EtherAuthority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	Fairception SKILLERZ
Website:	www.fairception.com
Platform:	Polygon
Language:	Solidity
Date:	April 1st, 2022

Table of contents

`

Introduction

EtherAuthority was contracted by the Fairception team to perform the Security audit of the Fairception SKILLERZ minting smart contract code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on April 1st, 2022.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

- The Skillerz are the first soft skills based NFT collection reinventing the future of work on the Ethereum/Polygon blockchain.
- For each soft skill a mini-collection of 300 NFTs will be available from the start.
- The whitelist price is approximately 0.03 ETH rounded off depending on the ETH/MATIC conversion and slippage volatility during the transaction.
- Those who do not have a crypto wallet yet and decide to not mint their NFTs themselves will only be able to purchase their NFTs post whitelist during the public sale using Debit/Credit card and will be charged one-off additional NFT minting, handling, storing and airdrop fees.
- Fairception Skillerz Contract is an NFT smart contract, having functions like mintNFT, mintWhitelistNFT, airdopNFT, initDict, getCollectionOfNFT, switchMintingActive, etc.
- The Fairception Skillerz NFT contract inherits ERC721Enumerable, ERC721, Ownable standard smart contracts from the OpenZeppelin library.
- These OpenZeppelin contracts are considered community audited and time tested, and hence are not part of the audit scope. However, we confirm that the modules/code selected are the right ones with regards to the intent of the minting process and that they are recognised to be flaw free and not malicious.

Audit scope

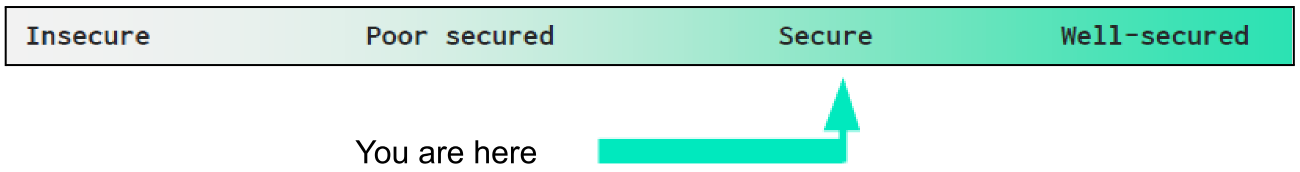
Name	**Code Review and Security Analysis Report for Fairception SKILLERZ minting Smart Contract**
Platform	**Polygon / Solidity**
File	Skillerz.sol
Online Code	0x15E6A1C5b54dD540797Dbb8Eb4Eb6F607642E7DF
Audit Date	April 1st, 2022
Revision Date	April 2nd, 2022

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Tokenomics: ● Name: Skillerz ● Symbol: SKZ ● Minting price: 0.01 Matic ● Whitelist minting price: 0.01 Matic ● Maximum mint in single transaction: 10 ● Maximum whitelist minting: 3 ● Maximum supply: 6000	**YES, This is valid.**
Ownership control: ● Owner is allowed to make airdrops to users. He can input a list of wallet addresses along with a collection number. These wallets then will receive the NFTs into their wallets. ● Owner can do reserved minting. Owner can change the number of reserved minting (by default 15). ● Owners can set prices, max minting, base URI, based extension, whitelisting, and many other things as mentioned in the centralization section below. ● Owner can withdraw the fund (Matic) out of the smart contract.	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **“secure”**. This token contract does contain owner control, which does not make it fully decentralized.



We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 1 critical, 0 high, 0 medium and 2 low and some very low level issues. These issues are resolved/acknowledged in the revised contract code.

Investors Advice: Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Passed
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Passed
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: PASSED

Code Quality

This audit scope has 1 smart contract file. Smart contract contains Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in the Fairception contract are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the Fairception Token.

The Fairception Token team has **not** provided scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Code parts are **well** commented on smart contracts. And, Ethereum's NatSpec style for commenting is used, which is a good thing.

Documentation

We were given a Fairception Token smart contract code in the form of File.The hash of that code is mentioned above in the table.

As mentioned above, code parts are **well** commented. So it is easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Another source of information was its official website https://www.fairception.com which provided rich information about the project architecture and tokenomics.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are based on well known industry standard open source projects.

Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	mintNFT	write	Passed	No Issue
3	mintWhitelistNFT	write	Passed	No Issue
4	tokenURI	read	Passed	No Issue
5	getNFTContract	read	Passed	No Issue
6	isWhitelisted	read	Passed	No Issue
7	getWhitelistClaim	read	Passed	No Issue
8	setmintingPrice	write	access only Owner	No Issue
9	setwhitelistPrice	write	access only Owner	No Issue
10	setmaxMintAmount	write	access only Owner	No Issue
11	setmaxWhitelistAmount	write	access only Owner	No Issue
12	setBaseURI	write	access only Owner	No Issue
13	setContractURI	write	access only Owner	No Issue
14	setBaseExtension	write	access only Owner	No Issue
15	switchMintingActive	write	access only Owner	No Issue
16	switchWhitelistActive	write	access only Owner	No Issue
17	switchPaused	write	access only Owner	No Issue
18	updateReserved	write	access only Owner	No Issue
19	withdraw	write	access only Owner	No Issue
20	bulkSetWhitelist	write	access only Owner	No Issue
21	mintReserved	write	access only Owner	No Issue
22	airdopNFT	write	access only Owner	No Issue
23	getCollectionOfNFT	write	Passed	No Issue
24	initDict	write	No need for empty assignment	No Big Issue

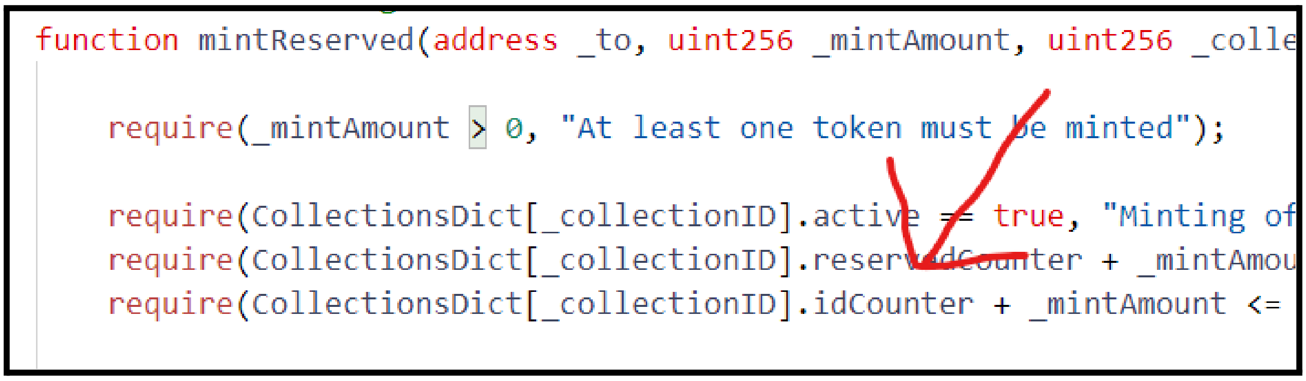
Severity Definitions

Risk Level	**Description**
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
`	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Medium	Medium-level vulnerabilities are important to fix; however, they can’t lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can’t have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can’t affect smart contract execution and can be ignored.

Audit Findings

Critical Severity

(1) Owner can not mint reserved tokens



```
function mintReserved(address _to, uint256 _mintAmount, uint256 _colle

    require(_mintAmount > 0, "At least one token must be minted");

    require(CollectionsDict[_collectionID].active == true, "Minting of
    require(CollectionsDict[_collectionID].reservedCounter + _mintAmou
    require(CollectionsDict[_collectionID].idCounter + _mintAmount <=
```

This condition will always fail for collection over 3, as the idCounter will be always more than maxSupply. More specifically, the owner will never be able to mint the reserved collection.

Resolution: Please use *CollectionsDict[_collectionID].counter* instead of *CollectionsDict[_collectionID].idCounter*

Status: Fixed

High Severity

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

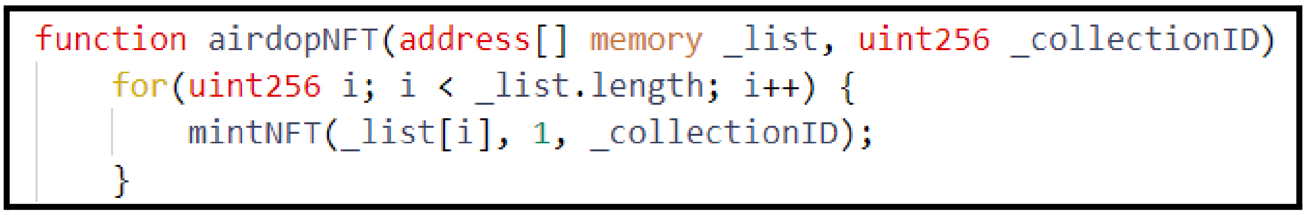
(1) Consider adding events when a significant state is changed.

