CERTIK

Security Assessment

BOSagora - Loyal Token

CertiK Assessed on Jun 9th, 2024

CertiK Assessed on Jun 9th, 2024

BOSagora - Loyal Token

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
ERC-20	Ethereum (ETH)	Formal Verification, Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 06/09/2024	N/A

CODEBASE

https://github.com/bosagora/loyalty-tokens/

View All in Codebase Page

COMMITS

- 3a8430174e88d5a62a1f1c12d61f4a29af32ed9c
- 609bf58cdcd4b7da1437342e51dd5744484b4632

View All in Codebase Page

Highlighted Centralization Risks

Privileged role can mint tokens

Vulnerability Summary

2	1	0	0	1	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined

0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
1	Major	1 Acknowledged	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
1	Medium	1 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
0	Minor		Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
0	Informational		Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | BOSAGORA - LOYAL TOKEN

CODEBASE | BOSAGORA - LOYAL TOKEN

Repository

https://github.com/bosagora/loyalty-tokens/

Commit

- 3a8430174e88d5a62a1f1c12d61f4a29af32ed9c
- 609bf58cdcd4b7da1437342e51dd5744484b4632

AUDIT SCOPE | BOSAGORA - LOYAL TOKEN

12 files audited ● 1 file with Acknowledged findings ● 1 file with Resolved findings ● 10 files without findings

ID	Repo	Commit	File	SHA256 Checksum
LTB	bosagora/loyalty-tokens	3a84301	LoyaltyToken.sol	7e77713a6ac81f85cdc3b57f47416f963d6a673e96bd322b0900b72391947093
BID	bosagora/loyalty-tokens	3a84301	BIP20/BIP20DelegatedTransfer.sol	e7c806b00f8f0ec887cd0cdecc4e96e4721911a49692d858e9bb8bda5f0b19c4
BIB	bosagora/loyalty-tokens	3a84301	BIP20/BIP20.sol	c54e02fb457526fbe46bcd435a88353dd02bf7806270633aeceab6a61a56de3b
IBP	bosagora/loyalty-tokens	3a84301	BIP20/IBIP20.sol	1c5d4aa60f56cb4f5672aab0e038c678ffff40ed650e8c9ed4ed6f68a28d9833
IBI	bosagora/loyalty-tokens	3a84301	BIP20/IBIP20DelegatedTransfer.sol	9d426bf71dc83b8e8339e18c815a9f4f28441d82402ff9c2c255a45562537d92
LYT	bosagora/loyalty-tokens	3a84301	LYT.sol	376af487a5cf4000edec6c5725123c06e711333cc7eb5301c928296bf08ae35c
BII	bosagora/loyalty-tokens	609bf58	BIP20/BIP20.sol	c54e02fb457526fbe46bcd435a88353dd02bf7806270633aeceab6a61a56de3b
BIT	bosagora/loyalty-tokens	609bf58	BIP20/BIP20DelegatedTransfer.sol	62db190271884589dcc5ebe75890ff10fe958cb475943b6751275ea2bcbff6b5
IBB	bosagora/loyalty-tokens	609bf58	BIP20/IBIP20.sol	1c5d4aa60f56cb4f5672aab0e038c678ffff40ed650e8c9ed4ed6f68a28d9833
IBD	bosagora/loyalty-tokens	609bf58	BIP20/IBIP20DelegatedTransfer.sol	a8472b42acc6b57b463206316e97b958e25733966b4126cb78c388bd82acf24e
LYC	bosagora/loyalty-tokens	609bf58	LYT.sol	3aa6077e02e2b7d68dd1a391a8183f5620798e58ea4240b1fa8610e7ae4eb274
LTU	bosagora/loyalty-tokens	609bf58	LoyaltyToken.sol	facbe5ff5a4de8a0355f3fcb79d633823d7dd34f708cbae641afba99267038bb

APPROACH & METHODS | BOSAGORA - LOYAL TOKEN

This report has been prepared for BOSagora to discover issues and vulnerabilities in the source code of the BOSagora - Loyal Token project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis, Formal Verification, 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.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

REVIEW NOTES | BOSAGORA - LOYAL TOKEN

Overview

The **BOSagora - Loyal Token** is a custom implementation of an ERC-20 token with additional delegated transfer functionality.

