CERTIK

Preliminary Comments

IHC Token

Sept 1st, 2021

Table of Contents

Summary

This report has been prepared for IHC Token to discover issues and vulnerabilities in the source code of the IHC 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 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:

- 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.

Overview

Project Summary

Project Name	IHC Token
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/IHC-Token/token-source-code/tree/e9a16634af369b390584bedf9a5766b5ff2d1ba6
Commit	e9a16634af369b390584bedf9a5766b5ff2d1ba6

Audit Summary

Delivery Date	Sept 01, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

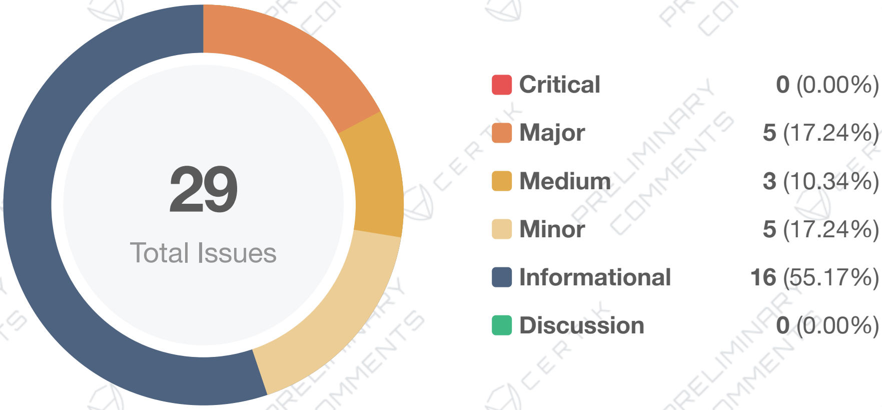
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
● Critical	0	0	0	0	0	0
● Major	5	0	0	3	0	2
● Medium	3	0	0	3	0	0
● Minor	5	1	0	4	0	0
● Informational	16	3	0	11	0	2
● Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
IHC	ihc_loan.sol	492a4ee9b402ba446c15d37ffda59a22f094375ae751dcd8a600e9e4dc7b72c1
IHT	ihc_time_lock.sol	931353304be95dccad208383701119f0aa370a2f2cc23cd28439f4270c2fb170
ICT	ihc_token.sol	cd5eeacc77c3c19ae7a6dc580e5ee051295189142567c8ef0713b532be19781c
HCT	ihc_yield_farm.sol	8b0327bac532dbb854ef6e00e666d5939b74b1e28b9200a8da8f7d222d7ae486

Findings



ID	Title	Category	Severity	Status
HCT-01	Missing Zero Address Validation	Logical Issue, Volatile Code	● Minor	ⓘ Acknowledged
HCT-02	Boolean Equality	Gas Optimization	● Informational	ⓘ Acknowledged
HCT-03	Solidity Version Should Remain Consistent	Inconsistency	● Informational	ⓘ Acknowledged
HCT-04	Code Reuse	Coding Style	● Informational	ⓘ Acknowledged
HCT-05	Public Function That Could Be Declared External	Gas Optimization	● Informational	✓ Resolved
HCT-06	Privileged Ownership	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
HCT-07	Missing Emit Events	Coding Style	● Informational	ⓘ Acknowledged
HCT-08	Transfer Amount Calculation Optimization	Gas Optimization	● Informational	! Pending
HCT-09	Typo in Variable Name	Coding Style	● Informational	! Pending
ICT-01	Missing Zero Address Validation	Logical Issue, Volatile Code	● Minor	ⓘ Acknowledged
ICT-02	Solidity Version Should Remain Consistent	Inconsistency	● Informational	ⓘ Acknowledged
ICT-03	Centralization Risk on `ihc_time_lock`	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
ICT-04	Incorrect BEP-20 Application	Volatile Code	● Medium	ⓘ Acknowledged

