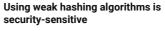


👬 Bug

C static code analysis

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





Tags

Analyze your code

Seerch by name.

cwe spring owasp sans-top25

Cryptographic hash algorithms such as MD2, MD4, MD5, MD6, HAVAL-128, HMAC-MD5, ${\tt DSA} \ (which \ uses \ {\tt SHA-1}), {\tt RIPEMD}, {\tt RIPEMD-128}, {\tt RIPEMD-160}, {\tt HMACRIPEMD160}$ and SHA-1 are no longer considered secure, because it is possible to have collisions (little computational effort is enough to find two or more different inputs that produce the same hash).

Ask Yourself Whether

The hashed value is used in a security context like:

- · User-password storage.
- Security token generation (used to confirm e-mail when registering on a website, reset password, etc ...).
- . To compute some message integrity.

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

Recommended Secure Coding Practices

Safer alternatives, such as SHA-256, SHA-512, SHA-3 are recommended, and for password hashing, it's even better to use algorithms that do not compute too "quickly", like bcrypt, scrypt, argon2 or pbkdf2 because it slows down brute force attacks.

Sensitive Code Example

```
#include <hotan/hash.h>
Botan::secure_vector<uint8_t> f(std::string input){
   std::unique_ptr<Botan::HashFunction> hash(Botan::HashFunc
   return hash->process(input);
```

Compliant Solution

```
#include <botan/hash.h>
Botan::secure_vector<uint8_t> f(std::string input){
    std::unique_ptr<Botan::HashFunction> hash(Botan::HashFunc
   return hash->process(input);
```

- OWASP Top 10 2021 Category A2 Cryptographic Failures
- OWASP Top 10 2017 Category A3 Sensitive Data Exposure
- OWASP Top 10 2017 Category A6 Security Misconfiguration
- Mobile AppSec Verification Standard Cryptography Requirements

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

OWASP Mobile Top 10 2016 Category M5 - Insufficient Cryptography

• MITRE, CWE-1240 - Use of a Risky Cryptographic Primitive

• SANS Top 25 - Porous Defenses

Available In:

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