External Dependencies

The following are external addresses used within the contracts:

- @openzeppelin/contracts

We assume these contracts or addresses are valid and non-vulnerable actors and implement proper logic to collaborate with the current project. It is recommended that the team actively monitor the changes in the aforementioned libraries to avoid unexpected security incidents.

Privileged Functions

In the **BOSagora - Loyal Token** project, the role `owner` is adopted to ensure the dynamic runtime updates of the project, which were specified in the finding *LTB-01*.

The advantage of this privileged role in the codebase is that the client reserves the ability to adjust the protocol according to the runtime required to best serve the community. It is also worth noting the potential drawbacks of these functions, which should be clearly stated through the client's action/plan. Additionally, if the private key of the privileged account is compromised, it could lead to devastating consequences for the project.

To improve the trustworthiness of the project, dynamic runtime updates in the project should be notified to the community. Any plan to invoke the aforementioned functions should also be considered to move to the execution queue of the `Timelock` contract.

FINDINGS | BOSAGORA - LOYAL TOKEN

2	0	1	1	0	0
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for BOSagora - Loyal Token. Through this audit, we have uncovered 2 issues ranging from different severity levels. Utilizing the techniques of Static Analysis, Formal Verification & Manual Review to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
LTB-01	**Initial Token Distribution And Mint Centralization Risk**	**Centralization**	**Major**	**Acknowledged**
BID-01	Potential Signature Replay Attack	Volatile Code	Medium	Resolved

LTB-01 INITIAL TOKEN DISTRIBUTION AND MINT CENTRALIZATION RISK

Category	Severity	Location	Status
Centralization	Major	LoyaltyToken.sol (loyalty-tokens (04/12-3a84301))): 38, 41	Acknowledged

Description

All of the "Loyalty Coin (LYT)" tokens (1e10 * 1e18) are sent to `owner` during the contract deployment. This is a centralization risk because the owner of the initial token supplements can distribute tokens without obtaining the consensus of the community. Any compromise to these addresses may allow a hacker to steal and sell tokens on the market, resulting in severe damage to the project.

The `owner` of `LoyaltyToken` / `LYT` has authority over the `mint` function, any compromises on the owner account would allow the attacker to mint unlimited LYT tokens to any address.

Recommendation

It is recommended that the team be transparent regarding the initial token distribution process. The token distribution plan should be published in a public location that the community can access. The team should make efforts to restrict access to the private keys of the deployer account or EOAs. A multi-signature (⅔, ⅗) wallet can be used to prevent a single point of failure due to a private key compromise. Additionally, the team can lock up a portion of tokens, release them with a vesting schedule for long-term success, and deanonymize the project team with a third-party KYC provider to create greater accountability.

If the team could provide the initial token distribution information such as the link to the token distribution plan, multi-sig wallet, and signer addresses, the information would be verified and updated in the report.

Alleviation

[BOSagora Team, 05/08/2024]: For the LYT token, the initial token distribution is removed from the constructor and the `mint` function can only mint tokens for the owner. The change is reflected in commit 3ecae40b6ea1adee4064381d544bf82c4bf2d393

1 million LYTs will be distributed to BOSagora Mainnet.

[BOSagora Team, 05/21/2024]: The distribution plan is presented on Page 32 of the Whitepaper: https://github.com/bosagora/loyalty-tokens/blob/v0.x.x/packages/contracts/docs/LYT_TokenWhitePaper_EN.pdf

[CertiK, 06/08/2024]: It is suggested to implement the recommended methods to avoid centralized failure. Also, it strongly encourages the project team to periodically revisit the private key security management of all addresses related to centralized roles.