ID	Title	Category	Severity	Status
IHC-01	Multiplication on the Result of a Division	Mathematical Operations, Language Specific	● Minor	Acknowledged
IHC-02	Solidity Version Should Remain Consistent	Inconsistency	● Informational	Acknowledged
IHC-03	Unused Variable	Gas Optimization	● Informational	Acknowledged
IHC-04	Code Reuse	Coding Style	● Informational	Acknowledged
IHC-05	Incompatibility With IHC Token	Volatile Code	● Major	Resolved
IHC-06	The design of the loan contract	Logical Issue	● Major	Acknowledged
IHC-07	Incorrect BEP-20 Application	Volatile Code	● Medium	Acknowledged
IHC-08	Transfer Amount Calculation Optimization	Gas Optimization	● Informational	Pending
IHC-09	State Variable Naming Inconsistency	Coding Style	● Minor	Pending
IHT-01	Multiplication on the Result of a Division	Mathematical Operations, Language Specific	● Minor	Acknowledged
IHT-02	Solidity Version Should Remain Consistent	Inconsistency	● Informational	Acknowledged
IHT-03	Unused Variable	Gas Optimization	● Informational	Acknowledged
IHT-04	Code Reuse	Coding Style	● Informational	Acknowledged
IHT-05	Public Function That Could Be Declared External	Gas Optimization	● Informational	Resolved
IHT-06	Incompatibility With IHC Token	Volatile Code	● Major	Resolved
IHT-07	Incorrect BEP-20 Application	Volatile Code	● Medium	Acknowledged

HCT-01 | Missing Zero Address Validation

Category	Severity	Location	Status
Logical Issue, Volatile Code	● Minor	ihc_token.sol (08/31/2021): 731, 723, 686	ⓘ Acknowledged

Description

Addresses should be checked before assignment to make sure they are not zero addresses.

Recommendation

We recommend considering adding a zero check.

HCT-02 | Boolean Equality

Category	Severity	Location	Status
Gas Optimization	● Informational	ihc_token.sol (08/31/2021): 800, 706, 695, 582, 580, 533, 531	ⓘ Acknowledged

Description

Boolean constants can be used directly and do not need to be compared to true or false.

Recommendation

We recommend removing the equality to the boolean constant.

HCT-03 | Solidity Version Should Remain Consistent

Category	Severity	Location	Status
Inconsistency	● Informational	ihc_token.sol (08/31/2021): 1	ⓘ Acknowledged

Description

The ihc_stake.sol, ihc_loan.sol, ihc_time_lock.sol use Solidity version `^0.5.16`, while ihc_token.sol uses Solidity version `0.5.16`. The Solidity version should remain consistent.

Recommendation

We recommend locking contract version on production environment for stability.

HCT-04 | Code Reuse

Category	Severity	Location	Status
Coding Style	● Informational	ihc_token.sol (08/31/2021): 132	ⓘ Acknowledged

Description

The library `SafeMath` has been reused in `ihc_loan.sol`, `ihc_stake.sol` and `ihc_token.sol`. We recommend reusing the library `SafeMath` to keep the concise.

Recommendation

We recommend reusing the library `SafeMath` of `ihc_token.sol` in `ihc_stake.sol` and `ihc_loan.sol`.

HCT-05 | Public Function That Could Be Declared External

Category	Severity	Location	Status
Gas Optimization	● Informational	ihc_token.sol (08/31/2021): 754, 746, 738, 730, 722, 705, 694, 685, 677 , 669, 661, 653, 645, 635, 627, 608, 578, 560, 549, 526, 514, 507, 500, 493, 486, 479, 472, 465, 458, 451, 444, 437, 430, 423, 416, 409, 402, 3 95, 388, 325, 316	✓ Resolved

Description

Following public functions that are never called by the contract internally should be declared with `external` visibility to save gas.

