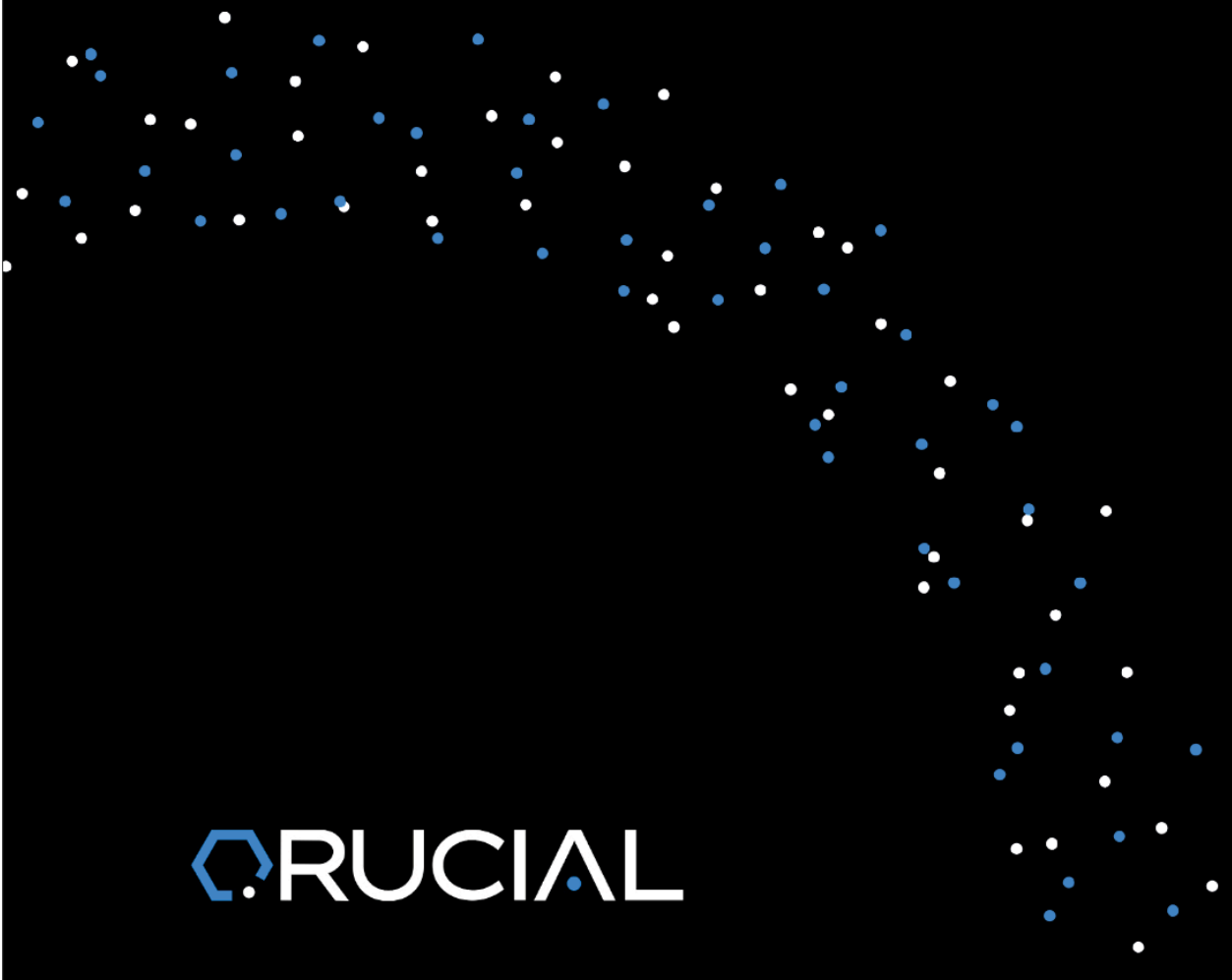
# SMART CONTRACT SECURITY REPORT.

Personally audited and created for:
NXTT.