BID-01 | POTENTIAL SIGNATURE REPLAY ATTACK

Category	Severity	Location	Status
Volatile Code	Medium	BIP20/BIP20DelegatedTransfer.sol (loyalty-tokens (04/12-3a84301))): 25~37	Resolved

Description

Repository:

- loyalty-tokens

Commit hash:

- `3a8430174e88d5a62a1f1c12d61f4a29af32ed9c`

Files:

- packages/contracts/contracts/BIP20/BIP20DelegatedTransfer.sol

In the contract `BIP20DelegatedTransfer`, the function `delegatedTransfer` allows the caller to transfer tokens with a valid signature issued by the `from` address`. The data for the signature includes the following elements:

```
31    bytes32 dataHash = keccak256(abi.encode(from, to, amount, block.chainid, nonce
[from]));
```

However, the `dataHash` for the signature does not include the token address, which would increase the risk that the signature can be reused when another token uses the same set of elements for the signature.

Recommendation

Recommend including the token address in the `dataHash` as well to avoid signature reuse. Also, for better security practices, it is recommended to add expiry time as well for the signature.

Alleviation

[BOSagora Team, 05/08/2024]: The team heeded the advice and resolved this issue in the commit af6ade6660d76dcc6479fa31f8ea7405e1fcf515 by adding validation on expiry time and including expiry time and token address in the data hash for the signature.

FORMAL VERIFICATION | BOSAGORA - LOYAL TOKEN

Formal guarantees about the behavior of smart contracts can be obtained by reasoning about properties relating to the entire contract (e.g. contract invariants) or to specific functions of the contract. Once such properties are proven to be valid, they guarantee that the contract behaves as specified by the property. As part of this audit, we applied formal verification to prove that important functions in the smart contracts adhere to their expected behaviors.

Considered Functions And Scope

In the following, we provide a description of the properties that have been used in this audit. They are grouped according to the type of contract they apply to.

Verification of ERC-20 Compliance

We verified properties of the public interface of those token contracts that implement the ERC-20 interface. This covers

- Functions `transfer` and `transferFrom` that are widely used for token transfers,
- functions `approve` and `allowance` that enable the owner of an account to delegate a certain subset of her tokens to another account (i.e. to grant an allowance), and
- the functions `balanceOf` and `totalSupply`, which are verified to correctly reflect the internal state of the contract.

The properties that were considered within the scope of this audit are as follows:

Property Name	Title
erc20-transferfrom-revert-zero-argument	`transferFrom` Fails for Transfers with Zero Address Arguments
erc20-transfer-revert-zero	`transfer` Prevents Transfers to the Zero Address
erc20-transfer-correct-amount	`transfer` Transfers the Correct Amount in Transfers
erc20-transferfrom-fail-exceed-allowance	`transferFrom` Fails if the Requested Amount Exceeds the Available Allowance
erc20-transferfrom-correct-amount	`transferFrom` Transfers the Correct Amount in Transfers
erc20-transferfrom-correct-allowance	`transferFrom` Updated the Allowance Correctly
erc20-transferfrom-fail-recipient-overflow	`transferFrom` Prevents Overflows in the Recipient's Balance
erc20-transfer-recipient-overflow	`transfer` Prevents Overflows in the Recipient's Balance
erc20-balanceof-succeed-always	`balanceOf` Always Succeeds
erc20-balanceof-correct-value	`balanceOf` Returns the Correct Value

Property Name	Title
erc20-allowance-succeed-always	`allowance` Always Succeeds
erc20-approve-false	If `approve` Returns `false`, the Contract's State Is Unchanged
erc20-approve-revert-zero	`approve` Prevents Approvals For the Zero Address
erc20-allowance-correct-value	`allowance` Returns Correct Value
erc20-approve-correct-amount	`approve` Updates the Approval Mapping Correctly
erc20-allowance-change-state	`allowance` Does Not Change the Contract's State
erc20-balanceof-change-state	`balanceOf` Does Not Change the Contract's State
erc20-transferfrom-never-return-false	`transferFrom` Never Returns `false`
erc20-totalsupply-succeed-always	`totalSupply` Always Succeeds
erc20-totalsupply-correct-value	`totalSupply` Returns the Value of the Corresponding State Variable
erc20-transferfrom-false	If `transferFrom` Returns `false`, the Contract's State Is Unchanged
erc20-transfer-false	If `transfer` Returns `false`, the Contract State Is Not Changed
erc20-transferfrom-fail-exceed-balance	`transferFrom` Fails if the Requested Amount Exceeds the Available Balance
erc20-transfer-never-return-false	`transfer` Never Returns `false`
erc20-transfer-exceed-balance	`transfer` Fails if Requested Amount Exceeds Available Balance
erc20-totalsupply-change-state	`totalSupply` Does Not Change the Contract's State
erc20-approve-never-return-false	`approve` Never Returns `false`
erc20-approve-succeed-normal	`approve` Succeeds for Valid Inputs