- `IHC_STAKE.getIhcTokenAddress()`
- `IHC_STAKE.getThisContractAddress()`
- `IHC_STAKE.getBalanceOfPool()`
- `IHC_STAKE.getStakeAmount()`
- `IHC_STAKE.getStakeApy()`
- `IHC_STAKE.getYieldAmount()`
- `IHC_STAKE.getWithdrawDeadlineByTimestamp()`
- `IHC_STAKE.stake(uint256,uint256)`
- `IHC_STAKE.withdraw()`
- `Ownable.renounceOwnership()`
- `Ownable.transferOwnership(address)`
- `IHC.getOwner()`
- `IHC.decimals()`
- `IHC.symbol()`
- `IHC.name()`
- `IHC.totalSupply()`
- `IHC.balanceOf(address)`
- `IHC.getApy()`
- `IHC.getLoanFeePercent()`
- `IHC.getLoanSizePercent()`
- `IHC.getTransactionPoolAddress()`
- `IHC.getYieldFarmPoolAddress()`
- `IHC.getLoanPoolAddress()`
- `IHC.getEndOfTime()`

- `IHC.getTransactionFeePercent()`
- `IHC.getBurnAmount()`
- `IHC.getBurnFlag()`
- `IHC.isExcludedTransactionFee(address)`
- `IHC.getYieldFarmMinAmount()`
- `IHC.getLoanMinAmount()`
- `IHC.transfer(address,uint256)`
- `IHC.allowance(address,address)`
- `IHC.approve(address,uint256)`
- `IHC.transferFrom(address,address,uint256)`
- `IHC.increaseAllowance(address,uint256)`
- `IHC.decreaseAllowance(address,uint256)`
- `IHC.setEndTime(uint256)`
- `IHC.setTransactionFeePercent(uint256)`
- `IHC.setApy(uint256)`
- `IHC.setLoanFeePercent(uint256)`
- `IHC.setLoanSizePercent(uint256)`
- `IHC.setBurnAmount(uint256)`
- `IHC.setTransactionPoolAddress(address)`
- `IHC.setExcludedAddressOfTransactionFee(address)`
- `IHC.popExcludedAddressOfTransactionFee(address)`
- `IHC.setYieldFarmPoolAddress(address)`
- `IHC.setLoanPoolAddress(address)`
- `IHC.setYieldFarmMinAmount(uint256)`
- `IHC.setLoanMinAmount(uint256)`
- `IHC.burn()`

Recommendation

We advise using the `external` attribute for the visibility of the listed functions as they are never called from the contract internally.

Alleviation

Fixed in commit hash `266b87324424c2232f89e5a4a628bc3f4dfbed02`. Also, `ihc_stake.sol` is renamed to `ihc_yield_farm.sol` in commit hash `b4e738995d8c2e57fc07c61575777c84515a4ecc`.

HCT-06 | Privileged Ownership

Category	Severity	Location	Status
Centralization / Privilege	● Major	ihc_token.sol (08/31/2021): 754, 746, 738, 730, 722, 705, 694, 685 , 677, 669, 661, 653, 645, 635	ⓘ Acknowledged

Description

The owner of contract `IHC` has the permission to:

1. set end time,
2. set transaction fee percent,
3. set the apy,
4. set loan fee percent,
5. set loan size percent,
6. set burn amount,
7. set the transactionPoolAddress,
8. set the address to exclude from transaction fee,
9. delete an address from transaction fee exclude list,
10. set yield farm pool address,
11. set loan pool address,
12. set yield farm min amount,
13. set the loanMinAmount,
14. set burn

without obtaining the consensus of the community.

Recommendation

This is the intended functionality of the protocol, however, users should be aware of this functionality.

We advise the client to carefully manage the owner account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract-based accounts with enhanced security practices, e.g. Multisignature wallets.

Here are some feasible solutions that would also mitigate the potential risk:

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;

- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

HCT-07 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	● Informational	ihc_token.sol (08/31/2021): 694, 685, 677, 669, 661, 653, 645, 63 5	ⓘ Acknowledged

Description

The function that affects the status of sensitive variables should be able to emit events as notifications to customers.

- `setEndTime()`
- `setTransactionFeePercent()`
- `setApy()`
- `setLoanFeePercent()`
- `setLoanSizePercent()`
- `setBurnAmount()`
- `setTransactionPoolAddress()`
- `setExcludedAddressOfTransactionFee()`

Recommendation

We advise adding events for sensitive actions, and emit them in the function.

