

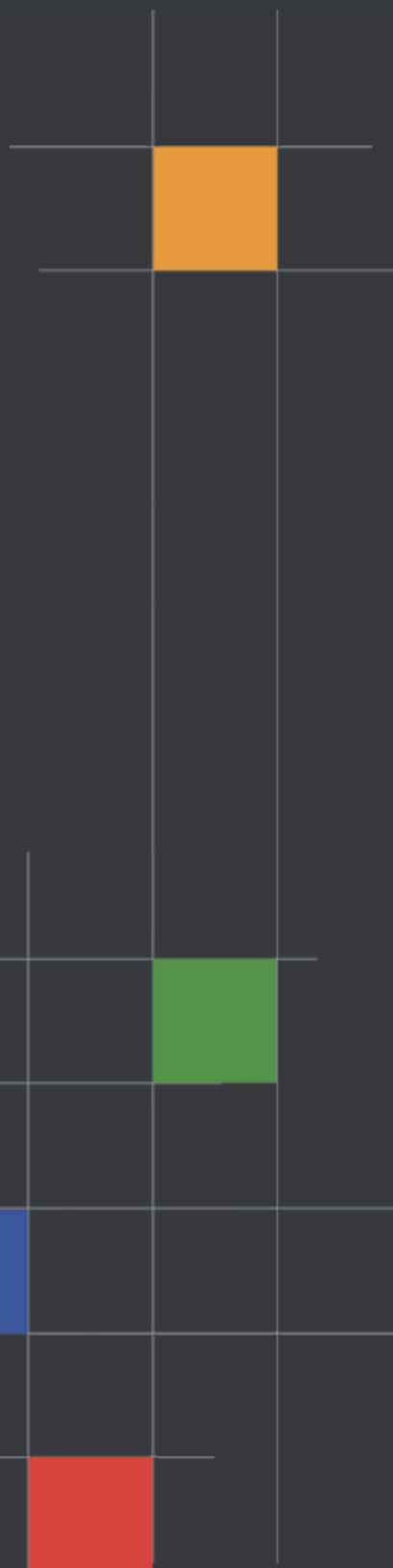


Radio Finance

Security Assessment

March 23, 2021

For :
Radio Finance





Disclaimer

CertiK reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts CertiK to perform a security review.

CertiK Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

CertiK Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

CertiK Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK’s position is that each company and individual are responsible for their own due diligence and continuous security. CertiK’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.
- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
- Representation that a Client of CertiK has indeed completed a round of auditing with the intention to increase the quality of the company/product’s IT infrastructure and or source code.

Overview





Project Summary

Project Name	radio-core
Platform	Heco; Solidity
Codebase	radio-core
Commit	6dc0723ba15c6dba09301ec783b5cf36f817e998a165bb2495e69c61909169f85d778ed1bb34e034

Audit Summary

Delivery Date	March. 23, 2021
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	2
Timeline	Mar. 19th, 2021 - Mar. 21th, 2021

Vulnerability Summary

Total Issues	9
 Total Critical	0
 Total Major	0
 Total Minor	3
 Total Informational	6



Executive Summary

This report has been prepared for **Radio Finance** smart contract to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract 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:

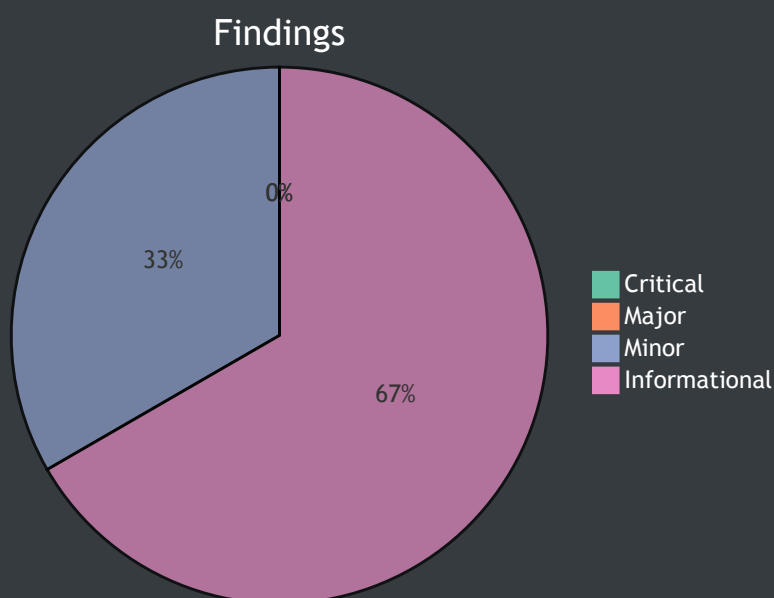
- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.



