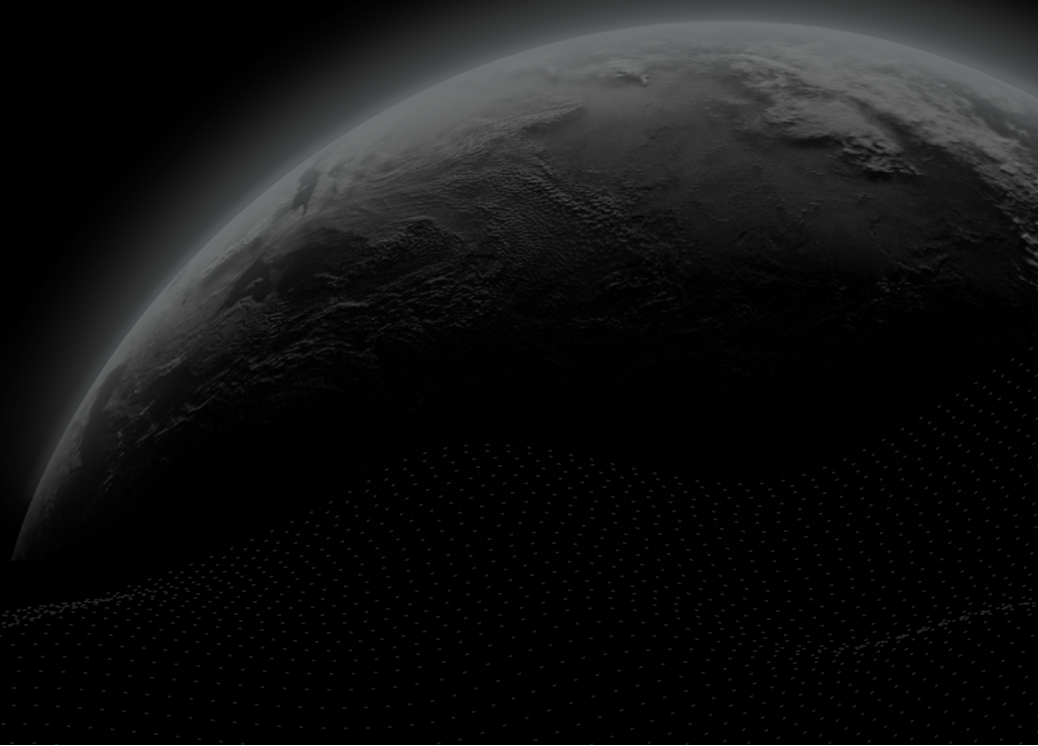
CERTIK

Security Assessment

Boundless Network Token - Audit

CertiK Assessed on Jul 16th, 2024



CertiK Assessed on Jul 16th, 2024

Boundless Network Token - Audit

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

Executive Summary

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

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 07/16/2024	N/A

CODEBASE

bc129f42c0be8181a8d0ad76e3642744a79af629
0xf3a1bf85fc8d739c343d75f9a998955c234743c4
0x6E0609352D29de397a9D4dBEf217004740DF9A3C
View All in Codebase Page

COMMITS

bc129f42c0be8181a8d0ad76e3642744a79af629
0xf3a1bf85fc8d739c343d75f9a998955c234743c4
0x6E0609352D29de397a9D4dBEf217004740DF9A3C
View All in Codebase Page

Vulnerability Summary

4	0	4	0	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined

0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
4	Major	4 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.
0	Minor		Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
0	Informational		Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | BOUNDLESS NETWORK TOKEN - AUDIT

CODEBASE | BOUNDLESS NETWORK TOKEN - AUDIT

Repository

bc129f42c0be8181a8d0ad76e3642744a79af629

0xf3a1bf85fc8d739c343d75f9a998955c234743c4

0x6E0609352D29de397a9D4dBEf217004740DF9A3C

18dd401f2cc6feea2c128c58310068e76118b0fd

Commit

bc129f42c0be8181a8d0ad76e3642744a79af629

0xf3a1bf85fc8d739c343d75f9a998955c234743c4

0x6E0609352D29de397a9D4dBEf217004740DF9A3C

18dd401f2cc6feea2c128c58310068e76118b0fd

AUDIT SCOPE | BOUNDLESS NETWORK TOKEN - AUDIT

2 files audited ● 1 file with Acknowledged findings ● 1 file without findings

ID	Repo	File	SHA256 Checksum
● BTB	rotonda1/bun-token	contracts/BunToken.sol	d46cd5c47bc78f61d57aa30e2b54fc43020bd421d7df8400259c6b170d9d63ef
● BTU	rotonda1/bun-token	contracts/BunToken.sol	b8d0db5c27e15f87afb4a52d85b7c46e61b2e95736f1c9adbaa4091609127b93

APPROACH & METHODS | BOUNDLESS NETWORK TOKEN - AUDIT

This report has been prepared for Boundless to discover issues and vulnerabilities in the source code of the Boundless Network Token - Audit 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:

- 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 | BOUNDLESS NETWORK TOKEN - AUDIT

Overview

The **Boundless Network Token** project contains two contracts: an ERC20 token contract and a locked token contract. All tokens are initially distributed to the deployer, and the ERC20 token supports batch transfers. The `SYSTEM_ROLE` can create a locked token contract and transfer tokens to it, allowing the `_beneficiary` to claim these tokens once the `releaseTime` is reached.

External Dependencies

the following library/contract are considered as the third-party dependencies:

- @openzeppelin/contracts/

The scope of the audit would treat those third-party entities as black boxes and assume their functional correctness and return honest results.

Privileged Functions

In the **Boundless Network Token** project, multiple roles are adopted to ensure the dynamic runtime updates of the project, which were specified in the centralization findings *BTB-03, BTB-04*.

