

- 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 **311**

Vulnerability **13**

Bug **74**

Security Hotspot **18**

Code Smell **206**

Quick Fix **14**

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

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

Bug

"pthread_mutex_t" should be properly initialized and destroyed

Bug

"pthread_mutex_t" should not be consecutively locked or unlocked twice

Bug

Functions with "noreturn" attribute should not return

Bug

"memcpy" should only be called with pointers to trivially copyable types with no padding

Bug

Using pseudorandom number generators (PRNGs) is security-sensitive

Analyze your code

Security Hotspot Critical cwe cert owasp

Using pseudorandom number generators (PRNGs) is security-sensitive. For example, it has led in the past to the following vulnerabilities:

- [CVE-2013-6386](#)
- [CVE-2006-3419](#)
- [CVE-2008-4102](#)

When software generates predictable values in a context requiring unpredictability, it may be possible for an attacker to guess the next value that will be generated, and use this guess to impersonate another user or access sensitive information.

As the functions rely on a pseudorandom number generator, they should not be used for security-critical applications or for protecting sensitive data.

Ask Yourself Whether

- the code using the generated value requires it to be unpredictable. It is the case for all encryption mechanisms or when a secret value, such as a password, is hashed.
- the function you use generates a value which can be predicted (pseudo-random).
- the generated value is used multiple times.
- an attacker can access the generated value.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

- Use functions which rely on a strong random number generator such as `randombytes_uniform()` or `randombytes_buf()` from `libsodium`, or `randomize()` from `Botan`.
- Use the generated random values only once.
- You should not expose the generated random value. If you have to store it, make sure that the database or file is secure.

Sensitive Code Example

```
#include <random>
// ...

void f() {
    int random_int = std::rand(); // Sensitive
}
```

Compliant Solution

```
#include <sodium.h>
#include <botan/system_rng.h>
// ...

void f() {
    char random_chars[10];
```

Stack allocated memory and non-owned 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

```
randombytes_buf(random_chars, 10); // Compliant
uint32_t random_int = randombytes_uniform(10); // Compliant

uint8_t random_chars[10];
Botan::System_RNG system;
system.randomize(random_chars, 10); // Compliant
}
```

See

- [OWASP Top 10 2021 Category A2](#) - Cryptographic Failures
- [OWASP Top 10 2017 Category A3](#) - Sensitive Data Exposure
- [Mobile AppSec Verification Standard](#) - Cryptography Requirements
- [OWASP Mobile Top 10 2016 Category M5](#) - Insufficient Cryptography
- [MITRE, CWE-338](#) - Use of Cryptographically Weak Pseudo-Random Number Generator (PRNG)
- [MITRE, CWE-330](#) - Use of Insufficiently Random Values
- [MITRE, CWE-326](#) - Inadequate Encryption Strength
- [MITRE, CWE-1241](#) - Use of Predictable Algorithm in Random Number Generator
- [CERT, MSC02-J.](#) - Generate strong random numbers
- [CERT, MSC30-C.](#) - Do not use the rand() function for generating pseudorandom numbers
- [CERT, MSC50-CPP.](#) - Do not use std::rand() for generating pseudorandom numbers
- Derived from FindSecBugs rule [Predictable Pseudo Random Number Generator](#)

Available In:

  Developer Edition