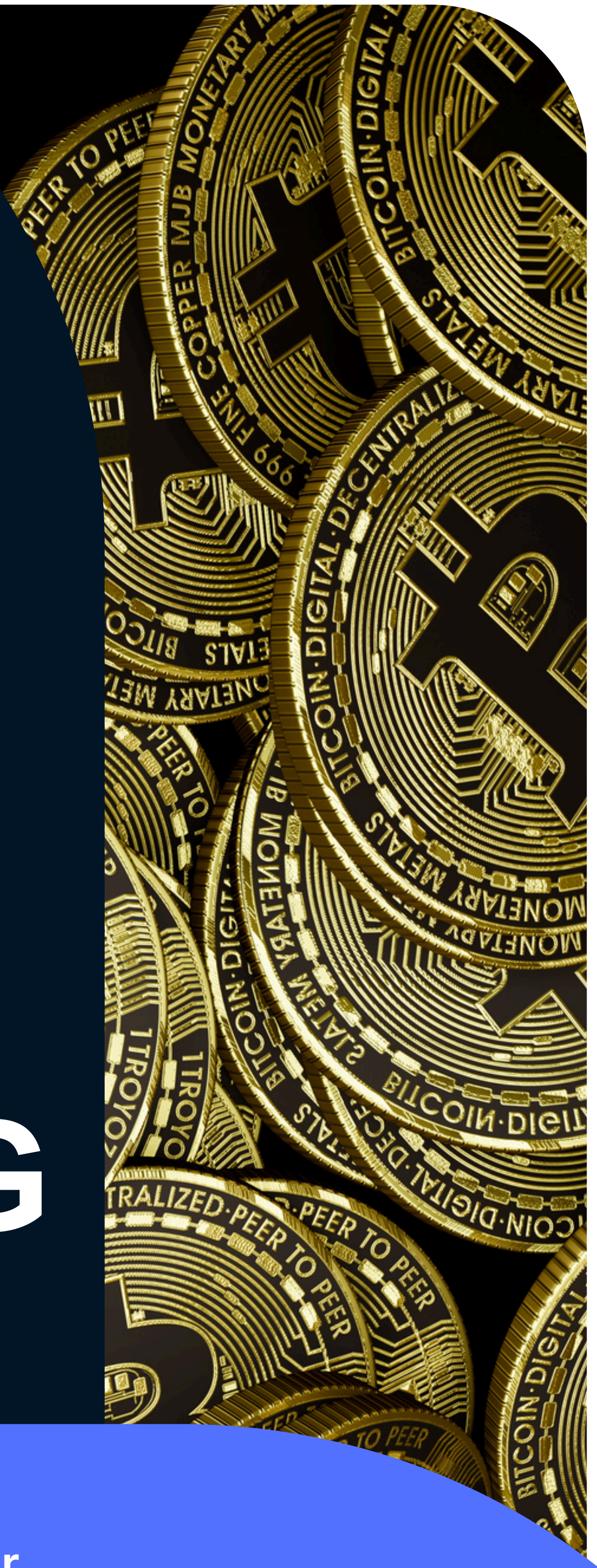
AUDITBLOCK
Premium

AUDIT REPORT

UCBI BANKING



v0.4.24+commit.e67f014

0x2adba23Cf1252dE095aCEd801e758b369EC10426

Disclaimer

UCBI BANKING

UCBI Banking is a pioneering initiative that serves as a blockchain data bank, integrating cryptocurrency rates with international blockchain services. This innovative concept aims to enhance financial interactions by creating synergies among trading partners worldwide, thus facilitating seamless transactions and fostering global economic collaboration.