HCT-08 | Transfer Amount Calculation Optimization

Category	Severity	Location	Status
Gas Optimization	● Informational	ihc_yield_farm.sol: 225	(!) Pending

Description

Currently the added function `calculateTransferAmount()` is calculating the fee first and then subtracting the fee from the original amount.

```
function calculateTransferAmount(uint256 originalAmount) internal returns(uint256) {
    uint256 feeAmount = (originalAmount.mul(transactionFeePercent)).div(100);
    return originalAmount.sub(feeAmount);
```

The Math calculation can be optimized by calculating the remaining percentage and returning the transfer amount directly. Also note that this function can be declared as a `view` function as this function is not aiming to modify any contract states.

HCT-09 | Typo in Variable Name

Category	Severity	Location	Status
Coding Style	● Informational	ihc_yield_farm.sol: 218~220, 193~195	ⓘ Pending

Description

In functions `getYieldAmount()` and `withdraw()` of contract `IHC_YIELD_FARM` (former name is `IHC_STAKE`). The local variable `yeildAmount` should be `yieldAmount` from its context and functionality.

ICT-01 | Missing Zero Address Validation

Category	Severity	Location	Status
Logical Issue, Volatile Code	● Minor	ihc_time_lock.sol (08/31/2021): 9	ⓘ Acknowledged

Description

Addresses should be checked before assignment to make sure they are not zero addresses.

Recommendation

We recommend considering adding a zero check.

ICT-02 | Solidity Version Should Remain Consistent

Category	Severity	Location	Status
Inconsistency	● Informational	ihc_time_lock.sol (08/31/2021): 1	ⓘ Acknowledged

Description

The ihc_stake.sol, ihc_loan.sol, ihc_time_lock.sol use Solidity version `^0.5.16`, while ihc_token.sol uses Solidity version `0.5.16`. The Solidity version should remain consistent.

Recommendation

We recommend locking contract version on production environment for stability.

ICT-03 | Centralization Risk on `ihc_time_lock`

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	ihc_time_lock.sol (08/31/2021): 19~20	ⓘ Acknowledged

Description

When the time lock ends, the owner of the `ihc_time_lock` contract can extract all assets to the `owner` address without obtaining the consensus of the community.

ICT-04 | Incorrect BEP-20 Application

Category	Severity	Location	Status
Volatile Code	● Medium	ihc_time_lock.sol (08/31/2021): 25	ⓘ Acknowledged

Description

According to BEP-20 and EIP-20, functions `transfer()`, `transferFrom()`, and `approve()` should always have a `bool` return value, for the ERC20 caller to handle, as the callers must not assume that `false` is never returned.

IHC-01 | Multiplication on the Result of a Division

Category	Severity	Location	Status
Mathematical Operations, Language Specific	● Minor	ihc_loan.sol (08/31/2021): 235	ⓘ Acknowledged

Description

Linked function performs a multiplication on the result of a division, which can truncate.

Recommendation

We would recommend to re-arrange arithmetic to perform multiplication before division.

IHC-02 | Solidity Version Should Remain Consistent

Category	Severity	Location	Status
Inconsistency	● Informational	ihc_loan.sol (08/31/2021): 2	ⓘ Acknowledged

Description

The ihc_stake.sol, ihc_loan.sol, ihc_time_lock.sol use Solidity version `^0.5.16`, while ihc_token.sol uses Solidity version `0.5.16`. The Solidity version should remain consistent.

Recommendation

We recommend locking contract version on production environment for stability.

IHC-03 | Unused Variable

Category	Severity	Location	Status
Gas Optimization	● Informational	ihc_loan.sol (08/31/2021): 158	ⓘ Acknowledged

Description

The unused variables `loanMinAmount` and `stakeMinAmount` are declared. Remove or comment out the variable name.

Recommendation

We recommend removing the unused variables.

IHC-04 | Code Reuse

Category	Severity	Location	Status
Coding Style	● Informational	ihc_loan.sol (08/31/2021): 5	ⓘ Acknowledged

Description

The library `SafeMath` has been reused in `ihc_loan.sol`, `ihc_stake.sol` and `ihc_token.sol`. We recommend reusing the library `SafeMath` to keep the concise.