Verification Results

For the following contracts, formal verification established that each of the properties that were in scope of this audit (see scope) are valid:

Detailed Results For Contract BIP20 (packages/contracts/contracts/BIP20/BIP20.sol) In Commit 609bf58cdcd4b7da1437342e51dd5744484b4632

Verification of ERC-20 Compliance

Detailed Results for Function `balanceOf`

Property Name	Final Result	Remarks
erc20-balanceof-correct-value	True	
erc20-balanceof-change-state	True	
erc20-balanceof-succeed-always	True	

Detailed Results for Function `approve`

Property Name	Final Result	Remarks
erc20-approve-false	True	
erc20-approve-revert-zero	True	
erc20-approve-correct-amount	True	
erc20-approve-never-return-false	True	
erc20-approve-succeed-normal	True	

Detailed Results for Function `transferFrom`

Property Name	Final Result	Remarks
erc20-transferfrom-false	True	
erc20-transferfrom-never-return-false	True	
erc20-transferfrom-revert-zero-argument	True	
erc20-transferfrom-fail-exceed-allowance	True	
erc20-transferfrom-fail-exceed-balance	True	
erc20-transferfrom-correct-amount	True	
erc20-transferfrom-correct-allowance	True	

Detailed Results for Function `totalSupply`

Property Name	Final Result	Remarks
erc20-totalsupply-change-state	True	
erc20-totalsupply-succeed-always	True	
erc20-totalsupply-correct-value	True	

Detailed Results for Function `allowance`

Property Name	Final Result	Remarks
erc20-allowance-change-state	True	
erc20-allowance-correct-value	True	
erc20-allowance-succeed-always	True	

Detailed Results for Function `transfer`

Property Name	Final Result	Remarks
erc20-transfer-never-return-false	True	
erc20-transfer-false	True	
erc20-transfer-revert-zero	True	
erc20-transfer-exceed-balance	True	
erc20-transfer-correct-amount	True	

In the remainder of this section, we list all contracts where formal verification of at least one property was not successful. There are several reasons why this could happen:

- False: The property is violated by the project.
- Inconclusive: The proof engine cannot prove or disprove the property due to timeouts or exceptions.
- Inapplicable: The property does not apply to the project.

Detailed Results For Contract BIP20DelegatedTransfer (packages/contracts/contracts/BIP20/BIP20DelegatedTransfer.sol) In Commit 609bf58cdcd4b7da1437342e51dd5744484b4632

Verification of ERC-20 Compliance

Detailed Results for Function `transferFrom`

Property Name	Final Result	Remarks
erc20-transferfrom-revert-zero-argument	True	
erc20-transferfrom-fail-exceed-allowance	True	
erc20-transferfrom-correct-amount	True	
erc20-transferfrom-correct-allowance	True	
erc20-transferfrom-fail-recipient-overflow	Inconclusive	
erc20-transferfrom-never-return-false	True	
erc20-transferfrom-false	True	
erc20-transferfrom-fail-exceed-balance	True	

Detailed Results for Function `transfer`

Property Name	Final Result	Remarks
erc20-transfer-revert-zero	True	
erc20-transfer-correct-amount	True	
erc20-transfer-recipient-overflow	Inconclusive	
erc20-transfer-false	True	
erc20-transfer-never-return-false	True	
erc20-transfer-exceed-balance	True	

