com/thoughtworks/go/security/AESEncrypter.java	62	I	SECURITY	CIPHER_INTEGRITY	
	85	I	SECURITY	CIPHER_INTEGRITY	

The ciphertext produced is susceptible to alteration by an adversary. This mean that the cipher provides no way to detect that the data has been tampered with. If the ciphertext can be controlled by an attacker, it could be altered without detection.

The solution is to use a cipher that includes a Hash based Message Authentication Code (HMAC) to sign the data. Combining a HMAC function to the existing cipher is prone to error [1]. Specifically, it is always recommended that you be able to verify the HMAC first, and only if the data is unmodified, do you then perform any cryptographic functions on the data.

The following modes are vulnerable because they don't provide a HMAC:

- CBC

- OFB

- CTR

- ECB

The following snippets code are some examples of vulnerable code.

Code at risk:

AES in CBC mode

Cipher c = Cipher.getInstance("AES/CBC/PKCS5Padding"); c.init(Cipher.ENCRYPT_MODE, k, iv); byte[] cipherText = c.doFinal(plainText);

Triple DES with ECB mode

Cipher c = Cipher.getInstance("DESede/ECB/PKCS5Padding"); c.init(Cipher.ENCRYPT_MODE, k, iv); byte[] cipherText = c.doFinal(plainText);

Solution:

Cipher c = Cipher.getInstance("AES/GCM/NoPadding"); c.init(Cipher.ENCRYPT_MODE, k, iv); byte[] cipherText = c.doFinal(plainText);

In the example solution above, the GCM mode introduces an HMAC into the resulting encrypted data, providing integrity of the result.

References

Wikipedia: Authenticated encryption

NIST: Authenticated Encryption Modes

Moxie Marlinspike's blog: The Cryptographic Doom Principle

CWE-353: Missing Support for Integrity Check

Block ciphers, symmetric, and asymmetric encryption schemes are accessible to an application through the `Cipher` API. To obtain an instance of a specific encryption scheme, the developer calls the `Cipher.getInstance` factory method and provides a *transformation* as the argument. A transformation is a string that specifies the name of the algorithm, the cipher mode, and padding scheme to use. For example, to obtain a cipher object that uses AES in CBC mode with PKCS5 padding the developer would specify the transformation as: `AES/CBC/PKCS5Padding`. Only the algorithm part is mandatory. The security provider maintains default values for the cipher mode as well as the padding scheme should the developer choose to omit these values. Unfortunately, Java as well as Android chose ECB and PKCS7Padding as default values in case only the AES encryption algorithm is specified. Thus, a developer who only specifies the arguably secure AES block cipher ends up in a potentially insecure situation where the ECB block cipher mode is used.

"An Empirical Study of Cryptographic Misuse in Android Applications"
https://dl.acm.org/citation.cfm?doid=2508859.2516693