code-injection-from-http-request-deepsemgrep	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	main	tainted-code-injection.java#L118

Finding Title	Finding Description & Remediation	severity	state	ref	location
tainted-code-injection-from-http-request	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	main	tainted-code-injection.java#L42
tainted-code-injection-from-http-request	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	main	tainted-code-injection.java#L52
command-injection-process-builder	A formatted or concatenated string was detected as input to a ProcessBuilder call. This is dangerous if a variable is controlled by user input and could result in a command injection. Ensure your variables are not controlled by users or sufficiently sanitized.	high	unresolved	refs/heads/master	tainted.java#L67
command-injection-process-builder	A formatted or concatenated string was detected as input to a ProcessBuilder call. This is dangerous if a variable is controlled by user input and could result in a command injection. Ensure your variables are not controlled by users or sufficiently sanitized.	high	unresolved	refs/heads/master	tainted.java#L259
command-injection-process-builder	A formatted or concatenated string was detected as input to a ProcessBuilder call. This is dangerous if a variable is controlled by user input and could result in a command injection. Ensure your variables are not controlled by users or sufficiently sanitized.	high	unresolved	refs/heads/master	tainted.java#L312
tainted-cmd-from-http-request	Detected input from a HTTPServletRequest going into a 'ProcessBuilder' or 'exec' command. This could lead to command injection if variables passed into the exec commands are not properly sanitized. Instead, avoid using these OS commands with user-supplied input, or, if you must use these commands, use a whitelist of specific values.	high	unresolved	refs/heads/master	tainted.java#L63
tainted-cmd-from-http-request	Detected input from a HTTPServletRequest going into a 'ProcessBuilder' or 'exec' command. This could lead to command injection if variables passed into the exec commands are not properly sanitized. Instead, avoid using these OS commands with user-supplied input, or, if you must use these commands, use a whitelist of specific values.	high	unresolved	refs/heads/master	tainted.java#L115
tainted-cmd-from-http-request	Detected input from a HTTPServletRequest going into a 'ProcessBuilder' or 'exec' command. This could lead to command injection if variables passed into the exec commands are not properly sanitized. Instead, avoid using these OS commands with user-supplied input, or, if you must use these commands, use a whitelist of specific values.	high	unresolved	refs/heads/master	tainted.java#L176
tainted-cmd-from-http-request	Detected input from a HTTPServletRequest going into a 'ProcessBuilder' or 'exec' command. This could lead to command injection if variables passed into the exec commands are not properly sanitized. Instead, avoid using these OS commands with user-supplied input, or, if you must use these commands, use a whitelist of specific values.	high	unresolved	refs/heads/master	tainted.java#L257
tainted-cmd-from-http-request-deepsemgrep	Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent this vulnerability, avoid executing OS commands with user input. If the execution of OS commands is unavoidable, validate and sanitize the user input, and use safe methods for executing the commands. For more information, see [Java command injection prevention](https://semgrep.dev/docs/cheat-sheets/java-command-injection/).	high	unresolved	refs/heads/master	tainted.java#L115
tainted-cmd-from-http-request-deepsemgrep	Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent this vulnerability, avoid executing OS commands with user input. If the execution of OS commands is unavoidable, validate and sanitize the user input, and use safe methods for executing the commands. For more information, see [Java command injection prevention](https://semgrep.dev/docs/cheat-sheets/java-command-injection/).	high	fixed	refs/heads/master	tainted.java#L176

Finding Title	Finding Description & Remediation	severity	state	ref	location
tainted-cmd-from-http-request	Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent command injection, avoid executing OS commands with user input. If this is unavoidable, validate and sanitize the user input, and use safe methods for executing the commands. For more information, see [JavaScript command injection prevention](https://semgrep.dev/docs/cheat-sheets/java-command-injection/)	high	unresolved	refs/heads/master	tainted.java#L63
tainted-cmd-from-http-request	Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent command injection, avoid executing OS commands with user input. If this is unavoidable, validate and sanitize the user input, and use safe methods for executing the commands. For more information, see [JavaScript command injection prevention](https://semgrep.dev/docs/cheat-sheets/java-command-injection/)	high	unresolved	refs/heads/master	tainted.java#L115
tainted-cmd-from-http-request	Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent command injection, avoid executing OS commands with user input. If this is unavoidable, validate and sanitize the user input, and use safe methods for executing the commands. For more information, see [JavaScript command injection prevention](https://semgrep.dev/docs/cheat-sheets/java-command-injection/)	high	unresolved	refs/heads/master	tainted.java#L176
tainted-cmd-from-http-request	Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent command injection, avoid executing OS commands with user input. If this is unavoidable, validate and sanitize the user input, and use safe methods for executing the commands. For more information, see [JavaScript command injection prevention](https://semgrep.dev/docs/cheat-sheets/java-command-injection/)	high	unresolved	refs/heads/master	tainted.java#L257
tainted-code-injection-from-http-request-deepsemgrep	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	refs/heads/master	tainted-code-injection.java#L42
tainted-code-injection-from-http-request-deepsemgrep	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	refs/heads/master	tainted-code-injection.java#L52
tainted-code-injection-from-http-request-deepsemgrep	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	refs/heads/master	tainted-code-injection.java#L80
tainted-code-injection-from-http-request-deepsemgrep	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	refs/heads/master	tainted-code-injection.java#L96
tainted-code-injection-from-http-request-deepsemgrep	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	refs/heads/master	tainted-code-injection.java#L118

Finding Title	Finding Description & Remediation	severity	state	ref	location
tainted-code-injection-from-http-request	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	refs/heads/master	tainted-code-injection.java#L42
tainted-code-injection-from-http-request	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	refs/heads/master	tainted-code-injection.java#L52
detected-github-token	GitHub Token detected	high	fixed	refs/heads/main	.github/workflows/scheduled_scan_all_repos.yml#L42

Findings Summary- MEDIUM Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/main	active-debug-code-getstacktrace.java#L10
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/main	active-debug-code-getstacktrace.java#L19
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/main	active-debug-code-getstacktrace.java#L28
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/main	active-debug-code-getstacktrace.java#L33
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/main	active-debug-code-getstacktrace.java#L35
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/main	active-debug-code-getstacktrace.java#L41
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	tainted.java#L141
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	tainted.java#L202
el-injection	An expression is built with a dynamic value. The source of the value(s) should be verified to avoid that unfiltered values fall into this risky code evaluation.	medium	unresolved	refs/heads/main	tainted-code-injection.java#L22
el-injection	An expression is built with a dynamic value. The source of the value(s) should be verified to avoid that unfiltered values fall into this risky code evaluation.	medium	unresolved	refs/heads/main	tainted-code-injection.java#L62
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	tainted.java#L141
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	tainted.java#L202
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	tainted.java#L141

Finding Title	Finding Description & Remediation	severity	state	ref	location
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	tainted.java#L202
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	main	tainted.java#L141
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	main	tainted.java#L202
el-injection	An expression is built with a dynamic value. The source of the value(s) should be verified to avoid that unfiltered values fall into this risky code evaluation.	medium	unresolved	main	tainted-code-injection.java#L22
el-injection	An expression is built with a dynamic value. The source of the value(s) should be verified to avoid that unfiltered values fall into this risky code evaluation.	medium	unresolved	main	tainted-code-injection.java#L62
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	main	tainted.java#L141
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	main	tainted.java#L202
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	main	tainted.java#L141
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	main	tainted.java#L202
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/master	tainted.java#L141
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/master	tainted.java#L202
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/master	tainted.java#L141

Finding Title	Finding Description & Remediation	severity	state	ref	location
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/master	tainted.java#L202
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/master	tainted.java#L141
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/master	tainted.java#L202
el-injection	An expression is built with a dynamic value. The source of the value(s) should be verified to avoid that unfiltered values fall into this risky code evaluation.	medium	unresolved	refs/heads/master	tainted-code-injection.java#L22
el-injection	An expression is built with a dynamic value. The source of the value(s) should be verified to avoid that unfiltered values fall into this risky code evaluation.	medium	unresolved	refs/heads/master	tainted-code-injection.java#L62
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	main	active-debug-code-getstacktrace.java#L10
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	main	active-debug-code-getstacktrace.java#L19
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	main	active-debug-code-getstacktrace.java#L28
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	main	active-debug-code-getstacktrace.java#L33
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	main	active-debug-code-getstacktrace.java#L35
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	main	active-debug-code-getstacktrace.java#L41
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/master	active-debug-code-getstacktrace.java#L10
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/master	active-debug-code-getstacktrace.java#L19

Finding Title	Finding Description & Remediation	severity	state	ref	location
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/master	active-debug-code-getstacktrace.java#L28
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/master	active-debug-code-getstacktrace.java#L33
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/master	active-debug-code-getstacktrace.java#L35
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/master	active-debug-code-getstacktrace.java#L41

Findings Summary- LOW Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location

Semgrep SAST Scan Report for Repository: Nitin-Personal/semgrep-appsec

Report Generated at 2024-02-28 09:27

SAST Scan Summary

Vulnerability Severity	Vulnerability Count
Findings- SAST High Severity	16
Findings- SAST Medium Severity	24
Findings- SAST Low Severity	0

Findings Summary- HIGH Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
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
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
tainted-file-path	Detected user input controlling a file path. An attacker could control the location of this file, to include going backwards in the directory with '../'. To address this, ensure that user-controlled variables in file paths are sanitized. You may also consider using a utility method such as org.apache.commons.io.FilenameUtils.getName(...) to only retrieve the file name from the path.	high	unresolved	refs/heads/main	src/main/java/hawk/service/tainted-1.java#L42
tainted-file-path	Detected user input controlling a file path. An attacker could control the location of this file, to include going backwards in the directory with '../'. To address this, ensure that user-controlled variables in file paths are sanitized. You may also consider using a utility method such as org.apache.commons.io.FilenameUtils.getName(...) to only retrieve the file name from the path.	high	unresolved	refs/heads/main	src/main/java/hawk/service/tainted-1.java#L59
tainted-file-path	Detected user input controlling a file path. An attacker could control the location of this file, to include going backwards in the directory with '../'. To address this, ensure that user-controlled variables in file paths are sanitized. You may also consider using a utility method such as org.apache.commons.io.FilenameUtils.getName(...) to only retrieve the file name from the path.	high	unresolved	refs/heads/main	src/main/java/hawk/service/tainted-1.java#L73
tainted-file-path	Detected user input controlling a file path. An attacker could control the location of this file, to include going backwards in the directory with '../'. To address this, ensure that user-controlled variables in file paths are sanitized. You may also consider using a utility method such as org.apache.commons.io.FilenameUtils.getName(...) to only retrieve the file name from the path.	high	unresolved	refs/heads/main	src/main/java/hawk/service/tainted-1.java#L117
tainted-file-path	Detected user input controlling a file path. An attacker could control the location of this file, to include going backwards in the directory with '../'. To address this, ensure that user-controlled variables in file paths are sanitized. You may also consider using a utility method such as org.apache.commons.io.FilenameUtils.getName(...) to only retrieve the file name from the path.	high	unresolved	refs/heads/main	src/main/java/hawk/service/tainted_file_path.java#L42
tainted-file-path	Detected user input controlling a file path. An attacker could control the location of this file, to include going backwards in the directory with '../'. To address this, ensure that user-controlled variables in file paths are sanitized. You may also consider using a utility method such as org.apache.commons.io.FilenameUtils.getName(...) to only retrieve the file name from the path.	high	unresolved	refs/heads/main	src/main/java/hawk/service/tainted_file_path.java#L59
tainted-file-path	Detected user input controlling a file path. An attacker could control the location of this file, to include going backwards in the directory with '../'. To address this, ensure that user-controlled variables in file paths are sanitized. You may also consider using a utility method such as org.apache.commons.io.FilenameUtils.getName(...) to only retrieve the file name from the path.	high	unresolved	refs/heads/main	src/main/java/hawk/service/tainted_file_path.java#L73
tainted-file-path	Detected user input controlling a file path. An attacker could control the location of this file, to include going backwards in the directory with '../'. To address this, ensure that user-controlled variables in file paths are sanitized. You may also consider using a utility method such as org.apache.commons.io.FilenameUtils.getName(...) to only retrieve the file name from the path.	high	unresolved	refs/heads/main	src/main/java/hawk/service/tainted_file_path.java#L117

Finding Title	Finding Description & Remediation	severity	state	ref	location
spring-tainted-path-traversal	The application builds a file path from potentially untrusted data, which can lead to a path traversal vulnerability. An attacker can manipulate the 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. In Java, you may also consider using a utility method such as `org.apache.commons.io.FilenameUtils.getName(...)` to only retrieve the file name from the path.	high	unresolved	refs/heads/main	src/main/java/hawk/service/tainted-1.java#L42
spring-tainted-path-traversal	The application builds a file path from potentially untrusted data, which can lead to a path traversal vulnerability. An attacker can manipulate the 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. In Java, you may also consider using a utility method such as `org.apache.commons.io.FilenameUtils.getName(...)` to only retrieve the file name from the path.	high	unresolved	refs/heads/main	src/main/java/hawk/service/tainted-1.java#L59
spring-tainted-path-traversal	The application builds a file path from potentially untrusted data, which can lead to a path traversal vulnerability. An attacker can manipulate the 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. In Java, you may also consider using a utility method such as `org.apache.commons.io.FilenameUtils.getName(...)` to only retrieve the file name from the path.	high	unresolved	refs/heads/main	src/main/java/hawk/service/tainted-1.java#L73
spring-tainted-path-traversal	The application builds a file path from potentially untrusted data, which can lead to a path traversal vulnerability. An attacker can manipulate the 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. In Java, you may also consider using a utility method such as `org.apache.commons.io.FilenameUtils.getName(...)` to only retrieve the file name from the path.	high	unresolved	refs/heads/main	src/main/java/hawk/service/tainted_file_path.java#L42
spring-tainted-path-traversal	The application builds a file path from potentially untrusted data, which can lead to a path traversal vulnerability. An attacker can manipulate the 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. In Java, you may also consider using a utility method such as `org.apache.commons.io.FilenameUtils.getName(...)` to only retrieve the file name from the path.	high	unresolved	refs/heads/main	src/main/java/hawk/service/tainted_file_path.java#L59
spring-tainted-path-traversal	The application builds a file path from potentially untrusted data, which can lead to a path traversal vulnerability. An attacker can manipulate the 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. In Java, you may also consider using a utility method such as `org.apache.commons.io.FilenameUtils.getName(...)` to only retrieve the file name from the path.	high	unresolved	refs/heads/main	src/main/java/hawk/service/tainted_file_path.java#L73

