CERTIK

CertiK Audit Report for ICON

Contents

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Verification Services Agreement between CertiK and ICON (the "Company"), or the scope of services/verification, and terms and conditions provided to the Company in connection with the verification (collectively, the "Agreement"). This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

About CertiK

CertiK is a technology-led blockchain security company founded by Computer Science professors from Yale University and Columbia University built to prove the security and correctness of smart contracts and blockchain protocols.

CertiK, in partnership with grants from IBM and the Ethereum Foundation, CertiK's mission of every audit is to apply different approaches and detection methods, ranging from manual, static, and dynamic analysis, to ensure that projects are checked against known attacks and potential vulnerabilities. CertiK leverages a team of seasoned engineers and security auditors to apply testing methodologies and assessments to each project, in turn creating a more secure and robust software system.

CertiK has served more than 100 clients with high quality auditing and consulting services, ranging from stablecoins such as Binance's BGBP and Paxos Gold to decentralized oracles

such as Band Protocol and Tellor. CertiK customizes its engineering tool kits, while applying cutting-edge research on smart contracts, for each client on its project to offer a high quality deliverable. For more information: https://certik.io.

Executive Summary

This report has been prepared for **ICON** to discover issues and vulnerabilities in the source code of their **ICON Service Engine** as well as any code dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Dynamic Analysis, Static Analysis, and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the service engine against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring codebase logic meets the specifications and intentions of the client.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Testing Summary

SECURITY LEVEL

TBA

Service Engine Audit

This report has been prepared as a product of the ICON Service Engine Audit request by ICON. This audit was conducted to discover issues and vulnerabilities in the source code of ICON's Service Engine.

TYPE	Service Engine
SOURCE CODE	https://github.com/icon-project/icon-service https://github.com/icon-project/icon-service-test-suite/
LANGUAGE	Python
REQUEST DATE	March 18, 2020
DELIVERY DATE	June 16, 2020
METHODS	A comprehensive examination has been performed using Dynamic Analysis, Static Analysis, and Manual Review.

Review Notes

Introduction

CertiK team was contracted by the ICON team to audit the design and implementations of their ICON Service Engine that is meant to be utilized as a dependency by other projects. The audited source code links are:

- ICON Service: https://github.com/icon-project/icon-service/tree/eefc37cdb6b2fe181853746d1ad24fc44b8a9fc9
- ICON Service Test Suit: https://github.com/icon-project/icon-service-test-suite/tree/48fb990cf4f430d514060f68333c11bc0187df9c

The goal of this audit was to review the Python implementation for its business model, study potential security vulnerabilities, its general design and architecture, and uncover bugs that could compromise the software in production.

Documentation

We used the following sources of truth about how the **ICON Service** should work:

1. Whitepaper: https://icon.foundation/resources/whitepaper/ICON_Whitepaper_EN.pdf
2. Developer Portal: https://www.icondev.io/

Summary

The codebase of the project successfully implements an engine that is used for Smart Contract creation through the python environment. The documentation was helpful and the test suite extensive with high coverage of both base and edge cases, which highlights the level of dedication of the team.

While **most of the issues pinpointed were of negligible importance** and mostly referred to coding standards and inefficiencies, **minor flaws** were identified that should be remediated as soon as possible to ensure the codebase of the ICON team are of the highest standard and quality.

These inefficiencies and flaws should be swiftly dealt with by the development team behind the ICON project.

Recommendations

With regards to the codebase, the main recommendation we can make **is to continue updating the codebase to match the latest major versions of Python and their respected standards**, so that officially recognized libraries, as well as the team's custom codebase, deliver **high code quality and security**.

Additionally, we advise that all our findings are carefully considered and assimilated in the codebase of the project to ensure the highest code standard is achieved.

Findings

Exhibit 1: iconservice/iconscore/icon_container_db.py

TITLE	TYPE	SEVERITY	LOCATION
1. Class “ContainerUtil” inherits from object, can be safely removed from bases	Language Specific Issue	Semantic	Line 40
2. Class “DictDB” inherits from object, can be safely removed from bases	Language Specific Issue	Semantic	Line 159
3. Unnecessary "else" statement	Optimization	Semantic	Line 199
4. Class “ArrayDB” inherits from object, can be safely removed from bases	Language Specific Issue	Semantic	Line 223
5. Unnecessary "else" statement	Optimization	Semantic	Line 276
6. Class “VarDB” inherits from object, can be safely removed from bases	Language Specific Issue	Semantic	Line 350

[Numbers 1, 2, 4, 6] Description:

All Classes in Python3 automatically inherit from “object”, unless specified otherwise. The keyword “object” can be safely removed from the Classes showcased above.

