Semgrep SAST Scan Report for Repository: local_scan/vulnado

Report Generated at 2024-09-04 21:52

SAST Scan Summary

Vulnerability Severity	Vulnerability Count
Findings- SAST High Severity	37
Findings- SAST Medium Severity	59
Findings- SAST Low Severity	0

Findings Summary- High Severity

Finding Title	Finding Description & Remediation	severity	status	ref	location
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	open	master	src/main/java/com/scalesec/vulnado/Cowsay.java#L11
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	open	master	src/main/java/com/scalesec/vulnado/User.java#L49
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	open	master	src/main/java/com/scalesec/vulnado/User.java#L49
tainted-system-command	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 this is unavoidable, validate and sanitize the input, and	high	open	master	src/main/java/com/scalesec/vulnado/Cowsay.java#L11

Finding Title	Finding Description & Remediation	severity	status	ref	location
	use safe methods for executing the commands. For more information, see: [Java command injection prevention](https://semgrep.dev/docs/cheat-sheets/java-command-injection/)				
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	open	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	open	master	src/main/java/com/scalesec/vulnado/LinkLister.java#L26
tainted-ssrf-spring-add	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.	high	open	master	src/main/java/com/scalesec/vulnado/LinkLister.java#L16

Finding Title	Finding Description & Remediation	severity	status	ref	location
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 Sequelize which will protect your queries.	high	open	master	sqli.js#L35
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 Sequelize which will protect your queries.	high	open	master	sqli.js#L38
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 Sequelize which will protect your queries.	high	open	master	sqli.js#L41
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	high	open	master	sqli.js#L44

Finding Title	Finding Description & Remediation	severity	status	ref	location
	object-relational mapper (ORM) such as Sequelize which will protect your queries.				
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 Sequelize which will protect your queries.	high	open	master	sqli.js#L47
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 Sequelize which will protect your queries.	high	open	master	sqli.js#L50
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 Sequelize which will protect your queries.	high	open	master	sqli.js#L53
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	high	open	master	v-sqli.js#L34

Finding Title	Finding Description & Remediation	severity	status	ref	location
	in most database engines. Alternatively, consider using an object-relational mapper (ORM) such as Sequelize which will protect your queries.				
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 Sequelize which will protect your queries.	high	open	master	v-sqli.js#L36
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 Sequelize which will protect your queries.	high	open	master	v-sqli.js#L39
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 Sequelize which will protect your queries.	high	open	master	vv-sqli.js#L34
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.	high	open	master	vv-sqli.js#L36

Finding Title	Finding Description & Remediation	severity	status	ref	location
	Instead, use a parameterized query which is available by default in most database engines. Alternatively, consider using an object-relational mapper (ORM) such as Sequelize which will protect your queries.				
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 Sequelize which will protect your queries.	high	open	master	vv-sqli.js#L39
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	open	master	sqli.js#L35
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	open	master	sqli.js#L38
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	open	master	sqli.js#L41
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	high	open	master	sqli.js#L44

Finding Title	Finding Description & Remediation	severity	status	ref	location
	recommended to use parameterized queries or prepared statements.				
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	open	master	sqli.js#L47
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	open	master	sqli.js#L50
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	open	master	sqli.js#L53
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	open	master	v-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	open	master	v-sqli.js#L36
	Detected a sequelize statement that is tainted by user-input. This could lead to SQL injection if the variable is user-controlled and	high	open	master	v-sqli.js#L39

Finding Title	Finding Description & Remediation	severity	status	ref	location
express-sequelize-injection	is not properly sanitized. In order to prevent SQL injection, it is recommended to use parameterized queries or prepared statements.				
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	open	master	vv-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	open	master	vv-sqli.js#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	open	master	vv-sqli.js#L39
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	open	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	high	open	master	client/index.html#L67

Finding Title	Finding Description & Remediation	severity	status	ref	location
	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`).				
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	open	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	open	master	client/index.html#L73

Findings Summary- Medium Severity

Finding Title	Finding Description & Remediation	severity	status	ref	location
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 'cookie.setSecure(true);'	medium	open	master	vulns/newnewnenwbadcookie.java#L23
cookie-secure-flag-false	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 'cookie.setSecure(true);'	medium	open	master	vulns/newnewnenwbadcookie.java#L23
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	open	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	medium	open	master	client/index.html#L67

Finding Title	Finding Description & Remediation	severity	status	ref	location
	(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`).				
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	open	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	open	master	client/index.html#L73
missing-integrity	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows	medium	open	master	client/index.html#L57

Finding Title	Finding Description & Remediation	severity	status	ref	location
	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.				
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	open	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	medium	open	master	client/login.html#L40

Finding Title	Finding Description & Remediation	severity	status	ref	location
	fetch in the 'integrity' attribute for all externally hosted files.				
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	open	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	open	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	open	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	open	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	open	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	open	master	vulns/new-vul.java#L35

Finding Title	Finding Description & Remediation	severity	status	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	open	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	open	master	vulns/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	open	master	vulns/new-vul.java#L49
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	open	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	open	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	open	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	open	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	open	master	vulns/very-new-vul.java#L35
		medium	open	master	vulns/very-new-vul.java#L41

Finding Title	Finding Description & Remediation	severity	status	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.				
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	open	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	open	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	open	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	open	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	open	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	open	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	open	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	open	master	vulns/vuln.java#L50

Finding Title	Finding Description & Remediation	severity	status	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	open	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	open	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	open	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	open	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	open	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	open	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	open	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	open	master	src/main/java/com/scalesec/vulnado/User.java#L58
	A cookie was detected without setting the 'secure' flag. The 'secure' flag for cookies prevents the client from transmitting the cookie over insecure	medium	open	master	vulns/cookie-secure-flag.java#L23

Finding Title	Finding Description & Remediation	severity	status	ref	location
cookie-missing-secure-flag	channels such as HTTP. Set the 'secure' flag by calling 'cookie.setSecure(true);'				
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	open	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	open	master	src/main/java/com/scalesec/vulnado/User.java#L49
cookie-secure-flag-false	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 'cookie.setSecure(true);'	medium	open	master	vulns/cookie-secure-flag.java#L23
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	open	master	src/main/java/com/scalesec/vulnado/CowController.java#L11
	Detected a method annotated with 'RequestMapping' that does not specify the HTTP	medium	open	master	src/main/java/com/scalesec/vulnado/LinksController.java#L15

Finding Title	Finding Description & Remediation	severity	status	ref	location
unrestricted-request-mapping	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').				
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	open	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	open	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	medium	open	master	vulns/hardcoded_secrets.swift#L60

Finding Title	Finding Description & Remediation	severity	status	ref	location
	the `SecCopyRandomBytes` API available from RandomizationServices.				
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	open	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	open	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	open	master	vulns/hardcoded_secrets.swift#L57
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	open	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	open	master	vulns/hardcoded_secrets.swift#L40

Finding Title	Finding Description & Remediation	severity	status	ref	location
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	open	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	open	master	vulns/hardcoded_secrets.swift#L34
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	medium	open	master	reverse_shell/main.tf#L33

Finding Title	Finding Description & Remediation	severity	status	ref	location
	`map_public_ip_on_launch` to false so that resources are not publicly-accessible.				
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	open	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	open	master	docker-compose.yml#L23

Findings Summary- Low Severity

Finding Title	Finding Description & Remediation	severity	status	ref	location