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

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

FINDINGS | BOUNDLESS NETWORK TOKEN - AUDIT

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

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

ID	Title	Category	Severity	Status
BTB-02	**Initial Token Distribution**	**Centralization**	**Major**	**Mitigated**
BTB-03	**Centralization Risks**	**Centralization**	**Major**	**Mitigated**
BTB-04	**Pausing Centralization Risks**	**Centralization**	**Major**	**Mitigated**
BTB-07	**Withdrawal Centralization Risk**	**Logical Issue, Centralization**	**Major**	**Mitigated**

BTB-02 | INITIAL TOKEN DISTRIBUTION

Category	Severity	Location	Status
Centralization	Major	contracts/BunToken.sol (bc129f42c0be8181a8d0ad76e3642744a79af629): 86	Mitigated

Description

All of the `BUN` tokens are sent to the contract deployer. This is a centralization risk because the deployer can distribute tokens without obtaining the consensus of the community. Any compromise to these addresses may allow a hacker to steal and sell tokens on the market, resulting in severe damage to the project.

Recommendation

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

Alleviation

[Boundless Team, 06/17/2024]: The team acknowledged the finding and stated that all token operations will be executed by multi-signature wallets.

[CertiK, 06/18/2024]: The team has yet to address the centralization-related risks. CertiK strongly encourages the project team to periodically revisit the private key security management of all addresses related to privileged roles.

[Boundless Team, 07/11/2024]: The team deployed the contract at the address 0xF3A1BF85fc8D739c343D75f9a998955c234743C4, transferred all the tokens to a multi-sig wallet, and provided the distribution plan: https://burrito-wallet.gitbook.io/boundlessnetwork/token-allocation-overview.

The multi-sig wallet : 0xe4e1153e0c6e9c51e86a58e4c8a36d8313863b3e:

- 0x68bCd7083973d0cADC78671df3Df6E327dD17AfA
- 0x480B49aC655967397B10f96654d3618Eb519797F
- 0x508ce057bf933391bbc7a7DEe0B960E2C75389A8

[CertiK, 07/11/2024] The team mitigated this finding at the specified address 0xF3A1BF85fc8D739c343D75f9a998955c234743C4 by transferring all undistributed tokens to the multi-signature wallet 0xe4e1153e0c6e9c51e86a58e4c8a36d8313863b3e.

[Boundless Team, 07/15/2024]: The team deployed the contract at the address 0x6E0609352D29de397a9D4dBEf217004740DF9A3C, transferred all the tokens to a multi-sig wallet, and provided the distribution plan: https://burrito-wallet.gitbook.io/boundlessnetwork/token-allocation-overview.

The multi-sig wallet : 0xe4e1153e0c6e9c51e86a58e4c8a36d8313863b3e:

- 0x68bCd7083973d0cADC78671df3Df6E327dD17AfA
- 0x480B49aC655967397B10f96654d3618Eb519797F
- 0x508ce057bf933391bbc7a7DEe0B960E2C75389A8

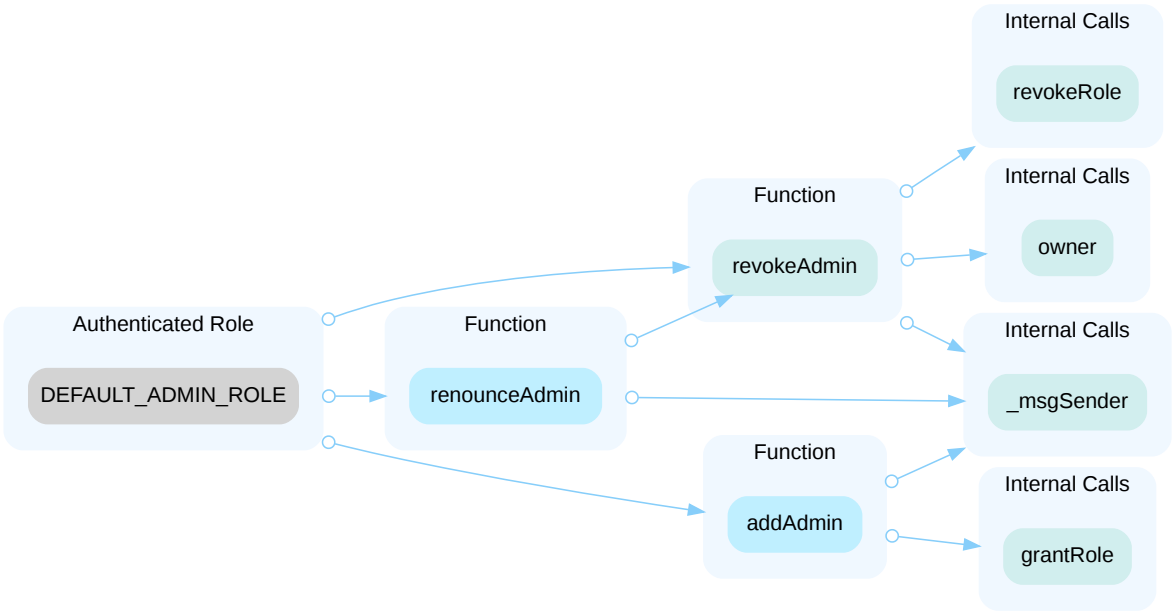
[CertiK, 07/15/2024] The team mitigated this finding at the specified address 0x6E0609352D29de397a9D4dBEf217004740DF9A3C by transferring all undistributed tokens to the multi-signature wallet 0xe4e1153e0c6e9c51e86a58e4c8a36d8313863b3e.