Findings Summary- MEDIUM Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/main	src/main/java/hawk/service/tainted-1.java#L86
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/main	src/main/java/hawk/service/tainted-1.java#L110
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/main	src/main/java/hawk/service/tainted_file_path.java#L86
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/main	src/main/java/hawk/service/tainted_file_path.java#L110
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-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
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
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
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#L181

Finding Title	Finding Description & Remediation	severity	state	ref	location
unrestricted-request-mapping	Detected a method annotated with 'RequestMapping' that does not specify the HTTP method. CSRF protections are not enabled for GET, HEAD, TRACE, or OPTIONS, and by default all HTTP methods are allowed when the HTTP method is not explicitly specified. This means that a method that performs state changes could be vulnerable to CSRF attacks. To mitigate, add the 'method' field and specify the HTTP method (such as 'RequestMethod.POST').	medium	unresolved	refs/heads/main	src/main/java/hawk/service/tainted-1.java#L36
unrestricted-request-mapping	Detected a method annotated with 'RequestMapping' that does not specify the HTTP method. CSRF protections are not enabled for GET, HEAD, TRACE, or OPTIONS, and by default all HTTP methods are allowed when the HTTP method is not explicitly specified. This means that a method that performs state changes could be vulnerable to CSRF attacks. To mitigate, add the 'method' field and specify the HTTP method (such as 'RequestMethod.POST').	medium	unresolved	refs/heads/main	src/main/java/hawk/service/tainted_file_path.java#L36
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
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
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
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

Findings Summary- LOW Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location

Semgrep SAST Scan Report for Repository: Nitin-Vulnerable-Flask-App

Report Generated at 2024-02-28 09:28

SAST Scan Summary

Vulnerability Severity	Vulnerability Count
Findings- SAST High Severity	11
Findings- SAST Medium Severity	17
Findings- SAST Low Severity	0

Findings Summary- HIGH Severity

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	master	app/static/loader.js#L153
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	master	app/static/loader.js#L153
tainted-sql-string	Detected user input used to manually construct a SQL string. This is usually bad practice because manual construction could accidentally result in a SQL injection. An attacker could use a SQL injection to steal or modify contents of the database. Instead, use a parameterized query which is available by default in most database engines. Alternatively, consider using the Django object-relational mappers (ORM) instead of raw SQL queries.	high	unresolved	master	app/app.py#L261
tainted-pyyaml-flask	Insecure deserialization (called pickling in python) is when user-controllable data is deserialized by an application. This potentially enables an attacker to manipulate serialized objects in order to pass harmful data into the application code and may result in arbitrary code execution, OS command injection or DoS. Many deserialization-based attacks are completed before deserialization is finished. This means that the deserialization process itself can initiate an attack, even if the app's own functionality does not directly interact with the malicious object. PyYAML's `yaml` module is as powerful as `pickle` and so may call any Python function. It is recommended to secure your application by using `yaml.SafeLoader` or `yaml.CSafeLoader`.	high	unresolved	master	app/app.py#L329
dangerous-template-string	Found a template created with string formatting. This is susceptible to server-side template injection and cross-site scripting attacks.	high	unresolved	master	app/app.py#L103
dangerous-template-string	Found a template created with string formatting. This is susceptible to server-side template injection and cross-site scripting attacks.	high	unresolved	master	app/app.py#L271
tainted-sql-string	Detected user input used to manually construct a SQL string. This is usually bad practice because manual construction could accidentally result in a SQL injection. An attacker could use a SQL injection to steal or modify contents of the database. Instead, use a parameterized query which is available by default in most database engines. Alternatively, consider using an object-relational mapper (ORM) such as SQLAlchemy which will protect your queries.	high	unresolved	master	app/app.py#L261
insecure-deserialization	Detected the use of an insecure deserialization library in a Flask route. These libraries are prone to code execution vulnerabilities. Ensure user data does not enter this function. To fix this, try to avoid serializing whole objects. Consider instead using a serializer such as JSON.	high	unresolved	master	app/app.py#L329
jwt-python-hardcoded-secret	Hardcoded JWT secret or private key is used. This is a Insufficiently Protected Credentials weakness: https://cwe.mitre.org/data/definitions/522.html Consider using an appropriate security mechanism to protect the credentials (e.g. keeping secrets in environment variables)	high	unresolved	master	app/app.py#L184
unverified-jwt-decode	Detected JWT token decoded with 'verify=False'. This bypasses any integrity checks for the token which means the token could be tampered with by malicious actors. Ensure that the JWT token is verified.	high	unresolved	master	app/app.py#L97
sqlalchemy-execute-raw-query	Avoiding SQL string concatenation: untrusted input concatenated with raw SQL query can result in SQL Injection. In order to execute raw query safely, prepared statement should be used. SQLAlchemy provides TextualSQL to easily used prepared statement with named parameters. For complex SQL composition, use SQL Expression Language or Schema Definition Language. In most cases, SQLAlchemy ORM will be a better option.	high	unresolved	master	app/app.py#L265

Findings Summary- MEDIUM Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
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	master	app/static/loader.js#L24
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	master	app/static/loader.js#L26
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	master	app/static/loader.js#L41
var-in-href	Detected a template variable used in an anchor tag with the 'href' attribute. This allows a malicious actor to input the 'javascript:' URI and is subject to cross-site scripting (XSS) attacks. If using a relative URL, start with a literal forward slash and concatenate the URL, like this: href='/{{link}}'. You may also consider setting the Content Security Policy (CSP) header.	medium	unresolved	master	app/templates/index.html#L12
detect-non-literal-regexp	RegExp() called with a `c` function argument, this might allow an attacker to cause a Denial of Service (DoS) within your application as RegExP which blocks the main thread.	medium	unresolved	master	app/static/loader.js#L55
detect-non-literal-regexp	RegExp() called with a `c` function argument, this might allow an attacker to cause a Denial of Service (DoS) within your application as RegExP which blocks the main thread.	medium	unresolved	master	app/static/loader.js#L55
detect-non-literal-regexp	RegExp() called with a `c` function argument, this might allow an attacker to cause a Denial of Service (DoS) within your application as RegExP which blocks the main thread.	medium	unresolved	master	app/static/loader.js#L59
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	master	app/static/loader.js#L3
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	master	app/static/loader.js#L106
template-href-var	Detected a template variable used in an anchor tag with the 'href' attribute. This allows a malicious actor to input the 'javascript:' URI and is subject to cross-site scripting (XSS) attacks. Use the 'url' template tag to safely generate a URL. You may also consider setting the Content Security Policy (CSP) header.	medium	unresolved	master	app/templates/index.html#L12
raw-html-format	Detected user input flowing into a manually constructed HTML string. You may be accidentally bypassing secure methods of rendering HTML by manually constructing HTML and this could create a cross-site scripting vulnerability, which could let attackers steal sensitive user data. To be sure this is safe, check that the HTML is rendered safely. Otherwise, use templates (`django.shortcuts.render`) which will safely render HTML instead.	medium	unresolved	master	app/app.py#L103
render-template-string	Found a template created with string formatting. This is susceptible to server-side template injection and cross-site scripting attacks.	medium	unresolved	master	app/app.py#L114

Finding Title	Finding Description & Remediation	severity	state	ref	location
render-template-string	Found a template created with string formatting. This is susceptible to server-side template injection and cross-site scripting attacks.	medium	unresolved	master	app/app.py#L281
raw-html-format	Detected user input flowing into a manually constructed HTML string. You may be accidentally bypassing secure methods of rendering HTML by manually constructing HTML and this could create a cross-site scripting vulnerability, which could let attackers steal sensitive user data. To be sure this is safe, check that the HTML is rendered safely. Otherwise, use templates (`flask.render_template`) which will safely render HTML instead.	medium	unresolved	master	app/app.py#L103
template-href-var	Detected a template variable used in an anchor tag with the 'href' attribute. This allows a malicious actor to input the 'javascript:' URI and is subject to cross-site scripting (XSS) attacks. Use 'url_for()' to safely generate a URL. You may also consider setting the Content Security Policy (CSP) header.	medium	unresolved	master	app/templates/index.html#L12
formatted-sql-query	Detected possible formatted SQL query. Use parameterized queries instead.	medium	unresolved	master	app/app.py#L265
md5-used-as-password	It looks like MD5 is used as a password hash. MD5 is not considered a secure password hash because it can be cracked by an attacker in a short amount of time. Use a suitable password hashing function such as scrypt. You can use `hashlib.scrypt`.	medium	unresolved	master	app/app.py#L141

Findings Summary- LOW Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location

Semgrep SAST Scan Report for Repository: javaspringvulny

Report Generated at 2024-02-28 09:28

SAST Scan Summary

Vulnerability Severity	Vulnerability Count
Findings- SAST High Severity	2
Findings- SAST Medium Severity	18
Findings- SAST Low Severity	0

Findings Summary- HIGH Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
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/master	src/main/resources/keyStore.pem#L5
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/master	src/main/java/hawk/api/jwt/JwtLog4jController.java#L24

Findings Summary- MEDIUM Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
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/master	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/master	src/main/java/hawk/controller/LoginController.java#L57
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/master	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/master	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/master	src/main/java/hawk/controller/LoginController.java#L57
spring-csrf-disabled	CSRF protection is disabled for this configuration. This is a security risk.	medium	unresolved	refs/heads/master	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/master	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/master	src/main/java/hawk/MultiHttpSecurityConfig.java#L110
spring-csrf-disabled	CSRF protection is disabled for this configuration. This is a security risk.	medium	unresolved	refs/heads/master	src/main/java/hawk/MultiHttpSecurityConfig.java#L181
django-no-csrf-token	Manually-created forms in django templates should specify a csrf_token to prevent CSRF attacks	medium	unresolved	refs/heads/master	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/master	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/master	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/master	src/main/resources/templates/search.html#L14

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/master	src/main/resources/templates/user-search.html#L14
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/master	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/master	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/master	docker-compose.yml#L3
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/master	docker-compose.yml#L12

Findings Summary- LOW Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location

Semgrep SAST Scan Report for Repository: mojito

Report Generated at 2024-02-28 09:28

SAST Scan Summary

Vulnerability Severity	Vulnerability Count
Findings- SAST High Severity	11
Findings- SAST Medium Severity	14
Findings- SAST Low Severity	1

Findings Summary- HIGH Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
missing-user-entrypoint	By not specifying a USER, a program in the container may run as 'root'. This is a security hazard. If an attacker can control a process running as root, they may have control over the container. Ensure that the last USER in a Dockerfile is a USER other than 'root'.	high	unresolved	master	webapp/src/main/docker/Dockerfile#L6
command-injection-process-builder	A formatted or concatenated string was detected as input to a ProcessBuilder call. This is dangerous if a variable is controlled by user input and could result in a command injection. Ensure your variables are not controlled by users or sufficiently sanitized.	high	unresolved	master	cli/src/main/java/com/box/l10n/mojito/cli/command/RetryCommand.java#L131
command-injection-process-builder	A formatted or concatenated string was detected as input to a ProcessBuilder call. This is dangerous if a variable is controlled by user input and could result in a command injection. Ensure your variables are not controlled by users or sufficiently sanitized.	high	unresolved	master	common/src/main/java/com/box/l10n/mojito/shell/Shell.java#L21
formatted-sql-string	Detected a formatted string in a SQL statement. This could lead to SQL injection if variables in the SQL statement are not properly sanitized. Use a prepared statements (java.sql.PreparedStatement) instead. You can obtain a PreparedStatement using 'connection.prepareStatement'.	high	unresolved	master	webapp/src/main/java/com/box/l10n/mojito/nativecriteria/JpaQueryProvider.java#L17
documentbuilderfactory-disallow-doctype-decl-missing	DOCTYPE declarations are enabled for this DocumentBuilderFactory. This is vulnerable to XML external entity attacks. Disable this by setting the feature "http://apache.org/xml/features/disallow-doctype-decl" to true. Alternatively, allow DOCTYPE declarations and only prohibit external entities declarations. This can be done by setting the features "http://xml.org/sax/features/external-general-entities" and "http://xml.org/sax/features/external-parameter-entities" to false.	high	unresolved	master	common/src/main/java/com/box/l10n/mojito/okapi/filters/XMLFilter.java#L95
documentbuilderfactory-disallow-doctype-decl-missing	DOCTYPE declarations are enabled for this DocumentBuilderFactory. This is vulnerable to XML external entity attacks. Disable this by setting the feature "http://apache.org/xml/features/disallow-doctype-decl" to true. Alternatively, allow DOCTYPE declarations and only prohibit external entities declarations. This can be done by setting the features "http://xml.org/sax/features/external-general-entities" and "http://xml.org/sax/features/external-parameter-entities" to false.	high	unresolved	master	webapp/src/main/java/com/box/l10n/mojito/android/strings/AndroidStringDocumentUtils.java#L15
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	master	webapp/src/main/java/com/box/l10n/mojito/security/ShowPageAuthenticationSuccessHandler.java#L30
var-in-script-tag	Detected a template variable used in a script tag. Although template variables are HTML escaped, HTML escaping does not always prevent cross-site scripting (XSS) attacks when used directly in JavaScript. If you need this data on the rendered page, consider placing it in the HTML portion (outside of a script tag). Alternatively, use a JavaScript-specific encoder, such as the one available in OWASP ESAPI. For Django, you may also consider using the 'json_script' template tag and retrieving the data in your script by using the element ID (e.g., `document.getElementById`).	high	unresolved	master	webapp/src/main/resources/templates/index.html#L8

