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

T-SQL static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your T-SQL code

All rules 80 | Vulnerability 1 | Bug 16 | Security Hotspot 4 | Code Smell 59

Tags

Search by name...

Using weak hashing algorithms is security-sensitive
Security Hotspot

Dynamically executing code is security-sensitive
Security Hotspot

"SELECT" statements used as argument of "EXISTS" statements should be selective
Code Smell

Size should be specified for "varchar" variables and parameters
Code Smell

Conditionally executed code should be denoted by either indentation or BEGIN...END block
Code Smell

Conditionals should start on new lines
Code Smell

"INSERT" statements should explicitly list the columns to be set
Code Smell

Output parameters should be assigned
Bug

Queries that use "TOP" should have an "ORDER BY"
Bug

All branches in a conditional structure should not have exactly the same implementation
Bug

"WHERE" clause conditions should not be contradictory
Bug

Unary prefix operators should not be repeated

Using weak hashing algorithms is security-sensitive

Analyze your code

Security Hotspot Critical cwe spring owasp sans-top25

Cryptographic hash algorithms such as `MD2`, `MD4`, `MD5`, `MD6`, `HAVAL-128`, `HMAC-MD5`, `DSA` (which uses `SHA-1`), `RIPEMD`, `RIPEMD-128`, `RIPEMD-160`, `HMACRIPEMD160` and `SHA-1` are no longer considered secure, because it is too easy to create hash collisions with them (little computational effort is enough to find two or more different inputs that produces 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` or `bcrypt` are recommended, and for password hashing, it's even better to use algorithms that not compute too "quickly", like `bcrypt` instead of `SHA-256`, because it slows brute force and dictionary based attacks.

Sensitive Code Example

```
SELECT HASHBYTES('SHA1', MyColumn) FROM dbo.MyTable;
```

Compliant Solution

```
SELECT HASHBYTES('SHA2_256', MyColumn) FROM dbo.MyTable;
```

See

- OWASP Top 10 2017 Category A3 - Sensitive Data Exposure
- OWASP Top 10 2017 Category A6 - Security Misconfiguration
- MITRE, CWE-327 - Use of a Broken or Risky Cryptographic Algorithm
- MITRE, CWE-916 - Use of Password Hash With Insufficient Computational Effort
- 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