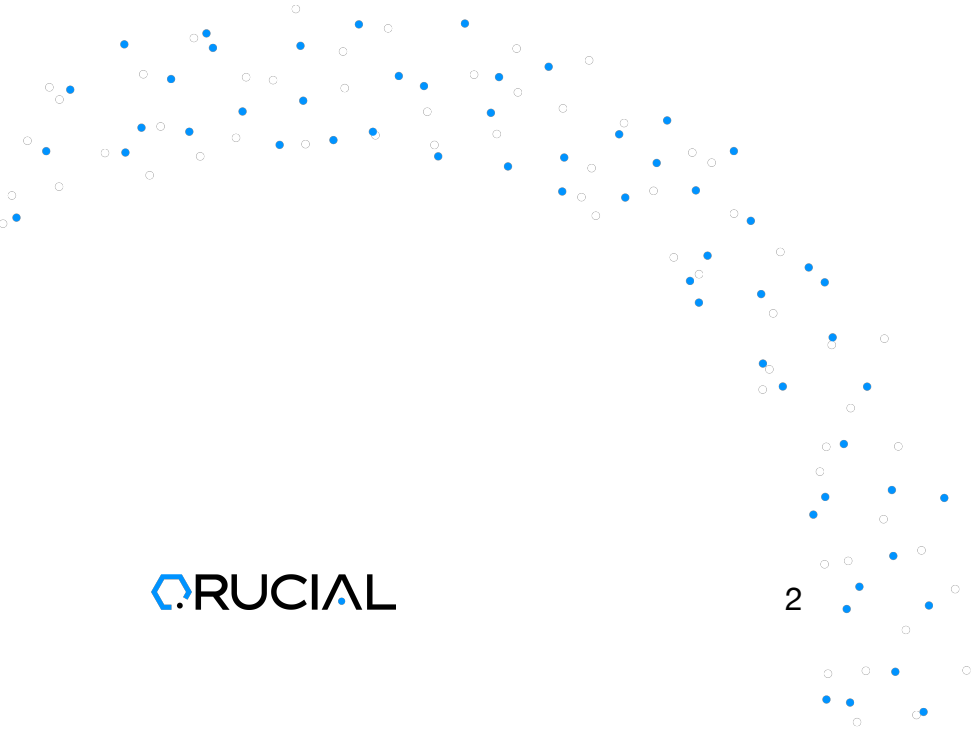


CRUCIAL

22th of January 2022

# TABLE OF CONTENTS





# Introduction

This report has been prepared for NXTT. An extensive analysis has been performed by manual review, static analysis and symbolic execution.

**In our Audits, we focus on the following areas:**

- Analyzing that the smart contract's logic meet the intentions of the client.
- Examining the smart contracts against conventional and unconventional attack vectors
- Verifying the codebase meets the current Industry standard and best practices
- Cross-referencing the Project against the implementation, contract, and structure of similar Industry-leading Projects
- Elaborate line by line manual review of the entire Codebase.

**The findings of the security evaluation resulted ranged from medium to informational.**
We advise you to address these findings as soon as possible to assure a foremost level of security for your project and community.

- Increase good coding practices for a better structure in the source code
- Increase the number of unit test to cover all possible angles of use cases
- Add more comments per function for readability.

# Overview

## Project Summary

| Project name | NXTT |
|---|---|
| Platform | Polygon |
| Language | Solidity |
| Codebase | 1next-earth-contracts.tar.gz.gpg |
| Hash | 8896f856923bcf04ed87946a7b15e61a125ad9b5820a7ee0 f2b095d414c54dd5 |

## Audit Summary

The NextEarth team has requested a security smart contract audit from QRUCIAL on their NXTT token. Two related files were in scope, which implement ERC20 capabilities and an IDO with vesting.

Based on the provided specifications, QRUCIAL assessed the smart contracts from a security perspective and found 5 informational, 7 low and 1 medium level risks. During the audit, these were immediately reported to the NextEarth team, fixes were applied.

The last tar.gz package QRUCIAL received was version 5 which includes fixes reported during the audit.

## Vulnerability Summary

| Level | Total | Pending | Rejected | Accepted | Partially fixed | Fixed |
|---|---|---|---|---|---|---|
| Critical | 0 | - | - | - | - | - |
| High | 0 | - | - | - | - | - |
| Medium | 1 | - | - | 1 | - | - |
| Low | 7 | - | - | 4 | - | 3 |
| Informational | 5 | - | - | 1 | - | 4 |

## Scope

| | |
|---|---|
| Audited Code: | NXTT |
| Blockchain Explorer Link: | - |
| Compiler: | 0.8.11 |
| Number of files: | 2 |
| Scope (list of files) | NXTT.sol<br>NXTTDistributor.sol |

## Risk Classifications

**Critical:**
Vulnerabilities that can lead to a loss of funds, impairment, or control over the system or its function.
We recommend that findings of this classification are fixed immediately.

**High:**
Findings of this classification can impact the flow of logic and can cause direct disruption in the system and the project's organization.
We recommend that issues of this classification are fixed as soon as possible.

**Medium:**
Vulnerabilities of this class have impact on the flow of logic, but does not cause any disturbance that would halt the system or organizational continuity.
We recommend that findings of this class are fixed nonetheless.

**Low:**
Bugs, or vulnerability that have minimal impact and do not pose a significant threat to the project or its users.
We recommend that issues of this class are fixed nonetheless because they increase the attack surface when your project is targeted by malicious actors.

**Informational:**
Findings of this class have a negligible risk factor but refer to best practices in syntax, style or general security.

## MEDIUM: Assignment of Tiers to arbitrary values – NXTTDistributor.sol

**Description:**
The project defines the tiers from 0-5.
However the code's logic allows assignments to tiers from tier 6 up to 255.

**Impact:**
Arbitrary assignments are possible, but only by the merchant role. If the risk is accepted and the merchant is aware, actual issues can be avoided.