Finding Title	Finding Description & Remediation	severity	state	ref	location
var-in-script-tag	Detected a template variable used in a script tag. Although template variables are HTML escaped, HTML escaping does not always prevent cross-site scripting (XSS) attacks when used directly in JavaScript. If you need this data on the rendered page, consider placing it in the HTML portion (outside of a script tag). Alternatively, use a JavaScript-specific encoder, such as the one available in OWASP ESAPI. For Django, you may also consider using the 'json_script' template tag and retrieving the data in your script by using the element ID (e.g., `document.getElementById`).	high	unresolved	master	webapp/src/main/resources/templates/index.html#L9
var-in-script-tag	Detected a template variable used in a script tag. Although template variables are HTML escaped, HTML escaping does not always prevent cross-site scripting (XSS) attacks when used directly in JavaScript. If you need this data on the rendered page, consider placing it in the HTML portion (outside of a script tag). Alternatively, use a JavaScript-specific encoder, such as the one available in OWASP ESAPI. For Django, you may also consider using the 'json_script' template tag and retrieving the data in your script by using the element ID (e.g., `document.getElementById`).	high	unresolved	master	webapp/src/main/resources/templates/index.html#L15
var-in-script-tag	Detected a template variable used in a script tag. Although template variables are HTML escaped, HTML escaping does not always prevent cross-site scripting (XSS) attacks when used directly in JavaScript. If you need this data on the rendered page, consider placing it in the HTML portion (outside of a script tag). Alternatively, use a JavaScript-specific encoder, such as the one available in OWASP ESAPI. For Django, you may also consider using the 'json_script' template tag and retrieving the data in your script by using the element ID (e.g., `document.getElementById`).	high	unresolved	master	webapp/src/main/resources/templates/index.html#L16

Findings Summary- MEDIUM Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
unsafe-reflection	If an attacker can supply values that the application then uses to determine which class to instantiate or which method to invoke, the potential exists for the attacker to create control flow paths through the application that were not intended by the application developers. This attack vector may allow the attacker to bypass authentication or access control checks or otherwise cause the application to behave in an unexpected manner.	medium	unresolved	master	cli/src/main/java/com/box/l10n/mojito/cli/command/checks/AbstractCliChecker.java#L16
unsafe-reflection	If an attacker can supply values that the application then uses to determine which class to instantiate or which method to invoke, the potential exists for the attacker to create control flow paths through the application that were not intended by the application developers. This attack vector may allow the attacker to bypass authentication or access control checks or otherwise cause the application to behave in an unexpected manner.	medium	unresolved	master	webapp/src/main/java/com/box/l10n/mojito/service/assetintegritychecker/integritychecker/IntegrityCheckerFactory.java#L84
unsafe-reflection	If an attacker can supply values that the application then uses to determine which class to instantiate or which method to invoke, the potential exists for the attacker to create control flow paths through the application that were not intended by the application developers. This attack vector may allow the attacker to bypass authentication or access control checks or otherwise cause the application to behave in an unexpected manner.	medium	unresolved	master	webapp/src/main/java/com/box/l10n/mojito/service/drop/exporter/DropExporterService.java#L87
unvalidated-redirect	Application redirects to a destination URL specified by a user-supplied parameter that is not validated. This could direct users to malicious locations. Consider using an allowlist to validate URLs.	medium	unresolved	master	webapp/src/main/java/com/box/l10n/mojito/rest/cli/CliWS.java#L52
unrestricted-request-mapping	Detected a method annotated with 'RequestMapping' that does not specify the HTTP method. CSRF protections are not enabled for GET, HEAD, TRACE, or OPTIONS, and by default all HTTP methods are allowed when the HTTP method is not explicitly specified. This means that a method that performs state changes could be vulnerable to CSRF attacks. To mitigate, add the 'method' field and specify the HTTP method (such as 'RequestMethod.POST').	medium	unresolved	master	webapp/src/main/java/com/box/l10n/mojito/react/ReactAppController.java#L58

Finding Title	Finding Description & Remediation	severity	state	ref	location
open-redirect-pathname	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	master	webapp/src/main/resources/public/js/app.js#L152
var-in-script-src	Detected a template variable used as the 'src' in a script tag. Although template variables are HTML escaped, HTML escaping does not always prevent malicious URLs from being injected and could results in a cross-site scripting (XSS) vulnerability. Prefer not to dynamically generate the 'src' attribute and use static URLs instead. If you must do this, carefully check URLs against an allowlist and be sure to URL-encode the result.	medium	unresolved	master	webapp/src/main/resources/templates/index.html#L15
var-in-script-src	Detected a template variable used as the 'src' in a script tag. Although template variables are HTML escaped, HTML escaping does not always prevent malicious URLs from being injected and could results in a cross-site scripting (XSS) vulnerability. Prefer not to dynamically generate the 'src' attribute and use static URLs instead. If you must do this, carefully check URLs against an allowlist and be sure to URL-encode the result.	medium	unresolved	master	webapp/src/main/resources/templates/index.html#L16
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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	master	webapp/src/main/resources/templates/index.html#L8
var-in-href	Detected a template variable used in an anchor tag with the 'href' attribute. This allows a malicious actor to input the 'javascript:' URI and is subject to cross- site scripting (XSS) attacks. If using a relative URL, start with a literal forward slash and concatenate the URL, like this: href='/{{link}}'. You may also consider setting the Content Security Policy (CSP) header.	medium	unresolved	master	webapp/src/main/resources/templates/email/sla/openIncident.html#L29
template-href-var	Detected a template variable used in an anchor tag with the 'href' attribute. This allows a malicious actor to input the 'javascript:' URI and is subject to cross- site scripting (XSS) attacks. Use the 'url' template tag to safely generate a URL. You may also consider setting the Content Security Policy (CSP) header.	medium	unresolved	master	webapp/src/main/resources/templates/email/sla/openIncident.html#L29

Finding Title	Finding Description & Remediation	severity	state	ref	location
template-href-var	Detected a template variable used in an anchor tag with the 'href' attribute. This allows a malicious actor to input the 'javascript:' URI and is subject to cross- site scripting (XSS) attacks. Use 'url_for()' to safely generate a URL. You may also consider setting the Content Security Policy (CSP) header.	medium	unresolved	master	webapp/src/main/resources/templates/email/sla/openIncident.html#L29
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	master	docker/docker-compose.yml#L3
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	master	docker/docker-compose.yml#L3

Findings Summary- LOW Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
xml-custom-entityresolver	The application is using an XML parser that has not been safely configured. This might lead to XML External Entity (XXE) vulnerabilities when parsing user-controlled input. An attacker can include document type definitions (DTDs) which can interact with internal or external hosts. XXE can lead to other vulnerabilities, such as Local File Inclusion (LFI), Remote Code Execution (RCE), and Server-side request forgery (SSRF), depending on the application configuration. An attacker can also use DTDs to expand recursively, leading to a Denial-of-Service (DoS) attack, also known as a Billion Laughs Attack. By setting a custom `EntityResolver` for all previous security configurations for are bypassed. It is your responsibility to handle security in the `EntityResolver` implementation instead. For more information, see: [Java XXE prevention](https://semgrep.dev/docs/cheat-sheets/java-xxe/)	low	unresolved	master	common/src/main/java/com/box/l10n/mojito/okapi/filters/XMLFilter.java#L101

Semgrep SAST Scan Report for Repository: nnayar-r2c/Skyscanner-backpack-foundations

Report Generated at 2024-02-28 09:28

SAST Scan Summary

Vulnerability Severity	Vulnerability Count
Findings- SAST High Severity	0
Findings- SAST Medium Severity	34
Findings- SAST Low Severity	0

Findings Summary- HIGH Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location

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	packages/bpk-foundations-android/gulpfile.babel.js#L85
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	packages/bpk-foundations-android/gulpfile.babel.js#L88
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	packages/bpk-foundations-android/gulpfile.babel.js#L88
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	packages/bpk-foundations-android/gulpfile.babel.js#L90
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	packages/bpk-foundations-android/gulpfile.babel.js#L91
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	packages/bpk-foundations-ios/gulpfile.babel.js#L92
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	packages/bpk-foundations-ios/gulpfile.babel.js#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	packages/bpk-foundations-ios/gulpfile.babel.js#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	packages/bpk-foundations-ios/gulpfile.babel.js#L97
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	packages/bpk-foundations-ios/gulpfile.babel.js#L98
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	packages/bpk-foundations-react-native/gulpfile.babel.js#L124
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	packages/bpk-foundations-react-native/gulpfile.babel.js#L127
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	packages/bpk-foundations-react-native/gulpfile.babel.js#L127

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	packages/bpk-foundations-react-native/gulpfile.babel.js#L129
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	packages/bpk-foundations-react-native/gulpfile.babel.js#L130
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	packages/bpk-foundations-web/gulpfile.babel.js#L89
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/ci.yml#L48
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	packages/bpk-foundations-android/gulpfile.babel.js#L85
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	packages/bpk-foundations-android/gulpfile.babel.js#L88
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	packages/bpk-foundations-android/gulpfile.babel.js#L88
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	packages/bpk-foundations-android/gulpfile.babel.js#L90
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	packages/bpk-foundations-android/gulpfile.babel.js#L91
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	packages/bpk-foundations-ios/gulpfile.babel.js#L92
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	packages/bpk-foundations-ios/gulpfile.babel.js#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	fixed	refs/heads/main	packages/bpk-foundations-ios/gulpfile.babel.js#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	fixed	refs/heads/main	packages/bpk-foundations-ios/gulpfile.babel.js#L97
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	packages/bpk-foundations-ios/gulpfile.babel.js#L98

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	packages/bpk-foundations-react-native/gulpfile.babel.js#L124
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	packages/bpk-foundations-react-native/gulpfile.babel.js#L127
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	packages/bpk-foundations-react-native/gulpfile.babel.js#L127
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	packages/bpk-foundations-react-native/gulpfile.babel.js#L129
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	packages/bpk-foundations-react-native/gulpfile.babel.js#L130
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	packages/bpk-foundations-web/gulpfile.babel.js#L89
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	packages/bpk-svgs/tasks/getIconFontMetadataProvider.js#L25

Findings Summary- LOW Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location

Semgrep SAST Scan Report for Repository: nnayar-r2c/adyen-node-api-library

Report Generated at 2024-02-28 09:28

SAST Scan Summary

Vulnerability Severity	Vulnerability Count
Findings- SAST High Severity	0
Findings- SAST Medium Severity	25
Findings- SAST Low Severity	0

Findings Summary- HIGH Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location

Findings Summary- MEDIUM Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
weak-symmetric-mode	Detected the use of `createCipheriv("aes-256-cbc")` which is considered a weak cryptographic mode. Where possible, leverage the industry standard recommendation which is to use a block cipher such as `AES` with at least `128-bit` strength, an example of a secure algorithm is `AES-256-GCM`. If your company has its own guidelines, you should follow your company's internal best practices.	medium	fixed	refs/heads/main	src/security/nexoCrypto.ts#L98
weak-symmetric-mode	Detected the use of `createDecipheriv("aes-256-cbc")` which is considered a weak cryptographic mode. Where possible, leverage the industry standard recommendation which is to use a block cipher such as `AES` with at least `128-bit` strength, an example of a secure algorithm is `AES-256-GCM`. If your company has its own guidelines, you should follow your company's internal best practices.	medium	fixed	refs/heads/main	src/security/nexoCrypto.ts#L99
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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	templates/typescript/api-single.mustache#L19
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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	templates/typescript/api-single.mustache#L25
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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	templates/typescript/api-single.mustache#L25
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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	templates/typescript/api-single.mustache#L25
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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	templates/typescript/api-single.mustache#L26
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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	templates/typescript/api-single.mustache#L30
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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	templates/typescript/api-single.mustache#L30
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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	templates/typescript/api-single.mustache#L32
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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	templates/typescript/api-single.mustache#L32
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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	templates/typescript/api-single.mustache#L38

Finding Title	Finding Description & Remediation	severity	state	ref	location
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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	templates/typescript/licenseInfo.mustache#L2
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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	templates/typescript/licenseInfo.mustache#L3
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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	templates/typescript/model.mustache#L10
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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	templates/typescript/model.mustache#L14
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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	templates/typescript/model.mustache#L18
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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	templates/typescript/model.mustache#L21
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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	templates/typescript/model.mustache#L21
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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	templates/typescript/model.mustache#L37
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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	templates/typescript/model.mustache#L37
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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	templates/typescript/model.mustache#L61
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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	templates/typescript/model.mustache#L74
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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	templates/typescript/models.mustache#L5
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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	templates/typescript/models.mustache#L13

