CERTIK

EstateX

Security Assessment

CertiK Assessed on Jun 13th, 2025

CertiK Assessed on Jun 13th, 2025

EstateX

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

Executive Summary

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

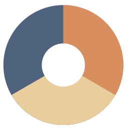
LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 06/13/2025	N/A

CODEBASE	COMMITS
base	0xc684edcb8b31f8960da6a59ac0898904107d7bf7
update_20250603	0x1b621d4d1f52e2487f92ee201cfb78e47458aaef
update_20250612	0x6a72d3A87f97a0fEE2c2ee4233BdAEBc32813D7a
View All in Codebase Page	View All in Codebase Page

Highlighted Centralization Risks

ⓘ Privileged role can remove users' tokens

Vulnerability Summary



9	7	0	2	0
Total Findings	Resolved	Partially Resolved	Acknowledged	Declined

🟧 0	Centralization		Centralization findings highlight privileged roles & functions and their capabilities, or instances where the project takes custody of users' assets.
🟥 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.
🟧 3	Major	1 Resolved, 2 Acknowledged	Major risks may include logical errors that, under specific circumstances, could result in fund losses or loss of project control.
🟨 0	Medium		Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.

🟨 3	Minor	3 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.
🟦 3	Informational	3 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 | ESTATEX

CODEBASE | ESTATEX

Repository

base

update_20250603

update_20250612

Commit

0xc684edcb8b31f8960da6a59ac0898904107d7bf7

0x1b621d4d1f52e2487f92ee201cfb78e47458aaef

0x6a72d3A87f97a0fEE2c2ee4233BdAEBc32813D7a

AUDIT SCOPE | ESTATEX

10 files audited ● 1 file with Acknowledged findings ● 6 files with Resolved findings ● 3 files without findings

ID	Repo	File	SHA256 Checksum
● EXC	mainnet	EstateX.sol	7f0d0ebe3b99be20e1e41152a3e50adfc8d7067006e80bcf1334bb9bf28a4808
● CON	mainnet	Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a
● ERC	mainnet	ERC20.sol	4eee086af7417003f5b7f5f26e7640ad63b0844efc18e45639e7d01cbeadb4a0
● IER	mainnet	IERC20.sol	94f23e4af51a18c2269b355b8c7cf4db8003d075c9c541019eb8dcf4122864d5
● IEC	mainnet	IERC20Metadata.sol	b10e2f8bcc3ed53a5d9a82a29b1ad3209225331bb4de4a0459862a762cf83a1a
● OWN	mainnet	Ownable.sol	661fd94b9274938bdc20e67a17c9eb5559ab2d75db6e5241bc7b58836b15c971
● EXS	testnet	contracts/EstateX.sol	bf2931f05d5a0c716acd98272d6250f87f5c8446116fec80742a9cf5b0c63dc7
● EXB	testnet	contracts/EstateX.sol	fe3249b76037262db7597da9dbaa5fad272bcffed5e3bcc25902b4b83d0d7548
● EXT	testnet	contracts/EstateX.sol	7e858adbd4fed731341d724ce54c4247eb0a0232f895767fa9aab5d4db3a9083
● EXE	testnet	contracts/EstateX.sol	874d9084a25309241352f0f22bad987e1b056e4835434a7a0013342721d16590

APPROACH & METHODS | ESTATEX

This report has been prepared for EstateX to discover issues and vulnerabilities in the source code of the EstateX 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.

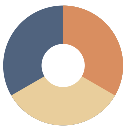
The auditing process pays special attention to the following considerations:

- 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.

FINDINGS | ESTATEX



9	0	0	3	0	3	3
Total Findings	Critical	Centralization	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for EstateX. Through this audit, we have uncovered 9 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
EXC-01	**Initial Token Distribution**	**Centralization**	**Major**	● **Acknowledged**
EXC-02	**Centralization Risks In EstateX.Sol**	**Centralization**	**Major**	● **Acknowledged**
GLOBAL-01	Improper Design Flow: TimelockController Should Be Standalone With Restricted Role Assignments	Design Issue	Major	● Resolved
EXC-04	Tax Precision Issue In `_transfer`	Coding Issue	Minor	● Resolved
EXC-05	Tax May Be Charged When Self Transfer	Design Issue	Minor	● Resolved
EXS-05	Lack Of Flexibility For Assigning Executors If List Is Empty	Coding Issue	Minor	● Resolved
0XC-01	Inconsistent Solidity Versions	Language Version	Informational	● Resolved
EXC-03	Usage Of Magic Numbers	Coding Issue	Informational	● Resolved
EXS-06	Potential Confusion In Balance Recalculation Due To Decimals Adjustment	Design Issue	Informational	● Resolved

EXC-01 | INITIAL TOKEN DISTRIBUTION

Category	Severity	Location	Status
Centralization	● **Major**	**EstateX.sol (base): 26**	● **Acknowledged**

