















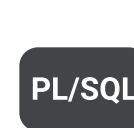


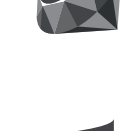




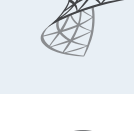









-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  Flex
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  Kubernetes
-  Objective C
-  PHP
-  PL/I
-  PL/SQL
-  Python
-  RPG
-  Ruby
-  Scala
-  Swift
-  Terraform
-  Text
-  TypeScript
-  **T-SQL**
-  VB.NET
-  VB6
-  XML














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... 🔍

SHA-1 and Message-Digest hash algorithms should not be used in secure contexts	 Vulnerability
"NOCOUNT" should be activated on "PROCEDURE" and "TRIGGER" definitions	 Code Smell
Control flow statements "IF", "WHILE" and "TRY" should not be nested too deeply	 Code Smell
"CASE" expressions should end with "ELSE" clauses	 Code Smell
"IF ... ELSEIF" constructs should end with "ELSE" clauses	 Code Smell
Control structures should use BEGIN...END blocks	 Code Smell
String literals should not be duplicated	 Code Smell
Expressions should not be too complex	 Code Smell
"WHERE" clauses should not contain redundant conditions	 Bug
Track lack of SQL Server session configuration	 Code Smell
Duplicate values should not be passed as arguments	 Code Smell

SHA-1 and Message-Digest hash algorithms should not be used in secure contexts

[Analyze your code](#)

 Vulnerability  Critical 

The MD5 algorithm and its successor, SHA-1, are no longer considered secure, because it is too easy to create hash collisions with them. That is, it takes too little computational effort to come up with a different input that produces the same MD5 or SHA-1 hash, and using the new, same-hash value gives an attacker the same access as if he had the originally-hashed value. This applies as well to the other Message-Digest algorithms: MD2, MD4, MD6, HAVAL-128, HMAC-MD5, DSA (which uses SHA-1), RIPEMD, RIPEMD-128, RIPEMD-160, HMACRIPEMD160.

Consider using safer alternatives, such as SHA-256, SHA-512 or SHA-3.

Noncompliant 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 A6](#) - Security Misconfiguration
- [MITRE, CWE-328](#) - Reversible One-Way Hash
- [MITRE, CWE-327](#) - Use of a Broken or Risky Cryptographic Algorithm
- [SANS Top 25](#) - Porous Defenses
- [SHAttered](#) - The first concrete collision attack against SHA-1.

Deprecated

This rule is deprecated; use {rule:tsql:S4790} instead.

Available In:

sonarlint  | **sonarcloud**  | **sonarqube**  Developer Edition