Findings Summary- LOW Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location

Semgrep SAST Scan Report for Repository: nnayar-r2c/deliveroo-jsonrpc-go

Report Generated at 2024-02-28 09:28

SAST Scan Summary

Vulnerability Severity	Vulnerability Count
Findings- SAST High Severity	0
Findings- SAST Medium Severity	3
Findings- SAST Low Severity	0

Findings Summary- HIGH Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location

Findings Summary- MEDIUM Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
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/master	.github/workflows/codeql-analysis.yml#L46
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/master	.github/workflows/codeql-analysis.yml#L59
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/master	.github/workflows/codeql-analysis.yml#L72

Findings Summary- LOW Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location

Semgrep SAST Scan Report for Repository: nnayar-r2c/deliveroo-safe-go

Report Generated at 2024-02-28 09:28

SAST Scan Summary

Vulnerability Severity	Vulnerability Count
Findings- SAST High Severity	0
Findings- SAST Medium Severity	5
Findings- SAST Low Severity	0

Findings Summary- HIGH Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location

Findings Summary- MEDIUM Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
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/master	.github/workflows/codeql-analysis.yml#L40
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/master	.github/workflows/codeql-analysis.yml#L47
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/master	.github/workflows/codeql-complete.yml#L33
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/master	.github/workflows/codeql-complete.yml#L42
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/master	.github/workflows/codeql-complete.yml#L51

Findings Summary- LOW Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location

Semgrep SAST Scan Report for Repository: nnayar-r2c/experian-nodejs

Report Generated at 2024-02-28 09:28

SAST Scan Summary

Vulnerability Severity	Vulnerability Count
Findings- SAST High Severity	0
Findings- SAST Medium Severity	5
Findings- SAST Low Severity	1

Findings Summary- HIGH Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location

Findings Summary- MEDIUM Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
bypass-tls-verification	Checks for setting the environment variable NODE_TLS_REJECT_UNAUTHORIZED to 0, which disables TLS verification. This should only be used for debugging purposes. Setting the option rejectUnauthorized to false bypasses verification against the list of trusted CAs, which also leads to insecure transport. These options lead to vulnerability to MTM attacks, and should not be used.	medium	unresolved	refs/heads/master	lib/businessOwners/index.js#L60
bypass-tls-verification	Checks for setting the environment variable NODE_TLS_REJECT_UNAUTHORIZED to 0, which disables TLS verification. This should only be used for debugging purposes. Setting the option rejectUnauthorized to false bypasses verification against the list of trusted CAs, which also leads to insecure transport. These options lead to vulnerability to MTM attacks, and should not be used.	medium	unresolved	refs/heads/master	lib/businessinformation/index.js#L289
bypass-tls-verification	Checks for setting the environment variable NODE_TLS_REJECT_UNAUTHORIZED to 0, which disables TLS verification. This should only be used for debugging purposes. Setting the option rejectUnauthorized to false bypasses verification against the list of trusted CAs, which also leads to insecure transport. These options lead to vulnerability to MTM attacks, and should not be used.	medium	unresolved	refs/heads/master	lib/consumerCreditProfileInformation/index.js#L50
bypass-tls-verification	Checks for setting the environment variable NODE_TLS_REJECT_UNAUTHORIZED to 0, which disables TLS verification. This should only be used for debugging purposes. Setting the option rejectUnauthorized to false bypasses verification against the list of trusted CAs, which also leads to insecure transport. These options lead to vulnerability to MTM attacks, and should not be used.	medium	unresolved	refs/heads/master	lib/experian.js#L132
bypass-tls-verification	Checks for setting the environment variable NODE_TLS_REJECT_UNAUTHORIZED to 0, which disables TLS verification. This should only be used for debugging purposes. Setting the option rejectUnauthorized to false bypasses verification against the list of trusted CAs, which also leads to insecure transport. These options lead to vulnerability to MTM attacks, and should not be used.	medium	unresolved	refs/heads/master	lib/sbcsinformation/index.js#L80

Findings Summary- LOW Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
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/master	examples/express/index.js#L4

Semgrep SAST Scan Report for Repository: nnayar-r2c/finos-common-domain-model

Report Generated at 2024-02-28 09:28

SAST Scan Summary

Vulnerability Severity	Vulnerability Count
Findings- SAST High Severity	1
Findings- SAST Medium Severity	4
Findings- SAST Low Severity	0

Findings Summary- HIGH Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
detected-jwt-token	JWT token detected	high	unresolved	refs/heads/master	documentation/source/cdm-overview.rst#L111

Findings Summary- MEDIUM Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/master	rosetta-project/src/main/java/com/regnosys/granite/projector/util/ExceptionUtils.java#L11
var-in-href	Detected a template variable used in an anchor tag with the 'href' attribute. This allows a malicious actor to input the 'javascript:' URI and is subject to cross- site scripting (XSS) attacks. If using a relative URL, start with a literal forward slash and concatenate the URL, like this: href='/{{link}}'. You may also consider setting the Content Security Policy (CSP) header.	medium	unresolved	refs/heads/master	documentation/site/_templates/header.html#L6
template-href-var	Detected a template variable used in an anchor tag with the 'href' attribute. This allows a malicious actor to input the 'javascript:' URI and is subject to cross- site scripting (XSS) attacks. Use the 'url' template tag to safely generate a URL. You may also consider setting the Content Security Policy (CSP) header.	medium	unresolved	refs/heads/master	documentation/site/_templates/header.html#L6
template-href-var	Detected a template variable used in an anchor tag with the 'href' attribute. This allows a malicious actor to input the 'javascript:' URI and is subject to cross- site scripting (XSS) attacks. Use 'url_for()' to safely generate a URL. You may also consider setting the Content Security Policy (CSP) header.	medium	unresolved	refs/heads/master	documentation/site/_templates/header.html#L6

Findings Summary- LOW Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location

Semgrep SAST Scan Report for Repository: nnayar-r2c/legend-pure

Report Generated at 2024-02-28 09:28

SAST Scan Summary

Vulnerability Severity	Vulnerability Count
Findings- SAST High Severity	5
Findings- SAST Medium Severity	14
Findings- SAST Low Severity	6

Findings Summary- HIGH Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
run-shell-injection	Using variable interpolation `${{...}}` with `github` context data in a `run:` step could allow an attacker to inject their own code into the runner. This would allow them to steal secrets and code. `github` context data can have arbitrary user input and should be treated as untrusted. Instead, use an intermediate environment variable with `env:` to store the data and use the environment variable in the `run:` script. Be sure to use double-quotes the environment variable, like this: "$ENVVAR".	high	unresolved	refs/heads/master	.github/workflows/ossrh-close-staging-repo.yml#L62
run-shell-injection	Using variable interpolation `${{...}}` with `github` context data in a `run:` step could allow an attacker to inject their own code into the runner. This would allow them to steal secrets and code. `github` context data can have arbitrary user input and should be treated as untrusted. Instead, use an intermediate environment variable with `env:` to store the data and use the environment variable in the `run:` script. Be sure to use double-quotes the environment variable, like this: "$ENVVAR".	high	unresolved	refs/heads/master	.github/workflows/release.yml#L74
run-shell-injection	Using variable interpolation `${{...}}` with `github` context data in a `run:` step could allow an attacker to inject their own code into the runner. This would allow them to steal secrets and code. `github` context data can have arbitrary user input and should be treated as untrusted. Instead, use an intermediate environment variable with `env:` to store the data and use the environment variable in the `run:` script. Be sure to use double-quotes the environment variable, like this: "$ENVVAR".	high	fixed	refs/heads/master	.github/workflows/ossrh-close-staging-repo.yml#L62
run-shell-injection	Using variable interpolation `${{...}}` with `github` context data in a `run:` step could allow an attacker to inject their own code into the runner. This would allow them to steal secrets and code. `github` context data can have arbitrary user input and should be treated as untrusted. Instead, use an intermediate environment variable with `env:` to store the data and use the environment variable in the `run:` script. Be sure to use double-quotes the environment variable, like this: "$ENVVAR".	high	fixed	refs/heads/master	.github/workflows/release.yml#L74
run-shell-injection	Using variable interpolation `${{...}}` with `github` context data in a `run:` step could allow an attacker to inject their own code into the runner. This would allow them to steal secrets and code. `github` context data can have arbitrary user input and should be treated as untrusted. Instead, use an intermediate environment variable with `env:` to store the data and use the environment variable in the `run:` script. Be sure to use double-quotes the environment variable, like this: "$ENVVAR".	high	unresolved	refs/heads/master	.github/workflows/release.yml#L83

Findings Summary- MEDIUM Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
jdbc-sqli	Detected a formatted string in a SQL statement. This could lead to SQL injection if variables in the SQL statement are not properly sanitized. Use a prepared statements (java.sql.PreparedStatement) instead. You can obtain a PreparedStatement using 'connection.prepareStatement'.	medium	unresolved	refs/heads/master	legend-pure-runtime-java-extension-store-relational/src/main/java/org/finos/legend/pure/runtime/java/extension/store/relational/interpreted/natives/ExecuteInDb.java#L179
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/master	.github/workflows/test-result.yml#L48
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/master	legend-pure-ide-light/src/main/java/org/finos/legend/pure/ide/light/helpers/JSONResponseTools.java#L66
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/master	legend-pure-ide-light/src/main/java/org/finos/legend/pure/ide/light/helpers/response/ExceptionTranslation.java#L72
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/master	legend-pure-m3-core/src/main/java/org/finos/legend/pure/m3/generator/par/PureJarGenerator.java#L51

Finding Title	Finding Description & Remediation	severity	state	ref	location
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/master	legend-pure-m3-core/src/main/java/org/finos/legend/pure/m3/serialization/filesystem/usercodestorage/composite/CompositeCodeStorage.java#L591
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/master	legend-pure-m3-core/src/main/java/org/finos/legend/pure/m3/serialization/runtime/cache/CacheState.java#L87
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/master	legend-pure-runtime-java-engine-compiled/src/main/java/org/finos/legend/pure/runtime/java/compiled/generation/orchestrator/JavaCodeGeneration.java#L66
jdbc-sqli	Detected a formatted string in a SQL statement. This could lead to SQL injection if variables in the SQL statement are not properly sanitized. Use a prepared statements (java.sql.PreparedStatement) instead. You can obtain a PreparedStatement using 'connection.prepareStatement'.	medium	unresolved	refs/heads/master	legend-pure-runtime-java-extension-store-relational/src/main/java/org/finos/legend/pure/runtime/java/extension/store/relational/compiled/natives/ResultSetRowIterableProvider.java#L88
jdbc-sqli	Detected a formatted string in a SQL statement. This could lead to SQL injection if variables in the SQL statement are not properly sanitized. Use a prepared statements (java.sql.PreparedStatement) instead. You can obtain a PreparedStatement using 'connection.prepareStatement'.	medium	fixed	refs/heads/master	legend-pure-runtime-java-extension-store-relational/src/main/java/org/finos/legend/pure/runtime/java/extension/store/relational/interpreted/natives/ExecuteInDb.java#L179

Finding Title	Finding Description & Remediation	severity	state	ref	location
unsafe-reflection	If an attacker can supply values that the application then uses to determine which class to instantiate or which method to invoke, the potential exists for the attacker to create control flow paths through the application that were not intended by the application developers. This attack vector may allow the attacker to bypass authentication or access control checks or otherwise cause the application to behave in an unexpected manner.	medium	unresolved	refs/heads/master	legend-pure-runtime-java-engine-compiled/src/main/java/org/finos/legend/pure/runtime/java/compiled/generation/orchestrator/JavaModelFactoryGenerator.java#L60
unsafe-reflection	If an attacker can supply values that the application then uses to determine which class to instantiate or which method to invoke, the potential exists for the attacker to create control flow paths through the application that were not intended by the application developers. This attack vector may allow the attacker to bypass authentication or access control checks or otherwise cause the application to behave in an unexpected manner.	medium	unresolved	refs/heads/master	legend-pure-runtime-java-engine-compiled/src/main/java/org/finos/legend/pure/runtime/java/compiled/testHelper/PureTestBuilderCompiled.java#L97
no-direct-response-writer	Detected a request with potential user-input going into a OutputStream or Writer object. This bypasses any view or template environments, including HTML escaping, which may expose this application to cross-site scripting (XSS) vulnerabilities. Consider using a view technology such as JavaServer Faces (JSFs) which automatically escapes HTML views.	medium	fixed	refs/heads/master	legend-pure-ide-light/src/main/java/org/finos/legend/pure/ide/light/api/concept/Concept.java#L115

Finding Title	Finding Description & Remediation	severity	state	ref	location
no-direct-response-writer	Detected a request with potential user-input going into a OutputStream or Writer object. This bypasses any view or template environments, including HTML escaping, which may expose this application to cross-site scripting (XSS) vulnerabilities. Consider using a view technology such as JavaServer Faces (JSFs) which automatically escapes HTML views.	medium	fixed	refs/heads/master	legend-pure-ide-light/src/main/java/org/finos/legend/pure/ide/light/api/find/FindPureFile.java#L85

