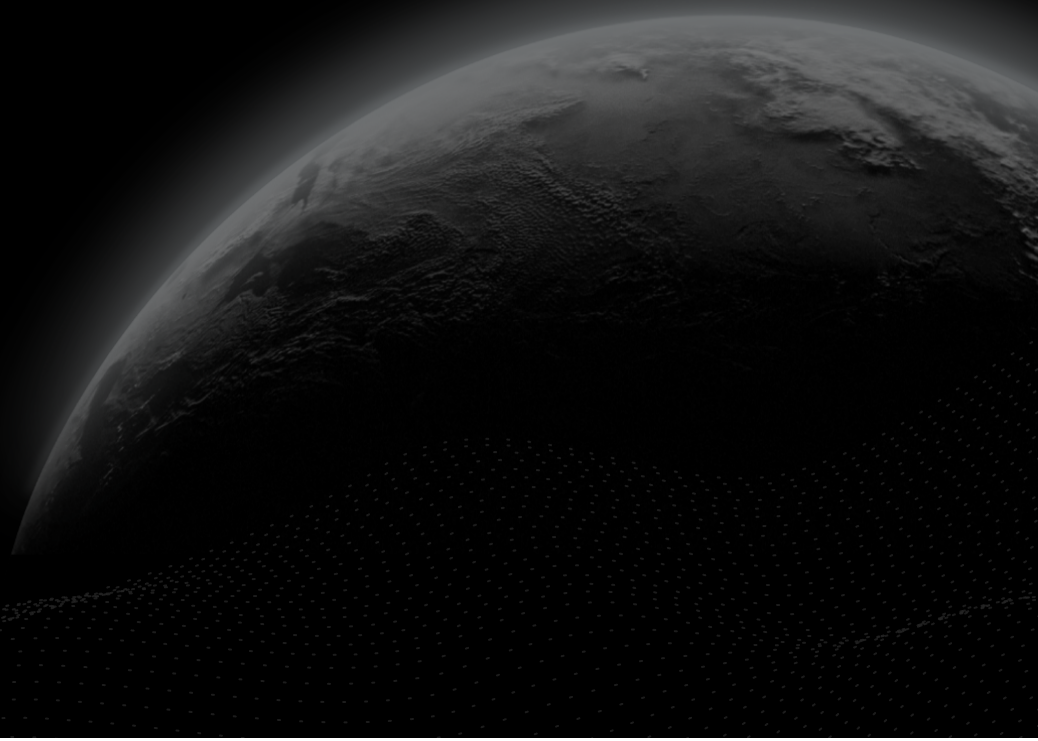
CERTIK

Security Assessment

INRx - Audit

CertiK Verified on Nov 1st, 2022



CertiK Verified on Nov 1st, 2022

INRx - Audit

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

Executive Summary

TYPES	ECOSYSTEM	METHODS
ERC-20	BSC	Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 11/01/2022	N/A

CODEBASE

https://github.com/inrxstablecoin/inrxtoken/tree/main/contracts

...View All

COMMITS

- 55e7b159832d5c870fd14c209253cd9976b27772
- 1579f9db009a9e6c1cc604e542a7db2275aa524e

...View All

Vulnerability Summary

4	3	1	0	0	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Unresolved

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 Mitigated	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.
0	Medium		Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
1	Minor	1 Resolved	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.
2	Informational	2 Resolved	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 | INRX - AUDIT

CODEBASE | INRX - AUDIT

Repository

https://github.com/inrxstablecoin/inrxtoken/tree/main/contracts

Commit

- 55e7b159832d5c870fd14c209253cd9976b27772
- 1579f9db009a9e6c1cc604e542a7db2275aa524e

AUDIT SCOPE | INRX - AUDIT

1 file audited ● 1 file with Mitigated findings

ID	Repo	File	SHA256 Checksum
● INT	CertiKProject/certik-audit-projects	INRxToken.sol	316193ff46ad552a0dac0bc14889b084ec7f45ad93efdb152d05f6142e7bc1b5

APPROACH & METHODS | INRX - AUDIT

This report has been prepared for INRx to discover issues and vulnerabilities in the source code of the INRx - Audit project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis 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 | INRX - AUDIT

Overview

The **INRx** contract is an ERC20 token `INRx`, whose owner has the ability to configure the minters, and pause/unpause the contract.