It is recommended to fire appropriate events when significant state is being changed. This helps UI clients to properly coordinate with the blockchain. we suggest to add events in the following functions:

- mintNFT
- mintWhitelistNFT
- setmintingPrice
- setwhitelistPrice
- setmaxMintAmount
- setmaxWhitelistAmount
- setBaseURI
- setContractURI
- setBaseExtension
- switchMintingActive
- switchWhitelistActive
- switchPaused
- updateReserved
- withdraw
- bulkSetWhitelist
- mintReserved

Status: Fixed / Acknowledged

(2) High gas consuming loops



```
function airdopNFT(address[] memory _list, uint256 _collectionID)
    for(uint256 i; i < _list.length; i++) {
        mintNFT(_list[i], 1, _collectionID);
    }
```

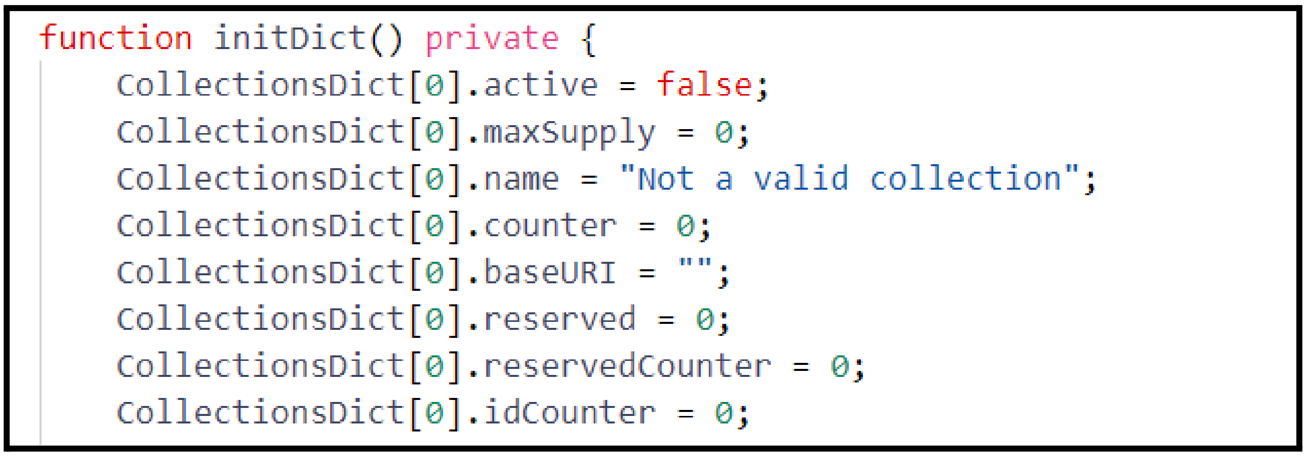
The functions airdropNFT and bulkSetWhitelist have infinite loop possibility. This is not an issue in case the owner inputs limited wallets. But it may hit the block's gas limit if there are lots of wallets used in the function call.

Resolution: We recommend adding some limit in the functions, or the owner can acknowledge that he will not input lots of wallets.

Status: Fixed

Very Low / Informational / Best practices:

(1) No need for assigning empty/zero value



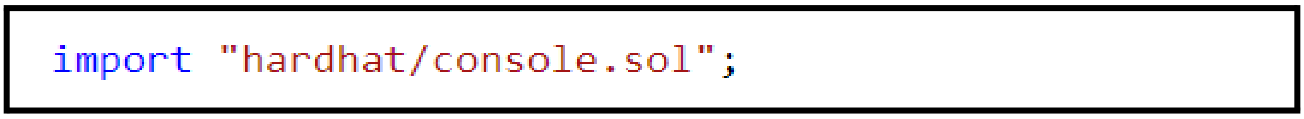
```
function initDict() private {
    CollectionsDict[0].active = false;
    CollectionsDict[0].maxSupply = 0;
    CollectionsDict[0].name = "Not a valid collection";
    CollectionsDict[0].counter = 0;
    CollectionsDict[0].baseURI = "";
    CollectionsDict[0].reserved = 0;
    CollectionsDict[0].reservedCounter = 0;
    CollectionsDict[0].idCounter = 0;
```

These variable's default values are the same as what were assigned. This is not a security or logical vulnerability. But it is better not to assign empty values explicitly to save the gas. More about empty/zero values: https://ethereum.stackexchange.com/a/40571/95100

Resolution: This variable assignments can be removed. or, this can be acknowledged as this does not raise any big problem.

Status: Fixed / Acknowledged

(2) Remove redundant code before mainnet deployment.



```
import "hardhat/console.sol";
```

The smart contract hardhat should be removed when deploying in the mainnet. This does not raise any vulnerabilities, as this is not used anywhere. But, it is best practice to remove it to make the code clean.

Status: Fixed

Centralization

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

- setmintingPrice: The Owner can set a new minting price.
- setwhitelistPrice: The Owner can set a new whitelist minting price.
- setmaxMintAmount: The Owner can set a maximum amount of NFT to mint.
- setmaxWhitelistAmount: The Owner can set the maximum amount of NFT to mint during the whitelist.
- setBaseURI: The Owner can change the collection of base uri.
- setContractURI: The Owner can change the contract URL to the contract.json file.
- setBaseExtension: The Owner can change the string ".json".
- switchMintingActive: The Owner can activate normal minting.
- switchWhitelistActive: The Owner can enable the whitelistminting state.
- switchPaused: The Owner can switch paused status.
- updateReserved: The Owner can update the reserved amount.
- withdraw: The Owner can withdraw all funds on the contract to his wallet.
- bulkSetWhitelist: The Owner can bulk upload of whitelist users accounts.
- mintReserved: The Owner can minting for the reserved NFTs. It requires the wallet address, the amount and the collection id.
- airdopNFT: The Owner is allowed to make airdrops to users.He can input a list of wallet addresses here and the target collection. These wallets then will receive the NFTs into their wallets.

Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We have observed some issues, but they are fixed / acknowledged. So, **it's good to go for the mainnet deployment**.

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed contract, based on standard audit procedure scope, is **"Secured".**

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:
In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:
Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

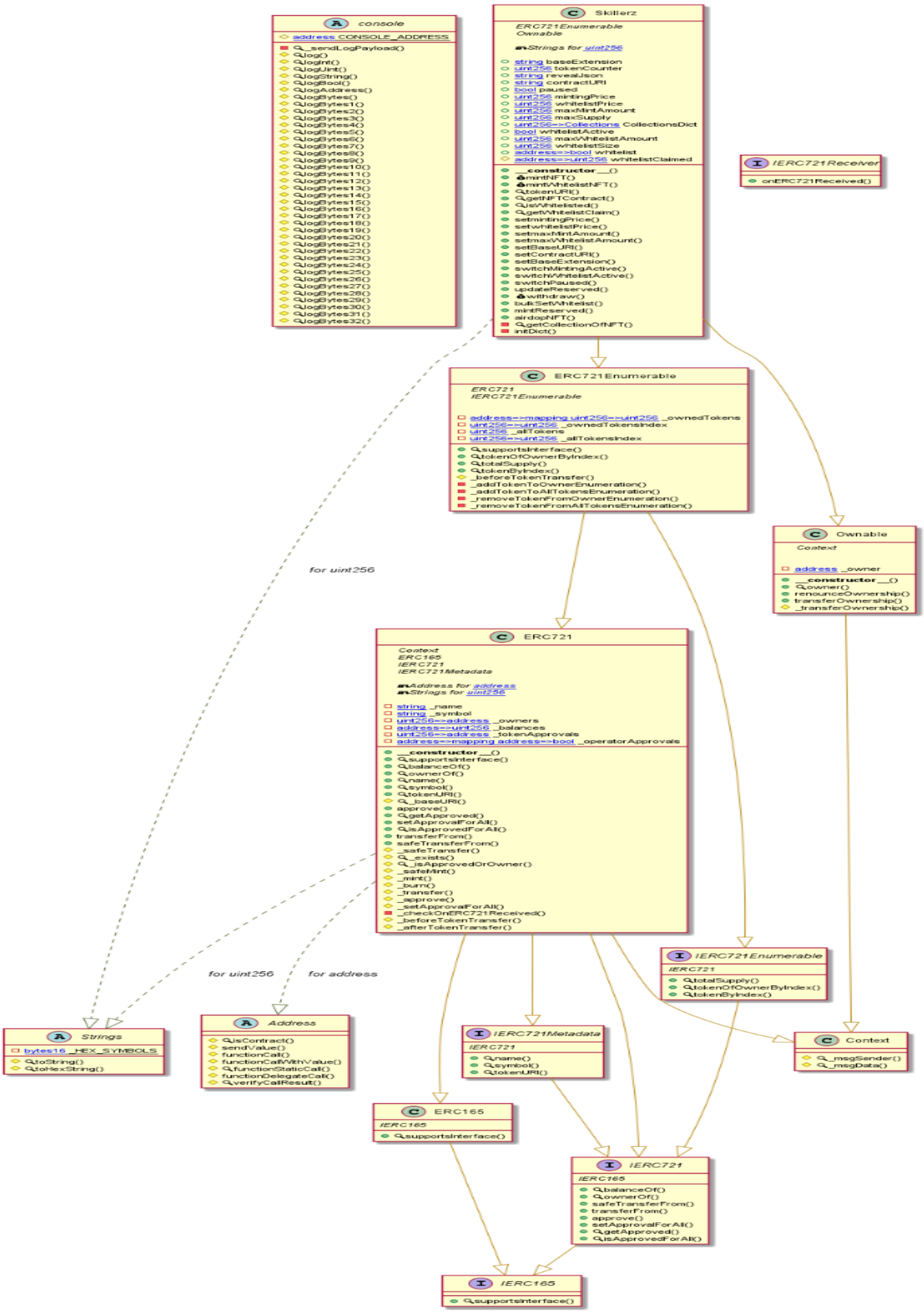
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

