|  |
| --- |
| TrialKey4  Version 1.0  Code analysis |

|  |
| --- |
| **By: default**  **2022-05-31** |

# Content

[Content 1](#_Toc97156258)

[Introduction 2](#_Toc97156259)

[Configuration 2](#_Toc97156260)

[Synthesis 3](#_Toc97156261)

[Analysis Status 3](#_Toc97156262)

[Quality gate status 3](#_Toc97156263)

[Metrics 3](#_Toc97156264)

[Tests 3](#_Toc97156265)

[Detailed technical debt 3](#_Toc97156266)

[Metrics Range 5](#_Toc97156267)

[Volume 5](#_Toc97156268)

[Issues 6](#_Toc97156269)

[Charts 6](#_Toc97156270)

[Issues count by severity and type 8](#_Toc97156271)

[Issues List 8](#_Toc97156272)

[Security Hotspots 9](#_Toc97156273)

[Security hotspots count by category and priority 9](#_Toc97156274)

[Security hotspots List 9](#_Toc97156275)

# Introduction

This document contains results of the code analysis of TrialKey4.

# Configuration

* Quality Profiles
  + Names: Sonar way [PHP]; Sonar way [Python];
  + Files: AYBqrPLIN0\_tCdZCZ0yT.json; AYBqrO2WN0\_tCdZCZ0EW.json;
* Quality Gate
  + Name: Sonar way
  + File: Sonar way.xml

# Synthesis

## Analysis Status

|  |  |  |  |
| --- | --- | --- | --- |
| Reliability | Security | Security Review | Maintainability |
| B.png | **A.png** | **E.png** | **A.png** |

## Quality gate status

|  |  |
| --- | --- |
| Quality Gate Status | **OK.png** |

|  |  |
| --- | --- |
| Metric | Value |
| Reliability Rating on New Code | OK |
| Security Rating on New Code | OK |
| Maintainability Rating on New Code | OK |

## Metrics

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Coverage | Duplication | Comment  density | Median number of lines of code per file | Adherence to coding standard |
| 0.0 % | **17.5 %** | **9.3 %** | **10.5** | **99.6 %** |

## Tests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Total | Success Rate | Skipped | Errors | Failures |
| 0 | **0 %** | **0** | **0** | **0** |

## Detailed technical debt

|  |  |  |  |
| --- | --- | --- | --- |
| Reliability | Security | Maintainability | Total |
| 0d 0h 5min | - | 0d 1h 0min | 0d 1h 5min |

## Metrics Range

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Cyclomatic  Complexity | Cognitive  Complexity | Lines of code per file | Comment  density (%) | Coverage | Duplication (%) |
| Min | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Max | 57.0 | 134.0 | 262.0 | 12.5 | 0.0 | 40.0 |

## Volume

|  |  |
| --- | --- |
| Language | Number |
| PHP | 252 |
| Python | 10 |
| Total | 262 |

# Issues

## Charts

## Issues count by severity and type

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Type / Severity | INFO | MINOR | MAJOR | CRITICAL | BLOCKER |
| BUG | 0 | 1 | 0 | 0 | 0 |
| VULNERABILITY | 0 | 0 | 0 | 0 | 0 |
| CODE\_SMELL | 0 | 1 | 5 | 2 | 0 |

## Issues List

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Description | Type | Severity | Number |
| Image, area and button with image tags should have an "alt" attribute |  | BUG | MINOR | 1 |
| Control structures should use curly braces | While not technically incorrect, the omission of curly braces can be misleading, and may lead to the introduction of errors during maintenance. Noncompliant Code Example if (condition) // Noncompliant executeSomething(); Compliant Solution if (condition) { executeSomething(); } See CERT, EXP19-C. - Use braces for the body of an if, for, or while statement CERT, EXP52-J. - Use braces for the body of an if, for, or while statement | CODE\_SMELL | CRITICAL | 2 |
| Source files should not have any duplicated blocks | An issue is created on a file as soon as there is at least one block of duplicated code on this file | CODE\_SMELL | MAJOR | 2 |
| Collapsible "if" statements should be merged | Merging collapsible if statements increases the code's readability. Noncompliant Code Example if (condition1) { if (condition2) { ... } } Compliant Solution if (condition1 &amp;&amp; condition2) { ... } | CODE\_SMELL | MAJOR | 2 |
| Sections of code should not be commented out | Programmers should not comment out code as it bloats programs and reduces readability. Unused code should be deleted and can be retrieved from source control history if required. | CODE\_SMELL | MAJOR | 1 |
| A close curly brace should be located at the beginning of a line | Shared coding conventions make it possible for a team to efficiently collaborate. This rule makes it mandatory to place a close curly brace at the beginning of a line. Noncompliant Code Example if(condition) { doSomething();} Compliant Solution if(condition) { doSomething(); } Exceptions When blocks are inlined (open and close curly braces on the same line), no issue is triggered. if(condition) {doSomething();} | CODE\_SMELL | MINOR | 1 |

# Security Hotspots

## Security hotspots count by category and priority

|  |  |  |  |
| --- | --- | --- | --- |
| Category / Priority | LOW | MEDIUM | HIGH |
| LDAP Injection | 0 | 0 | 0 |
| Object Injection | 0 | 0 | 0 |
| Server-Side Request Forgery (SSRF) | 0 | 0 | 0 |
| XML External Entity (XXE) | 0 | 0 | 0 |
| Insecure Configuration | 0 | 0 | 0 |
| XPath Injection | 0 | 0 | 0 |
| Authentication | 0 | 0 | 1 |
| Weak Cryptography | 0 | 0 | 0 |
| Denial of Service (DoS) | 0 | 0 | 0 |
| Log Injection | 0 | 0 | 0 |
| Cross-Site Request Forgery (CSRF) | 0 | 0 | 0 |
| Open Redirect | 0 | 0 | 0 |
| SQL Injection | 0 | 0 | 0 |
| Buffer Overflow | 0 | 0 | 0 |
| File Manipulation | 0 | 0 | 0 |
| Code Injection (RCE) | 0 | 0 | 0 |
| Cross-Site Scripting (XSS) | 0 | 0 | 0 |
| Command Injection | 0 | 0 | 0 |
| Path Traversal Injection | 0 | 0 | 0 |
| HTTP Response Splitting | 0 | 0 | 0 |
| Others | 0 | 0 | 0 |

## Security hotspots List

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Category | Name | Priority | Severity | Count |
| Authentication | Hard-coded credentials are security-sensitive | HIGH | BLOCKER | 1 |