Privileged Roles

In the **INRx** contract, the owner and the minters are adopted to ensure a good runtime behavior in the contract, which was specified in the finding *Centralization Risks in INRxToken.sol*.

The advantage of those privileged roles in the codebase is that the client reserves the ability to configure the minters, and pause/unpause the contract according to the runtime. 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 keys of the privileged accounts are compromised, the project could have devastating consequences.

FINDINGS | INRX - AUDIT

4	0	1	0	1	2
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for INRx - Audit. Through this audit, we have uncovered 4 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

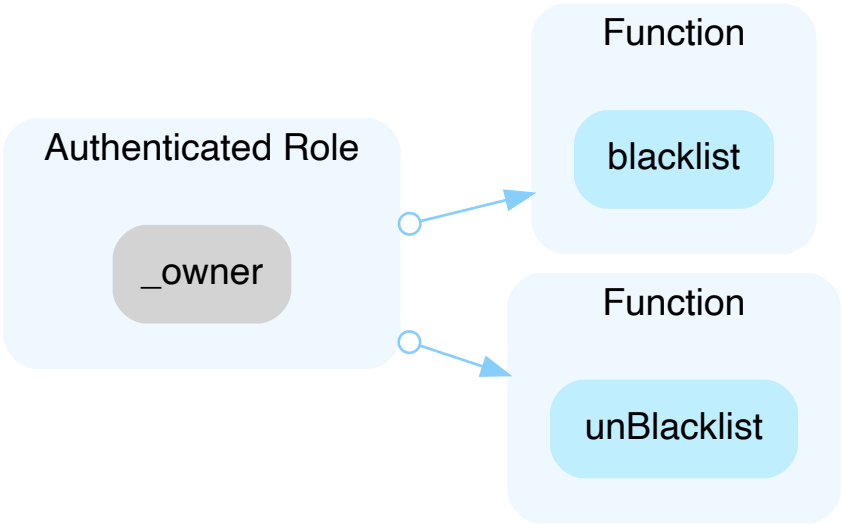
ID	Title	Category	Severity	Status
INT-01	**Centralization Risks In INRxToken.Sol**	**Centralization / Privilege**	**Major**	**Mitigated**
INT-02	Double-Spending Allowance To Mint Token	Logical Issue	Minor	Resolved
INT-03	Dead Code	Coding Style	Informational	Resolved
INT-04	Unused Event	Coding Style	Informational	Resolved