BTB-03 | CENTRALIZATION RISKS

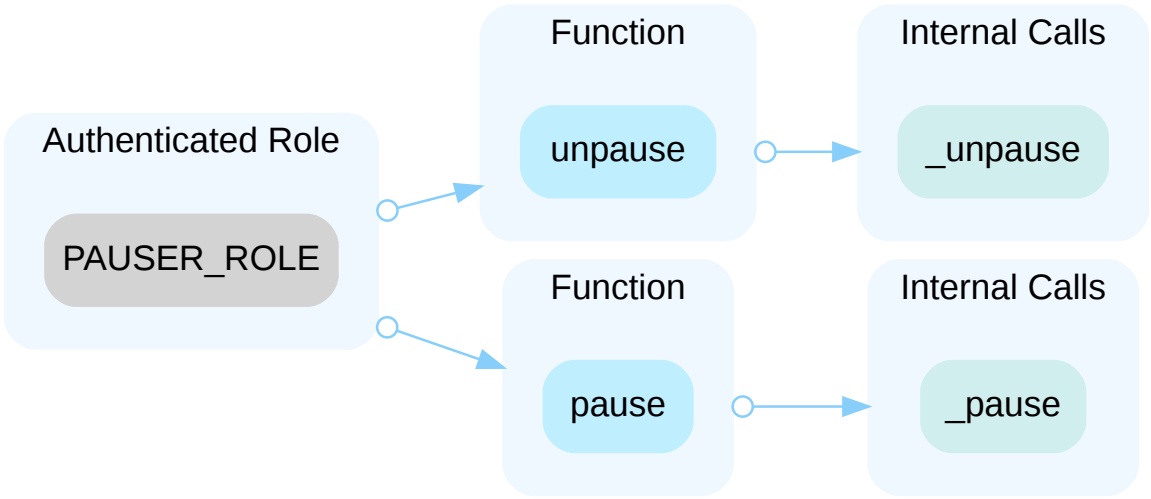
Category	Severity	Location	Status
Centralization	Major	contracts/BunToken.sol (bc129f42c0be8181a8d0ad76e3642744a79af629): 51, 89, 94, 98, 102, 148, 160, 166, 175, 179	Mitigated

Description

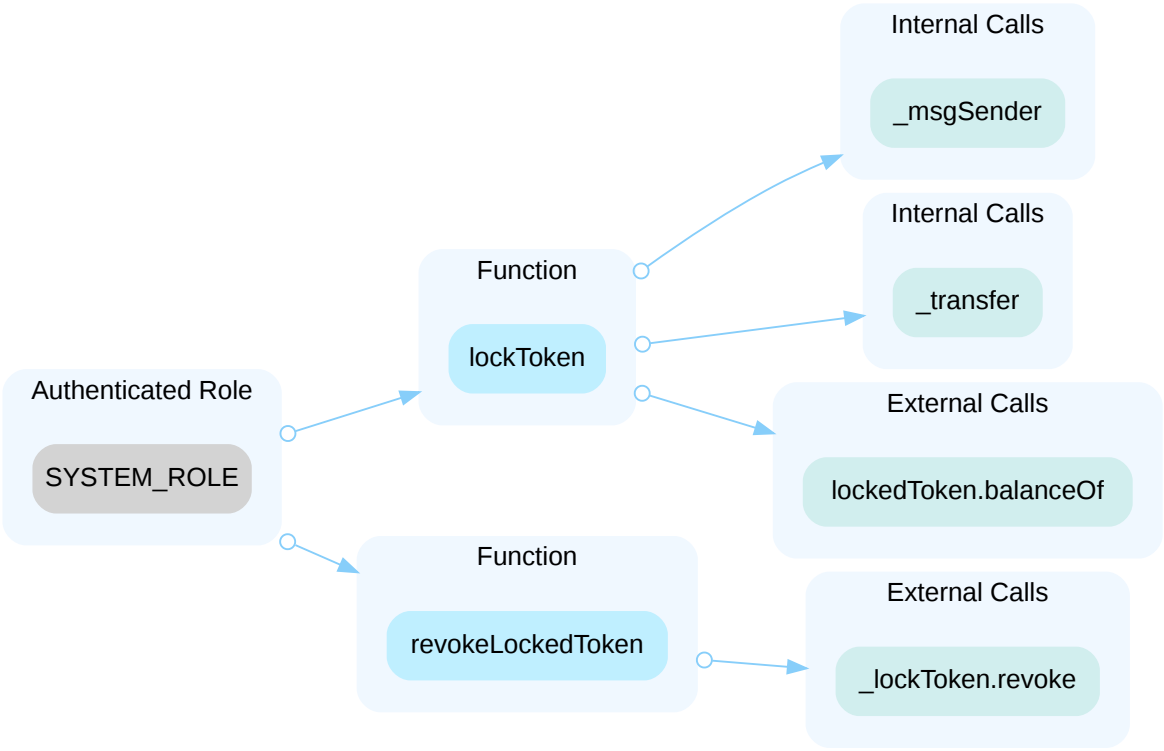
In the contract `BunToken` the role `DEFAULT_ADMIN_ROLE` has authority over the functions shown in the diagram below. Any compromise to the `DEFAULT_ADMIN_ROLE` account may allow the hacker to take advantage of this authority to grant the `DEFAULT_ADMIN_ROLE`, `PAUSER_ROLE`, `SYSTEM_ROLE` role.