Description

All of the `ESX` tokens are sent to the contract deployer or one or several externally-owned account (EOA) addresses. This is a centralization risk because the deployer or the owner(s) of the EOAs 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

`[EstateX, 02/04/2025]` : This was mitigated by the initial distribution being divided up during deployment time to individual wallet addresses. These deployment addresses will be MPA wallets using Gnosis Safe. New version is at https://testnet.bscscan.com/address/0x83f987FCC9762A9c6eC653f92Ea7fF47CCF5F626#code

`[CertiK, 02/05/2025]` : In the deployment at https://testnet.bscscan.com/address/0x83f987FCC9762A9c6eC653f92Ea7fF47CCF5F626. The tokens have been distributed to the following addresses:

- 0x13141D2B1a1Ce98b191895279DB9b27014a988fF: this is an EOA address which received 4,903,481,225 tokens, account for 77.8330% of total tokens
- 0x8ba4a6F0787902ec046f519dcbF471719623D9eE: this is an EOA address which received 618,288,136 tokens, account for 9.8141% of total tokens
- 0x12130a436fC9Ad68A927c5E82677939b124b5e5D: this is an EOA address which received 252,000,000 tokens, account for 4.0000% of total tokens
- 0x826ba16365bA370293EE774921C47D40d04b4266: this is an EOA address which received 242,730,639 tokens, account for 3.8529% of total tokens
- 0x42aA21d54cFa4Cd909f41316ad37a1822b5656d6: this is an EOA address which received 189,000,000 tokens, account for 3.0000% of total tokens

- 0xD76c8db3Ea33fCB605D5acA4415c21824AF33aa6: this is an EOA address which received 94,500,000 tokens, account for 1.5000% of total tokens

The finding status remains acknowledged according to the following facts:

1. No multi-sig wallet has been used to receive the initially distributed tokens
2. No public tokenomic plan for the token distribution

The finding and report will be revisited once more information is shared

`[CertiK, 06/03/2025]` : In the deployment at https://testnet.bscscan.com/address/0x1b621d4d1F52e2487f92Ee201cFB78E47458Aaef. The tokens have been distributed to the following addresses:

- 0x13141d2b1a1ce98b191895279db9b27014a988ff: this is an EOA address which received 3,178,141,225 tokens, account for 45.4020% of total tokens
- 0x212272cc5b7caa678164e7cd5ba9355c30453cb9: this is an EOA address which received 759,000,000 tokens, account for 10.8429% of total tokens
- 0xab33d7705743c5251878950a9791f553f113b61e: this is an EOA address which received 759,000,000 tokens, account for 10.8429% of total tokens
- 0x42aa21d54cfa4cd909f41316ad37a1822b5656d6: this is an EOA address which received 461,300,000 tokens, account for 6.5900% of total tokens
- 0x72f25a4d4b51f0a7ae9ff3e044ba2bc092e7b470: this is an EOA address which received 461,300,000 tokens, account for 6.5900% of total tokens
- 0x8ba4a6f0787902ec046f519dcbf471719623d9ee: this is an EOA address which received 311,720,339 tokens, account for 4.4531% of total tokens
- 0x18b9c1f2016c6efefea02048ad3de9bc20d845a5: this is an EOA address which received 311,720,339 tokens, account for 4.4531% of total tokens
- 0x826ba16365ba370293ee774921c47d40d04b4266: this is an EOA address which received 159,272,699 tokens, account for 2.2753% of total tokens
- 0x51a7ed09a4a29e1fa479508b0c469d17fcf2951c: this is an EOA address which received 159,272,699 tokens, account for 2.2753% of total tokens
- 0xa50d5780d58c92e2df5dad8a1c503434c6be7f19: this is an EOA address which received 159,272,699 tokens, account for 2.2753% of total tokens
- 0x12130a436fc9ad68a927c5e82677939b124b5e5d: this is an EOA address which received 105,000,000 tokens, account for 1.5000% of total tokens
- 0x691a7529c67ee735d2688052facd7226d63e4db1: this is an EOA address which received 105,000,000 tokens, account for 1.5000% of total tokens

The finding status remains acknowledged according to the following facts:

1. No multi-sig wallet has been used to receive the initially distributed tokens

2. No public tokenomic plan for the token distribution

`[EstateX, 06/12/2025]` : We have redeployed our contract to Base Sepolia testnet as we will be deploying to Base mainnet for our token launch. Can you please regenerate the report to align with our Base deployment and review our contracts on the Base Sepolia network.