INT-01 | CENTRALIZATION RISKS IN INRXTOKEN.SOL

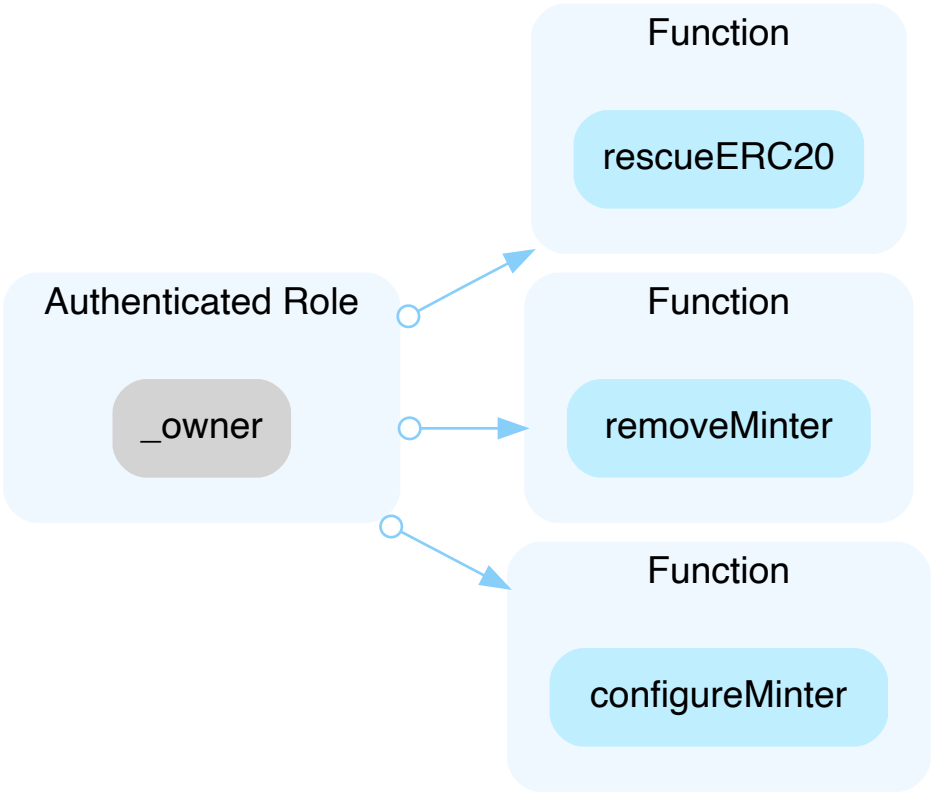
Category	Severity	Location	Status
Centralization / Privilege	**Major**	**INRxToken.sol: 280, 300, 306, 331, 336, 709, 713, 784, 791**	**Mitigated**

Description

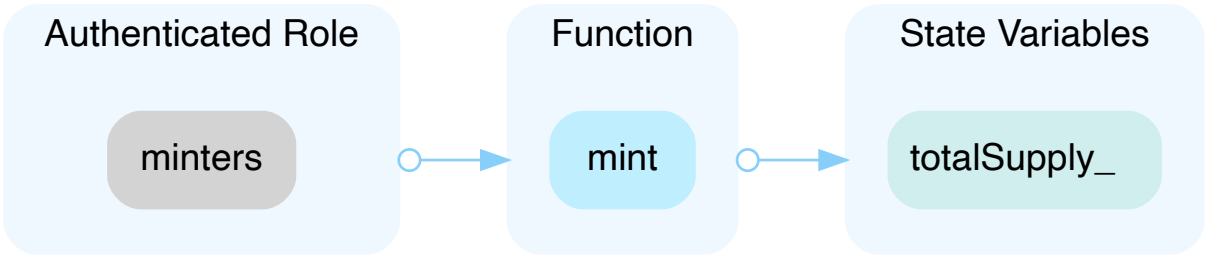
In the contract `Blacklistable` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.



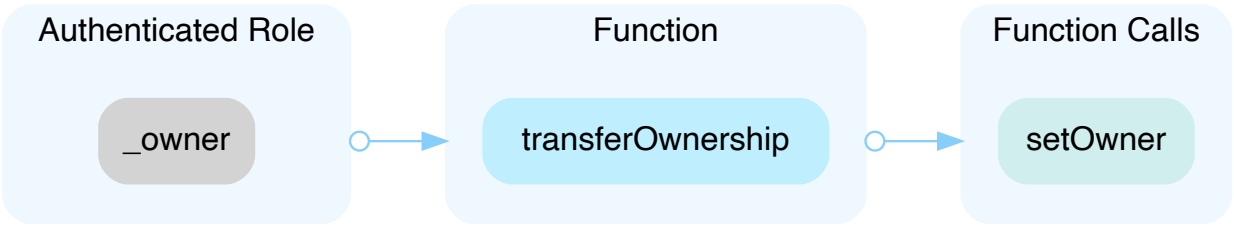
In the contract `INRxToken` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.