Findings Summary- LOW Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
gcm-detection	GCM detected, please check that IV/nonce is not reused, an Initialization Vector (IV) is a nonce used to randomize the encryption, so that even if multiple messages with identical plaintext are encrypted, the generated corresponding ciphertexts are different. Unlike the Key, the IV usually does not need to be secret, rather it is important that it is random and unique. Certain encryption schemes the IV is exchanged in public as part of the ciphertext. Reusing same Initialization Vector with the same Key to encrypt multiple plaintext blocks allows an attacker to compare the ciphertexts and then, with some assumptions on the content of the messages, to gain important information about the data being encrypted.	low	unresolved	refs/heads/master	legend-pure-runtime-java-extension-functions/src/main/java/org/finos/legend/pure/runtime/java/extension/functions/shared/cipher/AESCipherUtil.java#L52
gcm-detection	GCM detected, please check that IV/nonce is not reused, an Initialization Vector (IV) is a nonce used to randomize the encryption, so that even if multiple messages with identical plaintext are encrypted, the generated corresponding ciphertexts are different. Unlike the Key, the IV usually does not need to be secret, rather it is important that it is random and unique. Certain encryption schemes the IV is exchanged in public as part of the ciphertext. Reusing same Initialization Vector with the same Key to encrypt multiple plaintext blocks allows an attacker to compare the ciphertexts and then, with some assumptions on the content of the messages, to gain important information about the data being encrypted.	low	unresolved	refs/heads/master	legend-pure-runtime-java-extension-functions/src/main/java/org/finos/legend/pure/runtime/java/extension/functions/shared/cipher/AESCipherUtil.java#L76
gcm-detection	GCM detected, please check that IV/nonce is not reused, an Initialization Vector (IV) is a nonce used to randomize the encryption, so that even if multiple messages with identical plaintext are encrypted, the generated corresponding ciphertexts are different. Unlike the Key, the IV usually does not need to be secret, rather it is important that it is random and unique. Certain encryption schemes the IV is exchanged in public as part of the ciphertext. Reusing same Initialization Vector with the same Key to encrypt multiple plaintext blocks allows an attacker to compare the ciphertexts and then, with some assumptions on the content of the messages, to gain important information about the data being encrypted.	low	fixed	refs/heads/master	legend-pure-runtime-java-extension-functions/src/main/java/org/finos/legend/pure/runtime/java/extension/functions/shared/cipher/AESCipherUtil.java#L52

Finding Title	Finding Description & Remediation	severity	state	ref	location
gcm-detection	GCM detected, please check that IV/nonce is not reused, an Initialization Vector (IV) is a nonce used to randomize the encryption, so that even if multiple messages with identical plaintext are encrypted, the generated corresponding ciphertexts are different. Unlike the Key, the IV usually does not need to be secret, rather it is important that it is random and unique. Certain encryption schemes the IV is exchanged in public as part of the ciphertext. Reusing same Initialization Vector with the same Key to encrypt multiple plaintext blocks allows an attacker to compare the ciphertexts and then, with some assumptions on the content of the messages, to gain important information about the data being encrypted.	low	unresolved	refs/heads/master	legend-pure-runtime-java-extension-functions/src/main/java/org/finos/legend/pure/runtime/java/extension/functions/shared/cipher/AESCipherUtil.java#L53
gcm-detection	GCM detected, please check that IV/nonce is not reused, an Initialization Vector (IV) is a nonce used to randomize the encryption, so that even if multiple messages with identical plaintext are encrypted, the generated corresponding ciphertexts are different. Unlike the Key, the IV usually does not need to be secret, rather it is important that it is random and unique. Certain encryption schemes the IV is exchanged in public as part of the ciphertext. Reusing same Initialization Vector with the same Key to encrypt multiple plaintext blocks allows an attacker to compare the ciphertexts and then, with some assumptions on the content of the messages, to gain important information about the data being encrypted.	low	fixed	refs/heads/master	legend-pure-runtime-java-extension-functions/src/main/java/org/finos/legend/pure/runtime/java/extension/functions/shared/cipher/AESCipherUtil.java#L76
gcm-detection	GCM detected, please check that IV/nonce is not reused, an Initialization Vector (IV) is a nonce used to randomize the encryption, so that even if multiple messages with identical plaintext are encrypted, the generated corresponding ciphertexts are different. Unlike the Key, the IV usually does not need to be secret, rather it is important that it is random and unique. Certain encryption schemes the IV is exchanged in public as part of the ciphertext. Reusing same Initialization Vector with the same Key to encrypt multiple plaintext blocks allows an attacker to compare the ciphertexts and then, with some assumptions on the content of the messages, to gain important information about the data being encrypted.	low	unresolved	refs/heads/master	legend-pure-runtime-java-extension-functions/src/main/java/org/finos/legend/pure/runtime/java/extension/functions/shared/cipher/AESCipherUtil.java#L77

Semgrep SAST Scan Report for Repository: nnayar-r2c/new-vulnado

Report Generated at 2024-02-28 09:28

SAST Scan Summary

Vulnerability Severity	Vulnerability Count
Findings- SAST High Severity	7
Findings- SAST Medium Severity	25
Findings- SAST Low Severity	0

Findings Summary- HIGH Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
detected-jwt-token	JWT token detected	high	unresolved	master	exercises/02-xss.md#L65
command-injection-process-builder	A formatted or concatenated string was detected as input to a ProcessBuilder call. This is dangerous if a variable is controlled by user input and could result in a command injection. Ensure your variables are not controlled by users or sufficiently sanitized.	high	unresolved	master	src/main/java/com/scalesec/vulnado/Cowsay.java#L11
formatted-sql-string	Detected a formatted string in a SQL statement. This could lead to SQL injection if variables in the SQL statement are not properly sanitized. Use a prepared statements (java.sql.PreparedStatement) instead. You can obtain a PreparedStatement using 'connection.prepareStatement'.	high	unresolved	master	src/main/java/com/scalesec/vulnado/User.java#L49
var-in-script-tag	Detected a template variable used in a script tag. Although template variables are HTML escaped, HTML escaping does not always prevent cross-site scripting (XSS) attacks when used directly in JavaScript. If you need this data on the rendered page, consider placing it in the HTML portion (outside of a script tag). Alternatively, use a JavaScript-specific encoder, such as the one available in OWASP ESAPI. For Django, you may also consider using the 'json_script' template tag and retrieving the data in your script by using the element ID (e.g., `document.getElementById`).	high	unresolved	master	client/index.html#L63
var-in-script-tag	Detected a template variable used in a script tag. Although template variables are HTML escaped, HTML escaping does not always prevent cross-site scripting (XSS) attacks when used directly in JavaScript. If you need this data on the rendered page, consider placing it in the HTML portion (outside of a script tag). Alternatively, use a JavaScript-specific encoder, such as the one available in OWASP ESAPI. For Django, you may also consider using the 'json_script' template tag and retrieving the data in your script by using the element ID (e.g., `document.getElementById`).	high	unresolved	master	client/index.html#L67
var-in-script-tag	Detected a template variable used in a script tag. Although template variables are HTML escaped, HTML escaping does not always prevent cross-site scripting (XSS) attacks when used directly in JavaScript. If you need this data on the rendered page, consider placing it in the HTML portion (outside of a script tag). Alternatively, use a JavaScript-specific encoder, such as the one available in OWASP ESAPI. For Django, you may also consider using the 'json_script' template tag and retrieving the data in your script by using the element ID (e.g., `document.getElementById`).	high	unresolved	master	client/index.html#L67
var-in-script-tag	Detected a template variable used in a script tag. Although template variables are HTML escaped, HTML escaping does not always prevent cross-site scripting (XSS) attacks when used directly in JavaScript. If you need this data on the rendered page, consider placing it in the HTML portion (outside of a script tag). Alternatively, use a JavaScript-specific encoder, such as the one available in OWASP ESAPI. For Django, you may also consider using the 'json_script' template tag and retrieving the data in your script by using the element ID (e.g., `document.getElementById`).	high	unresolved	master	client/index.html#L73

Findings Summary- MEDIUM Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
aws-subnet-has-public-ip-address	Resources in the AWS subnet are assigned a public IP address. Resources should not be exposed on the public internet, but should have access limited to consumers required for the function of your application. Set `map_public_ip_on_launch` to false so that resources are not publicly-accessible.	medium	unresolved	master	reverse_shell/main.tf#L33
var-in-script-tag	Detected a template variable used in a script tag. Although template variables are HTML escaped, HTML escaping does not always prevent cross-site scripting (XSS) attacks when used directly in JavaScript. If you need this data on the rendered page, consider placing it in the HTML portion (outside of a script tag). Alternatively, use a JavaScript-specific encoder, such as the one available in OWASP ESAPI. For Django, you may also consider using the 'json_script' template tag and retrieving the data in your script by using the element ID (e.g., `document.getElementById`).	medium	unresolved	master	client/index.html#L63
var-in-script-tag	Detected a template variable used in a script tag. Although template variables are HTML escaped, HTML escaping does not always prevent cross-site scripting (XSS) attacks when used directly in JavaScript. If you need this data on the rendered page, consider placing it in the HTML portion (outside of a script tag). Alternatively, use a JavaScript-specific encoder, such as the one available in OWASP ESAPI. For Django, you may also consider using the 'json_script' template tag and retrieving the data in your script by using the element ID (e.g., `document.getElementById`).	medium	unresolved	master	client/index.html#L67
var-in-script-tag	Detected a template variable used in a script tag. Although template variables are HTML escaped, HTML escaping does not always prevent cross-site scripting (XSS) attacks when used directly in JavaScript. If you need this data on the rendered page, consider placing it in the HTML portion (outside of a script tag). Alternatively, use a JavaScript-specific encoder, such as the one available in OWASP ESAPI. For Django, you may also consider using the 'json_script' template tag and retrieving the data in your script by using the element ID (e.g., `document.getElementById`).	medium	unresolved	master	client/index.html#L67
var-in-script-tag	Detected a template variable used in a script tag. Although template variables are HTML escaped, HTML escaping does not always prevent cross-site scripting (XSS) attacks when used directly in JavaScript. If you need this data on the rendered page, consider placing it in the HTML portion (outside of a script tag). Alternatively, use a JavaScript-specific encoder, such as the one available in OWASP ESAPI. For Django, you may also consider using the 'json_script' template tag and retrieving the data in your script by using the element ID (e.g., `document.getElementById`).	medium	unresolved	master	client/index.html#L73
missing-integrity	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) youâ€™re telling the browser to fetch in the 'integrity' attribute for all externally hosted files.	medium	unresolved	master	client/index.html#L57
missing-integrity	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) youâ€™re telling the browser to fetch in the 'integrity' attribute for all externally hosted files.	medium	unresolved	master	client/index.html#L60
missing-integrity	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) youâ€™re telling the browser to fetch in the 'integrity' attribute for all externally hosted files.	medium	unresolved	master	client/login.html#L40
missing-integrity	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) youâ€™re telling the browser to fetch in the 'integrity' attribute for all externally hosted files.	medium	unresolved	master	client/login.html#L43

Finding Title	Finding Description & Remediation	severity	state	ref	location
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	master	src/main/java/com/scalesec/vulnado/Comment.java#L55
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	master	src/main/java/com/scalesec/vulnado/Comment.java#L70
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	master	src/main/java/com/scalesec/vulnado/Cowsay.java#L24
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	master	src/main/java/com/scalesec/vulnado/Postgres.java#L25
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	master	src/main/java/com/scalesec/vulnado/Postgres.java#L100
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	master	src/main/java/com/scalesec/vulnado/Postgres.java#L114
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	master	src/main/java/com/scalesec/vulnado/User.java#L34
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	master	src/main/java/com/scalesec/vulnado/User.java#L58
use-of-md5	Detected MD5 hash algorithm which is considered insecure. MD5 is not collision resistant and is therefore not suitable as a cryptographic signature. Use HMAC instead.	medium	unresolved	master	src/main/java/com/scalesec/vulnado/Postgres.java#L67
jdbc-sqli	Detected a formatted string in a SQL statement. This could lead to SQL injection if variables in the SQL statement are not properly sanitized. Use a prepared statements (java.sql.PreparedStatement) instead. You can obtain a PreparedStatement using 'connection.prepareStatement'.	medium	unresolved	master	src/main/java/com/scalesec/vulnado/User.java#L49
unrestricted-request-mapping	Detected a method annotated with 'RequestMapping' that does not specify the HTTP method. CSRF protections are not enabled for GET, HEAD, TRACE, or OPTIONS, and by default all HTTP methods are allowed when the HTTP method is not explicitly specified. This means that a method that performs state changes could be vulnerable to CSRF attacks. To mitigate, add the 'method' field and specify the HTTP method (such as 'RequestMethod.POST').	medium	unresolved	master	src/main/java/com/scalesec/vulnado/CowController.java#L11
unrestricted-request-mapping	Detected a method annotated with 'RequestMapping' that does not specify the HTTP method. CSRF protections are not enabled for GET, HEAD, TRACE, or OPTIONS, and by default all HTTP methods are allowed when the HTTP method is not explicitly specified. This means that a method that performs state changes could be vulnerable to CSRF attacks. To mitigate, add the 'method' field and specify the HTTP method (such as 'RequestMethod.POST').	medium	unresolved	master	src/main/java/com/scalesec/vulnado/LinksController.java#L15
unrestricted-request-mapping	Detected a method annotated with 'RequestMapping' that does not specify the HTTP method. CSRF protections are not enabled for GET, HEAD, TRACE, or OPTIONS, and by default all HTTP methods are allowed when the HTTP method is not explicitly specified. This means that a method that performs state changes could be vulnerable to CSRF attacks. To mitigate, add the 'method' field and specify the HTTP method (such as 'RequestMethod.POST').	medium	unresolved	master	src/main/java/com/scalesec/vulnado/LinksController.java#L19