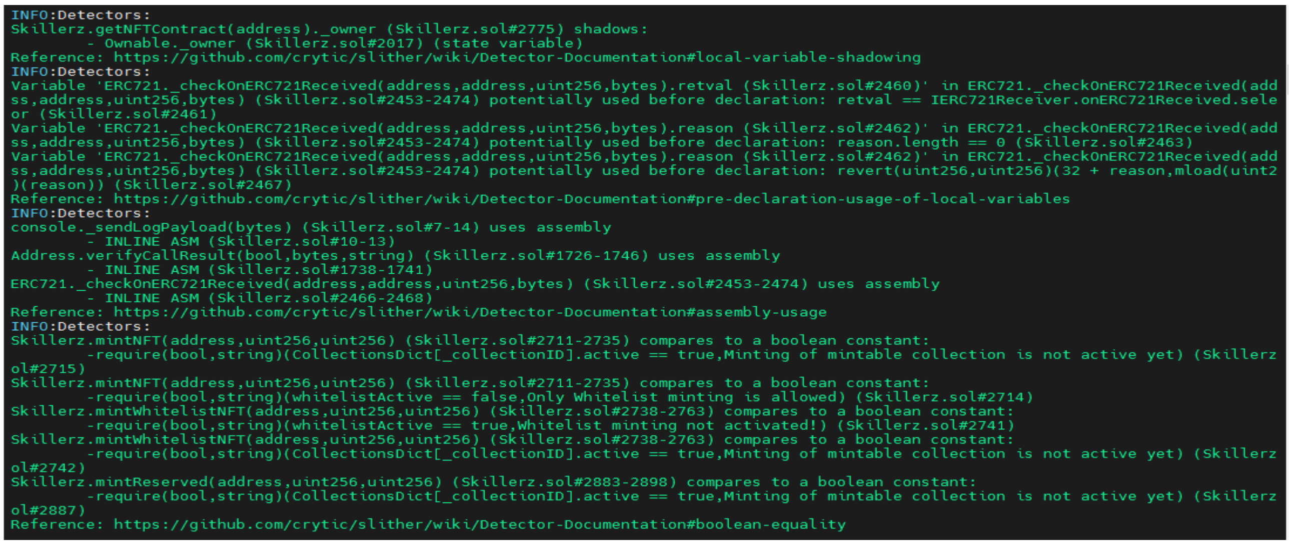
Appendix

Code Flow Diagram - Fairception Skillerz Token

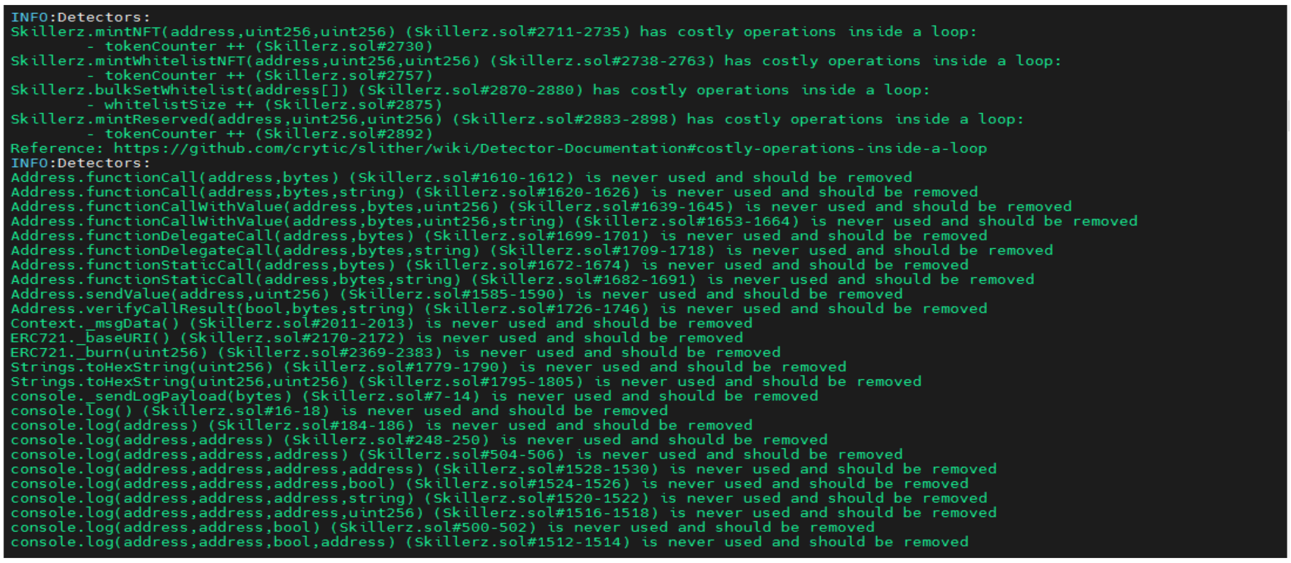


Slither Results Log

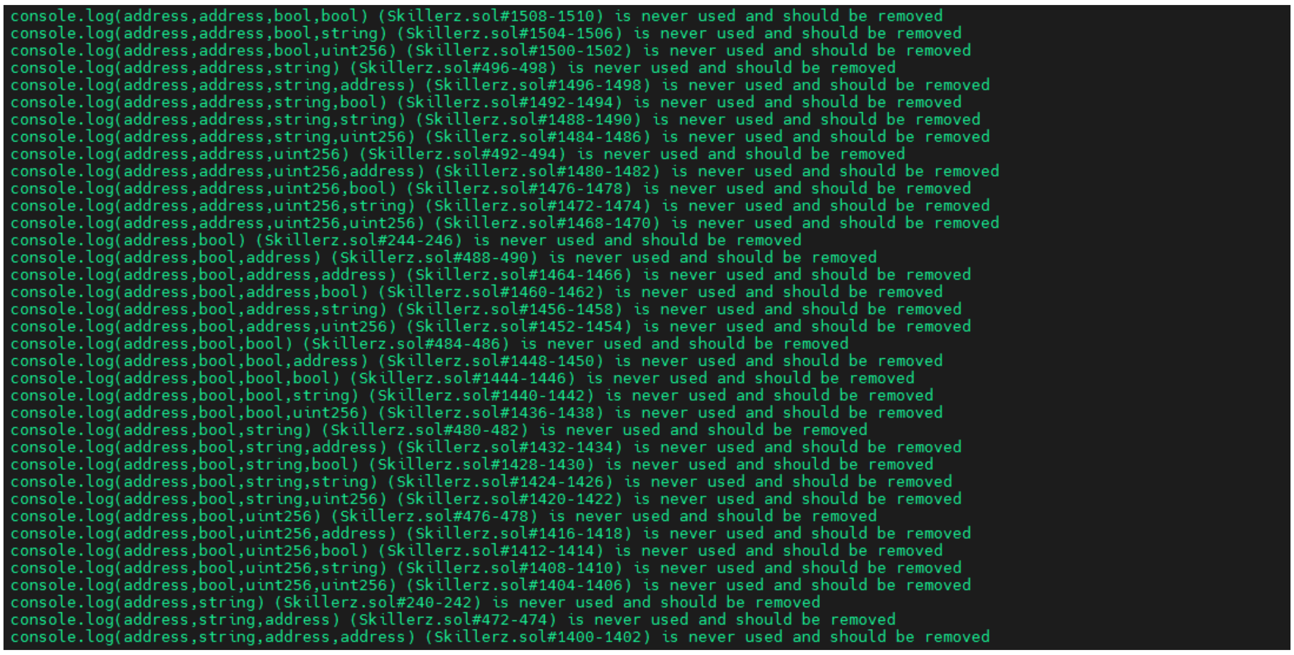
Slither log >> Skillerz.sol



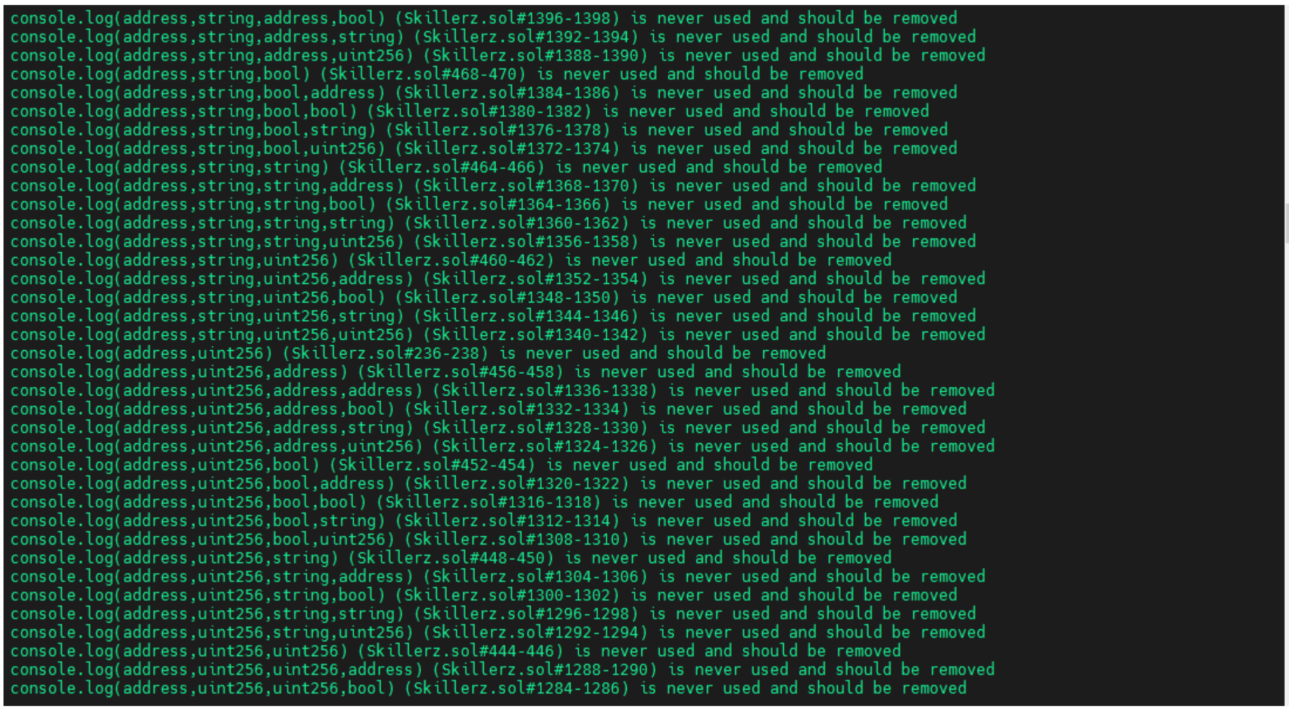
```
INFO:Detectors:
Skillerz.getNFTContract(address)._owner (Skillerz.sol#2775) shadows:
        - Ownable._owner (Skillerz.sol#2017) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (Skillerz.sol#2460)' in ERC721._checkOnERC721Received(add
ss,address,uint256,bytes) (Skillerz.sol#2453-2474) potentially used before declaration: retval == IERC721Receiver.onERC721Received.sele
or (Skillerz.sol#2461)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (Skillerz.sol#2462)' in ERC721._checkOnERC721Received(add
ss,address,uint256,bytes) (Skillerz.sol#2453-2474) potentially used before declaration: reason.length == 0 (Skillerz.sol#2463)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (Skillerz.sol#2462)' in ERC721._checkOnERC721Received(add
ss,address,uint256,bytes) (Skillerz.sol#2453-2474) potentially used before declaration: revert(uint256,uint256)(32 + reason,mload(uint2
)(reason)) (Skillerz.sol#2467)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
INFO:Detectors:
console._sendLogPayload(bytes) (Skillerz.sol#7-14) uses assembly
        - INLINE ASM (Skillerz.sol#10-13)
Address.verifyCallResult(bool,bytes,string) (Skillerz.sol#1726-1746) uses assembly
        - INLINE ASM (Skillerz.sol#1738-1741)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (Skillerz.sol#2453-2474) uses assembly
        - INLINE ASM (Skillerz.sol#2466-2468)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Skillerz.mintNFT(address,uint256,uint256) (Skillerz.sol#2711-2735) compares to a boolean constant:
        -require(bool,string)(CollectionsDict[_collectionID].active == true,Minting of mintable collection is not active yet) (Skillerz
ol#2715)
Skillerz.mintNFT(address,uint256,uint256) (Skillerz.sol#2711-2735) compares to a boolean constant:
        -require(bool,string)(whitelistActive == false,Only Whitelist minting is allowed) (Skillerz.sol#2714)
Skillerz.mintWhitelistNFT(address,uint256,uint256) (Skillerz.sol#2738-2763) compares to a boolean constant:
        -require(bool,string)(whitelistActive == true,Whitelist minting not activated!) (Skillerz.sol#2741)
Skillerz.mintWhitelistNFT(address,uint256,uint256) (Skillerz.sol#2738-2763) compares to a boolean constant:
        -require(bool,string)(CollectionsDict[_collectionID].active == true,Minting of mintable collection is not active yet) (Skillerz
ol#2742)
Skillerz.mintReserved(address,uint256,uint256) (Skillerz.sol#2883-2898) compares to a boolean constant:
        -require(bool,string)(CollectionsDict[_collectionID].active == true,Minting of mintable collection is not active yet) (Skillerz
ol#2887)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
```



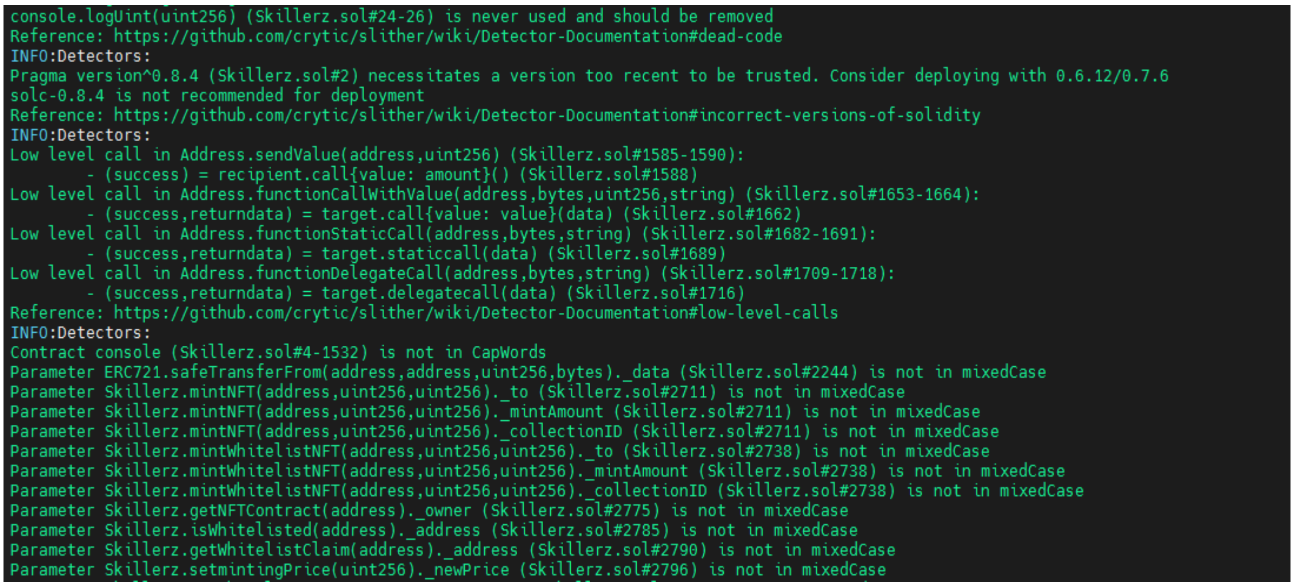
```
INFO:Detectors:
Skillerz.mintNFT(address,uint256,uint256) (Skillerz.sol#2711-2735) has costly operations inside a loop:
        - tokenCounter ++ (Skillerz.sol#2730)
Skillerz.mintWhitelistNFT(address,uint256,uint256) (Skillerz.sol#2738-2763) has costly operations inside a loop:
        - tokenCounter ++ (Skillerz.sol#2757)
Skillerz.bulkSetWhitelist(address[]) (Skillerz.sol#2870-2880) has costly operations inside a loop:
        - whitelistSize ++ (Skillerz.sol#2875)
Skillerz.mintReserved(address,uint256,uint256) (Skillerz.sol#2883-2898) has costly operations inside a loop:
        - tokenCounter ++ (Skillerz.sol#2892)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop
INFO:Detectors:
Address.functionCall(address,bytes) (Skillerz.sol#1610-1612) is never used and should be removed
Address.functionCall(address,bytes,string) (Skillerz.sol#1620-1626) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (Skillerz.sol#1639-1645) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (Skillerz.sol#1653-1664) is never used and should be removed
Address.functionDelegateCall(address,bytes) (Skillerz.sol#1699-1701) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (Skillerz.sol#1709-1718) is never used and should be removed
Address.functionStaticCall(address,bytes) (Skillerz.sol#1672-1674) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (Skillerz.sol#1682-1691) is never used and should be removed
Address.sendValue(address,uint256) (Skillerz.sol#1585-1590) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (Skillerz.sol#1726-1746) is never used and should be removed
Context._msgData() (Skillerz.sol#2011-2013) is never used and should be removed
ERC721._baseURI() (Skillerz.sol#2170-2172) is never used and should be removed
ERC721._burn(uint256) (Skillerz.sol#2369-2383) is never used and should be removed
Strings.toHexString(uint256) (Skillerz.sol#1779-1790) is never used and should be removed
Strings.toHexString(uint256,uint256) (Skillerz.sol#1795-1805) is never used and should be removed
console._sendLogPayload(bytes) (Skillerz.sol#7-14) is never used and should be removed
console.log() (Skillerz.sol#16-18) is never used and should be removed
console.log(address) (Skillerz.sol#184-186) is never used and should be removed
console.log(address,address) (Skillerz.sol#248-250) is never used and should be removed
console.log(address,address,address) (Skillerz.sol#504-506) is never used and should be removed
console.log(address,address,address,address) (Skillerz.sol#1528-1530) is never used and should be removed
console.log(address,address,address,bool) (Skillerz.sol#1524-1526) is never used and should be removed
console.log(address,address,address,string) (Skillerz.sol#1520-1522) is never used and should be removed
console.log(address,address,address,uint256) (Skillerz.sol#1516-1518) is never used and should be removed
console.log(address,address,bool) (Skillerz.sol#500-502) is never used and should be removed
console.log(address,address,bool,address) (Skillerz.sol#1512-1514) is never used and should be removed
```