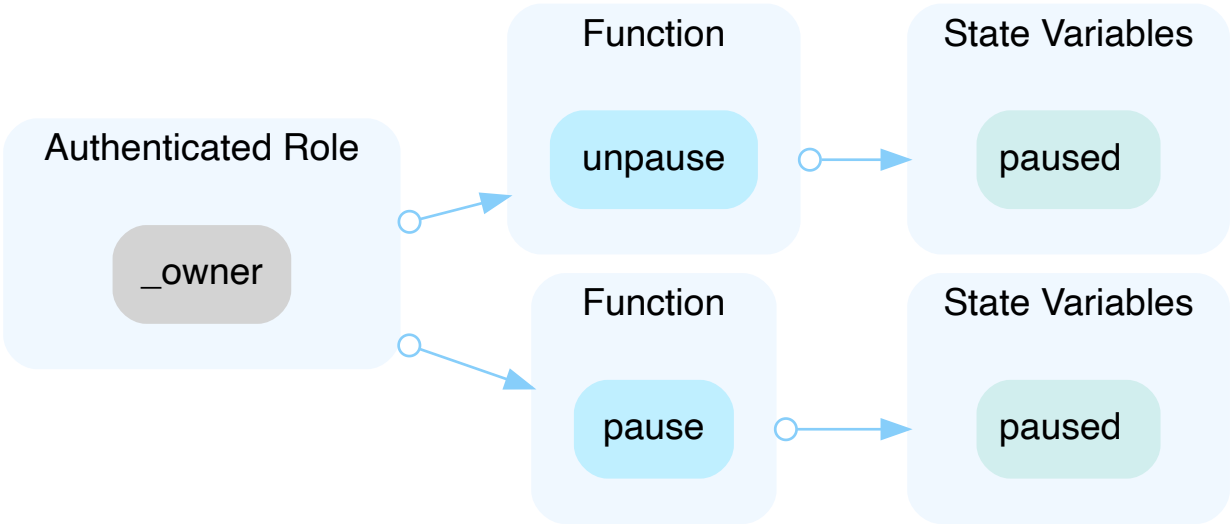
In the contract `INRxToken` the role `minters` has authority over the functions shown in the diagram below. Any compromise to the `minters` account may allow the hacker to take advantage of this authority.



In the contract `Ownable` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.



In the contract `Pausable` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.



Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND

- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

Alleviation

[INRx]: The contract `Blacklistable` and its corresponding code is removed in the commit 1579f9db009a9e6c1cc604e542a7db2275aa524e.

Furthermore, for the deployed contract 0xcD6970d5211EfDF2516A4fc73163db7bA812B212 | BSC the ownership was transferred to the gnosis-safe contract(0xbf5dbafdd31a487129fdb94057f84a0979838055) in the transaction 0xf0c3a8c114039010464d91effdb8fe8fdf665b8a0424a43b46f16c007a54d368.

The gnosis-safe contract requires 3 of 5 owners to confirm the transaction. Here are the 5 owners:

- 0x25AE640f362AeC6b105cD069dB1e09b25cFc5c31
- 0x710973C65277AFD47aC354F5f29E82135Ed19739
- 0x8e4A89e494d7d66046171eD234B1d6211EaD1434
- 0x90Aa066B6c101Bbb5C5585b88C3B35EDC3dc2593
- 0xce0b9307cE30Fe0e98Cf6Fc4fb147eb59310b6E4

INT-02 | DOUBLE-SPENDING ALLOWANCE TO MINT TOKEN

Category	Severity	Location	Status
Logical Issue	Minor	INRxToken.sol: 784	Resolved

Description

A minter may double-spending his/her allowance being authorized by the owner.

Example:

- The owner, Alice, configures the minter, Bob, 100 allowances to mint tokens.
- One day, Alice decides to decrease the allowance to Bob from 100 to 50.
- Before Alice decreases the allowance, Bob spends the 100 allowances by front-running.
- Alice continues to configure the minter, Bob, 50 allowances to mint tokens.
- Bob spends another 50 allowances to mint tokens.