Finding Title	Finding Description & Remediation	severity	state	ref	location
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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	master	client/index.html#L73
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	master	docker-compose.yml#L23
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	master	docker-compose.yml#L23

Findings Summary- LOW Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location

Semgrep SAST Scan Report for Repository: nnayar-r2c/onfido-java

Report Generated at 2024-02-28 09:28

SAST Scan Summary

Vulnerability Severity	Vulnerability Count
Findings- SAST High Severity	0
Findings- SAST Medium Severity	16
Findings- SAST Low Severity	0

Findings Summary- HIGH Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location

Findings Summary- MEDIUM Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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/fix/remove-challenge	onfido-java/src/main/templates/Model.mustache#L19
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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/fix/remove-challenge	onfido-java/src/main/templates/Model.mustache#L33
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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/fix/remove-challenge	onfido-java/src/main/templates/Model.mustache#L49
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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/fix/remove-challenge	onfido-java/src/main/templates/Model.mustache#L50
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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/fix/remove-challenge	onfido-java/src/main/templates/Model.mustache#L54
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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/fix/remove-challenge	onfido-java/src/main/templates/Model.mustache#L55
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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/fix/remove-challenge	onfido-java/src/main/templates/Model.mustache#L60
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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/fix/remove-challenge	onfido-java/src/main/templates/Model.mustache#L61
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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/fix/remove-challenge	onfido-java/src/main/templates/Model.mustache#L64
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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/fix/remove-challenge	onfido-java/src/main/templates/Model.mustache#L67
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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/fix/remove-challenge	onfido-java/src/main/templates/Model.mustache#L76
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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/fix/remove-challenge	onfido-java/src/main/templates/Model.mustache#L77
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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/fix/remove-challenge	onfido-java/src/main/templates/Model.mustache#L79

Finding Title	Finding Description & Remediation	severity	state	ref	location
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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/fix/remove-challenge	onfido-java/src/main/templates/Model.mustache#L91
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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/fix/remove-challenge	onfido-java/src/main/templates/Model.mustache#L92
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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/fix/remove-challenge	onfido-java/src/main/templates/Model.mustache#L94

Findings Summary- LOW Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location

Semgrep SAST Scan Report for Repository: nnayar-r2c/onfido-react-native-sdk

Report Generated at 2024-02-28 09:28

SAST Scan Summary

Vulnerability Severity	Vulnerability Count
Findings- SAST High Severity	1
Findings- SAST Medium Severity	12
Findings- SAST Low Severity	0

Findings Summary- HIGH Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
detected-jwt-token	JWT token detected	high	unresolved	refs/heads/master	js/__tests__/Onfido.spec.ts#L45

Findings Summary- MEDIUM Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
exported_activity	The application exports an activity. Any application on the device can launch the exported activity which may compromise the integrity of your application or its data. Ensure that any exported activities do not have privileged access to your application's control plane.	medium	unresolved	refs/heads/master	SampleApp/android/app/src/main/AndroidManifest.xml#L17
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/master	SampleApp/android/app/src/main/java/com/sampleapp/MainApplication.java#L64
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/master	SampleApp/android/app/src/main/java/com/sampleapp/MainApplication.java#L66
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/master	SampleApp/android/app/src/main/java/com/sampleapp/MainApplication.java#L68
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/master	SampleApp/android/app/src/main/java/com/sampleapp/MainApplication.java#L70
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/master	android/src/main/java/com/onfido/reactnative/sdk/OnfidoSdkModule.java#L114
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	refs/heads/master	android/src/main/java/com/onfido/reactnative/sdk/OnfidoSdkModule.java#L310
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/master	SampleApp/scripts/examples_postinstall.js#L42
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/master	SampleApp/scripts/examples_postinstall.js#L42
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/master	SampleApp/scripts/examples_postinstall.js#L50
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/master	SampleApp/scripts/examples_postinstall.js#L85
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/master	SampleApp/scripts/examples_postinstall.js#L86

Findings Summary- LOW Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location

Semgrep SAST Scan Report for Repository: nnayar-semgrep/Vulnerable-Flask-App

Report Generated at 2024-02-28 09:27

SAST Scan Summary

Vulnerability Severity	Vulnerability Count
Findings- SAST High Severity	12
Findings- SAST Medium Severity	11
Findings- SAST Low Severity	0

Findings Summary- HIGH Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
dangerous-template-string	Found a template created with string formatting. This is susceptible to server-side template injection and cross-site scripting attacks.	high	unresolved	main	vulnerable-flask-app.py#L43
path-traversal-open	Found request data in a call to 'open'. Ensure the request data is validated or sanitized, otherwise it could result in path traversal attacks.	high	unresolved	main	vulnerable-flask-app.py#L77
path-traversal-open	Found request data in a call to 'open'. Ensure the request data is validated or sanitized, otherwise it could result in path traversal attacks.	high	unresolved	main	vulnerable-flask-app.py#L129
subprocess-injection	Detected user input entering a `subprocess` call unsafely. This could result in a command injection vulnerability. An attacker could use this vulnerability to execute arbitrary commands on the host, which allows them to download malware, scan sensitive data, or run any command they wish on the server. Do not let users choose the command to run. In general, prefer to use Python API versions of system commands. If you must use subprocess, use a dictionary to allowlist a set of commands.	high	unresolved	main	vulnerable-flask-app.py#L58
subprocess-injection	Detected user input entering a `subprocess` call unsafely. This could result in a command injection vulnerability. An attacker could use this vulnerability to execute arbitrary commands on the host, which allows them to download malware, scan sensitive data, or run any command they wish on the server. Do not let users choose the command to run. In general, prefer to use Python API versions of system commands. If you must use subprocess, use a dictionary to allowlist a set of commands.	high	unresolved	main	vulnerable-flask-app.py#L119
tainted-sql-string	Detected user input used to manually construct a SQL string. This is usually bad practice because manual construction could accidentally result in a SQL injection. An attacker could use a SQL injection to steal or modify contents of the database. Instead, use a parameterized query which is available by default in most database engines. Alternatively, consider using an object-relational mapper (ORM) such as SQLAlchemy which will protect your queries.	high	unresolved	main	vulnerable-flask-app.py#L20
insecure-deserialization	Detected the use of an insecure deserialization library in a Flask route. These libraries are prone to code execution vulnerabilities. Ensure user data does not enter this function. To fix this, try to avoid serializing whole objects. Consider instead using a serializer such as JSON.	high	unresolved	main	vulnerable-flask-app.py#L97
subprocess-shell-true	Found 'subprocess' function 'check_output' with 'shell=True'. This is dangerous because this call will spawn the command using a shell process. Doing so propagates current shell settings and variables, which makes it much easier for a malicious actor to execute commands. Use 'shell=False' instead.	high	unresolved	main	vulnerable-flask-app.py#L58
subprocess-shell-true	Found 'subprocess' function 'check_output' with 'shell=True'. This is dangerous because this call will spawn the command using a shell process. Doing so propagates current shell settings and variables, which makes it much easier for a malicious actor to execute commands. Use 'shell=False' instead.	high	unresolved	main	vulnerable-flask-app.py#L119
dangerous-subprocess-use	Detected subprocess function 'check_output' with user controlled data. A malicious actor could leverage this to perform command injection. You may consider using 'shlex.escape()'.	high	unresolved	main	vulnerable-flask-app.py#L58
dangerous-subprocess-use	Detected subprocess function 'check_output' with user controlled data. A malicious actor could leverage this to perform command injection. You may consider using 'shlex.escape()'.	high	unresolved	main	vulnerable-flask-app.py#L119
sqlalchemy-execute-raw-query	Avoiding SQL string concatenation: untrusted input concatenated with raw SQL query can result in SQL Injection. In order to execute raw query safely, prepared statement should be used. SQLAlchemy provides TextualSQL to easily used prepared statement with named parameters. For complex SQL composition, use SQL Expression Language or Schema Definition Language. In most cases, SQLAlchemy ORM will be a better option.	high	unresolved	main	vulnerable-flask-app.py#L20

Findings Summary- MEDIUM Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
path-traversal-open	Found request data in a call to 'open'. Ensure the request data is validated or sanitized, otherwise it could result in path traversal attacks and therefore sensitive data being leaked. To mitigate, consider using os.path.abspath or os.path.realpath or the pathlib library.	medium	unresolved	main	vulnerable-flask-app.py#L76
path-traversal-open	Found request data in a call to 'open'. Ensure the request data is validated or sanitized, otherwise it could result in path traversal attacks and therefore sensitive data being leaked. To mitigate, consider using os.path.abspath or os.path.realpath or the pathlib library.	medium	unresolved	main	vulnerable-flask-app.py#L127
raw-html-format	Detected user input flowing into a manually constructed HTML string. You may be accidentally bypassing secure methods of rendering HTML by manually constructing HTML and this could create a cross-site scripting vulnerability, which could let attackers steal sensitive user data. To be sure this is safe, check that the HTML is rendered safely. Otherwise, use templates (`django.shortcuts.render`) which will safely render HTML instead.	medium	unresolved	main	vulnerable-flask-app.py#L43
request-data-write	Found user-controlled request data passed into '.write(...)'. This could be dangerous if a malicious actor is able to control data into sensitive files. For example, a malicious actor could force rolling of critical log files, or cause a denial-of-service by using up available disk space. Instead, ensure that request data is properly escaped or sanitized.	medium	unresolved	main	vulnerable-flask-app.py#L128
avoid_app_run_with_bad_host	Running flask app with host 0.0.0.0 could expose the server publicly.	medium	unresolved	main	vulnerable-flask-app.py#L228
directly-returned-format-string	Detected Flask route directly returning a formatted string. This is subject to cross-site scripting if user input can reach the string. Consider using the template engine instead and rendering pages with 'render_template()'.	medium	unresolved	main	vulnerable-flask-app.py#L37
directly-returned-format-string	Detected Flask route directly returning a formatted string. This is subject to cross-site scripting if user input can reach the string. Consider using the template engine instead and rendering pages with 'render_template()'.	medium	unresolved	main	vulnerable-flask-app.py#L62
render-template-string	Found a template created with string formatting. This is susceptible to server-side template injection and cross-site scripting attacks.	medium	unresolved	main	vulnerable-flask-app.py#L51
raw-html-format	Detected user input flowing into a manually constructed HTML string. You may be accidentally bypassing secure methods of rendering HTML by manually constructing HTML and this could create a cross-site scripting vulnerability, which could let attackers steal sensitive user data. To be sure this is safe, check that the HTML is rendered safely. Otherwise, use templates (`flask.render_template`) which will safely render HTML instead.	medium	unresolved	main	vulnerable-flask-app.py#L43
formatted-sql-query	Detected possible formatted SQL query. Use parameterized queries instead.	medium	unresolved	main	vulnerable-flask-app.py#L20
avoid-pickle	Avoid using `pickle`, which is known to lead to code execution vulnerabilities. When unpickling, the serialized data could be manipulated to run arbitrary code. Instead, consider serializing the relevant data as JSON or a similar text-based serialization format.	medium	unresolved	main	vulnerable-flask-app.py#L97

Findings Summary- LOW Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location

Semgrep SAST Scan Report for Repository: nnayar-semgrep/javaspringvulny

Report Generated at 2024-02-28 09:27

SAST Scan Summary

Vulnerability Severity	Vulnerability Count
Findings- SAST High Severity	14
Findings- SAST Medium Severity	31
Findings- SAST Low Severity	0

Findings Summary- HIGH Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
tainted-code-injection-from-http-request-deepsemgrep	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	main	src/main/java/hawk/code-inj.java#L67
tainted-code-injection-from-http-request	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	main	src/main/java/hawk/code-inj.java#L67
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	main	src/main/java/hawk/service/UserSearchService.java#L30
tainted-code-injection-from-http-request-deepsemgrep	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	main	src/main/java/hawk/code-inj.java#L42
tainted-code-injection-from-http-request-deepsemgrep	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	main	src/main/java/hawk/code-inj.java#L52
tainted-code-injection-from-http-request	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	main	src/main/java/hawk/code-inj.java#L42
tainted-code-injection-from-http-request	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	main	src/main/java/hawk/code-inj.java#L52
tainted-code-injection-from-http-request-deepsemgrep	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	refs/pull/2/merge	src/main/java/hawk/injection.java#L41
tainted-code-injection-from-http-request-deepsemgrep	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	refs/pull/2/merge	src/main/java/hawk/injection.java#L51
tainted-code-injection-from-http-request	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	refs/pull/2/merge	src/main/java/hawk/injection.java#L41
tainted-code-injection-from-http-request	Passing unsanitized user input to a Script Engine or other means of dynamic code evaluation is unsafe. This could lead to code injection with data leakage or arbitrary code execution as a result. Avoid this, or use proper sandboxing if user code evaluation is intended.	high	unresolved	refs/pull/2/merge	src/main/java/hawk/injection.java#L51
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	main	src/main/resources/keyStore.pem#L5

