

T-SQL

**VB.NET** 

VB6

XML



with arguments that trigger buffer

XML parsers should not be vulnerable

Function-like macros should not be

The address of an automatic object

object that may persist after the first

"pthread\_mutex\_t" should be unlocked

in the reverse order they were locked

"pthread\_mutex\_t" should be properly

"pthread\_mutex\_t" should not be

consecutively locked or unlocked

Functions with "noreturn" attribute

"memcmp" should only be called with pointers to trivially copyable types

initialized and destroyed

should not be assigned to another

object has ceased to exist

invoked without all of their arguments

overflows

Vulnerability

to XXE attacks

Vulnerability

₩ Bug

👬 Bug

₩ Bug

# Bua

₩ Bug

₩ Bua

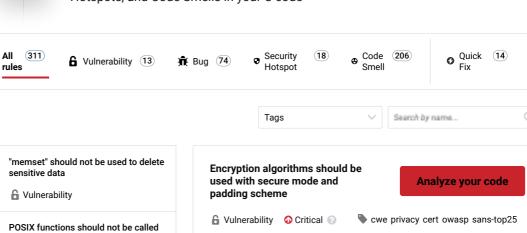
🖷 Bug

should not return

with no padding

# C static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your C code



Encryption operations should use a secure mode and padding scheme so that confidentiality and integrity can be guaranteed.

- For block cipher encryption algorithms (like AES):
  - o The ECB (Electronic Codebook) cipher mode doesn't provide serious message confidentiality: under a given key any given plaintext block always gets encrypted to the same ciphertext block. This mode never be used.
  - o The CBC (Cipher Block Chaining) mode by itself provides only data confidentiality. This cipher mode is also vulnerable to padding oracle attacks when used with padding. Using CBC along with Message Authentication Code can provide data integrity and should prevent such attacks. In practice the implementation has many pitfalls and it's recommended to avoid CBC with padding completely.
  - o The GCM (Galois Counter Mode) mode which works internally with zero/no padding scheme, is recommended, as it is designed to provide both data authenticity (integrity) and confidentiality. Other similar modes are CCM, CWC, EAX, IAPM and OCB.
- For RSA encryption algorithm, the recommended padding scheme is OAEP.

## Noncompliant Code Example

## hotan

```
#include <botan/cipher_mode.h>
 #include <botan/pubkey.h>
 #include <botan/rsa.h>
 // Example for a symmetric cipher: AES
 Botan::Cipher_Mode::create("AES-256/ECB", Botan::ENCRYPTION);
 Botan::Cipher_Mode::create("AES-256/CBC/PKCS7", Botan::ENCRYP
 // Example for a asymmetric cipher: RSA
 std::unique ptr<Botan::RandomNumberGenerator> rng(new Botan::
 Botan::RSA_PrivateKey rsaKey(*rng.get(), 2048);
 Botan::PK_Encryptor_EME(rsaKey, *rng.get(), "PKCS1v15"); // N
crypto++
```

```
#include <cryptopp/aes.h>
#include <cryptopp/modes.h>
#include <cryptopp/rsa.h>
// Example for a symmetric cipher: AES
CryptoPP::ECB Mode<CryptoPP::AES>::Encryption(); // Noncompli
CryptoPP::CBC_Mode<CryptoPP::AES>::Encryption(); // Noncompli
// Example for a asymmetric cipher: RSA
CryptoPP::RSAES<CryptoPP::PKCS1v15>::Encryptor(); // Noncompl
```

OpenSSL

Stack allocated memory and nonowned memory should not be freed

👬 Bug

Closed resources should not be accessed

# Bug

Dynamically allocated memory should be released

# Bug

Freed memory should not be used

```
#include <openss1/evp.h>
#include <openss1/rsa.h>

// Example for a symmetric cipher: AES
EVP_aes_128_ecb(); // Noncompliant
EVP_aes_128_cbc(); // Noncompliant

// Example for a asymmetric cipher: RSA
RSA_public_decrypt(flen, from, to, key, RSA_PKCS1_PADDING); //
RSA_public_decrypt(flen, from, to, key, RSA_SSLV23_PADDING);
RSA_public_decrypt(flen, from, to, key, RSA_NO_PADDING); // N
```

## Compliant Solution

#### botan

```
#include <botan/cipher_mode.h>
#include <botan/pubkey.h>
#include <botan/rsa.h>

// AES symmetric cipher is recommended to be used with GCM mo
Botan::Cipher_Mode::create("AES-256/GCM", Botan::ENCRYPTION);

// RSA asymmetric cipher is recommended to be used with OAEP
std::unique_ptr<Botan::RandomNumberGenerator> rng(new Botan::Botan::RSA_PrivateKey rsaKey(*rng.get(), 2048);

Botan::PK_Encryptor_EME(rsaKey, *rng.get(), "OAEP"); // Compl
```

## crypto++

```
#include <cryptopp/gcm.h>
// AES symmetric cipher is recommended to be used with GCM mo
CryptoPP::GCM<CryptoPP::AES>::Encryption(); // Compliant
// RSA asymmetric cipher is recommended to be used with OAEP
CryptoPP::RSAES<CryptoPP::OAEP<CryptoPP::SHA1>>::Encryptor();
```

## OpenSSL

```
#include <openssl/evp.h>

// AES symmetric cipher is recommended to be used with GCM mo
EVP_aes_128_gcm() // Compliant

// RSA asymmetric cipher is recommended be used with OAEP pad
RSA_public_decrypt(flen, from, to, key, RSA_PKCS1_OAEP_PADDIN
```

## See

- OWASP Top 10 2021 Category A2 Cryptographic Failures
- OWASP Top 10 2017 Category A6 Security Misconfiguration
- MITRE, CWE-327 Use of a Broken or Risky Cryptographic Algorithm
- SANS Top 25 Porous Defenses

## Available In:

sonarlint ⊖ sonarcloud ∴ sonarqube Developer Edition

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.

Privacy Policy