From the above example, we noticed that Bob spends a total of 150 allowances, instead of 50 allowances, even though Alice changed her mind to only give Bob 50 allowances.

Recommendation

Consider adding a check, in the `configureMinter()` function, to ensure that the minter is not a minter before.
Example:

```
require(!minters[minter],"should not be a minter before");
```

Alleviation

[INRx]: The team resolved this issue by adding a check on the `minter` parameter in commit 55e7b159832d5c870fd14c209253cd9976b27772.

INT-03 | DEAD CODE

Category	Severity	Location	Status
Coding Style	Informational	INRxToken.sol: 104, 243, 252, 257, 373, 387, 400, 446, 479 , 492, 507, 574, 588	Resolved

Description

One or more internal functions are not used.

```
104     function reset(Counter storage counter) internal {
```

```
243     function _disableInitializers() internal virtual {
```

```
252     function _getInitializedVersion() internal view returns (uint8) {
```

```
257     function _isInitializing() internal view returns (bool) {
```

```
373     function toHexString(uint256 value) internal pure returns (string memory)
{
```

```
387     function toHexString(uint256 value, uint256 length) internal pure returns
(string memory) {
```

```
400     function toHexString(address addr) internal pure returns (string memory) {
```

```
446     function tryRecover(bytes32 hash, bytes memory signature) internal pure
returns (address, RecoverError) {
```

```
479     function recover(bytes32 hash, bytes memory signature) internal pure
returns (address) {
```

```
492     function tryRecover(
```

```
507     function recover(
```

```
574     function toEthSignedMessageHash(bytes32 hash) internal pure returns
(bytes32) {
```

```
588     function toEthSignedMessageHash(bytes memory s) internal pure returns
(bytes32) {
```

Recommendation

We recommend removing those unused functions.

Alleviation

[INRx]: The team resolved this issue by removing the unused functions in commit 55e7b159832d5c870fd14c209253cd9976b27772.

INT-04 | UNUSED EVENT

Category	Severity	Location	Status
Coding Style	Informational	INRxToken.sol: 290, 319	Resolved

Description

```
290     event PauserChanged(address indexed newAddress);
```

- `PauserChanged` is declared in `Pausable` but never emitted.

```
319     event BlacklisterChanged(address indexed newBlacklister);
```

- `BlacklisterChanged` is declared in `Blacklistable` but never emitted.

Recommendation

We advise removing the unused events or emitting them in the intended functions.

Alleviation

[INRx]: The team resolved this issue by removing the unused events in commit 55e7b159832d5c870fd14c209253cd9976b27772.

OPTIMIZATIONS | INRX - AUDIT

ID	Title	Category	Severity	Status
INT-05	Unnecessary Use Of SafeMath	Gas Optimization	Optimization	Resolved
INT-06	Unused State Variable	Gas Optimization	Optimization	Resolved

INT-05 | UNNECESSARY USE OF SAFEMATH

Category	Severity	Location	Status
Gas Optimization	Optimization	INRxToken.sol: 5, 720, 721, 722, 765, 779, 780, 803, 804, 820, 824	Resolved

Description

The `SafeMath` library is used unnecessarily. With Solidity compiler versions 0.8.0 or newer, arithmetic operations will automatically revert in case of integer overflow or underflow.

```
5  library SafeMath {
```

- An implementation of `SafeMath` library is found.

```
671      using SafeMath for uint256;
```

- `SafeMath` library is used for `uint256` type in `INRxToken` contract.

```
720          totalSupply_ = totalSupply_.add(_amount);
```

- `SafeMath.add` is called in `mint` function of `INRxToken` contract.

Note: Only a sample of 2 `SafeMath` library usage in this contract (out of 11) are shown above.

Recommendation

We advise removing the usage of `SafeMath` library and using the built-in arithmetic operations provided by the Solidity programming language.