File in Scope

ID	Contract	SHA-256 Checksum
LP	contracts/LPTokenWrapper.sol	d4238bf7a738f08f09c1a473f36452b01a04a977b99d11e771918764c473ae85
PL	contracts/Pool.sol	2037cba0e5554bc8bde83b65567a2356b1687654da7aabcc08e8b92a869ecf74
RT	contracts/RadioToken.sol	8fe61c95ea8cadd5849b0d56304c5ea18caab70663338725521499c79796eee

Findings



ID	Title	Type	Severity
LP-01	Lack of Input Validation	Volatile Code	Informational
LP-02	Implementation of <code>transferBack()</code>	Optimization	Minor
LP-03	Missing Emit Events	Optimization	Informational
LP-04	Initialization of <code>Pool</code>	Optimization	Informational
LP-05	Check the Balance of Reward Token	Logical Issue	Minor
RT-01	Proper Usage of <code>public</code> And <code>external</code> Type	Optimization	Informational
RT-02	Typo	Optimization	Informational
RT-03	Redundant Code	Optimization	Informational
RT-04	Incorrect Value Read	Logical Issue	Minor



LP-01: Lack of Input Validation

Type	Severity	Location
Volatile Code	● Informational	Pool.sol

Description:

The assigned address of `radio` , `lpt` and `minerOwner` should be verified as non zero value to prevent being mistakenly assigned as `address(0)` in constructor of contract `Pool.sol` . Violation of this may cause errors.

Recommendation:

Check that the address is not zero by adding following checks in the constructor of contract `Pool.sol` , like:

```
require(radio_ != address(0), "radio_ is zero address");
require(lptoken_ != address(0), "lptoken_ is zero address");
require(minerOwner_ != address(0), "minerOwner_ is zero address");
```

Alleviation:

The development team heeded our advice and resolved this issue in commit [60c9128704314b351b3093b902df9d438166493d](#).



LP-02: Implementation of `transferBack()`

Type	Severity	Location
Optimization	● Minor	Pool.sol

Description:

Function `transferBack()` can transfer radio token to anyone by the owner.

Recommendation:

Consider emitting event for the function, or changing it to a recovery function in order to recovery ERC20 tokens which are transferred to this pool by mistake, like:

```
event Recovered(address token, address back, uint256 amount);

function recoverERC20(address _token, uint256 _amount)
    external
    onlyOwner()
{
    require(_token != address(lpt), "Cannot withdraw staking tokens");

    IERC20(_token).safeTransfer(back, _amount);
    emit Recovered(_token, _amount);
}
```

Alleviation:

The development team heeded our advice and resolved this issue in commit [60c9128704314b351b3093b902df9d438166493d](#).



LP-03 Missing Emit Events

Type	Severity	Location
Optimization	● Informational	Pool.sol

Description:

Several key actions are defined without event declarations.

Example:

- `initSet()` , `updateRewardRate()` in contract `Pool.sol` .
- `setTaxFeePercent()` , `setLiquidityFeePercent()` , `setMaxTxPercent()` in contract `RadioToken.sol` .

Recommendation:

Consider emitting events for key actions.

Alleviation:

No alleviation.



LP-04: Initialization of `Pool`

Type	Severity	Location
Optimization	● Informational	Pool.sol

Description:

Function `initSet()` can be called more than once, is that designed as expected?