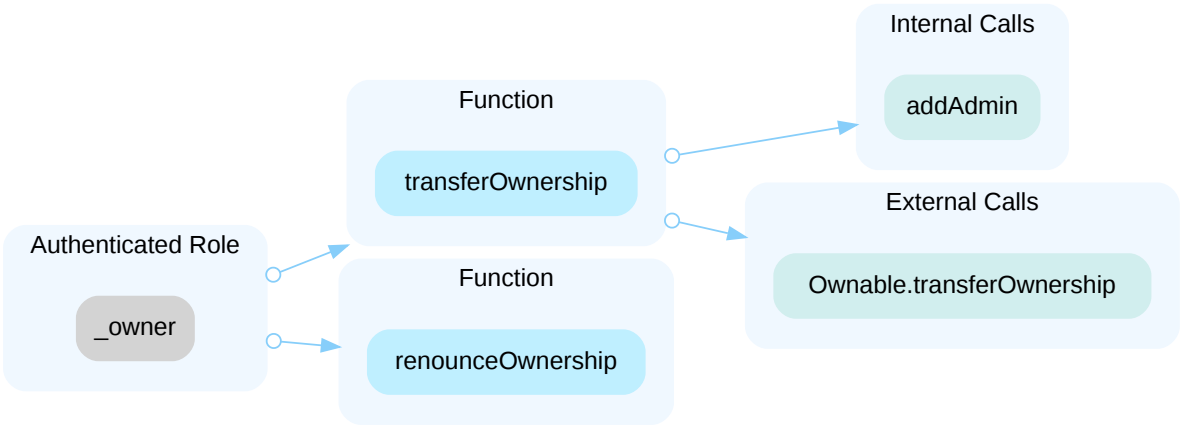
In the contract `BunToken` the role `PAUSER_ROLE` has authority over the functions shown in the diagram below. Any compromise to the `PAUSER_ROLE` account may allow the hacker to take advantage of this authority to pause the contract.



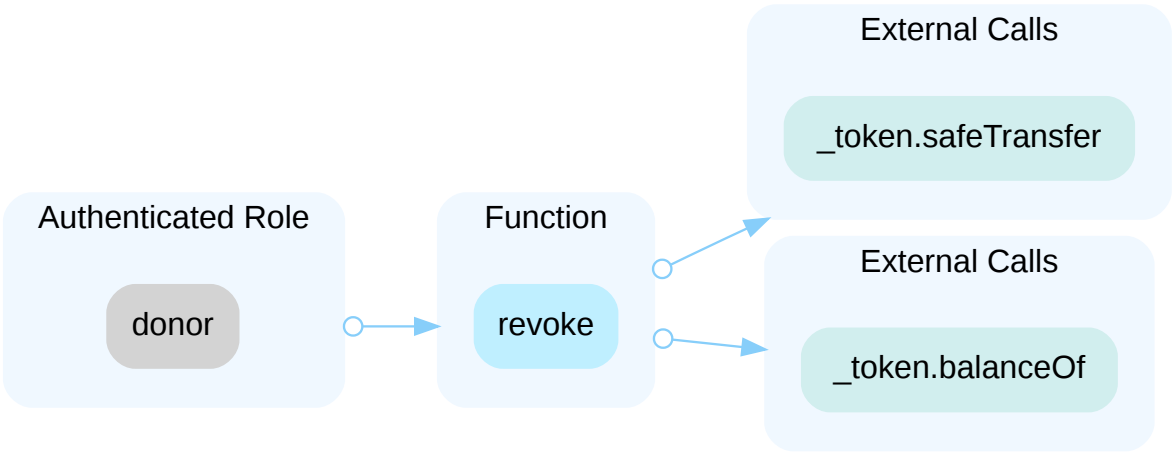
In the contract `BunToken` the role `SYSTEM_ROLE` has authority over the functions shown in the diagram below. Any compromise to the `SYSTEM_ROLE` account may allow the hacker to take advantage of this authority lock tokens and revoke the lock.



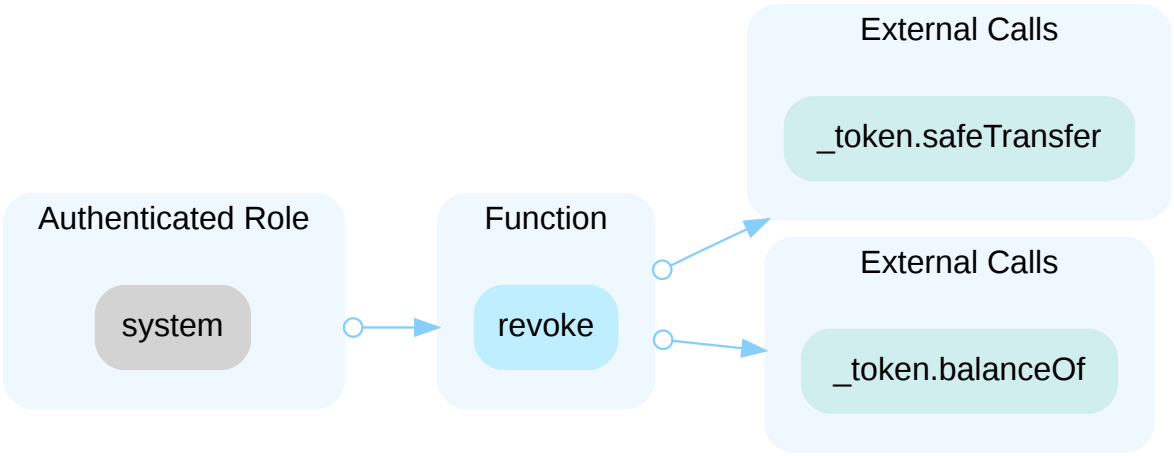
In the contract `BunToken` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority to transfer ownership.



In the contract `LockedToken` the role `donor` has authority over the functions shown in the diagram below. Any compromise to the `donor` account may allow the hacker to take advantage of this authority to unlock the tokens and withdraw tokens.