Alleviation

[INRx]: The team resolved this issue by removing the usage of `SafeMath` library in commit 55e7b159832d5c870fd14c209253cd9976b27772.

INT-06 | UNUSED STATE VARIABLE

Category	Severity	Location	Status
Gas Optimization	Optimization	INRxToken.sol: 688	Resolved

Description

One or more state variables are never used in the codebase.

Variable `_PERMIT_TYPEHASH_DEPRECATED_SLOT` in `INRxToken` is never used in `INRxToken`.

```
688     bytes32 private _PERMIT_TYPEHASH_DEPRECATED_SLOT;
```

```
670  contract INRxToken is  Ownable, Pausable, Blacklistable,Initializable,
IERC20Permit, EIP712  {
```

Recommendation

We advise removing the unused variables.

Alleviation

[INRx]: The team resolved this issue by removing the unused variables in commit 55e7b159832d5c870fd14c209253cd9976b27772.

FORMAL VERIFICATION | INRX - AUDIT

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 automated formal verification (symbolic model checking) to prove that well-known functions in the smart contracts adhere to their expected behavior.

Considered Functions And Scope

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-transfer-revert-zero	Function `transfer` Prevents Transfers to the Zero Address
erc20-transfer-succeed-normal	Function `transfer` Succeeds on Admissible Non-self Transfers
erc20-transfer-succeed-self	Function `transfer` Succeeds on Admissible Self Transfers
erc20-transfer-correct-amount	Function `transfer` Transfers the Correct Amount in Non-self Transfers
erc20-transfer-correct-amount-self	Function `transfer` Transfers the Correct Amount in Self Transfers
erc20-transfer-change-state	Function `transfer` Has No Unexpected State Changes
erc20-transfer-exceed-balance	Function `transfer` Fails if Requested Amount Exceeds Available Balance
erc20-transfer-recipient-overflow	Function `transfer` Prevents Overflows in the Recipient's Balance

Property Name	Title
erc20-transfer-false	If Function `transfer` Returns `false`, the Contract State Has Not Been Changed
erc20-transfer-never-return-false	Function `transfer` Never Returns `false`
erc20-transferfrom-revert-from-zero	Function `transferFrom` Fails for Transfers From the Zero Address
erc20-transferfrom-revert-to-zero	Function `transferFrom` Fails for Transfers To the Zero Address
erc20-transferfrom-succeed-normal	Function `transferFrom` Succeeds on Admissible Non-self Transfers
erc20-transferfrom-succeed-self	Function `transferFrom` Succeeds on Admissible Self Transfers
erc20-transferfrom-correct-amount	Function `transferFrom` Transfers the Correct Amount in Non-self Transfers
erc20-transferfrom-correct-amount-self	Function `transferFrom` Performs Self Transfers Correctly
erc20-transferfrom-correct-allowance	Function `transferFrom` Updated the Allowance Correctly
erc20-transferfrom-change-state	Function `transferFrom` Has No Unexpected State Changes
erc20-transferfrom-fail-exceed-allowance	Function `transferFrom` Fails if the Requested Amount Exceeds the Available Allowance
erc20-transferfrom-fail-exceed-balance	Function `transferFrom` Fails if the Requested Amount Exceeds the Available Balance
erc20-transferfrom-fail-recipient-overflow	Function `transferFrom` Prevents Overflows in the Recipient's Balance
erc20-transferfrom-false	If Function `transferFrom` Returns `false`, the Contract's State Has Not Been Changed
erc20-transferfrom-never-return-false	Function `transferFrom` Never Returns `false`
erc20-totalsupply-succeed-always	Function `totalSupply` Always Succeeds
erc20-totalsupply-correct-value	Function `totalSupply` Returns the Value of the Corresponding State Variable
erc20-totalsupply-change-state	Function `totalSupply` Does Not Change the Contract's State