Detailed Results for Function `balanceOf`

Property Name	Final Result	Remarks
erc20-balanceof-succeed-always	True	
erc20-balanceof-correct-value	True	
erc20-balanceof-change-state	True	

Detailed Results for Function `allowance`

Property Name	Final Result	Remarks
erc20-allowance-succeed-always	True	
erc20-allowance-correct-value	True	
erc20-allowance-change-state	True	

Detailed Results for Function `approve`

Property Name	Final Result	Remarks
erc20-approve-false	True	
erc20-approve-revert-zero	True	
erc20-approve-correct-amount	True	
erc20-approve-never-return-false	True	
erc20-approve-succeed-normal	True	

Detailed Results for Function `totalSupply`

Property Name	Final Result	Remarks
erc20-totalsupply-succeed-always	True	
erc20-totalsupply-correct-value	True	
erc20-totalsupply-change-state	True	

Detailed Results For Contract LoyaltyToken (packages/contracts/contracts/LoyaltyToken.sol) In Commit 609bf58cdcd4b7da1437342e51dd5744484b4632

Verification of ERC-20 Compliance

Detailed Results for Function `approve`

Property Name	Final Result	Remarks
erc20-approve-succeed-normal	True	
erc20-approve-correct-amount	True	
erc20-approve-false	True	
erc20-approve-never-return-false	True	
erc20-approve-revert-zero	True	

Detailed Results for Function `totalSupply`

Property Name	Final Result	Remarks
erc20-totalsupply-correct-value	True	
erc20-totalsupply-succeed-always	True	
erc20-totalsupply-change-state	True	

Detailed Results for Function `allowance`

Property Name	Final Result	Remarks
erc20-allowance-change-state	True	
erc20-allowance-succeed-always	True	
erc20-allowance-correct-value	True	

Detailed Results for Function `balanceOf`

Property Name	Final Result	Remarks
erc20-balanceof-succeed-always	True	
erc20-balanceof-correct-value	True	
erc20-balanceof-change-state	True	

Detailed Results for Function `transferFrom`

Property Name	Final Result	Remarks
erc20-transferfrom-never-return-false	True	
erc20-transferfrom-false	True	
erc20-transferfrom-revert-zero-argument	True	
erc20-transferfrom-fail-exceed-allowance	True	
erc20-transferfrom-fail-exceed-balance	True	
erc20-transferfrom-correct-amount	True	
erc20-transferfrom-correct-allowance	True	
erc20-transferfrom-fail-recipient-overflow	Inconclusive	

Detailed Results for Function `transfer`

Property Name	Final Result	Remarks
erc20-transfer-false	True	
erc20-transfer-never-return-false	True	
erc20-transfer-exceed-balance	True	
erc20-transfer-revert-zero	True	
erc20-transfer-correct-amount	True	
erc20-transfer-recipient-overflow	Inconclusive	

Detailed Results For Contract LYT (packages/contracts/contracts/LYT.sol) In Commit 609bf58cdcd4b7da1437342e51dd5744484b4632

Verification of ERC-20 Compliance

Detailed Results for Function `transferFrom`

Property Name	Final Result	Remarks
erc20-transferfrom-correct-allowance	True	
erc20-transferfrom-fail-recipient-overflow	Inconclusive	
erc20-transferfrom-false	True	
erc20-transferfrom-fail-exceed-balance	True	
erc20-transferfrom-fail-exceed-allowance	True	
erc20-transferfrom-correct-amount	True	
erc20-transferfrom-revert-zero-argument	True	
erc20-transferfrom-never-return-false	True	

Detailed Results for Function `transfer`

Property Name	Final Result	Remarks
erc20-transfer-recipient-overflow	Inconclusive	
erc20-transfer-never-return-false	True	
erc20-transfer-exceed-balance	True	
erc20-transfer-false	True	
erc20-transfer-revert-zero	True	
erc20-transfer-correct-amount	True	

