SYNOPSYS®

Coverity Support for OWASP Mobile Top 10 (2016)

Java

Coverity version 2020.09 — Java		
Category	**Description**	**Coverity checker**
M1: Improper Platform Usage	Misuse of a platform feature or failure to use platform security controls	ANDROID_CAPABILITY_LEAK, ANDROID_DEBUG_MODE, CONFIG.ANDROID_OUTDATED_ TARGETSDKVERSION, CONFIG. ANDROID_UNSAFE_ MINSDKVERSION, CONFIG.DWR_DEBUG_MODE, IMPLICIT_INTENT, MISSING_ PERMISSION_FOR_BROADCAST, MISSING_PERMISSION_ON_ EXPORTED_ COMPONENT, SENSITIVE_DATA_LEAK, URL_MANIPULATION, XML_ EXTERNAL_ ENTITY
M2: Insecure Data Storage	This covers misuse of insecure data storage, unintended data leakage, etc.	CONFIG.ANDROID_BACKUPS_ALLOWED, EXPOSED_PREFERENCES, HARDCODED_ CREDENTIALS, PATH_MANIPULATION, SENSITIVE_DATA_LEAK, UNLOGGED_ SECURITY_EXCEPTION, UNENCRYPTED_ SENSITIVE_DATA, UNRESTRICTED_ ACCESS_TO_FILE
M3: Insecure Communication	Poor handshaking, incorrect SSL, cleartext communication of sensitive information, etc.	BAD_CERT_VERIFICATION, CONFIG. SPRING_SECURITY_LOGIN_OVER_ HTTP, INSECURE_COMMUNICATION, RISKY_CRYPTO, SENSITIVE_DATA_LEAK, UNENCRYPTED_SENSITIVE_DATA
M4: Insecure Authentication	Exploitation of authentication vulnerabilities through failures in user identification and weaknesses in session management.	HARDCODED_CREDENTIALS, MOBILE_ ID_MISUSE, WEAK_GUARD
M5: Insufficient Cryptography	The code applies insufficient cryptography to protect sensitive data which can be misused	BAD_CERT_VERIFICATION, CONFIG. SPRING_BOOT_SSL_DISABLED, CONFIG. SPRING_SECURITY_WEAK_PASSWORD_ HASH, HARDCODED_CREDENTIALS, INSECURE_RANDOM, PREDICTABLE_ RANDOM_SEED, RISKY_CRYPTO, WEAK_ PASSWORD_HASH
M6: Insecure Authorization	Exploitation of authorization vulnerabilities (e.g., failures in client-side authorization decisions, forced browsing, etc.)	ANDROID_CAPABILITY_LEAK, ANDROID_ WEBVIEW_FILEACCESS, MISSING_ PERMISSION_ FOR_BROADCAST, MISSING_ PERMISSION_ON_EXPORTED_ COMPONENT

Coverity version 2020.09 — Java		
Category	**Description**	**Coverity checker**
M7: Client Code Quality	Code level implementation problems in the mobile client	CALL_SUPER, CHECKED_RETURN, CONFIG.MYBATIS_MAPPER_SQLI, CONSTANT_EXPRESSION_RESULT, COPY_PASTE_ERROR, DC.DANGEROUS, DEADCODE, DIVIDE_BY_ZERO, FORWARD_NULL, IDENTICAL_ BRANCHES, INFINITE_LOOP, INVALIDATE_ITERATOR, JSP_SQL_ INJECTION, MISSING_BREAK, MISSING_ RESTORE, MISSING_THROW, NESTING_ INDENT_MISMATCH, NULL_RETURNS, OS_CMD_ INJECTION, OVERFLOW_ BEFORE_ WIDEN, PATH_MANIPULATION, PREDICTABLE_RANDOM_SEED, PROPERTY_MIXUP, REGEX_CONFUSION, REGEX_ INJECTION, RESOURCE_ LEAK, REVERSE_INULL, SQLI, SQL_ NOT_CONSTANT, SWAPPED_ ARGUMENTS, UNEXPECTED_ CONTROL_FLOW, UNREACHABLE, UNSAFE_DESERIALIZATION, UNSAFE_ REFLECTION, UNUSED_ VALUE, URL_ MANIPULATION, USELESS_CALL, USE_ AFTER_ FREE, WRONG_METHOD, XSS, ATTRIBUTE_NAME_CONFLICT, CONFIG. SPRING_SECURITY_DEPRECATED_XSS_ HEADER, DC.DEADLOCK, GUARDED_BY_ VIOLATION, JAVA_CODE_INJECTION, LOCK_INVERSION, LOG_INJECTION, MISSING_HEADER_VALIDATION, NON_STATIC_GUARDING_STATIC, SCRIPT_CODE_INJECTION, SENSITIVE_ DATA_LEAK, SERVLET_ATOMICITY, SINGLETON_RACE, TAINTED_ ENVIRONMENT_WITH_EXECUTION, TAINT_ASSERT, TRUST_BOUNDARY_ VIOLATION, UNKNOWN_LANGUAGE_ INJECTION, UNSAFE_JNI, XML_ INJECTION, XPATH_INJECTION
M9: Reverse Engineering	Exploitation of final code binary to determine its string tables, libraries, algorithms etc.	CONFIG.HARDCODED_CREDENTIALS_ AUDIT
M10: Extraneous Functionality	Exploitation of unintended internal functionality (e.g., hardcoded credentials or disabling of authentication during testing)	CONFIG.SPRING_SECURITY_DEBUG_ MODE, HARDCODED_CREDENTIALS