Finding Title	Finding Description & Remediation	severity	state	ref	location
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	main	src/main/java/hawk/api/jwt/JwtLog4jController.java#L24
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	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	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	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	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	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	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	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	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	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	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	main	src/main/resources/templates/user-search.html#L14
el-injection	An expression is built with a dynamic value. The source of the value(s) should be verified to avoid that unfiltered values fall into this risky code evaluation.	medium	unresolved	main	src/main/java/hawk/code-inj.java#L22
el-injection	An expression is built with a dynamic value. The source of the value(s) should be verified to avoid that unfiltered values fall into this risky code evaluation.	medium	unresolved	main	src/main/java/hawk/code-inj.java#L62
el-injection	An expression is built with a dynamic value. The source of the value(s) should be verified to avoid that unfiltered values fall into this risky code evaluation.	medium	unresolved	refs/pull/2/merge	src/main/java/hawk/injection.java#L21
el-injection	An expression is built with a dynamic value. The source of the value(s) should be verified to avoid that unfiltered values fall into this risky code evaluation.	medium	unresolved	refs/pull/2/merge	src/main/java/hawk/injection.java#L61
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	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	main	src/main/java/hawk/controller/LoginController.java#L57

Finding Title	Finding Description & Remediation	severity	state	ref	location
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	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	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	main	src/main/java/hawk/controller/LoginController.java#L57
spring-csrf-disabled	CSRF protection is disabled for this configuration. This is a security risk.	medium	unresolved	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	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	main	src/main/java/hawk/MultiHttpSecurityConfig.java#L110
django-no-csrf-token	Manually-created forms in django templates should specify a csrf_token to prevent CSRF attacks	medium	fixed	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	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	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	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	main	src/main/resources/templates/user-search.html#L14
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	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	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	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	main	docker-compose.yml#L12

Findings Summary- LOW Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location

Semgrep SAST Scan Report for Repository: nnayar-semgrep/juice-shop

Report Generated at 2024-02-28 09:27

SAST Scan Summary

Vulnerability Severity	Vulnerability Count
Findings- SAST High Severity	19
Findings- SAST Medium Severity	75
Findings- SAST Low Severity	3

Findings Summary- HIGH Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
detected-generic-secret	Generic Secret detected	high	unresolved	bugfix/score-board-fixes	data/static/users.yml#L150
detected-jwt-token	JWT token detected	high	unresolved	bugfix/score-board-fixes	frontend/src/app/app.guard.spec.ts#L40
detected-jwt-token	JWT token detected	high	unresolved	bugfix/score-board-fixes	frontend/src/app/last-login-ip/last-login-ip.component.spec.ts#L50
detected-jwt-token	JWT token detected	high	unresolved	bugfix/score-board-fixes	frontend/src/app/last-login-ip/last-login-ip.component.spec.ts#L56
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	bugfix/score-board-fixes	frontend/src/hacking-instructor/index.ts#L107
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	bugfix/score-board-fixes	routes/dataExport.ts#L61
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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	routes/likeProductReviews.ts#L31
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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	routes/trackOrder.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	bugfix/score-board-fixes	routes/updateProductReviews.ts#L18
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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	data/static/codefixes/unionSqlInjectionChallenge_3.ts#L10
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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	routes/search.ts#L23

Findings Summary- MEDIUM Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
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	bugfix/score-board-fixes	server.ts#L245
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	bugfix/score-board-fixes	server.ts#L250
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	bugfix/score-board-fixes	server.ts#L254
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	bugfix/score-board-fixes	routes/fileServer.ts#L33
detected-private-key	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	bugfix/score-board-fixes	lib/insecurity.ts#L23
detected-private-key	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	bugfix/score-board-fixes	lib/insecurity.ts#L56
detected-private-key	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	bugfix/score-board-fixes	lib/insecurity.ts#L158
detect-non-literal-regexp	RegExp() called with a `file` 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	bugfix/score-board-fixes	lib/codingChallenges.ts#L76
detect-non-literal-regexp	RegExp() called with a `paths` 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	bugfix/score-board-fixes	lib/codingChallenges.ts#L76
detect-non-literal-regexp	RegExp() called with a `file` 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	bugfix/score-board-fixes	lib/codingChallenges.ts#L78

Finding Title	Finding Description & Remediation	severity	state	ref	location
detect-non-literal-regexp	RegExp() called with a `paths` 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	bugfix/score-board-fixes	lib/codingChallenges.ts#L78
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	bugfix/score-board-fixes	lib/insecurity.ts#L44
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	bugfix/score-board-fixes	lib/insecurity.ts#L158
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	bugfix/score-board-fixes	lib/codingChallenges.ts#L24
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	bugfix/score-board-fixes	routes/fileServer.ts#L33
missing-integrity	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) youâ€™re telling the browser to fetch in the 'integrity' attribute for all externally hosted files.	medium	unresolved	bugfix/score-board-fixes	frontend/src/index.html#L14
missing-integrity	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) youâ€™re telling the browser to fetch in the 'integrity' attribute for all externally hosted files.	medium	unresolved	bugfix/score-board-fixes	frontend/src/index.html#L15
missing-integrity	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) youâ€™re telling the browser to fetch in the 'integrity' attribute for all externally hosted files.	medium	unresolved	bugfix/score-board-fixes	frontend/src/index.html#L16
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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	routes/userProfile.ts#L36
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	bugfix/score-board-fixes	routes/profileImageFileUpload.ts#L28

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	bugfix/score-board-fixes	routes/profileImageUrlUpload.ts#L31
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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	routes/vulnCodeFixes.ts#L82
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	bugfix/score-board-fixes	routes/vulnCodeSnippet.ts#L93
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	bugfix/score-board-fixes	routes/vulnCodeSnippet.ts#L94
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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	routes/profileImageUrlUpload.ts#L23
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	bugfix/score-board-fixes	server.ts#L245

Finding Title	Finding Description & Remediation	severity	state	ref	location
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	bugfix/score-board-fixes	server.ts#L250
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	bugfix/score-board-fixes	server.ts#L254
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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	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	unresolved	bugfix/score-board-fixes	routes/redirect.ts#L19
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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	routes/fileServer.ts#L33
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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	routes/profileImageUrlUpload.ts#L23
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	bugfix/score-board-fixes	views/promotionVideo.pug#L79
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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	lib/insecurity.ts#L201
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	unresolved	bugfix/score-board-fixes	lib/insecurity.ts#L56
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	bugfix/score-board-fixes	lib/utils.ts#L97
detected-private-key	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	bugfix/score-board-fixes	lib/insecurity.ts#L23
detect-non-literal-regexp	RegExp() called with a `challengeKey` 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	bugfix/score-board-fixes	lib/codingChallenges.ts#L76
detect-non-literal-regexp	RegExp() called with a `challengeKey` 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	bugfix/score-board-fixes	lib/codingChallenges.ts#L78
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	bugfix/score-board-fixes	lib/insecurity.ts#L44
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	bugfix/score-board-fixes	lib/insecurity.ts#L158
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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	lib/codingChallenges.ts#L24
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	bugfix/score-board-fixes	lib/codingChallenges.ts#L24
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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	lib/startup/validatePreconditions.ts#L94
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	bugfix/score-board-fixes	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	fixed	bugfix/score-board-fixes	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	unresolved	bugfix/score-board-fixes	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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	routes/quarantineServer.ts#L14
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	bugfix/score-board-fixes	frontend/src/hacking-instructor/helpers/helpers.ts#L36

Finding Title	Finding Description & Remediation	severity	state	ref	location
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	bugfix/score-board-fixes	routes/videoHandler.ts#L57
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	bugfix/score-board-fixes	routes/videoHandler.ts#L69
vm-runincontext-context-injection	Make sure that unverified user data can not reach vm.runInContext.	medium	unresolved	bugfix/score-board-fixes	routes/b2bOrder.ts#L22
vm-runincontext-context-injection	Make sure that unverified user data can not reach vm.runInContext.	medium	unresolved	bugfix/score-board-fixes	routes/fileUpload.ts#L80
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	bugfix/score-board-fixes	models/index.ts#L31
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	bugfix/score-board-fixes	frontend/src/app/search-result/search-result.component.ts#L151
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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	docker-compose.test.yml#L7

Findings Summary- LOW Severity

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	bugfix/score-board-fixes	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('<', '<')`. 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	bugfix/score-board-fixes	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	bugfix/score-board-fixes	server.ts#L96

Semgrep SAST Scan Report for Repository: pre-commit-full

Report Generated at 2024-02-28 09:28

SAST Scan Summary

Vulnerability Severity	Vulnerability Count
Findings- SAST High Severity	13
Findings- SAST Medium Severity	52
Findings- SAST Low Severity	0

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	https://github.com/nnayar-r2c/vulnado.git/master	sqli.js#L34
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	https://github.com/nnayar-r2c/vulnado.git/master	sqli.js#L36
formatted-sql-string-deepsemgrep	Untrusted input might be used to build a database query, which can lead to a SQL injection vulnerability. An attacker can execute malicious SQL statements and gain unauthorized access to sensitive data, modify, delete data, or execute arbitrary system commands. To prevent this vulnerability, use prepared statements that do not concatenate user-controllable strings and use parameterized queries where SQL commands and user data are strictly separated. Also, consider using an object-relational (ORM) framework to operate with safer abstractions. To build SQL queries safely in Java, it is possible to adopt prepared statements by using the `java.sql.PreparedStatement` class with bind variables.	high	fixed	https://github.com/nnayar-r2c/vulnado.git/master	src/main/java/com/scalesec/vulnado/User.java#L49
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	https://github.com/nnayar-r2c/vulnado.git/master	src/main/java/com/scalesec/vulnado/User.java#L47
tainted-url-host	User data flows into the host portion of this manually-constructed URL. This could allow an attacker to send data to their own server, potentially exposing sensitive data such as cookies or authorization information sent with this request. They could also probe internal servers or other resources that the server runnig this code can access. (This is called server-side request forgery, or SSRF.) Do not allow arbitrary hosts. Instead, create an allowlist for approved hosts hardcode the correct host, or ensure that the user data can only affect the path or parameters.	high	fixed	https://github.com/nnayar-r2c/vulnado.git/master	src/main/java/com/scalesec/vulnado/LinkLister.java#L26
detected-jwt-token	JWT token detected	high	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	demo.json#L1
detected-jwt-token	JWT token detected	high	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	exercises/02-xss.md#L65
command-injection-process-builder	A formatted or concatenated string was detected as input to a ProcessBuilder call. This is dangerous if a variable is controlled by user input and could result in a command injection. Ensure your variables are not controlled by users or sufficiently sanitized.	high	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	src/main/java/com/scalesec/vulnado/Cowsay.java#L11
formatted-sql-string	Detected a formatted string in a SQL statement. This could lead to SQL injection if variables in the SQL statement are not properly sanitized. Use a prepared statements (java.sql.PreparedStatement) instead. You can obtain a PreparedStatement using 'connection.prepareStatement'.	high	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	src/main/java/com/scalesec/vulnado/User.java#L49
var-in-script-tag	Detected a template variable used in a script tag. Although template variables are HTML escaped, HTML escaping does not always prevent cross-site scripting (XSS) attacks when used directly in JavaScript. If you need this data on the rendered page, consider placing it in the HTML portion (outside of a script tag). Alternatively, use a JavaScript-specific encoder, such as the one available in OWASP ESAPI. For Django, you may also consider using the 'json_script' template tag and retrieving the data in your script by using the element ID (e.g., `document.getElementById`).	high	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	client/index.html#L63

Finding Title	Finding Description & Remediation	severity	state	ref	location
var-in-script-tag	Detected a template variable used in a script tag. Although template variables are HTML escaped, HTML escaping does not always prevent cross-site scripting (XSS) attacks when used directly in JavaScript. If you need this data on the rendered page, consider placing it in the HTML portion (outside of a script tag). Alternatively, use a JavaScript-specific encoder, such as the one available in OWASP ESAPI. For Django, you may also consider using the 'json_script' template tag and retrieving the data in your script by using the element ID (e.g., `document.getElementById`).	high	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	client/index.html#L67
var-in-script-tag	Detected a template variable used in a script tag. Although template variables are HTML escaped, HTML escaping does not always prevent cross-site scripting (XSS) attacks when used directly in JavaScript. If you need this data on the rendered page, consider placing it in the HTML portion (outside of a script tag). Alternatively, use a JavaScript-specific encoder, such as the one available in OWASP ESAPI. For Django, you may also consider using the 'json_script' template tag and retrieving the data in your script by using the element ID (e.g., `document.getElementById`).	high	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	client/index.html#L67
var-in-script-tag	Detected a template variable used in a script tag. Although template variables are HTML escaped, HTML escaping does not always prevent cross-site scripting (XSS) attacks when used directly in JavaScript. If you need this data on the rendered page, consider placing it in the HTML portion (outside of a script tag). Alternatively, use a JavaScript-specific encoder, such as the one available in OWASP ESAPI. For Django, you may also consider using the 'json_script' template tag and retrieving the data in your script by using the element ID (e.g., `document.getElementById`).	high	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	client/index.html#L73