Recommendations:

Remove the “object” keyword.

[Numbers 3, 5] Description:

Unnecessary "else" statement specified, the code in the "else" block will always execute if the “if” condition is not met.

Recommendations:

Remove the "else" statement.

Exhibit 2 : iconservice/iconscore/icon_score_api_generator.py

TITLE	TYPE	SEVERITY	LOCATION
1. Need type annotation for “api”	Optimization	Informational	Line 56
2. Consider merging these comparisons with "in" operator	Optimization	Informational	Line 72
3. Need type annotation for “info_list”	Optimization	Informational	Line 155
4. Unnecessary "else"	Optimization	Semantic	Line 187

statement			
5. Consider merging these comparisons with "in" operator	Optimization	Informational	Line 187
6. Need type annotation for “tmp_list”	Optimization	Informational	Line 194

[Numbers 1, 3, 6] Description:

Type annotation should be added in the variables

Recommendations:

Add the following code:

“variable_name: List[<type>] = …”

[Numbers 2, 5] Description:

Merging the targeted comparisons to a “tuple” data structure for future extendibility.

Recommendations:

Change the “if” statement to:

“if param_name in ('self', 'cls'): …”

[Number 4] Description:

See Exhibit 1, Finding number 3.

Recommendations:

See Exhibit 1, Finding number 3.

Exhibit 3 : iconservice/iconscore/icon_score_base.py

TITLE	TYPE	SEVERITY	LOCATION
1. Unnecessary "else" statement	Optimization	Semantic	Line 660
2. Unnecessary "else" statement	Optimization	Semantic	Line 667
3. “Optional” class is not imported/defined	Optimization	Minor	Line 705
4. Unnecessary "else" statement	Optimization	Semantic	Line 707
5. Unnecessary "else" statement	Optimization	Semantic	Line 715

[Numbers 1, 2, 4, 5] Description:

See Exhibit 1, Finding number 3.

Recommendations:

See Exhibit 1, Finding number 3.

[Number 3] Description:

“Optional” class of the “typing” library is not imported/defined.

Recommendations:

Add the statement "from typing import Optional".

Exhibit 4 : iconservice/iconscore/icon_score_context_util.py

TITLE	TYPE	SEVERITY	LOCATION
7. Class “ContainerUtil” inherits from object, can be safely removed from bases	Language Specific Issue	Semantic	Line 37

[Number 1] Description:

See Exhibit 1, Finding number 1.

Recommendations:

See Exhibit 1, Finding number 1.

Exhibit 5 : iconservice/iconscore/icon_score_context.py

TITLE	TYPE	SEVERITY	LOCATION

1. Class “ContextContainer” inherits from object, can be safely removed from bases	Language Specific Issue	Semantic	Line 47
2. Unnecessary "else" statement	Optimization	Semantic	Line 58
3. Unnecessary "else" statement	Optimization	Semantic	Line 79
4. Class “ContextGetter” inherits from object, can be safely removed from bases	Language Specific Issue	Semantic	Line 94
5. Class “IconScoreContext” inherits from object, can be safely removed from bases	Language Specific Issue	Semantic	Line 103
6. Need type annotation for “msg_stack”	Optimization	Informational	Line 152
7. Need type annotation for “event_log_stack”	Optimization	Informational	Line 153
8. Ambiguous naming convention for variable “decentralize_trigger”	Coding Style	Semantic	Line 166
9. Class “IconScoreContextFactory” inherits from object, can be safely removed from bases	Language Specific Issue	Semantic	Line 324

[Numbers 1, 4, 5, 9] Description:

See Exhibit 1, Finding number 1.

Recommendations:

See Exhibit 1, Finding number 1.

[Numbers 2, 3] Description:

See Exhibit 1, Finding number 3.

Recommendations:

See Exhibit 1, Finding number 3.