Recommendation

We recommend reusing the library `SafeMath` of `ihc_token.sol` in `ihc_stake.sol` and `ihc_loan.sol`.

IHC-05 | Incompatibility With IHC Token

Category	Severity	Location	Status
Volatile Code	● Major	ihc_loan.sol (08/31/2021)	✓ Resolved

Description

When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged transaction fee.

If a user stakes 100 IHC tokens inside the `ihc_stake.sol` contract, only `(100-transactionfee)%*100` tokens arrived in the `stakePoolAddress`. However, the user can still withdraw 100 tokens from the `stakePoolAddress`, which causes the contract to lose `transactionfee%*100` tokens in such a transaction.

Also, a similar scenario would happen in the `ihc_loan.sol` contract, that each `transfer()` and `transferFrom()` function call would lead to a loss of transaction fees. With that being said, when a `borrower` calls the `repay()` function, although the `loanFee` is added to the total amount to be repaid, because of the static interest and the transaction fee, the final amount received by the `lender` is possible to be less than the original amount.

Recommendation

We recommend using the amount the contract received instead of the amount user transferred as the `stakeamount`. Also, the percentages of the loan fee and the transaction fee need to be carefully thought and transparent to the community.

Alleviation

- `ihc_stake.sol` (renamed to `ihc_yield_farm.sol`) has the transaction fee issue fixed in commit hash `f0a9ee26d2cadde4da2b0d345a266e6841cb05b4` and `b4e738995d8c2e57fc07c61575777c84515a4ecc`.
- `ihc_loan.sol` has the transaction fee issue fixed in commit hash `832f8877c211518196289de4e7d38716d509ed0a`.

IHC-06 | The design of the loan contract

Category	Severity	Location	Status
Logical Issue	● Major	ihc_loan.sol (08/31/2021)	ⓘ Acknowledged

Description

The loan contract allows users to user IHC token as collateral token to borrow IHC token. If the `_collateralAmount` is larger than `loanAmount/(100-transactionFee)`, there is no point for a user to borrow token. If the `_collateralAmount` is smaller than `loanAmount/(100-transactionFee)`, the borrower could just not repaying the token, causing the lenders to lose their tokens, as the lenders can only liquidate `_collateralAmount*(100-transactionFee)` IHC token.

Recommendation

We recommend the team to provide further explanation regarding the contract.

IHC-07 | Incorrect BEP-20 Application

Category	Severity	Location	Status
Volatile Code	● Medium	ihc_loan.sol (08/31/2021): 227, 230, 236, 243	ⓘ Acknowledged

Description

According to BEP-20 and EIP-20, functions `transfer()`, `transferFrom()`, and `approve()` should always have a `bool` return value, for the ERC20 caller to handle, as the callers must not assume that `false` is never returned.

IHC-08 | Transfer Amount Calculation Optimization

Category	Severity	Location	Status
Gas Optimization	● Informational	ihc_loan.sol: 285	Pending

Description

Currently the added function `calculateTransferAmount()` is calculating the fee first and then subtracting the fee from the original amount.

```
function calculateTransferAmount(uint256 originalAmount) internal returns(uint256) {
    uint256 feeAmount = (originalAmount.mul(transactionFeePercent)).div(100);
    return originalAmount.sub(feeAmount);
```

The Math calculation can be optimized by calculating the remaining percentage and returning the transfer amount directly. Also note that this function can be declared as a `view` function as this function is not aiming to modify any contract states.

IHC-09 | State Variable Naming Inconsistency

Category	Severity	Location	Status
Coding Style	● Minor	ihc_loan.sol: 260~263	ⓘ Pending