Findings Summary- MEDIUM Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
var-in-script-tag	Detected a template variable used in a script tag. Although template variables are HTML escaped, HTML escaping does not always prevent cross-site scripting (XSS) attacks when used directly in JavaScript. If you need this data on the rendered page, consider placing it in the HTML portion (outside of a script tag). Alternatively, use a JavaScript-specific encoder, such as the one available in OWASP ESAPI. For Django, you may also consider using the 'json_script' template tag and retrieving the data in your script by using the element ID (e.g., `document.getElementById`).	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	client/index.html#L63
var-in-script-tag	Detected a template variable used in a script tag. Although template variables are HTML escaped, HTML escaping does not always prevent cross-site scripting (XSS) attacks when used directly in JavaScript. If you need this data on the rendered page, consider placing it in the HTML portion (outside of a script tag). Alternatively, use a JavaScript-specific encoder, such as the one available in OWASP ESAPI. For Django, you may also consider using the 'json_script' template tag and retrieving the data in your script by using the element ID (e.g., `document.getElementById`).	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	client/index.html#L67
var-in-script-tag	Detected a template variable used in a script tag. Although template variables are HTML escaped, HTML escaping does not always prevent cross-site scripting (XSS) attacks when used directly in JavaScript. If you need this data on the rendered page, consider placing it in the HTML portion (outside of a script tag). Alternatively, use a JavaScript-specific encoder, such as the one available in OWASP ESAPI. For Django, you may also consider using the 'json_script' template tag and retrieving the data in your script by using the element ID (e.g., `document.getElementById`).	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	client/index.html#L67
var-in-script-tag	Detected a template variable used in a script tag. Although template variables are HTML escaped, HTML escaping does not always prevent cross-site scripting (XSS) attacks when used directly in JavaScript. If you need this data on the rendered page, consider placing it in the HTML portion (outside of a script tag). Alternatively, use a JavaScript-specific encoder, such as the one available in OWASP ESAPI. For Django, you may also consider using the 'json_script' template tag and retrieving the data in your script by using the element ID (e.g., `document.getElementById`).	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	client/index.html#L73
missing-integrity	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) youâ€™re telling the browser to fetch in the 'integrity' attribute for all externally hosted files.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	client/index.html#L57
missing-integrity	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) youâ€™re telling the browser to fetch in the 'integrity' attribute for all externally hosted files.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	client/index.html#L60
missing-integrity	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) youâ€™re telling the browser to fetch in the 'integrity' attribute for all externally hosted files.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	client/login.html#L40

Finding Title	Finding Description & Remediation	severity	state	ref	location
missing-integrity	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) youâ€™re telling the browser to fetch in the 'integrity' attribute for all externally hosted files.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	client/login.html#L43
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/new-vul.java#L10
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/new-vul.java#L19
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/new-vul.java#L28
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/new-vul.java#L33
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/new-vul.java#L35
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/new-vul.java#L41
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/very-new-vul.java#L10
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/very-new-vul.java#L19
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/very-new-vul.java#L28
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/very-new-vul.java#L33
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/very-new-vul.java#L35
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/very-new-vul.java#L41

Finding Title	Finding Description & Remediation	severity	state	ref	location
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/very-new-vul.java#L47
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/very-new-vul.java#L54
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/very-new-vul.java#L61
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/vuln.java#L10
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/vuln.java#L20
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/vuln.java#L30
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/vuln.java#L40
active-debug-code-getstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/vuln.java#L50
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	src/main/java/com/scalesec/vulnado/Comment.java#L55
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	src/main/java/com/scalesec/vulnado/Comment.java#L70
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	src/main/java/com/scalesec/vulnado/Cowsay.java#L24
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	src/main/java/com/scalesec/vulnado/Postgres.java#L25
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	src/main/java/com/scalesec/vulnado/Postgres.java#L100
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	src/main/java/com/scalesec/vulnado/Postgres.java#L114

Finding Title	Finding Description & Remediation	severity	state	ref	location
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	src/main/java/com/scalesec/vulnado/User.java#L34
active-debug-code-printstacktrace	Possible active debug code detected. Deploying an application with debug code can create unintended entry points or expose sensitive information.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	src/main/java/com/scalesec/vulnado/User.java#L58
use-of-md5	Detected MD5 hash algorithm which is considered insecure. MD5 is not collision resistant and is therefore not suitable as a cryptographic signature. Use HMAC instead.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	src/main/java/com/scalesec/vulnado/Postgres.java#L67
jdbc-sqli	Detected a formatted string in a SQL statement. This could lead to SQL injection if variables in the SQL statement are not properly sanitized. Use a prepared statements (java.sql.PreparedStatement) instead. You can obtain a PreparedStatement using 'connection.prepareStatement'.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	src/main/java/com/scalesec/vulnado/User.java#L49
unrestricted-request-mapping	Detected a method annotated with 'RequestMapping' that does not specify the HTTP method. CSRF protections are not enabled for GET, HEAD, TRACE, or OPTIONS, and by default all HTTP methods are allowed when the HTTP method is not explicitly specified. This means that a method that performs state changes could be vulnerable to CSRF attacks. To mitigate, add the 'method' field and specify the HTTP method (such as 'RequestMethod.POST').	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	src/main/java/com/scalesec/vulnado/CowController.java#L11
unrestricted-request-mapping	Detected a method annotated with 'RequestMapping' that does not specify the HTTP method. CSRF protections are not enabled for GET, HEAD, TRACE, or OPTIONS, and by default all HTTP methods are allowed when the HTTP method is not explicitly specified. This means that a method that performs state changes could be vulnerable to CSRF attacks. To mitigate, add the 'method' field and specify the HTTP method (such as 'RequestMethod.POST').	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	src/main/java/com/scalesec/vulnado/LinksController.java#L15
unrestricted-request-mapping	Detected a method annotated with 'RequestMapping' that does not specify the HTTP method. CSRF protections are not enabled for GET, HEAD, TRACE, or OPTIONS, and by default all HTTP methods are allowed when the HTTP method is not explicitly specified. This means that a method that performs state changes could be vulnerable to CSRF attacks. To mitigate, add the 'method' field and specify the HTTP method (such as 'RequestMethod.POST').	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	src/main/java/com/scalesec/vulnado/LinksController.java#L19
template-explicit-unescape	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. 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	https://github.com/nnayar-r2c/vulnado.git/master	client/index.html#L73
hardcoded-salt	Cryptographic operations were identified that leverage a hardcoded salt/nonce. A salt does not need to remain secret, but should be random, generated from cryptographically secure sources of entropy, such as an CSPRNG. On iOS/macOS platforms, secure random data can be obtained via the `SecCopyRandomBytes` API available from RandomizationServices.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/hardcoded_secrets.swift#L60
hardcoded-salt	Cryptographic operations were identified that leverage a hardcoded salt/nonce. A salt does not need to remain secret, but should be random, generated from cryptographically secure sources of entropy, such as an CSPRNG. On iOS/macOS platforms, secure random data can be obtained via the `SecCopyRandomBytes` API available from RandomizationServices.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/hardcoded_secrets.swift#L74
hardcoded-symmetric-key	A hard-coded cryptographic key was detected. An attacker that obtains this key via reverse engineering or access to source code will be able to re-use this key to encrypt, decrypt, and/or sign data at will. Cryptographic keys should be unique, and randomly generated per user, per client.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/hardcoded_secrets.swift#L21
hardcoded-symmetric-key	A hard-coded cryptographic key was detected. An attacker that obtains this key via reverse engineering or access to source code will be able to re-use this key to encrypt, decrypt, and/or sign data at will. Cryptographic keys should be unique, and randomly generated per user, per client.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/hardcoded_secrets.swift#L57

Finding Title	Finding Description & Remediation	severity	state	ref	location
insecure-crypto-aes-keysize	AES symmetric cryptographic operations were identified using a key size of 128bit which is less than the industry standard recommendation of 256bit.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/hardcoded_secrets.swift#L22
insecure-crypto-aes-keysize	AES symmetric cryptographic operations were identified using a key size of 128bit which is less than the industry standard recommendation of 256bit.	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/hardcoded_secrets.swift#L40
insecure-crypto-cbc-mode	Symmetric cryptographic operations were identified that use Cipher Block Chaining (CBC) mode. AES in CBC mode provides unauthenticated cryptographic encryption. CBC is also malleable, meaning that an attacker can influence the decrypted plaintext by modifying bits of the ciphertext (bit flipping attacks). Consider using an authenticated encryption mechanism, such as AES-GCM or ChaChaPoly. If CBC mode is **required**, consider augmenting the encryption with authentication by signing the ciphertexts with a Message Authentication Code (e.g. HMAC).	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/hardcoded_secrets.swift#L16
insecure-crypto-cbc-mode	Symmetric cryptographic operations were identified that use Cipher Block Chaining (CBC) mode. AES in CBC mode provides unauthenticated cryptographic encryption. CBC is also malleable, meaning that an attacker can influence the decrypted plaintext by modifying bits of the ciphertext (bit flipping attacks). Consider using an authenticated encryption mechanism, such as AES-GCM or ChaChaPoly. If CBC mode is **required**, consider augmenting the encryption with authentication by signing the ciphertexts with a Message Authentication Code (e.g. HMAC).	medium	unresolved	https://github.com/nnayar-r2c/vulnado.git/master	vulns/hardcoded_secrets.swift#L34
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	https://github.com/nnayar-r2c/vulnado.git/master	docker-compose.yml#L23
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	https://github.com/nnayar-r2c/vulnado.git/master	docker-compose.yml#L23

Findings Summary- LOW Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location

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

Report Generated at 2024-02-28 09:28

SAST Scan Summary

Vulnerability Severity	Vulnerability Count
Findings- SAST High Severity	3
Findings- SAST Medium Severity	13
Findings- SAST Low Severity	0

Findings Summary- HIGH Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
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	main	src/main/resources/keyStore.pem#L5
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	main	src/main/java/hawk/api/jwt/JwtLog4jController.java#L24
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	main	src/main/resources/application-postgresql.properties#L36

Findings Summary- MEDIUM Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
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	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	main	src/main/java/hawk/controller/LoginController.java#L57
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	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	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	main	src/main/java/hawk/controller/LoginController.java#L57
spring-csrf-disabled	CSRF protection is disabled for this configuration. This is a security risk.	medium	unresolved	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	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	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	unresolved	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	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	main	docker-compose.yml#L3
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	main	docker-compose.yml#L12

Finding Title	Finding Description & Remediation	severity	state	ref	location
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	main	.github/workflows/hawkscan.yml#L23

Findings Summary- LOW Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location

Semgrep SAST Scan Report for Repository: projects

Report Generated at 2024-02-28 09:28

SAST Scan Summary

Vulnerability Severity	Vulnerability Count
Findings- SAST High Severity	12
Findings- SAST Medium Severity	8
Findings- SAST Low Severity	0

Findings Summary- HIGH Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
detected-generic-secret	Generic Secret detected	high	unresolved	feature-bad-password-vuln	findings-short.json#L1
detected-generic-secret	Generic Secret detected	high	unresolved	feature-bad-password-vuln	findings-shorter.json#L1
detected-generic-secret	Generic Secret detected	high	unresolved	feature-bad-password-vuln	findings.json#L1
detected-jwt-token	JWT token detected	high	unresolved	feature-bad-password-vuln	findings-short.json#L1
detected-jwt-token	JWT token detected	high	unresolved	feature-bad-password-vuln	findings-short.json#L1
detected-jwt-token	JWT token detected	high	unresolved	feature-bad-password-vuln	findings-short.json#L1
detected-jwt-token	JWT token detected	high	unresolved	feature-bad-password-vuln	findings-shorter.json#L1
detected-jwt-token	JWT token detected	high	unresolved	feature-bad-password-vuln	findings-shorter.json#L1
detected-jwt-token	JWT token detected	high	unresolved	feature-bad-password-vuln	findings-shorter.json#L1
detected-jwt-token	JWT token detected	high	unresolved	feature-bad-password-vuln	findings.json#L1
detected-jwt-token	JWT token detected	high	unresolved	feature-bad-password-vuln	findings.json#L1
detected-jwt-token	JWT token detected	high	unresolved	feature-bad-password-vuln	findings.json#L1

Findings Summary- MEDIUM Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location
missing-integrity	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) youâ€™re telling the browser to fetch in the 'integrity' attribute for all externally hosted files.	medium	unresolved	feature-bad-password-vuln	semgrep_sast_findings_legend-engine_20230727-1821.html#L4
missing-integrity	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) youâ€™re telling the browser to fetch in the 'integrity' attribute for all externally hosted files.	medium	unresolved	feature-bad-password-vuln	semgrep_sast_findings_legend-engine_20230727-1821.html#L126
missing-integrity	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) youâ€™re telling the browser to fetch in the 'integrity' attribute for all externally hosted files.	medium	unresolved	feature-bad-password-vuln	semgrep_sast_findings_legend-engine_20230727-1821.html#L709
missing-integrity	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) youâ€™re telling the browser to fetch in the 'integrity' attribute for all externally hosted files.	medium	unresolved	feature-bad-password-vuln	semgrep_sast_findings_legend-engine_20230727-1821.html#L764
missing-integrity	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) youâ€™re telling the browser to fetch in the 'integrity' attribute for all externally hosted files.	medium	unresolved	feature-bad-password-vuln	semgrep_sast_findings_legend-engine_20230727-1822.html#L4
missing-integrity	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) youâ€™re telling the browser to fetch in the 'integrity' attribute for all externally hosted files.	medium	unresolved	feature-bad-password-vuln	semgrep_sast_findings_legend-engine_20230727-1822.html#L126
missing-integrity	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) youâ€™re telling the browser to fetch in the 'integrity' attribute for all externally hosted files.	medium	unresolved	feature-bad-password-vuln	semgrep_sast_findings_legend-engine_20230727-1822.html#L709
missing-integrity	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) youâ€™re telling the browser to fetch in the 'integrity' attribute for all externally hosted files.	medium	unresolved	feature-bad-password-vuln	semgrep_sast_findings_legend-engine_20230727-1822.html#L764

Findings Summary- LOW Severity

Finding Title	Finding Description & Remediation	severity	state	ref	location