Detailed Results for Function `approve`

Property Name	Final Result	Remarks
erc20-approve-succeed-normal	True	
erc20-approve-revert-zero	True	
erc20-approve-never-return-false	True	
erc20-approve-false	True	
erc20-approve-correct-amount	True	

Detailed Results for Function `totalSupply`

Property Name	Final Result	Remarks
erc20-totalsupply-correct-value	True	
erc20-totalsupply-change-state	True	
erc20-totalsupply-succeed-always	True	

Detailed Results for Function `allowance`

Property Name	Final Result	Remarks
erc20-allowance-correct-value	True	
erc20-allowance-change-state	True	
erc20-allowance-succeed-always	True	

Detailed Results for Function `balanceOf`

Property Name	Final Result	Remarks
erc20-balanceof-correct-value	True	
erc20-balanceof-change-state	True	
erc20-balanceof-succeed-always	True	

APPENDIX | BOSAGORA - LOYAL TOKEN

Finding Categories

Categories	Description
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases and may result in vulnerabilities.
Centralization	Centralization findings detail the design choices of designating privileged roles or other centralized controls over the code.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Details on Formal Verification

Some Solidity smart contracts from this project have been formally verified. Each such contract was compiled into a mathematical model that reflects all its possible behaviors with respect to the property. The model takes into account the semantics of the Solidity instructions found in the contract. All verification results that we report are based on that model.

The following assumptions and simplifications apply to our model:

- Certain low-level calls and inline assembly are not supported and may lead to a contract not being formally verified.
- We model the semantics of the Solidity source code and not the semantics of the EVM bytecode in a compiled contract.

Formalism for property specifications

All properties are expressed in a behavioral interface specification language that CertiK has developed for Solidity, which allows us to specify the behavior of each function in terms of the contract state and its parameters and return values, as well as contract properties that are maintained by every observable state transition. Observable state transitions occur when the contract's external interface is invoked and the invocation does not revert, and when the contract's Ether balance is changed by the EVM due to another contract's "self-destruct" invocation. The specification language has the usual Boolean connectives, as well as the operator `\old` (used to denote the state of a variable before a state transition), and several types of specification clause:

Apart from the Boolean connectives and the modal operators "always" (written `[]`) and "eventually" (written `<>`), we use the following predicates to reason about the validity of atomic propositions. They are evaluated on the contract's state

whenever a discrete time step occurs:

- `requires [cond]` - the condition `cond`, which refers to a function's parameters, return values, and contract state variables, must hold when a function is invoked in order for it to exhibit a specified behavior.
- `ensures [cond]` - the condition `cond`, which refers to a function's parameters, return values, and both `\old` and current contract state variables, is guaranteed to hold when a function returns if the corresponding requires condition held when it was invoked.
- `invariant [cond]` - the condition `cond`, which refers only to contract state variables, is guaranteed to hold at every observable contract state.
- `constraint [cond]` - the condition `cond`, which refers to both `\old` and current contract state variables, is guaranteed to hold at every observable contract state except for the initial state after construction (because there is no previous state); constraints are used to restrict how contract state can change over time.

Description of the Analyzed ERC-20 Properties

Properties related to function `transferFrom`

erc20-transferfrom-correct-allowance

All non-reverting invocations of `transferFrom(from, dest, amount)` that return `true` must decrease the allowance for address `msg.sender` over address `from` by the value in `amount`.

Specification:

```
ensures \result ==> allowance(\old(sender), msg.sender) == \old(allowance(sender,
msg.sender)) - \old(amount)
                  || (allowance(\old(sender), msg.sender) == \old(allowance(sender,
msg.sender)) && \old(allowance(sender, msg.sender)) == type(uint256).max);
```

erc20-transferfrom-correct-amount

All invocations of `transferFrom(from, dest, amount)` that succeed and that return `true` subtract the value in `amount` from the balance of address `from` and add the same value to the balance of address `dest`.