Kotlin

Coverity version 2020.09 — Kotlin		
Category	**Description**	**Coverity checker**
M1: Improper Platform Usage	Misuse of a platform feature or failure to use platform security controls	ANDROID_CAPABILITY_LEAK, ANDROID_DEBUG_MODE, IMPLICIT_INTENT, MISSING_ PERMISSION_FOR_BROADCAST, MISSING_PERMISSION_ON_ EXPORTED_COMPONENT, SENSITIVE_ DATA_LEAK, URL_MANIPULATION, XML_EXTERNAL_ENTITY, CONFIG. ANDROID_BACKUPS_ALLOWED, CONFIG.ANDROID_OUTDATED_ TARGETSDKVERSION, CONFIG. ANDROID_UNSAFE_MINSDKVERSION
M2: Insecure Data Storage	This covers misuse of insecure data storage, unintended data leakage, etc.	CONFIG.ANDROID_BACKUPS_ ALLOWED, EXPOSED_PREFERENCES, HARDCODED_CREDENTIALS, PATH_ MANIPULATION, SENSITIVE_DATA_LEAK, UNENCRYPTED_SENSITIVE_DATA, UNRESTRICTED_ACCESS_TO_ FILE
M3: Insecure Communication	Poor handshaking, incorrect SSL, cleartext communication of sensitive information, etc.	BAD_CERT_VERIFICATION, INSECURE_COMMUNICATION, RISKY_ CRYPTO, SENSITIVE_DATA_LEAK, UNENCRYPTED_SENSITIVE_DATA
M4: Insecure Authentication	Exploitation of authentication vulnerabilities through failures in user identification and weaknesses in session management.	HARDCODED_CREDENTIALS, MOBILE_ ID_MISUSE
M5: Insufficient Cryptography	The code applies insufficient cryptography to protect sensitive data which can be misused	BAD_CERT_VERIFICATION, HARDCODED_CREDENTIALS, INSECURE_RANDOM, PREDICTABLE_ RANDOM_SEED, RISKY_CRYPTO, WEAK_ PASSWORD_HASH
M6: Insecure Authorization	Exploitation of authorization vulnerabilities (e.g., failures in client-side authorization decisions, forced browsing, etc.)	ANDROID_CAPABILITY_LEAK, MISSING_ PERMISSION_FOR_BROADCAST, MISSING_PERMISSION_ON_EXPORTED_ COMPONENT
M7: Client Code Quality	Code level implementation problems in the mobile client	OS_CMD_INJECTION, PATH_ MANIPULATION, PREDICTABLE_ RANDOM_SEED, SENSITIVE_DATA_LEAK, SQLI, UNSAFE_DESERIALIZATION, URL_ MANIPULATION
M10: Extraneous Functionality	Exploitation of unintended internal functionality (e.g., hardcoded credentials or disabling of authentication during testing)	HARDCODED_CREDENTIALS