**Recommendations:**
Use require statements for no other tiers can be created.

**References:**
https://nextearth.io/faq

## Technical Details:

NXTTDistributor.sol, code line 49

```
function assignTier(uint8[] calldata _tiers, uint256 _tickets, bytes calldata
sig)
external whenNotPaused payable {
  {
      bytes32 hash = keccak256(abi.encodePacked(msg.sender,
                                  _tiers, _tickets));
      bytes32 prefixedHash = ECDSA.toEthSignedMessageHash(hash);
      address signer = ECDSA.recover(prefixedHash, sig);
      //require(signer == merchant, 'invalid merchant signature');
  }
  require(_tickets > 0 && _tiers.length > 0, 'invalid params');
  require(!saleStarted && !saleEnded, "sale already started");

  userTiers[msg.sender] = _tiers;
  userTickets[msg.sender] = _tickets;
  uint256 i;
  for(i=0; i<_tiers.length; i++) {
    tierTickets[_tiers[i]] += _tickets;
  }

  userLockedMatic[msg.sender] += msg.value;
  emit TierAssigned(msg.sender, _tiers, _tickets);
  emit FundsLocked(msg.value);
}
```

## LOW: userTickets and tierTickers limits are not specified - NXTTDistribution.sol

**Description:**
In the assignTier() function it is only checked that the passed value is greater than 0. Up to uint256 tickets can be provided by the merchant. This might be logical to do, but can also be questionable by the community.

**Impact:**
The merchant can create large tickets up to uint256.

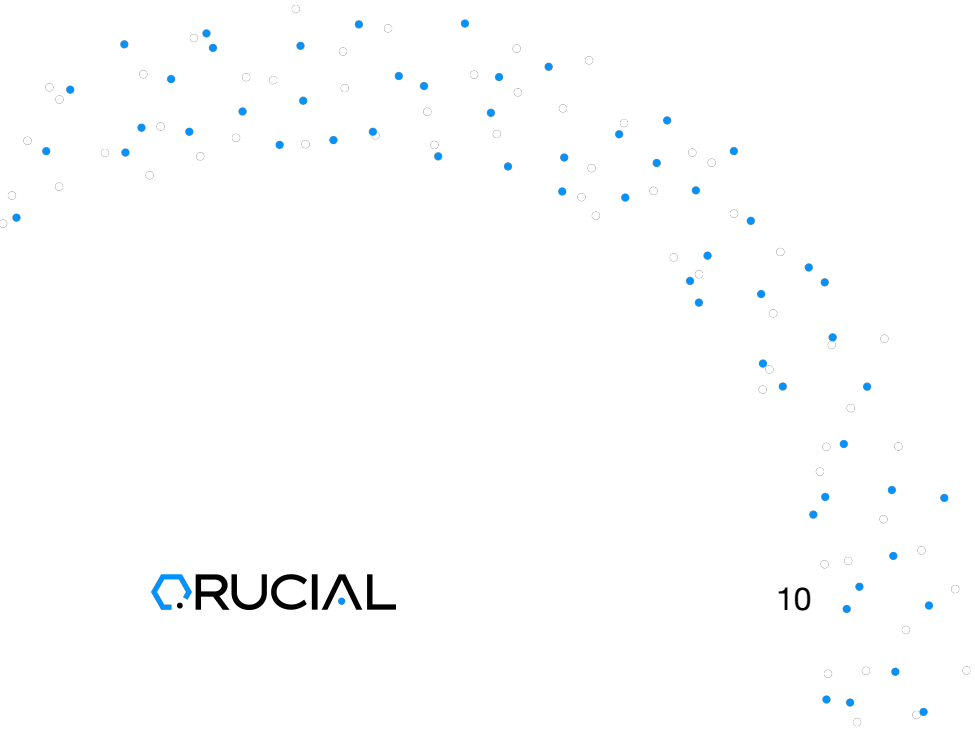
**Recommendations:**
Implement a require for ticket size checks.

**References:**
https://nextearth.io/faq

### Technical Details:

Line 58, the only check:

```
    require(_tickets > 0 && _tiers.length > 0, 'invalid params');
```



## LOW: Lack of decentralization by single point of failure

**Description:**
During the audit, it was found that the project have single point of failures in the system: the smart contract's onlyOwner modifier and the centralized ChainLink oracle. These are common for most projects in 2022 - for the aforementioned one, NextEarth applies a Shamir's Secret based solution, so a single failing person cannot cause direct breach on the admin account. For this reason, we consider this only a low level vulnerability. This finding is valid for both smart contracts.

**Impact:**
In case the admin account is breached, the project might be taken down as a whole. It can happen through multiple scenarios, examples are the following:

- Stealing the devices physically that stores the private keys or Shamir's secrets
- Exploitation of the system that stores the Shamir's secrets
- By human errors, losing the devices that store Shamir's secrets
- System errors, eg. ssd/disk failure and lack of usable backup
- Insider threat
- Incident of the owners of the devices and having no possibility to restore the private keys