In the contract `LockedToken` the role `system` has authority over the functions shown in the diagram below. Any compromise to the `system` account may allow the hacker to take advantage of this authority to revoke the lock and transfer tokens to the `donor` .



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

[CertiK, 06/17/2024]: The team has removed the `PAUSER_ROLE` . The functions are previously accessible by the `PAUSER_ROLE` can now be called by the `DEFAULT_ADMIN_ROLE` , the changes were reflected in the commitc50928da7d5dfca468b65ea65a38f8e7cca64887.

[Boundless Team, 06/17/2024]: The team acknowledged the finding and we will use a multi-sig contract wallet for all operations.

[CertiK, 06/18/2024]: The team has yet to address the centralization-related risks. CertiK strongly encourages the project team to periodically revisit the private key security management of all addresses related to privileged roles.

[CertiK, 07/11/2024]: The team used the combination of Timelock and multi-sig wallet to mitigate the centralized risk.

The token contract address: 0xF3A1BF85fc8D739c343D75f9a998955c234743C4.

The timelock contract address: (0xc7d74e1905487ecb4bf657ee878ce339549e645f

The multi-sig wallet(Threshold: 2 out of 3 owner(s)): 0xe4e1153e0c6e9c51e86a58e4c8a36d8313863b3e:

- 0x68bCd7083973d0cADC78671df3Df6E327dD17AfA

- 0x480B49aC655967397B10f96654d3618Eb519797F
- 0x508ce057bf933391bbc7a7DEe0B960E2C75389A8

The team transferred the roles `DEFAULT_ADMIN_ROLE` and `SYSTEM_ROLE` of the token contract to the timelock address.

Transaction Hash: 0x3a3034a8162fdd29664f66f4ab55c24b9e485217eee4c569c7d1097538dd5bdf

Upon deployment of the timelock contract, the `PROPOSER_ROLE` was transferred to the multi-signature wallet.

The team renounced admin role from multi-sig wallet via the transaction hash 0x212d935485f7bb506af5d617ebcf172b1e8c79200221e360ec8f7050b4eb6b63.

[CertiK, 07/15/2024]: The team used the combination of Timelock and multi-sig wallet to mitigate the centralized risk.

The token contract address: 0x6E0609352D29de397a9D4dBEf217004740DF9A3C.

The timelock contract address: (0xc7d74e1905487ecb4bf657ee878ce339549e645f

The multi-sig wallet(Threshold: 2 out of 3 owner(s)): 0xe4e1153e0c6e9c51e86a58e4c8a36d8313863b3e:

- 0x68bCd7083973d0cADC78671df3Df6E327dD17AfA
- 0x480B49aC655967397B10f96654d3618Eb519797F
- 0x508ce057bf933391bbc7a7DEe0B960E2C75389A8

The team transferred the roles `DEFAULT_ADMIN_ROLE` and `SYSTEM_ROLE` of the token contract to the timelock address.

Transaction Hash: 0x60bec246fb7ed3b506cefa35bda57ee1ae7f0b0c3941e188e4bf673ea5191fbd

Upon deployment of the timelock contract, the `PROPOSER_ROLE` was transferred to the multi-signature wallet.

The team renounced admin role from multi-sig wallet via the transaction hash 0x1758c9f2b460487512e1c46bbe22c95db46015e14c34b0e4b5dcf23a646852ce.

BTB-04 | PAUSING CENTRALIZATION RISKS

Category	Severity	Location	Status
Centralization	Major	contracts/BunToken.sol (bc129f42c0be8181a8d0ad76e3642744a79af629): 98, 102	Mitigated

Description

In the contract `BunToken`, the `PAUSER_ROLE` has the authority to update the status of the `_paused` and further pause/resume the functionality of the token transfers.

```
98      function pause() public onlyRole(PAUSER_ROLE) {
99          _pause();
100     }
101
102     function unpause() public onlyRole(PAUSER_ROLE) {
103         _unpause();
104     }
```

Any compromise to the private key of the `PAUSER_ROLE` may allow hackers to take advantage of this authority and allow/prevent user access to token transfer functionalities.

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., multi-signature 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.

Note: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

[Boundless Team, 06/17/2024]: The team acknowledged the finding and we will use a multi-sig contract wallet for all operations.

[CertiK, 06/18/2024]: The team has yet to address the centralization-related risks. CertiK strongly encourages the project team to periodically revisit the private key security management of all addresses related to privileged roles.

[CertiK, 07/11/2024]: The team used the combination of Timelock and multi-sig wallet to mitigate the centralized risk.

The token contract address: 0xF3A1BF85fc8D739c343D75f9a998955c234743C4.

The timelock contract address: (0xc7d74e1905487ecb4bf657ee878ce339549e645f

The multi-sig wallet(Threshold: 2 out of 3 owner(s)): 0xe4e1153e0c6e9c51e86a58e4c8a36d8313863b3e:

- 0x68bCd7083973d0cADC78671df3Df6E327dD17AfA
- 0x480B49aC655967397B10f96654d3618Eb519797F
- 0x508ce057bf933391bbc7a7DEe0B960E2C75389A8

The team transferred the roles `DEFAULT_ADMIN_ROLE` and `SYSTEM_ROLE` of the token contract to the timelock address.

Transaction Hash: 0x3a3034a8162fdd29664f66f4ab55c24b9e485217eee4c569c7d1097538dd5bdf

Upon deployment of the timelock contract, the `PROPOSER_ROLE` was transferred to the multi-signature wallet.

The team renounced admin role from multi-sig wallet via the transaction hash 0x212d935485f7bb506af5d617ebcf172b1e8c79200221e360ec8f7050b4eb6b63.

[CertiK, 07/15/2024]: The team used the combination of Timelock and multi-sig wallet to mitigate the centralized risk.