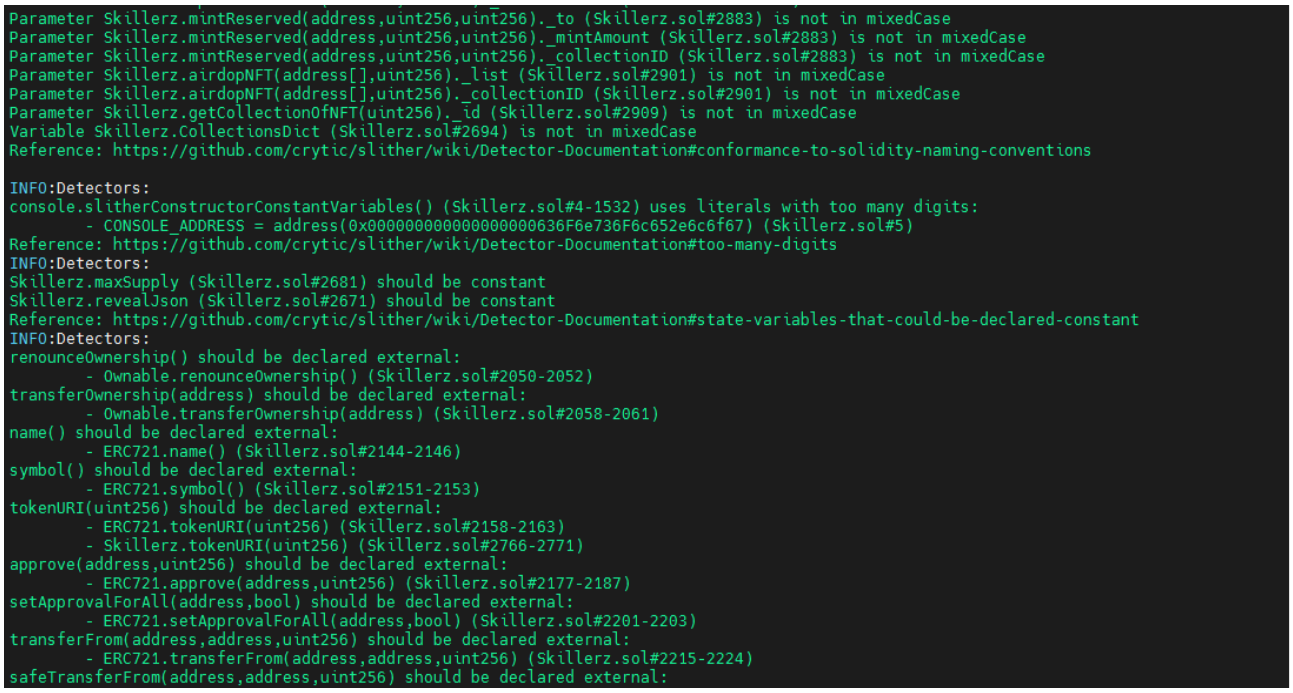
```
console.log(address,address,bool,bool) (Skillerz.sol#1508-1510) is never used and should be removed
console.log(address,address,bool,string) (Skillerz.sol#1504-1506) is never used and should be removed
console.log(address,address,bool,uint256) (Skillerz.sol#1500-1502) is never used and should be removed
console.log(address,address,string) (Skillerz.sol#496-498) is never used and should be removed
console.log(address,address,string,address) (Skillerz.sol#1496-1498) is never used and should be removed
console.log(address,address,string,bool) (Skillerz.sol#1492-1494) is never used and should be removed
console.log(address,address,string,string) (Skillerz.sol#1488-1490) is never used and should be removed
console.log(address,address,string,uint256) (Skillerz.sol#1484-1486) is never used and should be removed
console.log(address,address,uint256) (Skillerz.sol#492-494) is never used and should be removed
console.log(address,address,uint256,address) (Skillerz.sol#1480-1482) is never used and should be removed
console.log(address,address,uint256,bool) (Skillerz.sol#1476-1478) is never used and should be removed
console.log(address,address,uint256,string) (Skillerz.sol#1472-1474) is never used and should be removed
console.log(address,address,uint256,uint256) (Skillerz.sol#1468-1470) is never used and should be removed
console.log(address,bool) (Skillerz.sol#244-246) is never used and should be removed
console.log(address,bool,address) (Skillerz.sol#488-490) is never used and should be removed
console.log(address,bool,address,address) (Skillerz.sol#1464-1466) is never used and should be removed
console.log(address,bool,address,bool) (Skillerz.sol#1460-1462) is never used and should be removed
console.log(address,bool,address,string) (Skillerz.sol#1456-1458) is never used and should be removed
console.log(address,bool,address,uint256) (Skillerz.sol#1452-1454) is never used and should be removed
console.log(address,bool,bool) (Skillerz.sol#484-486) is never used and should be removed
console.log(address,bool,bool,address) (Skillerz.sol#1448-1450) is never used and should be removed
console.log(address,bool,bool,bool) (Skillerz.sol#1444-1446) is never used and should be removed
console.log(address,bool,bool,string) (Skillerz.sol#1440-1442) is never used and should be removed
console.log(address,bool,bool,uint256) (Skillerz.sol#1436-1438) is never used and should be removed
console.log(address,bool,string) (Skillerz.sol#480-482) is never used and should be removed
console.log(address,bool,string,address) (Skillerz.sol#1432-1434) is never used and should be removed
console.log(address,bool,string,bool) (Skillerz.sol#1428-1430) is never used and should be removed
console.log(address,bool,string,string) (Skillerz.sol#1424-1426) is never used and should be removed
console.log(address,bool,string,uint256) (Skillerz.sol#1420-1422) is never used and should be removed
console.log(address,bool,uint256) (Skillerz.sol#476-478) is never used and should be removed
console.log(address,bool,uint256,address) (Skillerz.sol#1416-1418) is never used and should be removed
console.log(address,bool,uint256,bool) (Skillerz.sol#1412-1414) is never used and should be removed
console.log(address,bool,uint256,string) (Skillerz.sol#1408-1410) is never used and should be removed
console.log(address,bool,uint256,uint256) (Skillerz.sol#1404-1406) is never used and should be removed
console.log(address,string) (Skillerz.sol#240-242) is never used and should be removed
console.log(address,string,address) (Skillerz.sol#472-474) is never used and should be removed
console.log(address,string,address,address) (Skillerz.sol#1400-1402) is never used and should be removed
```



```
console.log(address,string,address,bool) (Skillerz.sol#1396-1398) is never used and should be removed
console.log(address,string,address,string) (Skillerz.sol#1392-1394) is never used and should be removed
console.log(address,string,address,uint256) (Skillerz.sol#1388-1390) is never used and should be removed
console.log(address,string,bool) (Skillerz.sol#468-470) is never used and should be removed
console.log(address,string,bool,address) (Skillerz.sol#1384-1386) is never used and should be removed
console.log(address,string,bool,bool) (Skillerz.sol#1380-1382) is never used and should be removed
console.log(address,string,bool,string) (Skillerz.sol#1376-1378) is never used and should be removed
console.log(address,string,bool,uint256) (Skillerz.sol#1372-1374) is never used and should be removed
console.log(address,string,string) (Skillerz.sol#464-466) is never used and should be removed
console.log(address,string,string,address) (Skillerz.sol#1368-1370) is never used and should be removed
console.log(address,string,string,bool) (Skillerz.sol#1364-1366) is never used and should be removed
console.log(address,string,string,string) (Skillerz.sol#1360-1362) is never used and should be removed
console.log(address,string,string,uint256) (Skillerz.sol#1356-1358) is never used and should be removed
console.log(address,string,uint256) (Skillerz.sol#460-462) is never used and should be removed
console.log(address,string,uint256,address) (Skillerz.sol#1352-1354) is never used and should be removed
console.log(address,string,uint256,bool) (Skillerz.sol#1348-1350) is never used and should be removed
console.log(address,string,uint256,string) (Skillerz.sol#1344-1346) is never used and should be removed
console.log(address,string,uint256,uint256) (Skillerz.sol#1340-1342) is never used and should be removed
console.log(address,uint256) (Skillerz.sol#236-238) is never used and should be removed
console.log(address,uint256,address) (Skillerz.sol#456-458) is never used and should be removed
console.log(address,uint256,address,address) (Skillerz.sol#1336-1338) is never used and should be removed
console.log(address,uint256,address,bool) (Skillerz.sol#1332-1334) is never used and should be removed
console.log(address,uint256,address,string) (Skillerz.sol#1328-1330) is never used and should be removed
console.log(address,uint256,address,uint256) (Skillerz.sol#1324-1326) is never used and should be removed
console.log(address,uint256,bool) (Skillerz.sol#452-454) is never used and should be removed
console.log(address,uint256,bool,address) (Skillerz.sol#1320-1322) is never used and should be removed
console.log(address,uint256,bool,bool) (Skillerz.sol#1316-1318) is never used and should be removed
console.log(address,uint256,bool,string) (Skillerz.sol#1312-1314) is never used and should be removed
console.log(address,uint256,bool,uint256) (Skillerz.sol#1308-1310) is never used and should be removed
console.log(address,uint256,string) (Skillerz.sol#448-450) is never used and should be removed
console.log(address,uint256,string,address) (Skillerz.sol#1304-1306) is never used and should be removed
console.log(address,uint256,string,bool) (Skillerz.sol#1300-1302) is never used and should be removed
console.log(address,uint256,string,string) (Skillerz.sol#1296-1298) is never used and should be removed
console.log(address,uint256,string,uint256) (Skillerz.sol#1292-1294) is never used and should be removed
console.log(address,uint256,uint256) (Skillerz.sol#444-446) is never used and should be removed
console.log(address,uint256,uint256,address) (Skillerz.sol#1288-1290) is never used and should be removed
console.log(address,uint256,uint256,bool) (Skillerz.sol#1284-1286) is never used and should be removed
```