Swift

Coverity version 2020.09 — Swift		
Category	**Description**	**Coverity checker**
M1: Improper Platform Usage	Misuse of a platform feature or failure to use platform security controls	CONFIG.ATS_INSECURE, CUSTOM_ KEYBOARD_DATA_LEAK, WEAK_ BIOMETRIC_AUTH, XML_EXTERNAL_ ENTITY
M2: Insecure Data Storage	This covers misuse of insecure data storage, unintended data leakage, etc.	PATH_MANIPULATION, SENSITIVE_ DATA_LEAK, UNENCRYPTED_ SENSITIVE_DATA
M3: Insecure Communication	Poor handshaking, incorrect SSL, cleartext communication of sensitive information, etc.	BAD_CERT_VERIFICATION, CONFIG. ATS_INSECURE, INSECURE_ COMMUNICATION, INSECURE_ MULTIPEER_CONECTION, RISKY_ CRYPTO, SENSITIVE_DATA_LEAK, UNENCRYPTED_SENSITIVE_DATA
M4: Insecure Authentication	Exploitation of authentication vulnerabilities through failures in user identification and weaknesses in session management.	HARDCODED_CREDENTIALS, WEAK_ BIOMETRIC_AUTH
M5: Insufficient Cryptography	The code applies insufficient cryptography to protect sensitive data which can be misused	BAD_CERT_VERIFICATION, HARDCODED_CREDENTIALS, RISKY_ CRYPTO
M7: Client Code Quality	Code level implementation problems in the mobile client	CONSTANT_EXPRESSION_ RESULT, COPY_PASTE_ERROR, DEADCODE, FORWARD_NULL, IDENTICAL_ BRANCHES, PATH_ MANIPULATION, PROPERTY_ MIXUP, REGEX_INJECTION, REVERSE_INULL, SCRIPT_CODE_ INJECTION, SENSITIVE_DATA_LEAK, SQLI, UNEXPECTED_CONTROL_FLOW, XPATH_INJECTION
M10: Extraneous Functionality	Exploitation of unintended internal functionality (e.g., hardcoded credentials or disabling of authentication during testing).	HARDCODED_CREDENTIALS

Appendix: Detailed description of OWASP Mobile Top 10 categories.

Category	Description
M1: Improper Platform Usage	This category covers misuse of a platform feature or failure to use platform security controls. It might include Android intents, platform permissions, misuse of TouchID, the Keychain, or some other security control that is part of the mobile operating system.
M2: Insecure Data Storage	This covers insecure data storage and unintended data leakage. Insecure data storage vulnerabilities occur when development teams assume that users or malware will not have access to a mobile device's filesystem and subsequent sensitive information in data-stores on the device. Unintended data leakage (formerly side-channel data leakage) includes vulnerabilities from the OS, frameworks, compiler environment, new hardware, etc. without a developer's knowledge.
M3: Insecure Communication	This covers poor handshaking, incorrect SSL versions, weak negotiation, cleartext communication of sensitive assets, etc.
M4: Insecure Authentication	This category captures notions of authenticating the end user or bad session management. This can include failing to identify the user at all when that should be required, failure to maintain the user's identity when it is required, and weaknesses in session management.

Category	Description
M5: Insufficient Cryptography	The code applies cryptography to a sensitive information asset. However, the cryptography is insufficient in some way. Note that anything and everything related to TLS or SSL goes in M3. Also, if the app fails to use cryptography at all when it should, that probably belongs in M2. This category is for issues where cryptography was attempted, but it wasn't done correctly.
M6: Insecure Authorization	This is a category to capture any failures in authorization (e.g., authorization decisions in the client side, forced browsing, etc.). It is distinct from authentication issues (e.g., device enrolment, user identification, etc.). If the app does not authenticate users at all in a situation where it should (e.g., granting anonymous access to some resource or service when authenticated and authorized access is required), then that is an authentication failure not an authorization failure.
M7: Client Code Quality	This would be the catch-all for code-level implementation problems in the mobile client. This would capture things like buffer overflows, format string vulnerabilities, and various other code-level mistakes where the solution is to rewrite some code that's running on the mobile device.
M8: Code Tampering	This category covers binary patching, local resource modification, method hooking, method swizzling, and dynamic memory modification. Once the application is delivered to the mobile device, the code and data resources are resident there. An attacker can either directly modify the code, change the contents of memory dynamically, change or replace the system APIs that the application uses, or modify the application's data and resources. This can provide the attacker a direct method of subverting the intended use of the software for personal or monetary gain.
M9: Reverse Engineering	This category includes analysis of the final core binary to determine its source code, libraries, algorithms, and other assets. Software such as IDA Pro, Hopper, otool, and other binary inspection tools give the attacker insight into the inner workings of the application. This may be used to exploit other nascent vulnerabilities in the application, as well as revealing information about back end servers, cryptographic constants and ciphers, and intellectual property.
M10: Extraneous Functionality	Often, developers include hidden backdoor functionality or other internal development security controls that are not intended to be released into a production environment. For example, a developer may accidentally include a password as a comment in a hybrid app. Another example includes disabling of 2-factor authentication during testing.

The Synopsys difference

Synopsys helps development teams build secure, high-quality software, minimizing risks while maximizing speed and productivity. Synopsys, a recognized leader in application security, provides static analysis, software composition analysis, and dynamic analysis solutions that enable teams to quickly find and fix vulnerabilities and defects in proprietary code, open source components, and application behavior.

For more information about the Synopsys Software Integrity Group, visit us online at www.synopsys.com/software.

Synopsys, Inc.
185 Berry Street, Suite 6500
San Francisco, CA 94107 USA

U.S. Sales: 800.873.8193
International Sales: +1 415.321.5237
Email: sig-info@synopsys.com