The token contract address: 0x6E0609352D29de397a9D4dBEf217004740DF9A3C.

The timelock contract address: (0xc7d74e1905487ecb4bf657ee878ce339549e645f

The multi-sig wallet(Threshold: 2 out of 3 owner(s)): 0xe4e1153e0c6e9c51e86a58e4c8a36d8313863b3e:

- 0x68bCd7083973d0cADC78671df3Df6E327dD17AfA
- 0x480B49aC655967397B10f96654d3618Eb519797F
- 0x508ce057bf933391bbc7a7DEe0B960E2C75389A8

The team transferred the roles `DEFAULT_ADMIN_ROLE` and `SYSTEM_ROLE` of the token contract to the timelock address.

Transaction Hash: 0x60bec246fb7ed3b506cefa35bda57ee1ae7f0b0c3941e188e4bf673ea5191fbd

Upon deployment of the timelock contract, the `PROPOSER_ROLE` was transferred to the multi-signature wallet.

The team renounced admin role from multi-sig wallet via the transaction hash 0x1758c9f2b460487512e1c46bbe22c95db46015e14c34b0e4b5dcf23a646852ce.

BTB-07 | WITHDRAWAL CENTRALIZATION RISK

Category	Severity	Location	Status
Logical Issue, Centralization	**Major**	**contracts/BunToken.sol (bc129f42c0be8181a8d0ad76e3642744a79af629): 51, 160**	**Mitigated**

Description

In the contract `LockedToken` , the `donor` or the `system` role has the authority to withdraw the beneficiary's locked tokens from the contract.

```
 51      function revoke() public {
 52          require(revocable, "L: not revocable");
 53          require((msg.sender == donor) || (msg.sender == system),
"L: no permission");
 54
 55          uint256 amount = _token.balanceOf(address(this));
 56          require(amount > 0, "L: no tokens");
 57
 58          _token.safeTransfer(donor, amount);
 59          emit Revoke(donor, amount);
 60      }
```

```
 160      function revokeLockedToken(LockedToken _lockToken) public onlyRole(
SYSTEM_ROLE) {
 161          _lockToken.revoke();
 162      }
```

Any compromise to the account may allow a hacker to take advantage of this authority and withdraw the beneficiary's locked tokens.

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

[Boundless Team, 06/17/2024]: The team acknowledged the finding and we will use multi-sig contract wallet for all operations.

[CertiK, 06/18/2024]: The team has yet to address the centralization-related risks. CertiK strongly encourages the project team to periodically revisit the private key security management of all addresses related to privileged roles.

[CertiK, 07/11/2024]: The team used the combination of Timelock and multi-sig wallet to mitigate the centralized risk.

The token contract address: 0xF3A1BF85fc8D739c343D75f9a998955c234743C4.

The timelock contract address: (0xc7d74e1905487ecb4bf657ee878ce339549e645f

The multi-sig wallet(Threshold: 2 out of 3 owner(s)): 0xe4e1153e0c6e9c51e86a58e4c8a36d8313863b3e:

- 0x68bCd7083973d0cADC78671df3Df6E327dD17AfA
- 0x480B49aC655967397B10f96654d3618Eb519797F
- 0x508ce057bf933391bbc7a7DEe0B960E2C75389A8

The team transferred the roles `DEFAULT_ADMIN_ROLE` and `SYSTEM_ROLE` of the token contract to the timelock address.

Transaction Hash: 0x3a3034a8162fdd29664f66f4ab55c24b9e485217eee4c569c7d1097538dd5bdf

Upon deployment of the timelock contract, the `PROPOSER_ROLE` was transferred to the multi-signature wallet.

The team renounced admin role from multi-sig wallet via the transaction hash 0x212d935485f7bb506af5d617ebcf172b1e8c79200221e360ec8f7050b4eb6b63.

[CertiK, 07/15/2024]: The team used the combination of Timelock and multi-sig wallet to mitigate the centralized risk.

The token contract address: 0x6E0609352D29de397a9D4dBEf217004740DF9A3C.

The timelock contract address: (0xc7d74e1905487ecb4bf657ee878ce339549e645f

The multi-sig wallet(Threshold: 2 out of 3 owner(s)): 0xe4e1153e0c6e9c51e86a58e4c8a36d8313863b3e:

- 0x68bCd7083973d0cADC78671df3Df6E327dD17AfA
- 0x480B49aC655967397B10f96654d3618Eb519797F
- 0x508ce057bf933391bbc7a7DEe0B960E2C75389A8

The team transferred the roles `DEFAULT_ADMIN_ROLE` and `SYSTEM_ROLE` of the token contract to the timelock address.

Transaction Hash: 0x60bec246fb7ed3b506cefa35bda57ee1ae7f0b0c3941e188e4bf673ea5191fbd

Upon deployment of the timelock contract, the `PROPOSER_ROLE` was transferred to the multi-signature wallet.

The team renounced admin role from multi-sig wallet via the transaction hash 0x1758c9f2b460487512e1c46bbe22c95db46015e14c34b0e4b5dcf23a646852ce.