- Timelock contract -> https://sepolia.basescan.org/address/0x552ce105c3d3442501D4176E17B2533916972d54#code
- Token contract -> https://sepolia.basescan.org/address/0x6a72d3A87f97a0fEE2c2ee4233BdAEBc32813D7a#code

`[CertiK, 06/03/2025]` : In the deployment at https://sepolia.basescan.org/address/0x6a72d3A87f97a0fEE2c2ee4233BdAEBc32813D7a. The tokens have been distributed to the following addresses:

- 0x13141d2b1a1ce98b191895279db9b27014a988ff: this is an EOA address which received 3,178,141,225 tokens, account for 45.4020% of total tokens
- 0x212272cc5b7caa678164e7cd5ba9355c30453cb9: this is an EOA address which received 759,000,000 tokens, account for 10.8429% of total tokens
- 0xab33d7705743c5251878950a9791f553f113b61e: this is an EOA address which received 759,000,000 tokens, account for 10.8429% of total tokens
- 0x42aa21d54cfa4cd909f41316ad37a1822b5656d6: this is an EOA address which received 461,300,000 tokens, account for 6.5900% of total tokens
- 0x72f25a4d4b51f0a7ae9ff3e044ba2bc092e7b470: this is an EOA address which received 461,300,000 tokens, account for 6.5900% of total tokens
- 0x8ba4a6f0787902ec046f519dcbf471719623d9ee: this is an EOA address which received 311,720,339 tokens, account for 4.4531% of total tokens
- 0x18b9c1f2016c6efefea02048ad3de9bc20d845a5: this is an EOA address which received 311,720,339 tokens, account for 4.4531% of total tokens
- 0x826ba16365ba370293ee774921c47d40d04b4266: this is an EOA address which received 159,272,699 tokens, account for 2.2753% of total tokens
- 0x51a7ed09a4a29e1fa479508b0c469d17fcf2951c: this is an EOA address which received 159,272,699 tokens, account for 2.2753% of total tokens
- 0xa50d5780d58c92e2df5dad8a1c503434c6be7f19: this is an EOA address which received 159,272,699 tokens, account for 2.2753% of total tokens
- 0x12130a436fc9ad68a927c5e82677939b124b5e5d: this is an EOA address which received 105,000,000 tokens, account for 1.5000% of total tokens
- 0x691a7529c67ee735d2688052facd7226d63e4db1: this is an EOA address which received 105,000,000 tokens, account for 1.5000% of total tokens
- 0x68E94FcA96536CE72c02AE1DE65E7E13BdB807F8: this is an EOA address which received 35,000,000 tokens, account for 0.5000% of total tokens

- 0xD76c8db3Ea33fCB605D5acA4415c21824AF33aa6: this is an EOA address which received 35,000,000 tokens, account for 0.5000% of total tokens

The finding status remains acknowledged according to the following facts:

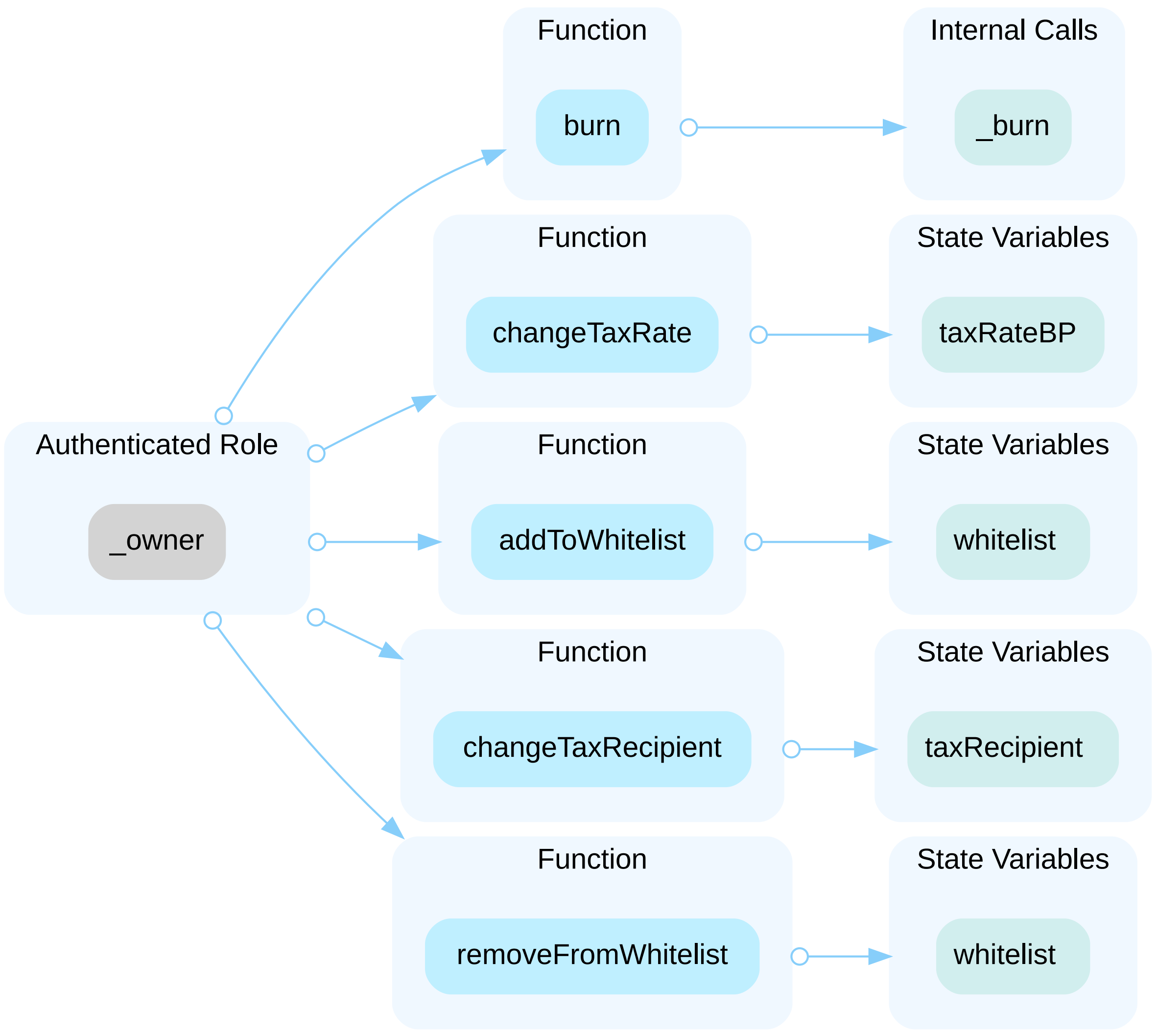
1. No multi-sig wallet has been used to receive the initially distributed tokens
2. No public tokenomic plan for the token distribution