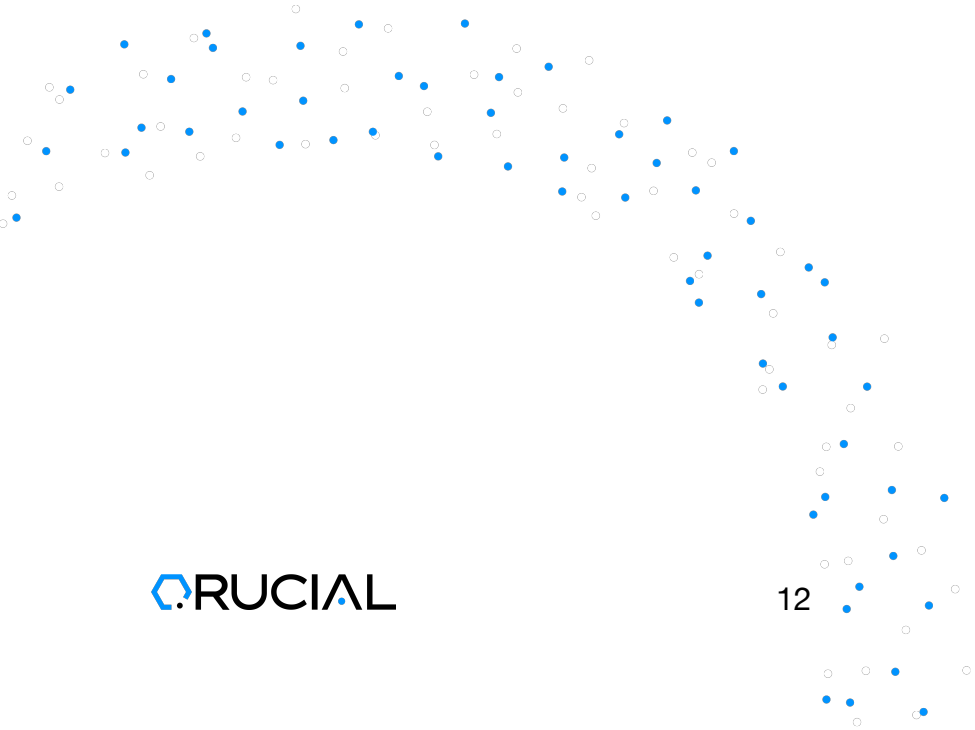
**Recommendations:**
Implement decentralization for the admin functions. Use multiple sources for the price and in case of too high difference, handle the exception.

**References:**
https://github.com/Qrucial/Voronoi
Threshold ECDSA: https://eprint.iacr.org/2019/114.pdf

## Technical Details:

- Reliance on a single account for managing functions on the smart contract (eg. withdrawMATIC() or pause())

- The use of *@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol* makes the prices reliant on a single point.



## LOW: Visibility of functions and could be stricter (optimization)

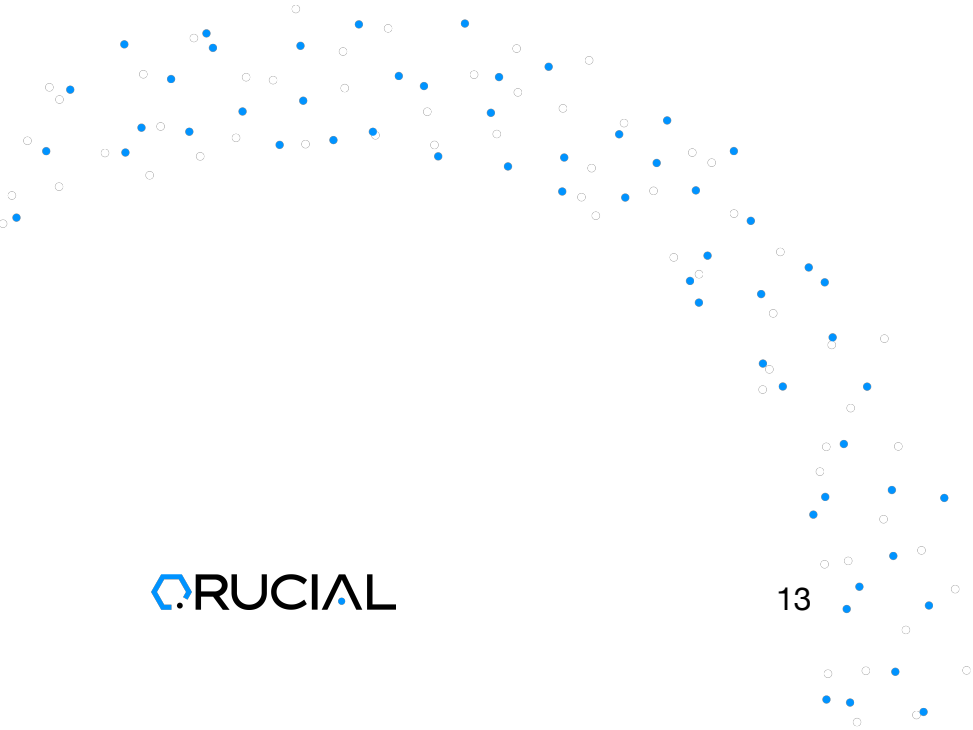
**Description:**
If a function does not require to be called internally, it is possible to save gas costs and improve security by changing their visibility from public to external.