Recommendation:

Consider adding logical to prevent repeated call on the function, like applying modifier `initializer` of contract `Initializable` of `openzeppelin`.

Alleviation:

No alleviation.



LP-05: Check the Balance of Reward Token

Type	Severity	Location
Optimization	● Minor	Pool.sol

Description:

Function `updateRewardRate()` updates the reward distributed to accounts but missing check whether the balance of reward token is enough or not.

Recommendation:

Consider checking the balance of reward token is enough or not to distribute the reward.

Alleviation:

No alleviation.



RT-01: Proper Usage of `public` And `external` Type

Type	Severity	Location
Optimization	● Informational	RadioToken.sol

Description:

`public` functions that are never called by the contract could be declared `external`. When the inputs are arrays `external` functions are more efficient than `public` functions.

Examples:

- `name()`, `symbol()`, `decimals()`, `totalSupply()`, `transfer(address,uint256)`, `allowance(address,address)`, `approve(address,uint256)`, `transferFrom(address,address,uint256)`, `increaseAllowance(address,uint256)`, `decreaseAllowance(address,uint256)`, `isExcludedFromReward(address)`, `totalFees()`, `deliver(uint256)`, `reflectionFromToken(uint256,bool)`, `excludeFromReward(address)`, `excludeFromFee(address)`, `includeInFee(address)`, `setSwapAndLiquifyEnabled(bool)` and `isExcludedFromFee(address)`.

Recommendation:

Consider using the `external` attribute for functions never called from the contract.

Alleviation:

No alleviation.



RT-02: Typo

Type	Severity	Location
Optimization	● Informational	RadioToken.sol

Description:

The error message in `require(!_isExcluded[account], "Account is already excluded")` of function `includeInReward()` does not describe the error correctly.

Recommendation:

Consider changing the message `"Account is already excluded"` to `"Account is already included"`.

Alleviation:

The development team heeded our advice and resolved this issue in commit [60c9128704314b351b3093b902df9d438166493d](#).



RT-03: Redundant Code

Type	Severity	Location
Logical Issue	● Informational	RadioToken.sol L395-L396

Description:

The condition `!_isExcluded[sender] && !_isExcluded[recipient]` can be included in else.

Recommendation:

The following code can be removed:

```
... else if (!_isExcluded[sender] && !_isExcluded[recipient]) {  
    _transferStandard(sender, recipient, amount);  
} ...
```

Alleviation:

The development team heeded our advice and resolved this issue in commit [60c9128704314b351b3093b902df9d438166493d](#).



RT-04: Incorrect Value Read

Type	Severity	Location
Logical Issue	● Minor	RadioToken.sol L347 L351

Description:

In the function `swapAndLiquify()`, value `address(this).balance` i.e. balance of `ETH` is assigned to `initialBalance`, however, what swaps for in the function `swapTokensForEth()` is `WHT`.

The name of function `swapTokensForEth()` does not describe what it does.

Recommendation:

Consider querying the balance of `WHT` instead of `ETH`, in the function `swapAndLiquify()`.

Consider renaming the function `swapTokensForEth()` to `swapTokensForHt()`.

Alleviation:

The development team responded that the smart contract is universally adapted to all Ethereum-family blockchains and maybe deployed to more blockchains further, although the native token/currency is HT on Heco. The term Eth in function name `swapTokensForEth()` is as above, not designated for native currency, which is Eth of Ethereum. For now, the smart contract is planed to be deployed on Heco.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how `block.timestamp` works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a `struct` assignment operation affecting an in-memory `struct` rather than an instorage one.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of `private` or `delete` .

Coding Style

Coding Style findings usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a `constructor` assignment imposing different `require` statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as `constant` contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Dead Code

Code that otherwise does not affect the functionality of the codebase and can be safely omitted.

Icons explanation



: Issue resolved



: Issue not resolved / Acknowledged. The team will be fixing the issues in the own timeframe.



: Issue partially resolved. Not all instances of an issue was resolved.