Property Name	Title
erc20-balanceof-succeed-always	Function `balanceOf` Always Succeeds
erc20-balanceof-correct-value	Function `balanceOf` Returns the Correct Value
erc20-allowance-succeed-always	Function `allowance` Always Succeeds
erc20-balanceof-change-state	Function `balanceOf` Does Not Change the Contract's State
erc20-allowance-correct-value	Function `allowance` Returns Correct Value
erc20-allowance-change-state	Function `allowance` Does Not Change the Contract's State
erc20-approve-revert-zero	Function `approve` Prevents Giving Approvals For the Zero Address
erc20-approve-succeed-normal	Function `approve` Succeeds for Admissible Inputs
erc20-approve-correct-amount	Function `approve` Updates the Approval Mapping Correctly
erc20-approve-change-state	Function `approve` Has No Unexpected State Changes
erc20-approve-false	If Function `approve` Returns `false`, the Contract's State Has Not Been Changed
erc20-approve-never-return-false	Function `approve` Never Returns `false`

Verification Results

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

- Model checking reports a counterexample that violates the property. Depending on the counterexample,this occurs if
    - The specification of the property is too generic and does not accurately capture the intended behavior of the smart contract. In that case, the counterexample does not indicate a problem in the underlying smart contract. We report such instances as being "inapplicable".
    - The property is applicable to the smart contract. In that case, the counterexample showcases a problem in the smart contract and a correspond finding is reported separately in the Findings section of this report. In the following tables, we report such instances as "invalid". The distinction between spurious and actual counterexamples is done manually by the auditors.
- The model checking result is inconclusive. Such a result does not indicate a problem in the underlying smart contract. An inconclusive result may occur if

- The model checking engine fails to construct a proof. This can happen if the logical deductions necessary are beyond the capabilities of the automated reasoning tool. It is a technical limitation of all proof engines and cannot be avoided in general.
- The model checking engine runs out of time or memory and did not produce a result. This can happen if automatic abstraction techniques are ineffective or of the state space is too big.

Contract INRx (Source File projects/inrx/contracts/INRxToken.sol)

Detailed results for function `transfer`

Property Name	Final Result	Remarks
erc20-transfer-revert-zero	Inconclusive	
erc20-transfer-succeed-normal	Inconclusive	
erc20-transfer-succeed-self	Inconclusive	
erc20-transfer-correct-amount	Inconclusive	
erc20-transfer-correct-amount-self	Inconclusive	
erc20-transfer-change-state	Inconclusive	
erc20-transfer-exceed-balance	Inconclusive	
erc20-transfer-recipient-overflow	Inconclusive	
erc20-transfer-false	Inconclusive	
erc20-transfer-never-return-false	Inconclusive	

Detailed results for function `transferFrom`

Property Name	Final Result	Remarks
erc20-transferfrom-revert-from-zero	Inconclusive	
erc20-transferfrom-revert-to-zero	Inconclusive	
erc20-transferfrom-succeed-normal	Inconclusive	
erc20-transferfrom-succeed-self	Inconclusive	
erc20-transferfrom-correct-amount	Inconclusive	
erc20-transferfrom-correct-amount-self	Inconclusive	
erc20-transferfrom-correct-allowance	Inconclusive	
erc20-transferfrom-change-state	Inconclusive	
erc20-transferfrom-fail-exceed-allowance	Inconclusive	
erc20-transferfrom-fail-exceed-balance	Inconclusive	
erc20-transferfrom-fail-recipient-overflow	Inconclusive	
erc20-transferfrom-false	Inconclusive	
erc20-transferfrom-never-return-false	Inconclusive	

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 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-revert-zero	Inconclusive	
erc20-approve-succeed-normal	Inconclusive	
erc20-approve-correct-amount	Inconclusive	
erc20-approve-change-state	Inconclusive	
erc20-approve-false	Inconclusive	
erc20-approve-never-return-false	Inconclusive	

APPENDIX | INRX - AUDIT

Finding Categories

Categories	Description
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.
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.
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.

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 | 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.