OPTIMIZATIONS | BOUNDLESS NETWORK TOKEN - AUDIT

ID	Title	Category	Severity	Status
BTB-01	Inefficient Memory Parameter	Inconsistency	Optimization	Resolved
BTB-05	Lack Of Balance And Allowance Check Before Batch Token Distribution	Logical Issue	Optimization	Acknowledged
BTB-06	Potential Out-Of-Gas Exception	Logical Issue	Optimization	Acknowledged

BTB-01 | INEFFICIENT MEMORY PARAMETER

Category	Severity	Location	Status
Inconsistency	Optimization	contracts/BunToken.sol (bc129f42c0be8181a8d0ad76e3642744a79af629): 116, 116, 124, 124, 124	Resolved

Description

One or more parameters with `memory` data location are never modified in their functions and those functions are never called internally within the contract. Thus, their data location can be changed to `calldata` to avoid the gas consumption copying from calldata to memory.

```
116     function batchTransfers(address[] memory recipients, uint256[] memory
amount) public returns (bool) {
```

`batchTransfers` has memory location parameters: `recipients`, `amount`.

```
124     function batchTransferFroms(address[] memory senders, address[] memory
recipients, uint256[] memory amount) public returns (bool) {
```

`batchTransferFroms` has memory location parameters: `senders`, `recipients`, `amount`.

Recommendation

We recommend changing the parameter's data location to `calldata` to save gas.

Alleviation

[Boundless Team, 06/17/2024]: The team resolved this issue at commit: 59ccdf9931de5fe3c7bb4fa4ee1272742fc48be8.

BTB-05 LACK OF BALANCE AND ALLOWANCE CHECK BEFORE BATCH TOKEN DISTRIBUTION

Category	Severity	Location	Status
Logical Issue	Optimization	contracts/BunToken.sol (bc129f42c0be8181a8d0ad76e3642744a79af629): 127	Acknowledged

Description

The root cause of this issue lies in the lack of sufficient balance and allowance checks before invoking the token transfer function.

This oversight results in reverting the transfer method, particularly impacting batch token transfers where the failure of a single transfer leads to the entire batch transfer failing. Importantly, any gas consumed during the failed transfers will not be refunded to the caller. For instance, in an extreme scenario, the failure of the 10,000th transfer would cause the rollback of the preceding 9,999 transfers, with the consumed gas irretrievably lost.

Recommendation

We recommend implementing the following mechanism to mitigate potential gas loss and enhance contract robustness:

1. Ensure thorough balance and authorization checks before initiating token transfers.
2. Consider implementing batch transfer mechanisms that allow for error recovery and partial execution, minimizing the impact of individual transfer failures on the overall transaction.

Alleviation

[Boundless Team, 06/17/2024]: The batch series of functions were created to minimize gas costs in case of operational airdrops.

What you pointed out is correct, but it does not happen often because it is basically checked in advance on the offchain before executing the transaction. In order to modify that part, we need to run a loop for the length of the array in advance and check all balances and allowances. This will likely increase the gas cost for most successful transactions. Therefore, it seems right to proceed as is.

BTB-06 | POTENTIAL OUT-OF-GAS EXCEPTION

Category	Severity	Location	Status
Logical Issue	Optimization	contracts/BunToken.sol (bc129f42c0be8181a8d0ad76e3642744a79af629): 118, 126	Acknowledged

Description

When a loop allows an arbitrary number of iterations or accesses state variables in its body, the function may run out of gas and revert the transaction.

```
118         for (uint256 i = 0; i < recipients.length; i++) {
```

Function `BunToken.batchTransfers` contains a loop and its loop condition depends on parameters: `recipients`.

```
126         for (uint256 i = 0; i < senders.length; i++) {
```

Function `BunToken.batchTransferFroms` contains a loop and its loop condition depends on parameters: `senders`.

Scenario

The number of transfers is limited to a maximum of 100

Recommendation

It is recommended to either 1) place limitations on the loop's bounds or 2) optimize the loop.

Alleviation

[Boundless Team, 06/17/2024]: The team resolved the issue by restricting the number of transfers to a maximum of 100. The changes were reflected in the commit c50928da7d5dfca468b65ea65a38f8e7cca64887.

[CertiK, 07/15/2024]: The team reverted the code changes and redeployed the contract at the address 0x6E0609352D29de397a9D4dBEf217004740DF9A3C.

[Boundless Team, 07/15/2024]: The code limiting 100 internal transactions in batch functions was removed, because it is ambiguous to the caller side. Because the primary goal is to reduce costs, we believe it would be better to perform the necessary work off-chain before making an on-chain call.

FORMAL VERIFICATION | BOUNDLESS NETWORK TOKEN - 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 formal verification to prove that important functions in the smart contracts adhere to their expected behaviors.

Considered Functions And Scope

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

Verification of Pausable ERC-20 Compliance

We verified properties of the public interface of those token contracts that implement the pausable 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-balanceof-correct-value	`balanceOf` Returns the Correct Value
erc20-transferfrom-false	If `transferFrom` Returns `false`, the Contract's State Is Unchanged
erc20-approve-correct-amount	`approve` Updates the Approval Mapping Correctly
erc20-transferfrom-never-return-false	`transferFrom` Never Returns `false`
erc20-approve-succeed-normal	`approve` Succeeds for Valid Inputs
erc20pausable-transferfrom-revert-paused	`transferFrom` Fails for a Paused Contract
erc20-totalsupply-succeed-always	`totalSupply` Always Succeeds
erc20-allowance-correct-value	`allowance` Returns Correct Value
erc20-transfer-never-return-false	`transfer` Never Returns `false`
erc20-transferfrom-revert-zero-argument	`transferFrom` Fails for Transfers with Zero Address Arguments