Specification:

```
requires recipient != sender;
requires balanceOf(recipient) + amount <= type(uint256).max;
ensures \result ==> balanceOf(\old(recipient)) == \old(balanceOf(recipient) +
amount)
                  && balanceOf(\old(sender)) == \old(balanceOf(sender) - amount);
  also
requires recipient == sender;
ensures \result ==> balanceOf(\old(recipient)) == \old(balanceOf(recipient));
```

erc20-transferfrom-fail-exceed-allowance

Any call of the form `transferFrom(from, dest, amount)` with a value for `amount` that exceeds the allowance of address `msg.sender` must fail.

Specification:

```
requires msg.sender != sender;
requires amount > allowance(sender, msg.sender);
ensures !\result;
```

erc20-transferfrom-fail-exceed-balance

Any call of the form `transferFrom(from, dest, amount)` with a value for `amount` that exceeds the balance of address `from` must fail.

Specification:

```
requires amount > balanceOf(sender);
ensures !\result;
```

erc20-transferfrom-fail-recipient-overflow

Any call of `transferFrom(from, dest, amount)` with a value in `amount` whose transfer would cause an overflow of the balance of address `dest` must fail.

Specification:

```
requires recipient != sender;
requires balanceOf(recipient) + amount > type(uint256).max;
ensures !\result;
```

erc20-transferfrom-false

If `transferFrom` returns `false` to signal a failure, it must undo all incurred state changes before returning to the caller.

Specification:

```
ensures !\result ==> \assigned (\nothing);
```

erc20-transferfrom-never-return-false

The `transferFrom` function must never return `false` .

Specification:

```
ensures \result;
```

erc20-transferfrom-revert-zero-argument

All calls of the form `transferFrom(from, dest, amount)` must fail for transfers from or to the zero address.

Specification:

```
ensures \old(sender) == address(0) ==> !\result;
also
ensures \old(recipient) == address(0) ==> !\result;
```

Properties related to function `transfer`

erc20-transfer-correct-amount

All non-reverting invocations of `transfer(recipient, amount)` that return `true` must subtract the value in `amount` from the balance of `msg.sender` and add the same value to the balance of the `recipient` address.

Specification:

```
requires recipient != msg.sender;
requires balanceOf(recipient) + amount <= type(uint256).max;
ensures \result ==> balanceOf(recipient) == \old(balanceOf(recipient) + amount)
&& balanceOf(msg.sender) == \old(balanceOf(msg.sender) - amount);
  also
requires recipient == msg.sender;
ensures \result ==> balanceOf(msg.sender) == \old(balanceOf(msg.sender));
```

erc20-transfer-exceed-balance

Any transfer of an amount of tokens that exceeds the balance of `msg.sender` must fail.

Specification:

```
requires amount > balanceOf(msg.sender);
ensures !\result;
```

erc20-transfer-false

If the `transfer` function in contract `BIP20DelegatedTransfer` fails by returning `false`, it must undo all state changes it incurred before returning to the caller.

Specification:

```
ensures !\result ==> \assigned (\nothing);
```

erc20-transfer-false

If the `transfer` function in contract `BIP20` fails by returning `false`, it must undo all state changes it incurred before returning to the caller.

Specification:

```
ensures !\result ==> \assigned (\nothing);
```

erc20-transfer-false

If the `transfer` function in contract `LoyaltyToken` fails by returning `false`, it must undo all state changes it incurred before returning to the caller.

Specification:

```
ensures !\result ==> \assigned (\nothing);
```

erc20-transfer-false

If the `transfer` function in contract `LYT` fails by returning `false`, it must undo all state changes it incurred before returning to the caller.

Specification:

```
ensures !\result ==> \assigned (\nothing);
```

erc20-transfer-never-return-false

The transfer function must never return `false` to signal a failure.

Specification:

```
ensures \result;
```

erc20-transfer-recipient-overflow

Any invocation of `transfer(recipient, amount)` must fail if it causes the balance of the `recipient` address to overflow.

Specification:

```
requires recipient != msg.sender;
requires balanceOf(recipient) + amount > type(uint256).max;
ensures !\result;
```

erc20-transfer-revert-zero

Any call of the form `transfer(recipient, amount)` must fail if the recipient address is the zero address.