**Impact:**
Function calls will cost less gas (both deploy and call times) and security will be slightly improved.

**Recommendations:**
Replace “public“ to “external“ in all functions listed above.

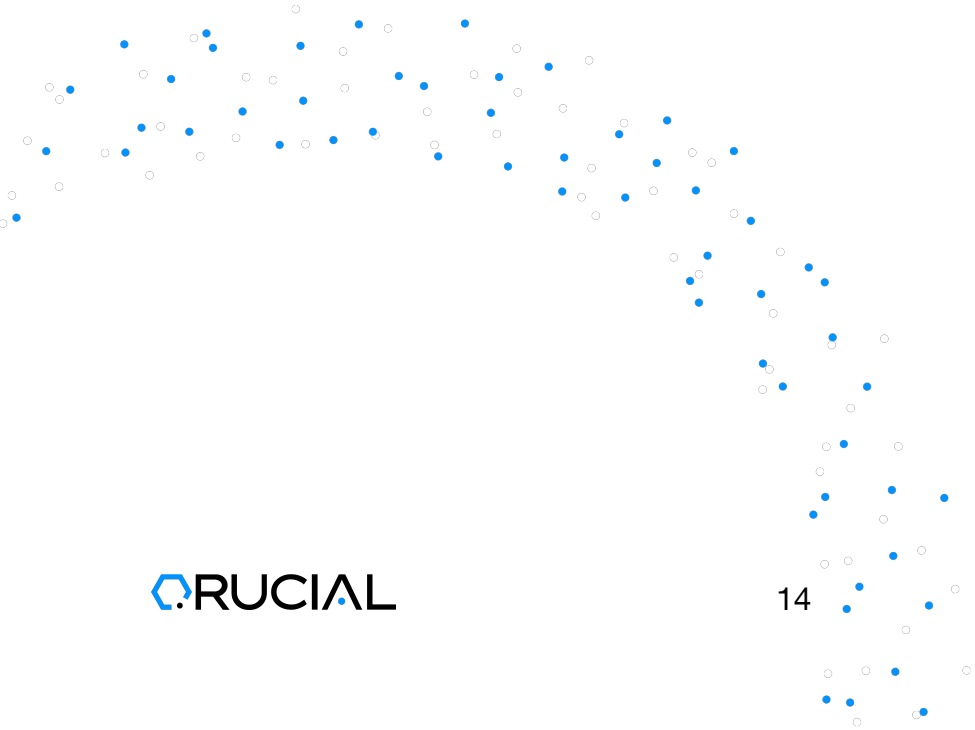
**References:**
https://ezcook.de/2018/01/29/Gas-Used-by-Public-and-External-Function-in-Solidity/

https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices



### Technical Details:

```
function snapshot() public
function pause() public
function unpause() public
```



## LOW: Unused variable, feeGuard - NXTT.sol

**Description:**
The feeGuard variable is not used at all.

**Impact:**
The code won't compile correctly. This issue has been fixed during the audit.

**Recommendations:**
Remove or use the variable.

**References:**
https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

### Technical Details:

`NXTT.feeGuard (1Nxt.sol#28) is never used in NXTT (1Nxt.sol#14-176)`

## LOW: Unused variable, startedAt - NXTTDistributor.sol

**Description:**
The startedAt variable is not used at all.

**Impact:**
The code won't compile correctly. This issue has been fixed during the audit.

**Recommendations:**
Remove or use the variable.

**References:**
https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

### Technical Details:

```
NXTTDistributor.sol, line 135: Unused local variable -> startedAt.
```

## LOW: The used solidity compiler version has known security issues

**Description:**
The solidity compiler version which was sent to QRUCIAL uses 0.8.2 and it has three issues that are considered low risk. Based on the solc security documentation these are not exploitable, but better to be addressed.

**Impact:**
A few low level risks can be avoided by using the latest compiler version.

**Recommendations:**
Use the latest version (0.8.11) of the compiler or the one before.

**References:**
https://blog.soliditylang.org/category/security-alerts/

### Technical Details:

```
pragma solidity ^0.8.2;
```

## LOW: Lack of comments and NatSpec in the code

**Description:**
The code provided by NextEarth is not well commented, nor does it include NatSpec.

This issue applies for both smart contracts.

**Impact:**
This makes the code logic more complicated to follow for developers and audits. At extremes, the Nextearthian community might find it not transparent enough.

**Recommendations:**
Provide comments for general logic and especially for the critical parts of the code.

**References:**
https://docs.soliditylang.org/en/v0.8.11/natspec-format.html

**Technical Details:**

Example:

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.11;