EXC-02 | CENTRALIZATION RISKS IN ESTATEX.SOL

Category	Severity	Location	Status
Centralization	● Major	EstateX.sol (base): 39, 44, 50, 56, 62	● Acknowledged

Description

In the contract `EstateX`, 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 and burn tokens from the owner's account, change the tax rate, add or remove addresses from the whitelist, and change the tax recipient address.



- **burn(uint256 amount)**: Burns a specific amount of tokens from the owner's balance. Privileged role: `onlyOwner`.
- **changeTaxRecipient(address newTaxRecipient)**: Updates the address that receives transfer taxes. Privileged role: `onlyOwner`.

- **changeTaxRate(uint256 newTaxRateBP)**: Changes the tax rate (in basis points) applied to transfers. Privileged role: `onlyOwner` .
- **addToWhitelist(address account)**: Adds an address to the whitelist, exempting it from the transfer tax. Privileged role: `onlyOwner` .
- **removeFromWhitelist(address account)**: Removes an address from the whitelist, making it subject to the transfer tax. Privileged role: `onlyOwner` .

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

`[EstateX, 02/04/2025]` : This has been mitigated via the use of an MPA wallet as the contract owner. All admin functions will require 3/5 multisig to execute. New contract is at https://testnet.bscscan.com/address/0x83f987FCC9762A9c6eC653f92Ea7fF47CCF5F626#code

`[CertiK, 02/05/2025]` : In the deployment at https://testnet.bscscan.com/address/0x83f987FCC9762A9c6eC653f92Ea7fF47CCF5F626. The owner privilege was granted to 0x13141D2B1a1Ce98b191895279DB9b27014a988fF which is an EoA account.

The finding status remains acknowledged according to the following facts:

1. No Time-lock with reasonable latency, e.g., 48 hours, for awareness of privileged operations
2. No multi-sig wallet has been used to manage the owner privilege
3. No medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

The finding and report will be revisited once more information is shared

`[EstateX, 02/09/2025]` : We have added timelock functionality to the contract to allow for 48 hours to elapse before certain administrative functions can be executed such as "burn", "addTimelockProposer", "removeTimelockProposer", "addTimelockExecutor", and "removeTimelockExecutor". When scheduled by a proposer, the contract emits an event to notify users. The events are "TimelockOperationScheduled" and "BurnScheduled". All functionality related to transfer tax has been removed. We are working on creating MPA wallets for mainnet deployment. We would appreciate it if you could please review our timelock code and let us know if this new code in combination with MPA wallets for deployment would satisfy this condition. Thank you.

The new code is located on BSC Testnet at https://testnet.bscscan.com/address/0x5ff605042cd48Cf72d8b3ab1fCE9Ee7424423C3d#code

`[EstateX, 06/12/2025]` : We have redeployed our contract to Base Sepolia testnet as we will be deploying to Base mainnet for our token launch. Can you please regenerate the report to align with our Base deployment and review our contracts on the Base Sepolia network.

- Timelock contract -> https://sepolia.basescan.org/address/0x552ce105c3d3442501D4176E17B2533916972d54#code
- Token contract -> https://sepolia.basescan.org/address/0x6a72d3A87f97a0fEE2c2ee4233BdAEBc32813D7a#code

`[CertiK, 06/12/2025]` : In the deployment at https://sepolia.basescan.org/address/0x6a72d3A87f97a0fEE2c2ee4233BdAEBc32813D7a. The owner privilege was granted to 0x13141D2B1a1Ce98b191895279DB9b27014a988fF which is an EoA account.