Specification:

```
ensures \old(recipient) == address(0) ==> !\result;
```

Properties related to function `balanceOf`

erc20-balanceof-change-state

Function `balanceOf` must not change any of the contract's state variables.

Specification:

```
assignable \nothing;
```

erc20-balanceof-correct-value

Invocations of `balanceOf(owner)` must return the value that is held in the contract's balance mapping for address `owner`.

Specification:

```
ensures \result == balanceOf(\old(account));
```

erc20-balanceof-succeed-always

Function `balanceOf` must always succeed if it does not run out of gas.

Specification:

```
reverts_only_when false;
```

Properties related to function `allowance`

erc20-allowance-change-state

Function `allowance` must not change any of the contract's state variables.

Specification:

```
assignable \nothing;
```

erc20-allowance-correct-value

Invocations of `allowance(owner, spender)` must return the allowance that address `spender` has over tokens held by address `owner`.

Specification:

```
ensures \result == allowance(\old(owner), \old(spender));
```

erc20-allowance-succeed-always

Function `allowance` must always succeed, assuming that its execution does not run out of gas.

Specification:

```
reverts_only_when false;
```

Properties related to function `approve`

erc20-approve-correct-amount

All non-reverting calls of the form `approve(spender, amount)` that return `true` must correctly update the allowance mapping according to the address `msg.sender` and the values of `spender` and `amount`.

Specification:

```
requires spender != address(0);
ensures \result ==> allowance(msg.sender, \old(spender)) == \old(amount);
```

erc20-approve-false

If function `approve` returns `false` to signal a failure, it must undo all state changes that it incurred before returning to the caller.

Specification:

```
ensures !\result ==> \assigned (\nothing);
```

erc20-approve-never-return-false

The function `approve` must never returns `false`.

Specification:

```
ensures \result;
```

erc20-approve-revert-zero

All calls of the form `approve(spender, amount)` must fail if the address in `spender` is the zero address.

Specification:

```
ensures \old(spender) == address(0) ==> !\result;
```

erc20-approve-succeed-normal

All calls of the form `approve(spender, amount)` must succeed, if

- the address in `spender` is not the zero address and
- the execution does not run out of gas.

Specification:

```
requires spender != address(0);
ensures \result;
reverts_only_when false;
```

Properties related to function `totalSupply`

erc20-totalsupply-change-state

The `totalSupply` function in contract BIP20DelegatedTransfer must not change any state variables.

Specification:

```
assignable \nothing;
```

erc20-totalsupply-change-state

The `totalSupply` function in contract BIP20 must not change any state variables.

Specification:

```
assignable \nothing;
```

erc20-totalsupply-change-state

The `totalSupply` function in contract LoyaltyToken must not change any state variables.

Specification:

```
assignable \nothing;
```

erc20-totalsupply-change-state

The `totalSupply` function in contract LYT must not change any state variables.

Specification:

```
assignable \nothing;
```

erc20-totalsupply-correct-value

The `totalSupply` function must return the value that is held in the corresponding state variable of contract BIP20DelegatedTransfer.

Specification:

```
ensures \result == totalSupply();
```

erc20-totalsupply-correct-value

The `totalSupply` function must return the value that is held in the corresponding state variable of contract BIP20.

Specification:

```
ensures \result == totalSupply();
```

erc20-totalsupply-correct-value

The `totalSupply` function must return the value that is held in the corresponding state variable of contract LoyaltyToken.

Specification:

```
ensures \result == totalSupply();
```

erc20-totalsupply-correct-value

The `totalSupply` function must return the value that is held in the corresponding state variable of contract LYT.

Specification:

```
ensures \result == totalSupply();
```

erc20-totalsupply-succeed-always

The function `totalSupply` must always succeeds, assuming that its execution does not run out of gas.

Specification:

```
reverts_only_when false;
```

DISCLAIMER | CERTIK

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 Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with 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, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is 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 assessment. This report does 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.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR

UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

CertiK | Securing the Web3 World

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.