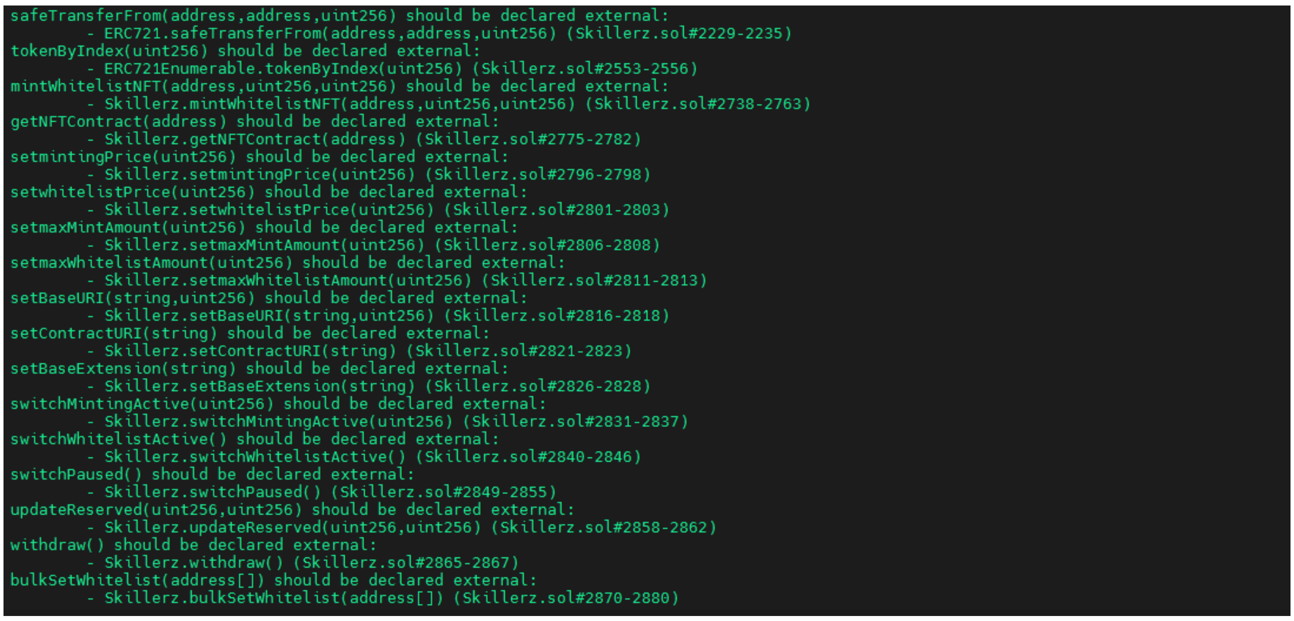


```
console.logUint(uint256) (Skillerz.sol#24-26) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.4 (Skillerz.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (Skillerz.sol#1585-1590):
        - (success) = recipient.call{value: amount}() (Skillerz.sol#1588)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (Skillerz.sol#1653-1664):
        - (success,returndata) = target.call{value: value}(data) (Skillerz.sol#1662)
Low level call in Address.functionStaticCall(address,bytes,string) (Skillerz.sol#1682-1691):
        - (success,returndata) = target.staticcall(data) (Skillerz.sol#1689)
Low level call in Address.functionDelegateCall(address,bytes,string) (Skillerz.sol#1709-1718):
        - (success,returndata) = target.delegatecall(data) (Skillerz.sol#1716)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Contract console (Skillerz.sol#4-1532) is not in CapWords
Parameter ERC721.safeTransferFrom(address,address,uint256,bytes)._data (Skillerz.sol#2244) is not in mixedCase
Parameter Skillerz.mintNFT(address,uint256,uint256)._to (Skillerz.sol#2711) is not in mixedCase
Parameter Skillerz.mintNFT(address,uint256,uint256)._mintAmount (Skillerz.sol#2711) is not in mixedCase
Parameter Skillerz.mintNFT(address,uint256,uint256)._collectionID (Skillerz.sol#2711) is not in mixedCase
Parameter Skillerz.mintWhitelistNFT(address,uint256,uint256)._to (Skillerz.sol#2738) is not in mixedCase
Parameter Skillerz.mintWhitelistNFT(address,uint256,uint256)._mintAmount (Skillerz.sol#2738) is not in mixedCase
Parameter Skillerz.mintWhitelistNFT(address,uint256,uint256)._collectionID (Skillerz.sol#2738) is not in mixedCase
Parameter Skillerz.getNFTContract(address)._owner (Skillerz.sol#2775) is not in mixedCase
Parameter Skillerz.isWhitelisted(address)._address (Skillerz.sol#2785) is not in mixedCase
Parameter Skillerz.getWhitelistClaim(address)._address (Skillerz.sol#2790) is not in mixedCase
Parameter Skillerz.setmintingPrice(uint256)._newPrice (Skillerz.sol#2796) is not in mixedCase
```

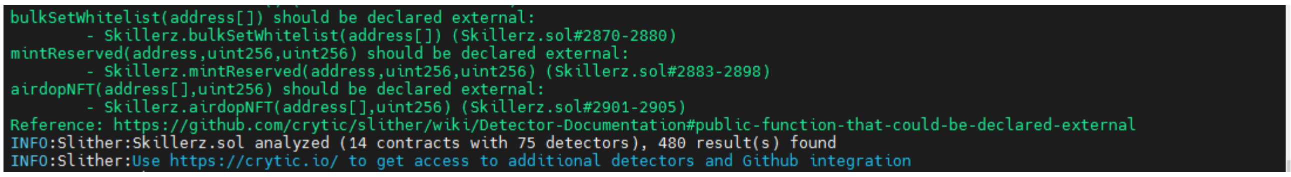


```
Parameter Skillerz.mintReserved(address,uint256,uint256)._to (Skillerz.sol#2883) is not in mixedCase
Parameter Skillerz.mintReserved(address,uint256,uint256)._mintAmount (Skillerz.sol#2883) is not in mixedCase
Parameter Skillerz.mintReserved(address,uint256,uint256)._collectionID (Skillerz.sol#2883) is not in mixedCase
Parameter Skillerz.airdopNFT(address[],uint256)._list (Skillerz.sol#2901) is not in mixedCase
Parameter Skillerz.airdopNFT(address[],uint256)._collectionID (Skillerz.sol#2901) is not in mixedCase
Parameter Skillerz.getCollectionOfNFT(uint256)._id (Skillerz.sol#2909) is not in mixedCase
Variable Skillerz.CollectionsDict (Skillerz.sol#2694) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

INFO:Detectors:
console.slitherConstructorConstantVariables() (Skillerz.sol#4-1532) uses literals with too many digits:
        - CONSOLE_ADDRESS = address(0x000000000000000000636F6e736F6c652e6c6f67) (Skillerz.sol#5)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
Skillerz.maxSupply (Skillerz.sol#2681) should be constant
Skillerz.revealJson (Skillerz.sol#2671) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Skillerz.sol#2050-2052)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Skillerz.sol#2058-2061)
name() should be declared external:
        - ERC721.name() (Skillerz.sol#2144-2146)
symbol() should be declared external:
        - ERC721.symbol() (Skillerz.sol#2151-2153)
tokenURI(uint256) should be declared external:
        - ERC721.tokenURI(uint256) (Skillerz.sol#2158-2163)
        - Skillerz.tokenURI(uint256) (Skillerz.sol#2766-2771)
approve(address,uint256) should be declared external:
        - ERC721.approve(address,uint256) (Skillerz.sol#2177-2187)
setApprovalForAll(address,bool) should be declared external:
        - ERC721.setApprovalForAll(address,bool) (Skillerz.sol#2201-2203)
transferFrom(address,address,uint256) should be declared external:
        - ERC721.transferFrom(address,address,uint256) (Skillerz.sol#2215-2224)
safeTransferFrom(address,address,uint256) should be declared external:
```