The finding status remains acknowledged according to the following facts:

1. No multi-sig wallet has been used to manage the owner privilege
2. No medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

The finding and report will be revisited once more information is shared

GLOBAL-01 | IMPROPER DESIGN FLOW: TIMELOCKCONTROLLER SHOULD BE STANDALONE WITH RESTRICTED ROLE ASSIGNMENTS

Category	Severity	Location	Status
Design Issue	● Major		● Resolved

Description

The issue lies in the design flow of the contract. The `TimelockController` should be deployed separately from the Token contract, as best practice dictates that it operates as a standalone contract. The `EXECUTOR_ROLE` should only be granted to a multisig wallet address, and the `PROPOSER_ROLE` should be assigned to a limited group of administrators, without any functionality to grant additional roles. This design ensures proper separation of concerns and minimizes the risk of unauthorized role assignments or excessive privileges, which could lead to potential security vulnerabilities.

Recommendation

We recommend redesigning the contract to deploy the `TimelockController` separately from the Token, assigning the `EXECUTOR_ROLE` to a multisig wallet and the `PROPOSER_ROLE` to a limited group of admins, without granting role assignment capabilities.

Alleviation

[**Certik** 03 Mar 2025]: All changes implemented in the new contract.

EXC-04 | TAX PRECISION ISSUE IN `_transfer`

Category	Severity	Location	Status
Coding Issue	● Minor	EstateX.sol (base): 72~77	● Resolved

Description

The `require(amount >= tax, "Transactional amount not enough to cover tax.");` check ensures that amount covers the tax, but this doesn't guarantee that there won't be issues for very small transactions. If `amount < 10000 / taxRateBP` , integer truncation might cause unexpected behavior, resulting in `remainingAmount` = 0.

For very small amount values, the tax may become larger than amount due to integer precision loss, leading to failed transfers or unexpected results.

Recommendation

Ensure that the minimum amount is large enough to avoid integer truncation issues. Add checks to prevent transfers of amounts that would result in remainingAmount = 0.

Alleviation

`[EstateX, 02/04/2025]` : Tax feature has been removed. New contract is at https://testnet.bscscan.com/address/0x83f987FCC9762A9c6eC653f92Ea7fF47CCF5F626#code

`[CertiK, 02/05/2025]` : The finding has been resolved in the deployment https://testnet.bscscan.com/address/0x83f987FCC9762A9c6eC653f92Ea7fF47CCF5F626

EXC-05 | TAX MAY BE CHARGED WHEN SELF TRANSFER

Category	Severity	Location	Status
Design Issue	● Minor	EstateX.sol (base): 68~80	● Resolved

Description

In the `_transfer` function, the tax is calculated as:

```
tax = (amount * taxRateBP) / 10000;
```

Even when `from` equals `to`, the contract still calculates and applies the tax. The sender might lose tokens to the `taxRecipient` during a self-transfer. This could confuse users, as self-transfers are typically expected to maintain the same token balance.

If the tax is applied to self-transfers, users may lose tokens unintentionally.

Recommendation

Add a check to skip tax calculation for self-transfers:

```
if (from == to) {
    super._transfer(from, to, amount);
    return;
}
```

Alleviation

`[EstateX, 02/04/2025]` : Tax feature has been removed. New contract is at https://testnet.bscscan.com/address/0x83f987FCC9762A9c6eC653f92Ea7fF47CCF5F626#code

`[CertiK, 02/05/2025]` : The finding has been resolved in the deployment https://testnet.bscscan.com/address/0x83f987FCC9762A9c6eC653f92Ea7fF47CCF5F626

EXS-05 | LACK OF FLEXIBILITY FOR ASSIGNING EXECUTORS IF LIST IS EMPTY

Category	Severity	Location	Status
Coding Issue	● Minor	contracts/EstateX.sol (update_20250214): 81~84	● Resolved

Description

If the `executor` s list passed to the constructor is empty, there is no way to assign `executors` later, which would render the contract non-functional. The contract relies on this list to initialize the `TimelockController` , and an empty list would result in an invalid state, preventing the contract from operating as intended. This lack of flexibility poses a potential issue if `executors` are not provided at deployment.

Recommendation

We recommend adding a mechanism to allow executors to be assigned or updated after contract deployment, ensuring the contract remains functional even if the executors list is initially empty. This provides flexibility and prevents the contract from being stuck in an invalid state.

Alleviation

[**Certik** 03 Mar 2025]: All changes implemented in the new contract.

0XC-01 | INCONSISTENT SOLIDITY VERSIONS

Category	Severity	Location	Status
Language Version	● Informational	Context.sol (base): 4; ERC20.sol (base): 4; EstateX.sol (base): 4; IERC20.sol (base): 4; IERC20Metadata.sol (base): 4; Ownable.sol (base): 4	● Resolved

Description

The codebase contains multiple Solidity versions, which can lead to unexpected behavior, potential vulnerabilities, difficulties in maintaining the code, and inconsistencies in the execution of the smart contract. Using different versions may also result in increased complexity during code auditing, as different security features and bug fixes are present in different versions of the compiler.