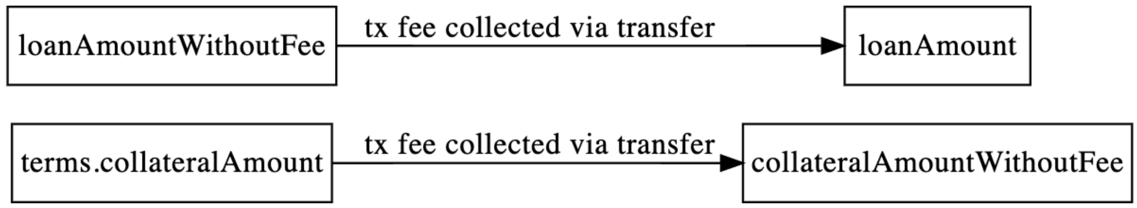
Description

In contract `IHC_TEST_LOAN`, there are two newly added state variable `loanAmountWithoutFee` and `collateralAmountWithoutFee`, as a mitigation of IHC-05(Incompatibility With IHC Token). However, the two variables with suffix `WithoutFee` do not have the same use cases. As the code block shows:

```
loanAmountWithoutFee = terms.collateralAmount * IHC(ihcTokenAddress).getLoanSizePercent()
/ 100;
loanAmount = calculateTransferAmount(loanAmountWithoutFee);

collateralAmountWithoutFee = calculateTransferAmount(terms.collateralAmount);
```

`loanAmountWithoutFee` represents the original amount to be transferred without subtracting the transaction fee; while `collateralAmountWithoutFee` represents the amount received, where the transaction fee is already collected.


loanAmountWithoutFee
tx fee collected via transfer
loanAmount
terms.collateralAmount
tx fee collected via transfer
collateralAmountWithoutFee

Recommendation

Recommend keeping consistent on the naming convention to avoid future ambiguity.

IHT-01 | Multiplication on the Result of a Division

Category	Severity	Location	Status
Mathematical Operations, Language Specific	● Minor	ihc_stake.sol (08/31/2021): 213, 191	ⓘ Acknowledged

Description

Linked function performs a multiplication on the result of a division, which can truncate.

Recommendation

We would recommend to re-arrange arithmetic to perform multiplication before division.

IHT-02 | Solidity Version Should Remain Consistent

Category	Severity	Location	Status
Inconsistency	● Informational	ihc_stake.sol (08/31/2021): 1	ⓘ Acknowledged

Description

The ihc_stake.sol, ihc_loan.sol, ihc_time_lock.sol use Solidity version `^0.5.16`, while ihc_token.sol uses Solidity version `0.5.16`. The Solidity version should remain consistent.

Recommendation

We recommend locking contract version on production environment for stability.

IHT-03 | Unused Variable

Category	Severity	Location	Status
Gas Optimization	● Informational	ihc_stake.sol (08/31/2021): 152	ⓘ Acknowledged

Description

The unused variables `loanMinAmount` and `stakeMinAmount` are declared. Remove or comment out the variable name.

Recommendation

We recommend removing the unused variables.

IHT-04 | Code Reuse

Category	Severity	Location	Status
Coding Style	● Informational	ihc_stake.sol (08/31/2021): 4	ⓘ Acknowledged

Description

The library `SafeMath` has been reused in `ihc_loan.sol`, `ihc_stake.sol` and `ihc_token.sol`. We recommend reusing the library `SafeMath` to keep the concise.

Recommendation

We recommend reusing the library `SafeMath` of `ihc_token.sol` in `ihc_stake.sol` and `ihc_loan.sol`.

IHT-05 | Public Function That Could Be Declared External

Category	Severity	Location	Status
Gas Optimization	● Informational	ihc_stake.sol (08/31/2021): 209, 199, 195, 190, 186, 181, 176, 171, 166	✓ Resolved

Description

Following public functions that are never called by the contract internally should be declared with `external` visibility to save gas.

- `IHC_STAKE.getIhcTokenAddress()`
- `IHC_STAKE.getThisContractAddress()`
- `IHC_STAKE.getBalanceOfPool()`
- `IHC_STAKE.getStakeAmount()`
- `IHC_STAKE.getStakeApy()`
- `IHC_STAKE.getYieldAmount()`
- `IHC_STAKE.getWithdrawDeadlineByTimestamp()`
- `IHC_STAKE.stake(uint256,uint256)`
- `IHC_STAKE.withdraw()`
- `Ownable.renounceOwnership()`
- `Ownable.transferOwnership(address)`
- `IHC.getOwner()`
- `IHC.decimals()`
- `IHC.symbol()`
- `IHC.name()`
- `IHC.totalSupply()`
- `IHC.balanceOf(address)`
- `IHC.getApy()`
- `IHC.getLoanFeePercent()`
- `IHC.getLoanSizePercent()`
- `IHC.getTransactionPoolAddress()`
- `IHC.getYieldFarmPoolAddress()`
- `IHC.getLoanPoolAddress()`
- `IHC.getEndOfTime()`
- `IHC.getTransactionFeePercent()`
- `IHC.getBurnAmount()`