/// @title NXTT
/// @author Silur
/// @notice ...
/// @dev …..
```

# Appendix

## INFORMATIONAL: Unused code and state variables - NXTT.sol

**Description:**
The more functions are available, the larger the attack surface is. Also, the code is more complex by that, hence the chance for bugs, errors are increased.

**Impact:**
Increased complexity and attack surface.

**Recommendations:**
Use optimization so it automatically removes dead code or remove the unused code manually.

**References:**
https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

## Technical Details:

Address.functionCall(address,bytes) (openzeppelin/contracts/utils/Address.sol#85-87)

Address.functionCallWithValue(address,bytes,uint256) (openzeppelin/contracts/utils/Address.sol#114-120)

Address.functionDelegateCall(address,bytes) (openzeppelin/contracts/utils/Address.sol#174-176)

Address.functionDelegateCall(address,bytes,string) (openzeppelin/contracts/utils/Address.sol#184-193)

Address.functionStaticCall(address,bytes) (openzeppelin/contracts/utils/Address.sol#147-149)

Address.functionStaticCall(address,bytes,string) (openzeppelin/contracts/utils/Address.sol#157-166)

Address.sendValue(address,uint256) (openzeppelin/contracts/utils/Address.sol#60-65)

Context._msgData() (openzeppelin/contracts/utils/Context.sol#21-23)

Counters.decrement(Counters.Counter) (openzeppelin/contracts/utils/Counters.sol#32-38)

Counters.reset(Counters.Counter) (openzeppelin/contracts/utils/Counters.sol#40-42)

ECDSA.recover(bytes32,bytes) (openzeppelin/contracts/utils/cryptography/ECDSA.sol#102-106)

ECDSA.recover(bytes32,bytes32,bytes32) (openzeppelin/contracts/utils/cryptography/ECDSA.sol#130-138)

ECDSA.toEthSignedMessageHash(bytes) (openzeppelin/contracts/utils/cryptography/ECDSA.sol#214-216)

ECDSA.toEthSignedMessageHash(bytes32) (openzeppelin/contracts/utils/cryptography/ECDSA.sol#200-204)

ECDSA.tryRecover(bytes32,bytes) (openzeppelin/contracts/utils/cryptography/ECDSA.sol#57-86)

ECDSA.tryRecover(bytes32,bytes32,bytes32) (openzeppelin/contracts/utils/cryptography/ECDSA.sol#115-123)

ERC20._burn(address,uint256) (openzeppelin/contracts/token/ERC20/ERC20.sol#283-298)

ERC20Votes._add(uint256,uint256) (openzeppelin/contracts/token/ERC20/extensions/ERC20Votes.sol#242-244)

ERC20Votes._burn(address,uint256) (openzeppelin/contracts/token/ERC20/extensions/ERC20Votes.sol#172-176)

ERC20Votes._subtract(uint256,uint256) (openzeppelin/contracts/token/ERC20/extensions/ERC20Votes.sol#246-248)

Math.ceilDiv(uint256,uint256) (openzeppelin/contracts/utils/math/Math.sol#39-42)

Math.max(uint256,uint256) (openzeppelin/contracts/utils/math/Math.sol#13-15)

Math.min(uint256,uint256) (openzeppelin/contracts/utils/math/Math.sol#20-22)

NXTT._burn(address,uint256) (NXTT.sol#112-117)

SafeCast.toInt128(int256) (openzeppelin/contracts/utils/math/SafeCast.sol#152-155)

SafeCast.toInt16(int256) (openzeppelin/contracts/utils/math/SafeCast.sol#206-209)

SafeCast.toInt256(uint256) (openzeppelin/contracts/utils/math/SafeCast.sol#236-240)

SafeCast.toInt32(int256) (openzeppelin/contracts/utils/math/SafeCast.sol#188-191)

SafeCast.toInt64(int256) (openzeppelin/contracts/utils/math/SafeCast.sol#170-173)

SafeCast.toInt8(int256) (openzeppelin/contracts/utils/math/SafeCast.sol#224-227)

SafeCast.toUint128(uint256) (openzeppelin/contracts/utils/math/SafeCast.sol#47-50)

SafeCast.toUint16(uint256) (openzeppelin/contracts/utils/math/SafeCast.sol#107-110)

SafeCast.toUint256(int256) (openzeppelin/contracts/utils/math/SafeCast.sol#134-137)

SafeCast.toUint64(uint256) (openzeppelin/contracts/utils/math/SafeCast.sol#77-80)

SafeCast.toUint8(uint256) (openzeppelin/contracts/utils/math/SafeCast.sol#122-125)

SafeCast.toUint96(uint256) (openzeppelin/contracts/utils/math/SafeCast.sol#62-65)

SafeERC20.safeApprove(IERC20,address,uint256) (openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#45-58)

SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#69-80)

SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#60-67)

SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#29-36)

Strings.toHexString(uint256) (openzeppelin/contracts/utils/Strings.sol#40-51)

Strings.toHexString(uint256,uint256) (openzeppelin/contracts/utils/Strings.sol#56-66)

Strings.toString(uint256) (openzeppelin/contracts/utils/Strings.sol#15-35)

## INFORMATIONAL: Unused code and state variables - NXTTDistributor.sol

**Description:**
The more functions are available, the larger the attack surface is. Also, the code is more complex by that, hence the chance for bugs, errors are increased.

**Impact:**
Increased complexity and attack surface.

**Recommendations:**
Use optimization so it automatically removes dead code or remove the unused code manually.

**References:**
https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

### Technical Details:

Address.functionCall(address,bytes) (openzeppelin/contracts/utils/Address.sol#85-87)

Address.functionCallWithValue(address,bytes,uint256) (openzeppelin/contracts/utils/Address.sol#114-120)

Address.functionDelegateCall(address,bytes) (openzeppelin/contracts/utils/Address.sol#174-176)

Address.functionDelegateCall(address,bytes,string) (openzeppelin/contracts/utils/Address.sol#184-193)

Address.functionStaticCall(address,bytes) (openzeppelin/contracts/utils/Address.sol#147-149)

Address.functionStaticCall(address,bytes,string) (openzeppelin/contracts/utils/Address.sol#157-166)

Address.sendValue(address,uint256) (openzeppelin/contracts/utils/Address.sol#60-65)

Context._msgData() (openzeppelin/contracts/utils/Context.sol#21-23)

Counters.decrement(Counters.Counter) (openzeppelin/contracts/utils/Counters.sol#32-38)

Counters.reset(Counters.Counter) (openzeppelin/contracts/utils/Counters.sol#40-42)

ECDSA.recover(bytes32,bytes32,bytes32) (openzeppelin/contracts/utils/cryptography/ECDSA.sol#130-138)

ECDSA.toEthSignedMessageHash(bytes) (openzeppelin/contracts/utils/cryptography/ECDSA.sol#214-216)

ERC20._burn(address,uint256) (openzeppelin/contracts/token/ERC20/ERC20.sol#283-298)

ERC20Votes._add(uint256,uint256) (openzeppelin/contracts/token/ERC20/extensions/ERC20Votes.sol#242-244)

ERC20Votes._burn(address,uint256) (openzeppelin/contracts/token/ERC20/extensions/ERC20Votes.sol#172-176)

ERC20Votes._subtract(uint256,uint256) (openzeppelin/contracts/token/ERC20/extensions/ERC20Votes.sol#246-248)

Math.ceilDiv(uint256,uint256) (openzeppelin/contracts/utils/math/Math.sol#39-42)

Math.max(uint256,uint256) (openzeppelin/contracts/utils/math/Math.sol#13-15)

Math.min(uint256,uint256) (openzeppelin/contracts/utils/math/Math.sol#20-22)

NXTT._burn(address,uint256) (NXTT.sol#112-117)

SafeCast.toInt128(int256) (openzeppelin/contracts/utils/math/SafeCast.sol#152-155)

SafeCast.toInt16(int256) (openzeppelin/contracts/utils/math/SafeCast.sol#206-209)

SafeCast.toInt256(uint256) (openzeppelin/contracts/utils/math/SafeCast.sol#236-240)

SafeCast.toInt32(int256) (openzeppelin/contracts/utils/math/SafeCast.sol#188-191)

SafeCast.toInt64(int256) (openzeppelin/contracts/utils/math/SafeCast.sol#170-173)

SafeCast.toInt8(int256) (openzeppelin/contracts/utils/math/SafeCast.sol#224-227)

SafeCast.toUint128(uint256) (openzeppelin/contracts/utils/math/
SafeCast.sol#47-50)

SafeCast.toUint16(uint256) (openzeppelin/contracts/utils/math/
SafeCast.sol#107-110)

SafeCast.toUint64(uint256) (openzeppelin/contracts/utils/math/
SafeCast.sol#77-80)

SafeCast.toUint8(uint256) (openzeppelin/contracts/utils/math/
SafeCast.sol#122-125)

SafeCast.toUint96(uint256) (openzeppelin/contracts/utils/math/
SafeCast.sol#62-65)

SafeERC20.safeApprove(IERC20,address,uint256) (openzeppelin/contracts/token/
ERC20/utils/SafeERC20.sol#45-58)

SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (openzeppelin/contracts/
token/ERC20/utils/SafeERC20.sol#69-80)

SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (openzeppelin/contracts/
token/ERC20/utils/SafeERC20.sol#60-67)

SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (openzeppelin/
contracts/token/ERC20/utils/SafeERC20.sol#29-36)

Strings.toHexString(uint256) (openzeppelin/contracts/utils/Strings.sol#40-51)

Strings.toHexString(uint256,uint256) (openzeppelin/contracts/utils/
Strings.sol#56-66)

Strings.toString(uint256) (openzeppelin/contracts/utils/Strings.sol#15-35)

## INFORMATIONAL: Compiled bytecode is larger than 24576 bytes

**Description:**
Ethereum based systems limit the smart contracts' size to 24576 bytes. Compiling NXTT.sol and NXTTDistributor.sol becomes larger without optimization.

**Impact:**
Optimization needs to be enabled.

**Recommendations:**
Use optimization.

**References:**
https://ethereum.org/mr/developers/tutorials/downsizing-contracts-to-fight-the-contract-size-limit/

### Technical Details:

```
Contract code size exceeds 24576 bytes (a limit introduced in Spurious Dragon).
```

## INFORMATIONAL: Using the snapshot() function modifies gas fees - NXTT.sol

**Description:**
Comment from ERC20Snapshot.sol the code:

"While an open way of calling {_snapshot} is required for certain trust minimization mechanisms such as forking, you must consider that it can potentially be used by attackers in two ways.

First, it can be used to increase the cost of retrieval of values from snapshots, although it will grow logarithmically thus rendering this attack ineffective in the long term. Second, it can be used to target specific accounts and increase the cost of ERC20 transfers for them, in the ways specified in the Gas Costs section above."

**Impact:**
Gas fees can be increased for single accounts.
Taking snapshots can be only done by accounts in the canSnapshot role. Only the ERC20 transfers are affected.

**Recommendations:**
Stay aware of the warning when adding accounts to the canSnapshot role.

**References:**
https://ethereum.org/mr/developers/tutorials/downsizing-contracts-to-fight-the-contract-size-limit/

### Technical Details:

The increased gas fee is not statistically verified yet and such an attack might be unfeasible from a practical point of view.

## INFORMATIONAL: Assembly and low level calls

**Description:**
Using low-level calls and assembly in smart contracts highly increase project complexity and possibilities for error, hence they are be er to be avoided if not required.

**Impact:**
Even if not directly a vulnerability, low-level calls might open up the surface for high-impact attacks.

**Recommendations:**
Avoid low-level calls whenever possible.

**References:**
https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

https://docs.soliditylang.org/en/v0.8.9/control-structures.html#error-handling-assert-require-revert-and-exceptions

### Technical Details:

```
Low level call in NXTT.withdrawMatic() (1Nxt.sol#164-167):

      - (ok) = msg.sender.call{value: address(this).balance}() (1Nxt.sol#165)

Low level call in Address.sendValue(address,uint256) (openzeppelin/contracts/
utils/Address.sol#60-65):

      - (success) = recipient.call{value: amount}() (openzeppelin/contracts/
utils/Address.sol#63)

Low level call in Address.functionCallWithValue(address,bytes,uint256,string)
(openzeppelin/contracts/utils/Address.sol#128-139):

      - (success,returndata) = target.call{value: value}(data) (openzeppelin/
contracts/utils/Address.sol#137)

Low level call in Address.functionStaticCall(address,bytes,string)
(openzeppelin/contracts/utils/Address.sol#157-166):

      - (success,returndata) = target.staticcall(data) (openzeppelin/contracts/
utils/Address.sol#164)

Low level call in Address.functionDelegateCall(address,bytes,string)
(openzeppelin/contracts/utils/Address.sol#184-193):

- (success,returndata) = target.delegatecall(data) (openzeppelin/contracts/
utils/Address.sol#191)



Low level call in NXTTDistributor.release() (1NXTTDist.sol#82-94):

      - (ok) = msg.sender.call{value: leftover}() (1NXTTDist.sol#91)

Low level call in NXTTDistributor.withdrawMatic() (1NXTTDist.sol#165-168):

      - (ok) = msg.sender.call{value: address(this).balance}()
(1NXTTDist.sol#166)

Low level call in NXTT.withdrawMatic() (NXTT.sol#160-163):

      - (ok) = msg.sender.call{value: address(this).balance}() (NXTT.sol#161)

Low level call in Address.sendValue(address,uint256) (openzeppelin/contracts/
utils/Address.sol#60-65):

      - (success) = recipient.call{value: amount}() (openzeppelin/contracts/
utils/Address.sol#63)

Low level call in Address.functionCallWithValue(address,bytes,uint256,string)
(openzeppelin/contracts/utils/Address.sol#128-139):

      - (success,returndata) = target.call{value: value}(data) (openzeppelin/
contracts/utils/Address.sol#137)

Low level call in Address.functionStaticCall(address,bytes,string)
(openzeppelin/contracts/utils/Address.sol#157-166):

      - (success,returndata) = target.staticcall(data) (openzeppelin/contracts/
utils/Address.sol#164)

Low level call in Address.functionDelegateCall(address,bytes,string)
(openzeppelin/contracts/utils/Address.sol#184-193):

      - (success,returndata) = target.delegatecall(data) (openzeppelin/
contracts/utils/Address.sol#191)
```

# DISCLAIMER

This report is fixed to the scope and subject to terms and conditions of the service agreement provided to the customer. This report must not be referred, transmitted or disclosed to a third party without QRUCIAL's prior written consent.

This report is not an endorsement disapproval of a team, a product, a service, a company, or an individual. This report should not be considered as financial advice and does not indicate any financial or economic value in an asset, an asset class a product or service. This report is not to be seen as an indication of the legal compliance regarding of a project an asset, an asset class or a business model.

This report does not provide the guarantee, that a project is without bugs, errors vulnerabilities or code that is harmful to machines, software, or data. This report is also no indication of the validity of any business model or technology. Each individual organization is responsible to do their own due diligence or security assessment. This report is not to be seen as a guarantee of the functionality of a technology or its security. The use of access or information in this audit is used on the risk of the reader or user of this document.

This report holds no guarantee that the given information meets requirements of any kind, is compatible with applications,  any software or systems. It is also not guaranteed that this audit is free of errors or harmful code or will cause interruptions of any software or systems. We do not give any guarantee of accuracy, reliability, or correctness of the information given in this audit. All third-party material provided to the client may be subject to the terms and conditions of third parties. All third-party material provided is provided without the guarantee of correctness. No third-party has the right to use the trademark QRUCIAL, its products or services as a reference or endorsement of its own products or services without prior written consent.

# THANK YOU.

Our team gave their full commitment to craft this audit with precision and focus on details with the goal to support you improving your work.

With this audit we want to provide you a guidance to make your project more
secure and for presenting your
community a product they can trust in.

We make security our priority, so you don‘t have to.

CRUCIAL


----Polkadot{.js} account validation address----

5EHagRYLNsCJUx5bA5Y8MZWLqPVzqQbFMC76V68Wzv7GHHXD

----Polkadot{.js} account validation address----

QRUCIAL

QRUCIAL OÜ c/o Gate to Baltics
Narva mnt 7-652, 10117 | Tallinn, Estonia