Versions used: `^0.8.0`, `0.8.14`

```
pragma solidity ^0.8.0;
```

`^0.8.0` is used in Ownable.sol file.

```
pragma solidity ^0.8.0;
```

```
pragma solidity 0.8.14;
```

`0.8.14` is used in EstateX.sol file.

```
pragma solidity 0.8.14;
```

Recommendation

It is recommended to standardize on a single, up-to-date Solidity version throughout the codebase to ensure consistent behavior, benefit from the latest security features, and improve maintainability.

Alleviation

`[EstateX, 02/04/2025]`: Fixed. New contract is at https://testnet.bscscan.com/address/0x83f987FCC9762A9c6eC653f92Ea7fF47CCF5F626#code

`[CertiK, 02/05/2025]`: The finding has been resolved in the deployment https://testnet.bscscan.com/address/0x83f987FCC9762A9c6eC653f92Ea7fF47CCF5F626

EXC-03 | USAGE OF MAGIC NUMBERS

Category	Severity	Location	Status
Coding Issue	● Informational	EstateX.sol (base): 24, 51, 73	● Resolved

Description

The contract contains "magic numbers" (hardcoded numeric values) without any explanation or constants to define their purpose. This reduces code readability and maintainability, making auditing harder and potentially hiding unintended logic or vulnerabilities.

Recommendation

We recommend to define all numeric values as named constants with descriptive names that explain their purpose.

Alleviation

`[EstateX, 02/04/2025]` : Fixed. New contract is at https://testnet.bscscan.com/address/0x83f987FCC9762A9c6eC653f92Ea7fF47CCF5F626#code

`[CertiK, 02/05/2025]` : The finding has been resolved in the deployment https://testnet.bscscan.com/address/0x83f987FCC9762A9c6eC653f92Ea7fF47CCF5F626

EXS-06 POTENTIAL CONFUSION IN BALANCE RECALCULATION DUE TO DECIMALS ADJUSTMENT

Category	Severity	Location	Status
Design Issue	● Informational	contracts/EstateX.sol (update_20250214): 37~43	● Resolved

Description

The issue in the contract is that all allocation balances, including `ESX_TOTAL_SUPPLY`, are being recalculated by multiplying with `10 ** decimals()`, which adjusts for token precision. While this is necessary for ERC-20 compliance, it can lead to confusion when interpreting the actual token distribution, as the real-world cost or intended allocation might not be immediately clear. This could cause miscalculations or misunderstandings, especially when comparing predefined values with the final adjusted balances.

Recommendation

We recommend clarifying the impact of `10 ** decimals()` in the allocation calculations by documenting it explicitly or defining pre-adjusted values. This prevents confusion and ensures accurate interpretation of real token costs.

Alleviation

[**Certik** 06 Mar 2025]: Previously, the total supply was calculated by modifying the `ESX_TOTAL_SUPPLY` variable based on the token's decimal places. Now, the logic has been separated into two distinct variables: `BASE_TOTAL_SUPPLY`, which holds the base value, and `ESX_TOTAL_SUPPLY`, which applies the scaling factor based on the decimals. Additionally, a comment has been added to explicitly describe the logic behind the operation, indicating that the decimal precision is set to `9`.

OPTIMIZATIONS | ESTATEX

ID	Title	Category	Severity	Status
EXS-01	Simplifying Total Supply Declaration For Better Readability	Code Optimization	Optimization	● Resolved
EXS-02	Redundant Require Check Due To Predefined Allocation Limits	Gas Optimization, Code Optimization	Optimization	● Resolved
EXS-03	Improving Readability By Using Time Units Instead Of Numeric Values	Code Optimization	Optimization	● Resolved
EXS-04	Redundant Existence Check In Role Assignment Function	Gas Optimization, Code Optimization	Optimization	● Resolved

EXS-01 | SIMPLIFYING TOTAL SUPPLY DECLARATION FOR BETTER READABILITY

Category	Severity	Location	Status
Code Optimization	● Optimization	contracts/EstateX.sol (update_20250214): 12~13	● Resolved

Description

The total supply calculation is unnecessarily complex, using a mathematical expression `(63 * BILLION / 10)` instead of directly assigning the intended value. While functionally correct, this approach reduces readability and may confuse developers reviewing the code. A simpler and more intuitive way to declare the total supply is to replace both lines with `uint256 private ESX_TOTAL_SUPPLY = 6_300_000_000;` , making the value immediately clear without requiring extra computation.

Recommendation

We recommend replacing `63 * BILLION / 10` with `6_300_000_000` for better readability and clarity. This simplifies the code, removes unnecessary computation, and makes the total supply immediately understandable.

Alleviation

[**Certik** 03 Mar 2025]: All changes implemented in the new contract.

