































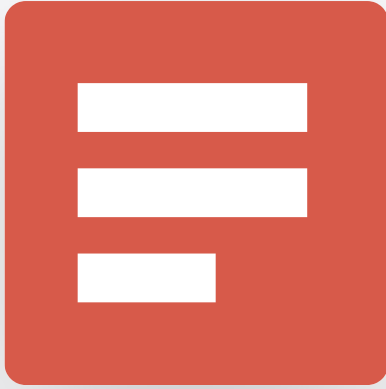



-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  Flex
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  Kubernetes
-  Objective C
-  PHP
-  PL/I
-  PL/SQL
-  Python
-  RPG
-  Ruby
-  Scala
-  Swift
-  Terraform
-  **Text**
-  TypeScript
-  T-SQL
-  VB.NET
-  VB6
-  XML




# Text static code analysis

Unique rules to find Security Hotspots in any files

- All rules 1
-  Security Hotspot 1

Tags 

Search by name... 

## Using bidirectional characters is security-sensitive

 Security Hotspot

## Using bidirectional characters is security-sensitive

Analyze your code

 Security Hotspot  Major   cwe

Using bidirectional (BIDI) characters can lead to incomprehensible code.

The Unicode encoding contains BIDI control characters that are used to display text right-to-left (RTL) instead of left-to-right (LTR). This is necessary for certain languages that use RTL text. The BIDI characters can be used to create a difference in the code between what a human sees and what a compiler or interpreter sees. An advisory might use this feature to hide a backdoor in the code that will not be spotted by a human reviewer as it is not visible.

This can lead to supply chain attacks since the backdoored code might persist over a long time without being detected and can even be included in other projects, for example in the case of libraries.

### Ask Yourself Whether

- This text requires a right-to-left writing system (to use Arabic or Hebrew words, for example).
- The author of this text is a legitimate user.
- This text contains a standard instruction, comment or sequence of characters.

There is a risk if you answered no to any of these questions.

### Recommended Secure Coding Practices

Open the file in an editor that reveals non-ASCII characters and remove all BIDI control characters that are not intended.

If hidden characters are illegitimate, this issue could indicate a potential ongoing attack on the code. Therefore, it would be best to warn your organization's security team about this issue.

Required opening BIDI characters should be explicitly closed with the PDI character.

### Sensitive Code Example

A hidden BIDI character is present in front of `return`:

```
def subtract_funds(account: str, amount: int):  
    ''' Subtract funds from bank account then return; '''  
    bank[account] -= amount  
    return
```

The executed code looks like the following:

```
def subtract_funds(account: str, amount: int):  
    ''' Subtract funds from bank account then <RLI>''' ;return  
    bank[account] -= amount  
    return
```

### Compliant Solution

No hidden BIDI characters are present:

```
def subtract_funds(account: str, amount: int):  
    ''' Subtract funds from bank account then return; '''  
    bank[account] -= amount  
    return
```

### See

- [Bidirectional Algorithm](#) - Unicode Standard
- [Wikipedia](#) - Bidirectional Text
- [Trojan Source Report](#)
- [Trojan Source Report](#) - IDEs revealing hidden characters
- [MITRE, CWE-94](#) - Improper Control of Generation of Code ('Code Injection')

Available In:  
**sonarcloud**  | **sonarqube** 