[Numbers 6, 7] Description:

See Exhibit 2, Finding number 1.

Recommendations:

See Exhibit 2, Finding number 1.

[Number 8] Description:

Ambiguous naming convention for variable “decentralize_trigger”

Recommendations:

Change the name of the variable so that there is no confusion when you read the code.

Exhibit 6 : iconservice/iconscore/icon_score_mapper_object.py

TITLE	TYPE	SEVERITY	LOCATION
10. Class “IconScoreInfo” inherits from object, can be safely removed from bases	Language Specific Issue	Semantic	Line 28

[Number 1] Description:

See Exhibit 1, Finding number 1.

Recommendations:

See Exhibit 1, Finding number 1.

Exhibit 7 : iconservice/iconscore/icon_score_mapper.py

TITLE	TYPE	SEVERITY	LOCATION
1. Class “IconScoreMapper” inherits from object, can be safely removed from bases	Language Specific Issue	Semantic	Line 27

[Number 1] Description:

See Exhibit 1, Finding number 1.

Recommendations:

See Exhibit 1, Finding number 1.

Exhibit 8 : iconservice/iconscore/icx.py

TITLE	TYPE	SEVERITY	LOCATION
1. Class “Icx” inherits from object, can be safely removed from bases	Language Specific Issue	Semantic	Line 29
2. No exception type(s) specified	Optimization	Informative	Line 64

[Number 1] Description:

See Exhibit 1, Finding number 1.

Recommendations:

See Exhibit 1, Finding number 1.

[Number 2] Description:

Exception has no exception type(s) specified when raised.

Recommendations:

Specify the type(s) of exception.

Exhibit 9 : iconservice/icx/base_part.py

TITLE	TYPE	SEVERITY	LOCATION
1. Class “BasePart” inherits from object, can be safely removed from bases	Language Specific Issue	Semantic	Line 27

[Number 1] Description:

See Exhibit 1, Finding number 1.

Recommendations:

See Exhibit 1, Finding number 1.

Exhibit 10 : iconservice/icx/coin_part.py

TITLE	TYPE	SEVERITY	LOCATION
Unnecessary "else"	Optimization	Semantic	Line 82

statement			
Unnecessary "else" statement	Optimization	Semantic	Line 186
Unused variable ”version”	Optimization	Informational	Line 193
Unnecessary "else" statement	Optimization	Semantic	Line 217

[Numbers 1, 2, 4] Description:

See Exhibit 1, Finding number 3.

Recommendations:

See Exhibit 1, Finding number 3.

[Number 3] Description:

Unused variable “version”.

Recommendations:

Change the name of the variable to “_” (underscore), indicating that this variable will not be used.

Exhibit 11 : tests/test_utils.py

TITLE	TYPE	SEVERITY	LOCATION
Unused imported class “PenaltyReason”	Optimization	Informational	Line 20

[Number 1] Description:

Unused class “PenaltyReason” imported in this module.

Recommendations:

Remove the statement “from iconservice.icon_constant import PenaltyReason” or use the class “PenaltyReason” in this test case.

Exhibit 12 : tests/test_icon_container_db.py

TITLE	TYPE	SEVERITY	LOCATION
1. Unnecessary "else" statement	Optimization	Semantic	Line 191

[Number 1] Description:

See Exhibit 1, Finding number 3.

Recommendations:

See Exhibit 1, Finding number 3.

Exhibit 13 : tests/test_icon_score_step.py

TITLE	TYPE	SEVERITY	LOCATION
1. Redefining built-in “type”	Language Specific Issue	Minor	Line 744
2. Unused variable “get”	Optimization	Informational	Line 795

[Number 1] Description:

Keyword “type” is reserved in Python and should not be used as a variable name.

Recommendations:

Choose a different variable name.

[Number 2] Description:

Unused variable “get”.

Recommendations:

Return the “get” variable after assignment to fulfill the purpose of the “get_db” function.

Exhibit 14 : tests/test_icon_score_trace.py

TITLE	TYPE	SEVERITY	LOCATION
1. Multiple statements on one line	Coding Style	Semantic	Line 273

[Number 1] Description:

Compound statements (multiple statements on the same line) are generally discouraged in the Python language.

Recommendations:

Formatting the code following the language conventions and formatting standards.