EXS-02 REDUNDANT REQUIRE CHECK DUE TO PREDEFINED ALLOCATION LIMITS

Category	Severity	Location	Status
Gas Optimization, Code Optimization	● Optimization	contracts/EstateX.sol (update_20250214): 46, 48~58	● Resolved

Description

The `require` statement checking whether the sum of allocated balances exceeds `ESX_TOTAL_SUPPLY` is redundant.

```
require(devOps.balance + team.balance + strategicReserve.balance + marketing.balance
+ charity.balance <= ESX_TOTAL_SUPPLY, "Allocated funds exceeds total supply");
```

Moreover, `staking.balance` is not considered in this check.

Even if `staking.balance` is considered, each individual allocation has a predefined maximum percentage of the total supply. As a result, the maximum sum is always `(ESX_TOTAL_SUPPLY * (5 + 15 + 5 + 10 + 40 + 2) / 100)`, which equals `ESX_TOTAL_SUPPLY * 77 / 100`. This guarantees that the sum of the allocations will never exceed the total supply, making the require check unnecessary and adding computational overhead without providing any real benefit.

Recommendation

We recommend removing the redundant require statement.

Alleviation

[**Certik** 03 Mar 2025]: All changes implemented in the new contract.

EXS-03 | IMPROVING READABILITY BY USING TIME UNITS INSTEAD OF NUMERIC VALUES

Category	Severity	Location	Status
Code Optimization	● Optimization	contracts/EstateX.sol (update_20250214): 84, 104, 111, 118, 125, 132	● Resolved

Description

Using the value `172800` directly in the constructor, which represents a 48-hour delay, reduces readability and relies on comments for clarification. Instead of using a numeric value with an explanatory comment, it would be more readable to use the Solidity's built-in time units, such as `48 hours` , to make the code self-explanatory. This approach eliminates the need for additional comments and improves the clarity of the time duration being used, making the code more intuitive and maintainable.

Recommendation

We recommend replacing the numeric value `172800` with the time unit `48 hours` to improve readability and eliminate the need for comments.

Alleviation

[**Certik** 03 Mar 2025]: All changes implemented in the new contract.

EXS-04 | REDUNDANT EXISTENCE CHECK IN ROLE ASSIGNMENT FUNCTION

Category	Severity	Location	Status
Gas Optimization, Code Optimization	● Optimization	contracts/EstateX.sol (update_20250214): 16~17, 140~141, 153~154, 171~189	● Resolved

Description

The function `_existsInArray(_timelockProposers, account)` is being used to check if an account exists in the `_timelockProposers` array, but this approach is inefficient as it requires iterating through the entire array, leading to increased gas costs. A more optimal solution is to use `timelock.hasRole()`, which directly checks for the role assignment in a more efficient way, leveraging Solidity's mapping-based access control mechanism. By replacing `_existsInArray()` with `timelock.hasRole()`, the function `_existsInArray()` becomes redundant and can be removed, improving both contract efficiency and maintainability. Since there is no longer a need to keep track of `_timelockProposers` and `_timelockExecutors`, the function `_removeAddressFromArray()` can also be removed.

Recommendation

We recommend removing the redundant `_existsInArray()` and `_removeAddressFromArray()` functions. Replace `_existsInArray()` with `timelock.hasRole()` and eliminate the `_timelockProposers` and `_timelockExecutors` state variables.

Alleviation

[**Certik** 03 Mar 2025]: All changes implemented in the new contract.

FORMAL VERIFICATION | ESTATEX

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

Considered Functions And Scope

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

Verification of ERC-20 Compliance

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

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

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

Property Name	Title
erc20-transferfrom-correct-allowance	`transferFrom` Updated the Allowance Correctly
erc20-transferfrom-correct-amount	`transferFrom` Transfers the Correct Amount in Transfers
erc20-approve-false	If `approve` Returns `false`, the Contract's State Is Unchanged
erc20-balanceof-succeed-always	`balanceOf` Always Succeeds
erc20-totalsupply-succeed-always	`totalSupply` Always Succeeds
erc20-balanceof-correct-value	`balanceOf` Returns the Correct Value
erc20-approve-revert-zero	`approve` Prevents Approvals For the Zero Address
erc20-totalsupply-correct-value	`totalSupply` Returns the Value of the Corresponding State Variable
erc20-approve-succeed-normal	`approve` Succeeds for Valid Inputs
erc20-allowance-correct-value	`allowance` Returns Correct Value
erc20-transferfrom-false	If `transferFrom` Returns `false`, the Contract's State Is Unchanged