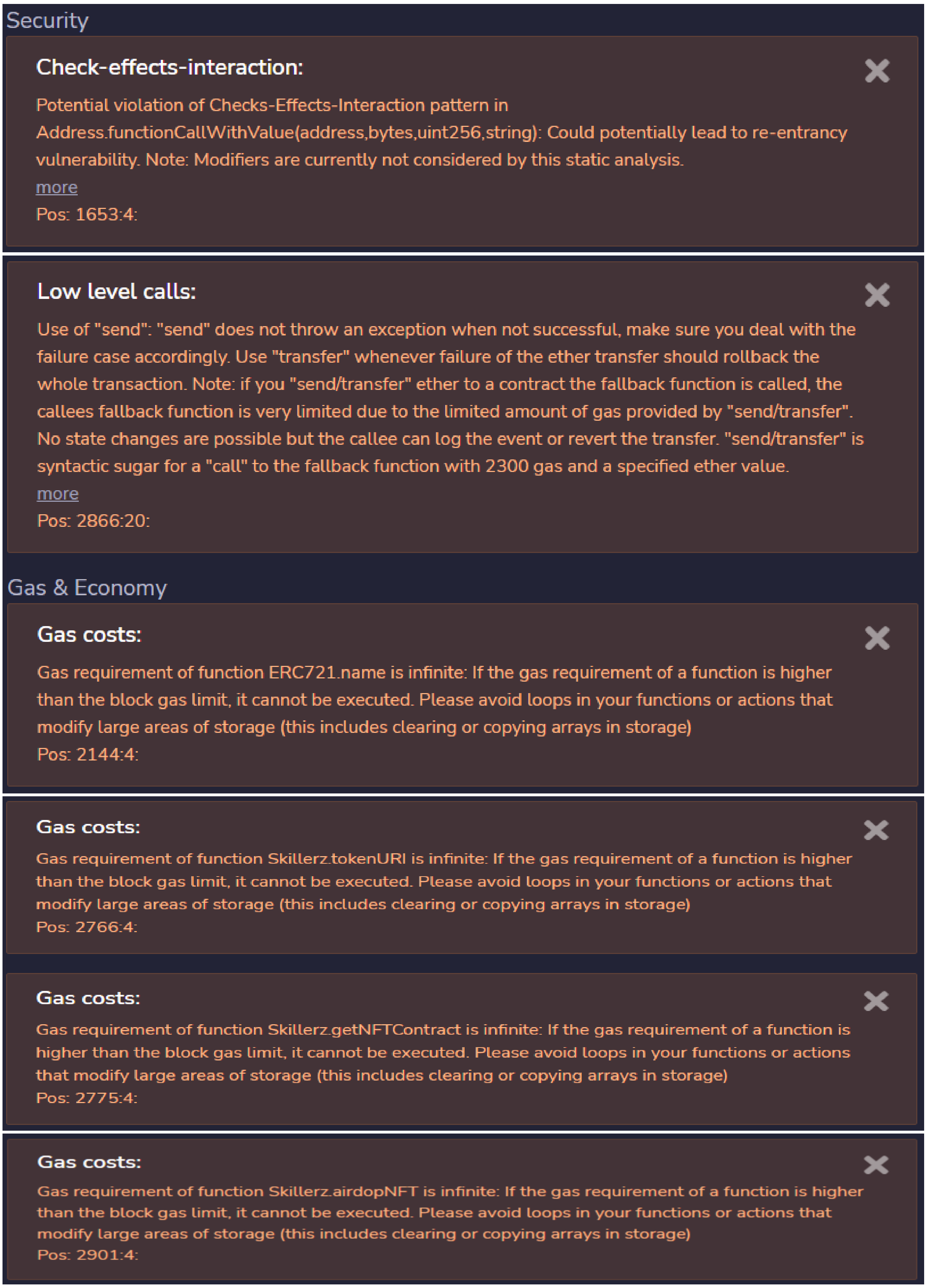
```
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721.safeTransferFrom(address,address,uint256) (Skillerz.sol#2229-2235)
tokenByIndex(uint256) should be declared external:
        - ERC721Enumerable.tokenByIndex(uint256) (Skillerz.sol#2553-2556)
mintWhitelistNFT(address,uint256,uint256) should be declared external:
        - Skillerz.mintWhitelistNFT(address,uint256,uint256) (Skillerz.sol#2738-2763)
getNFTContract(address) should be declared external:
        - Skillerz.getNFTContract(address) (Skillerz.sol#2775-2782)
setmintingPrice(uint256) should be declared external:
        - Skillerz.setmintingPrice(uint256) (Skillerz.sol#2796-2798)
setwhitelistPrice(uint256) should be declared external:
        - Skillerz.setwhitelistPrice(uint256) (Skillerz.sol#2801-2803)
setmaxMintAmount(uint256) should be declared external:
        - Skillerz.setmaxMintAmount(uint256) (Skillerz.sol#2806-2808)
setmaxWhitelistAmount(uint256) should be declared external:
        - Skillerz.setmaxWhitelistAmount(uint256) (Skillerz.sol#2811-2813)
setBaseURI(string,uint256) should be declared external:
        - Skillerz.setBaseURI(string,uint256) (Skillerz.sol#2816-2818)
setContractURI(string) should be declared external:
        - Skillerz.setContractURI(string) (Skillerz.sol#2821-2823)
setBaseExtension(string) should be declared external:
        - Skillerz.setBaseExtension(string) (Skillerz.sol#2826-2828)
switchMintingActive(uint256) should be declared external:
        - Skillerz.switchMintingActive(uint256) (Skillerz.sol#2831-2837)
switchWhitelistActive() should be declared external:
        - Skillerz.switchWhitelistActive() (Skillerz.sol#2840-2846)
switchPaused() should be declared external:
        - Skillerz.switchPaused() (Skillerz.sol#2849-2855)
updateReserved(uint256,uint256) should be declared external:
        - Skillerz.updateReserved(uint256,uint256) (Skillerz.sol#2858-2862)
withdraw() should be declared external:
        - Skillerz.withdraw() (Skillerz.sol#2865-2867)
bulkSetWhitelist(address[]) should be declared external:
        - Skillerz.bulkSetWhitelist(address[]) (Skillerz.sol#2870-2880)
```



```
bulkSetWhitelist(address[]) should be declared external:
        - Skillerz.bulkSetWhitelist(address[]) (Skillerz.sol#2870-2880)
mintReserved(address,uint256,uint256) should be declared external:
        - Skillerz.mintReserved(address,uint256,uint256) (Skillerz.sol#2883-2898)
airdopNFT(address[],uint256) should be declared external:
        - Skillerz.airdopNFT(address[],uint256) (Skillerz.sol#2901-2905)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Skillerz.sol analyzed (14 contracts with 75 detectors), 480 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

Skillerz.sol



Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address.functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1653:4:

Low level calls:

Use of "send": "send" does not throw an exception when not successful, make sure you deal with the failure case accordingly. Use "transfer" whenever failure of the ether transfer should rollback the whole transaction. Note: if you "send/transfer" ether to a contract the fallback function is called, the callees fallback function is very limited due to the limited amount of gas provided by "send/transfer". No state changes are possible but the callee can log the event or revert the transfer. "send/transfer" is syntactic sugar for a "call" to the fallback function with 2300 gas and a specified ether value.
more
Pos: 2866:20:

Gas & Economy

Gas costs:

Gas requirement of function ERC721.name is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 2144:4:

Gas costs:

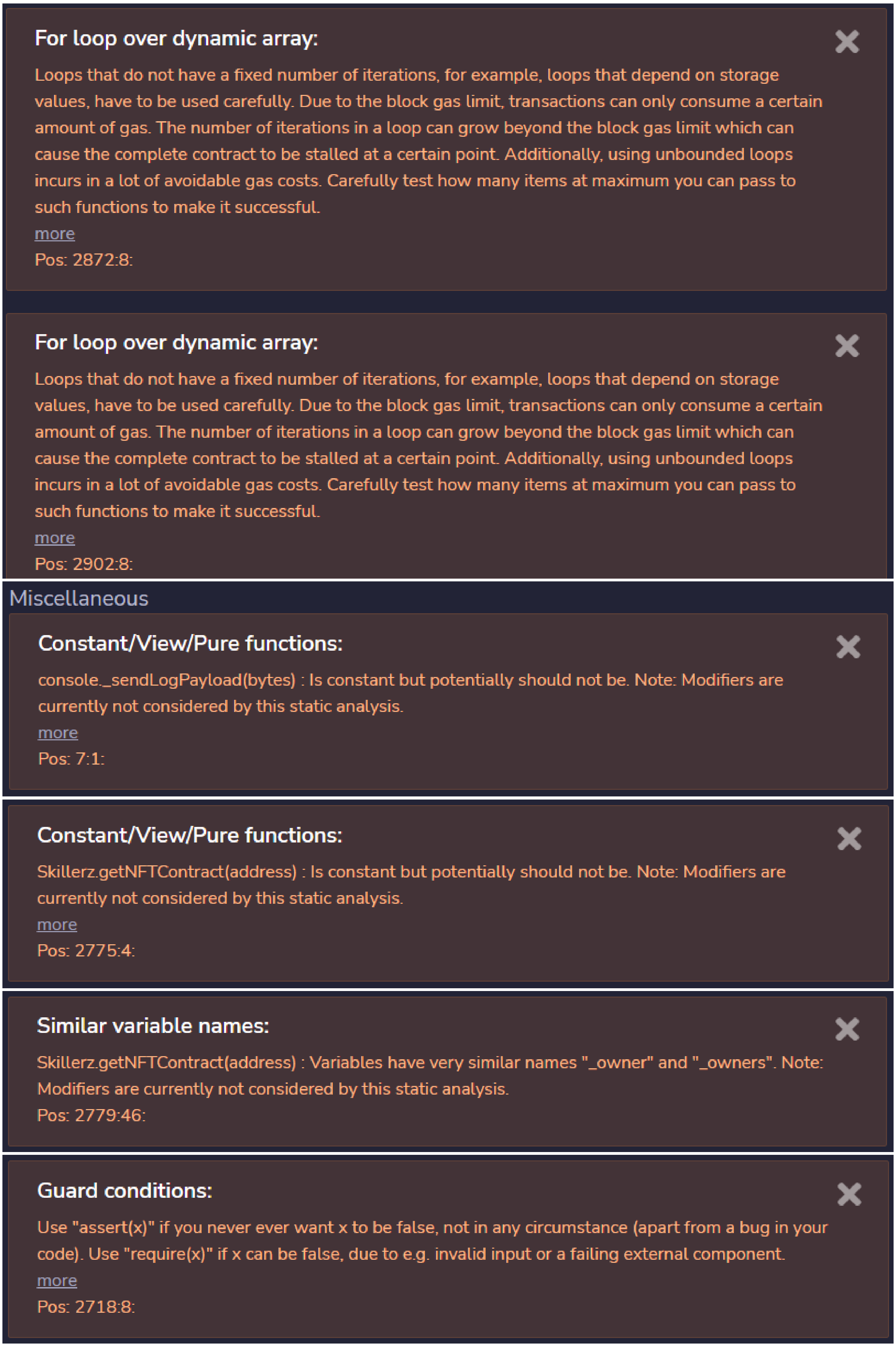
Gas requirement of function Skillerz.tokenURI is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 2766:4:

Gas costs:

Gas requirement of function Skillerz.getNFTContract is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 2775:4:

Gas costs:

Gas requirement of function Skillerz.airdopNFT is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 2901:4:



For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.

more

Pos: 2872:8:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.

more

Pos: 2902:8:

Miscellaneous

Constant/View/Pure functions:

console._sendLogPayload(bytes) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 7:1:

Constant/View/Pure functions:

Skillerz.getNFTContract(address) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 2775:4:

Similar variable names:

Skillerz.getNFTContract(address) : Variables have very similar names "_owner" and "_owners". Note: Modifiers are currently not considered by this static analysis.

Pos: 2779:46:

Guard conditions:

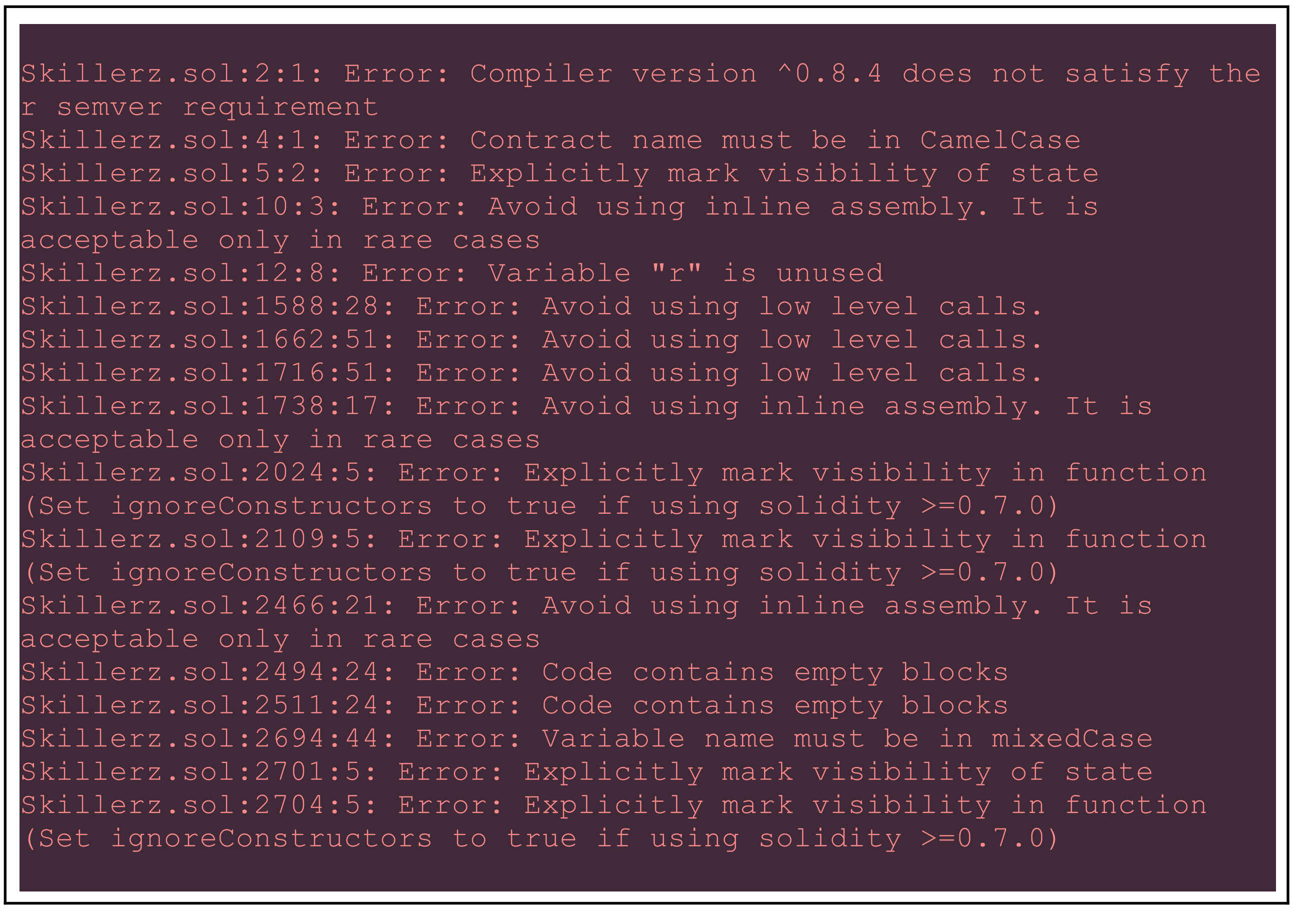
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 2718:8:

Solhint Linter

Skillerz.sol