Property Name	Title
erc20-transfer-false	If `transfer` Returns `false`, the Contract State Is Not Changed
erc20-transfer-revert-zero	`transfer` Prevents Transfers to the Zero Address
erc20-allowance-change-state	`allowance` Does Not Change the Contract's State
erc20-transferfrom-fail-exceed-allowance	`transferFrom` Fails if the Requested Amount Exceeds the Available Allowance
erc20-transferfrom-fail-exceed-balance	`transferFrom` Fails if the Requested Amount Exceeds the Available Balance
erc20-transfer-exceed-balance	`transfer` Fails if Requested Amount Exceeds Available Balance
erc20-balanceof-change-state	`balanceOf` Does Not Change the Contract's State
erc20-transfer-correct-amount	`transfer` Transfers the Correct Amount in Transfers
erc20-totalsupply-change-state	`totalSupply` Does Not Change the Contract's State
erc20-transferfrom-correct-amount	`transferFrom` Transfers the Correct Amount in Transfers
erc20-transferfrom-correct-allowance	`transferFrom` Updated the Allowance Correctly
erc20-transferfrom-fail-recipient-overflow	`transferFrom` Prevents Overflows in the Recipient's Balance
erc20-transfer-recipient-overflow	`transfer` Prevents Overflows in the Recipient's Balance
erc20-approve-false	If `approve` Returns `false`, the Contract's State Is Unchanged
erc20pausable-transfer-revert-paused	`transfer` Fails for a Paused Contract
erc20-approve-revert-zero	`approve` Prevents Approvals For the Zero Address
erc20-allowance-succeed-always	`allowance` Always Succeeds
erc20-approve-never-return-false	`approve` Never Returns `false`
erc20-balanceof-succeed-always	`balanceOf` Always Succeeds
erc20-totalsupply-correct-value	`totalSupply` Returns the Value of the Corresponding State Variable

Verification Results

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

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

Detailed Results For Contract BunToken (contracts/BunToken.sol) In Commit bc129f42c0be8181a8d0ad76e3642744a79af629

Verification of Pausable ERC-20 Compliance

Detailed Results for Function `balanceOf`

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

Detailed Results for Function `transferFrom`

Property Name	Final Result	Remarks
erc20-transferfrom-false	True	
erc20-transferfrom-never-return-false	True	
erc20pausable-transferfrom-revert-paused	True	
erc20-transferfrom-revert-zero-argument	True	
erc20-transferfrom-fail-exceed-allowance	True	
erc20-transferfrom-fail-exceed-balance	True	
erc20-transferfrom-correct-amount	True	
erc20-transferfrom-correct-allowance	True	
erc20-transferfrom-fail-recipient-overflow	Inconclusive	

Detailed Results for Function `approve`

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

Detailed Results for Function `totalSupply`

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

Detailed Results for Function `allowance`

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

Detailed Results for Function `transfer`

Property Name	Final Result	Remarks
erc20-transfer-never-return-false	True	
erc20-transfer-false	True	
erc20-transfer-revert-zero	True	
erc20-transfer-exceed-balance	True	
erc20-transfer-correct-amount	True	
erc20-transfer-recipient-overflow	Inconclusive	
erc20pausable-transfer-revert-paused	True	

APPENDIX | BOUNDLESS NETWORK TOKEN - AUDIT

Finding Categories

Categories	Description
Inconsistency	Inconsistency findings refer to different parts of code that are not consistent or code that does not behave according to its specification.
Logical Issue	Logical Issue findings indicate general implementation issues related to the program logic.
Centralization	Centralization findings detail the design choices of designating privileged roles or other centralized controls over the code.

Checksum Calculation Method

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

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

Details on Formal Verification

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

The following assumptions and simplifications apply to our model:

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

Formalism for property specifications

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

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

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

Description of the Analyzed ERC-20-Pausable Properties

Properties related to function `balanceOf`

erc20-balanceof-change-state

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

Specification:

```
assignable \nothing;
```

erc20-balanceof-correct-value

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

Specification:

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

erc20-balanceof-succeed-always

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

Specification:

```
reverts_only_when false;
```

Properties related to function `transferFrom`

erc20-transferfrom-correct-allowance

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

Specification:

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

erc20-transferfrom-correct-amount

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

Specification:

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

erc20-transferfrom-fail-exceed-allowance

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

Specification:

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

erc20-transferfrom-fail-exceed-balance

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

Specification:

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

erc20-transferfrom-fail-recipient-overflow

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

Specification:

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

erc20-transferfrom-false

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

Specification:

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

erc20-transferfrom-never-return-false

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

Specification:

```
ensures \result;
```

erc20-transferfrom-revert-zero-argument

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

Specification:

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

erc20pausable-transferfrom-revert-paused

Any call of the form `transferFrom(from, dest, amount)` must fail for a paused contract.

Specification:

```
reverts_when paused();
```

Properties related to function `approve`

erc20-approve-correct-amount

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

Specification:

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

erc20-approve-false

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

Specification:

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

erc20-approve-never-return-false

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

Specification:

```
ensures \result;
```

erc20-approve-revert-zero

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

Specification:

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

erc20-approve-succeed-normal

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

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

Specification:

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

Properties related to function `totalSupply`

erc20-totalsupply-change-state

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

Specification:

```
assignable \nothing;
```

erc20-totalsupply-correct-value

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

Specification:

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

erc20-totalsupply-succeed-always

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

Specification:

```
reverts_only_when false;
```

Properties related to function `allowance`

erc20-allowance-change-state

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

Specification:

```
assignable \nothing;
```

erc20-allowance-correct-value

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

Specification:

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

erc20-allowance-succeed-always

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

Specification:

```
reverts_only_when false;
```

Properties related to function `transfer`

erc20-transfer-correct-amount

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

Specification:

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

erc20-transfer-exceed-balance

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

Specification:

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

erc20-transfer-false

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

Specification:

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

erc20-transfer-never-return-false

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

Specification:

```
ensures \result;
```

erc20-transfer-recipient-overflow

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

Specification:

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

erc20-transfer-revert-zero

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

Specification:

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

erc20pausable-transfer-revert-paused

Any invocation of `transfer(recipient, amount)` must fail if the contract is paused.

Specification:

```
reverts_when paused();
```

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.