Property Name	Title
erc20-allowance-succeed-always	`allowance` Always Succeeds
erc20-approve-never-return-false	`approve` Never Returns `false`
erc20-approve-correct-amount	`approve` Updates the Approval Mapping Correctly
erc20-transfer-exceed-balance	`transfer` Fails if Requested Amount Exceeds Available Balance
erc20-transferfrom-never-return-false	`transferFrom` Never Returns `false`
erc20-transferfrom-revert-zero-argument	`transferFrom` Fails for Transfers with Zero Address Arguments
erc20-transfer-never-return-false	`transfer` Never Returns `false`
erc20-transferfrom-fail-exceed-allowance	`transferFrom` Fails if the Requested Amount Exceeds the Available Allowance
erc20-transfer-false	If `transfer` Returns `false` , the Contract State Is Not Changed
erc20-transferfrom-fail-exceed-balance	`transferFrom` Fails if the Requested Amount Exceeds the Available Balance
erc20-balanceof-change-state	`balanceOf` Does Not Change the Contract's State
erc20-transfer-correct-amount	`transfer` Transfers the Correct Amount in Transfers
erc20-transfer-revert-zero	`transfer` Prevents Transfers to the Zero Address
erc20-allowance-change-state	`allowance` Does Not Change the Contract's State
erc20-totalsupply-change-state	`totalSupply` Does Not Change the Contract's State
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

Verification Results

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

Detailed Results For Contract EstateX (contracts/EstateX.sol) In Commit 0xcf8ffbf8f337ae42c08c10c05f8c38c686dc8a18

Verification of ERC-20 Compliance

Detailed Results for Function `transferFrom`

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

Detailed Results for Function `approve`

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

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

Detailed Results for Function `transfer`

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

Detailed Results For Contract EstateX (contracts/EstateX.sol) In Commit 0x22baff866646da99de4f1b8b0d36f0d7fa640f71

Verification of ERC-20 Compliance

Detailed Results for Function `transfer`

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

Detailed Results for Function `totalSupply`

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

Detailed Results for Function `allowance`

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

Detailed Results for Function `balanceOf`

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

Detailed Results for Function `transferFrom`

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

Detailed Results for Function `approve`

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

Detailed Results For Contract EstateX (contracts/EstateX.sol) In Commit 0x7721b2a6b03c201e71becfcb79e82778b35c1836

Verification of ERC-20 Compliance

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 `approve`

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

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

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 `transfer`

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

Detailed Results For Contract ERC20 (ERC20.sol) In Commit 0xc684edcb8b31f8960da6a59ac0898904107d7bf7

Verification of ERC-20 Compliance

Detailed Results for Function `transferFrom`

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

Detailed Results for Function `allowance`

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

Detailed Results for Function `transfer`

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

Detailed Results for Function `totalSupply`

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

Detailed Results for Function `approve`

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

Detailed Results for Function `balanceOf`

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

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

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

Detailed Results For Contract EstateX (EstateX.sol) In Commit 0xc684edcb8b31f8960da6a59ac0898904107d7bf7

Verification of ERC-20 Compliance

Detailed Results for Function `allowance`

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

Detailed Results for Function `totalSupply`

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

Detailed Results for Function `transfer`

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

Detailed Results for Function `approve`

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

Detailed Results for Function `transferFrom`

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

Detailed Results for Function `balanceOf`

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

APPENDIX | ESTATEX

Finding Categories

Categories	Description
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.
Language Version	Language Version findings indicate that the code uses certain compiler versions or language features with known security issues.
Coding Issue	Coding Issue findings are about general code quality including, but not limited to, coding mistakes, compile errors, and performance issues.
Centralization	Centralization findings detail the design choices of designating privileged roles or other centralized controls over the code.
Design Issue	Design Issue findings indicate general issues at the design level beyond program logic that are not covered by other finding categories.

Checksum Calculation Method

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

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

Details on Formal Verification

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

The following assumptions and simplifications apply to our model:

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

Formalism for property specifications

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

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

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

Description of the Analyzed ERC-20 Properties

Properties related to function `transferFrom`

erc20-transferfrom-correct-allowance

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

Specification:

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

erc20-transferfrom-correct-amount

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

Specification:

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

erc20-transferfrom-fail-exceed-allowance

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

Specification:

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

erc20-transferfrom-fail-exceed-balance

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

Specification:

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

erc20-transferfrom-fail-recipient-overflow

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

Specification:

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

erc20-transferfrom-false

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

Specification:

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

erc20-transferfrom-never-return-false

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

Specification:

```
ensures \result;
```

erc20-transferfrom-revert-zero-argument

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

Specification:

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

Properties related to function `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 `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 `totalSupply`

erc20-totalsupply-change-state

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

Specification:

```
assignable \nothing;
```

erc20-totalsupply-change-state

The `totalSupply` function in contract ERC20 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 EstateX.

Specification:

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

erc20-totalsupply-correct-value

The `totalSupply` function must return the value that is held in the corresponding state variable of contract ERC20.

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 `EstateX` fails by returning `false`, it must undo all state changes it incurred before returning to the caller.

Specification:

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

erc20-transfer-false

If the `transfer` function in contract `ERC20` 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;
```

DISCLAIMER | CERTIK

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.

Elevating Your Entire Web3 Journey

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.