```
Skillerz.sol:2:1: Error: Compiler version ^0.8.4 does not satisfy the
r semver requirement
Skillerz.sol:4:1: Error: Contract name must be in CamelCase
Skillerz.sol:5:2: Error: Explicitly mark visibility of state
Skillerz.sol:10:3: Error: Avoid using inline assembly. It is
acceptable only in rare cases
Skillerz.sol:12:8: Error: Variable "r" is unused
Skillerz.sol:1588:28: Error: Avoid using low level calls.
Skillerz.sol:1662:51: Error: Avoid using low level calls.
Skillerz.sol:1716:51: Error: Avoid using low level calls.
Skillerz.sol:1738:17: Error: Avoid using inline assembly. It is
acceptable only in rare cases
Skillerz.sol:2024:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
Skillerz.sol:2109:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
Skillerz.sol:2466:21: Error: Avoid using inline assembly. It is
acceptable only in rare cases
Skillerz.sol:2494:24: Error: Code contains empty blocks
Skillerz.sol:2511:24: Error: Code contains empty blocks
Skillerz.sol:2694:44: Error: Variable name must be in mixedCase
Skillerz.sol:2701:5: Error: Explicitly mark visibility of state
Skillerz.sol:2704:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
```

Software analysis result:

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.

EtherAuthority