- `IHC.getBurnFlag()`
- `IHC.isExcludedTransactionFee(address)`
- `IHC.getYieldFarmMinAmount()`
- `IHC.getLoanMinAmount()`
- `IHC.transfer(address,uint256)`
- `IHC.allowance(address,address)`
- `IHC.approve(address,uint256)`
- `IHC.transferFrom(address,address,uint256)`
- `IHC.increaseAllowance(address,uint256)`
- `IHC.decreaseAllowance(address,uint256)`
- `IHC.setEndTime(uint256)`
- `IHC.setTransactionFeePercent(uint256)`
- `IHC.setApy(uint256)`
- `IHC.setLoanFeePercent(uint256)`
- `IHC.setLoanSizePercent(uint256)`
- `IHC.setBurnAmount(uint256)`
- `IHC.setTransactionPoolAddress(address)`
- `IHC.setExcludedAddressOfTransactionFee(address)`
- `IHC.popExcludedAddressOfTransactionFee(address)`
- `IHC.setYieldFarmPoolAddress(address)`
- `IHC.setLoanPoolAddress(address)`
- `IHC.setYieldFarmMinAmount(uint256)`
- `IHC.setLoanMinAmount(uint256)`
- `IHC.burn()`

Recommendation

We advise using the `external` attribute for the visibility of the listed functions as they are never called from the contract internally.

Alleviation

Fixed in commit hash `266b87324424c2232f89e5a4a628bc3f4dfbed02`. Also, `ihc_stake.sol` is renamed to `ihc_yield_farm.sol` in commit hash `b4e738995d8c2e57fc07c61575777c84515a4ecc`.

IHT-06 | Incompatibility With IHC Token

Category	Severity	Location	Status
Volatile Code	● Major	ihc_stake.sol (08/31/2021)	✓ Resolved

Description

When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged transaction fee.

If a user stakes 100 IHC tokens inside the `ihc_stake.sol` contract, only `(100-transactionfee)%*100` tokens arrived in the `stakePoolAddress`. However, the user can still withdraw 100 tokens from the `stakePoolAddress`, which causes the contract to lose `transactionfee%*100` tokens in such a transaction.

Also, a similar scenario would happen in the `ihc_loan.sol` contract, that each `transfer()` and `transferFrom()` function call would lead to a loss of transaction fees. With that being said, when a `borrower` calls the `repay()` function, although the `loanFee` is added to the total amount to be repaid, because of the static interest and the transaction fee, the final amount received by the `lender` is possible to be less than the original amount.

Recommendation

We recommend using the amount the contract received instead of the amount user transferred as the `stakeamount`. Also, the percentages of the loan fee and the transaction fee need to be carefully thought and transparent to the community.

Alleviation

- `ihc_stake.sol` (renamed to `ihc_yield_farm.sol`) has the transaction fee issue fixed in commit hash `f0a9ee26d2cadde4da2b0d345a266e6841cb05b4` and `b4e738995d8c2e57fc07c61575777c84515a4ecc`.
- `ihc_loan.sol` has the transaction fee issue fixed in commit hash `832f8877c211518196289de4e7d38716d509ed0a`.

IHT-07 | Incorrect BEP-20 Application

Category	Severity	Location	Status
Volatile Code	● Medium	ihc_stake.sol (08/31/2021): 206, 214	ⓘ Acknowledged

Description

According to BEP-20 and EIP-20, functions `transfer()`, `transferFrom()`, and `approve()` should always have a `bool` return value, for the ERC20 caller to handle, as the callers must not assume that `false` is never returned.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings 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 findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

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

Volatile Code

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

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 but rather 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.

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.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the 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.

About

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.