Secrets
ABAP
Apex
C
C++
CloudFormation
COBOL
C#
CSS
Flex
Go
HTML
Java
JavaScript
Kotlin
Kubernetes
Objective C
PHP
PL/I
PL/SQL
Python
RPG
Ruby
Scala
Swift
Terraform
Text
TypeScript
T-SQL
VB.NET
VB6
XML

C++ static code analysis

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

All rules 578 | Vulnerability 13 | Bug 111 | Security Hotspot 18 | Code Smell 436 | Quick Fix 68

Tags

Search by name...

"memset" should not be used to delete sensitive data
Vulnerability

POSIX functions should not be called with arguments that trigger buffer overflows
Vulnerability

XML parsers should not be vulnerable to XXE attacks
Vulnerability

Function-like macros should not be invoked without all of their arguments
Bug

The address of an automatic object should not be assigned to another object that may persist after the first object has ceased to exist
Bug

Assigning to an optional should directly target the optional
Bug

Result of the standard remove algorithms should not be ignored
Bug

"std::scoped_lock" should be created with constructor arguments
Bug

Objects should not be sliced
Bug

Immediately dangling references should not be created
Bug

"pthread_mutex_t" should be unlocked in the reverse order they were locked
Bug

"pthread_mutex_t" should be properly

Encryption algorithms should be used with secure mode and padding scheme

Analyze your code

Vulnerability | Critical | cwe privacy cert owasp sans-top25

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):
  - 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.
  - 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.
  - 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

botan

```
#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

initialized and destroyed

Bug

"pthread_mutex_t" should not be consecutively locked or unlocked twice

Bug

"std::move" and "std::forward" should not be confused

Bug

A call to "wait()" on a "std::condition_variable" should have a

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