Key Features of UCBI Banking

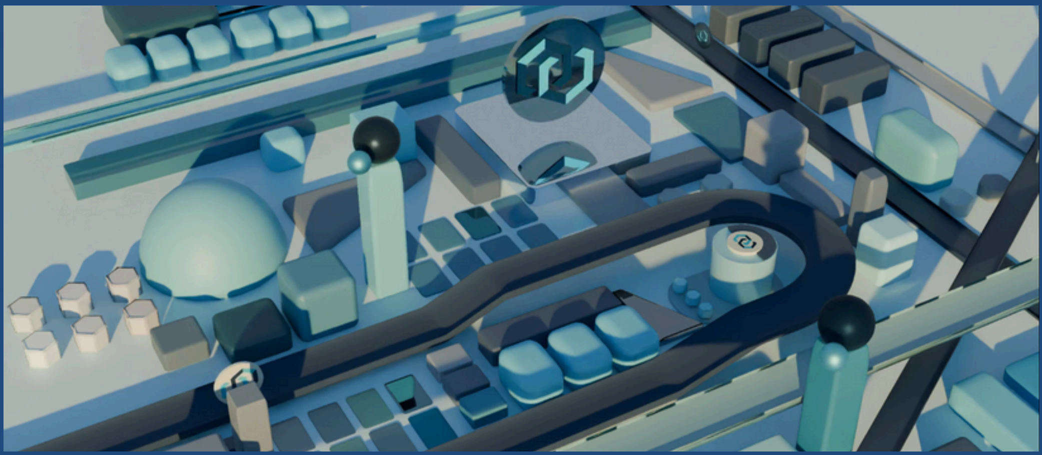
01 INTEGRATION OF CRYPTOCURRENCY AND TRADITIONAL BANKING

02 GLOBAL REACH AND ACCESSIBILITY

03 ENHANCED SECURITY AND COMPLIANCE

04 INNOVATIVE FINANCIAL PRODUCTS

SOCIAL LINKS



AUDITBLOCK
Premium

INFORMALITIES

01. High:

High-severity vulnerabilities can compromise your smart contract's security and functionality. Prioritize fixing these critical issues before deploying to a live network.

03. Low:

Low-severity issues might cause minor problems or are simply warnings that can be addressed later.

02. Medium:

Medium-severity issues can lead to potential problems in your smart contract. Addressing these code errors and deficiencies is essential for optimal performance and security.

04. Informational:

These low-severity issues are suggestions for improvement, such as documentation errors or cosmetic changes. While not critical, addressing them can enhance the overall quality and user experience.

TECHNIQUES AND METHODS

The overall quality of code.

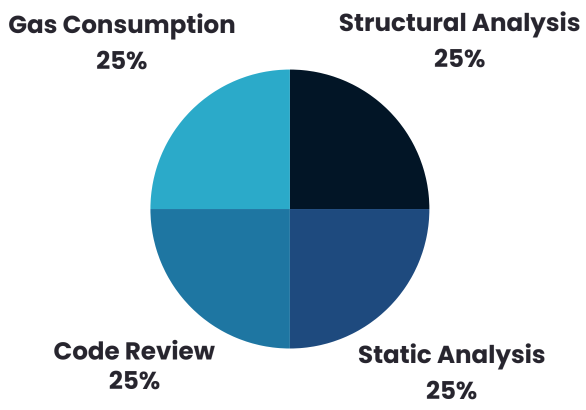
- Use of best practices.
- Code documentation and comments match logic and expected behavior.
- Token distribution and calculations are as per the intended behavior mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrance and other vulnerabilities.

01 **STRUCTURAL ANALYSIS**

02 **STATIC ANALYSIS:**

04 **GAS CONSUMPTION:**

03 **CODE REVIEW / MANUAL ANALYSIS:**



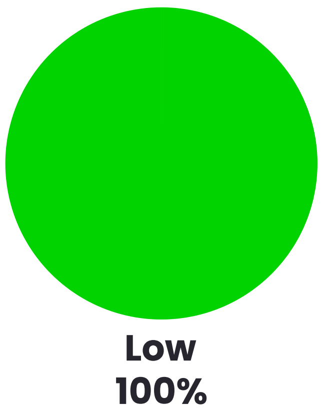
TOOLS AND PLATFORMS USED FOR AUDIT:

To ensure a rigorous and thorough audit of the smart contracts, the following tools were employed:

- **Development Environments:**
  - Remix IDE and Truffle Suite were used to facilitate efficient contract development, testing, and debugging.
- **Static Analysis Tools:**
  - Solhint, Mythril, and Slither were utilized to identify potential vulnerabilities, coding errors, and security weaknesses in the smart contract code.
- **Code Analysis Tools:**
  - Solidity Statistics was employed to analyze code complexity, maintainability, and potential optimization opportunities

ABSTRACT

By leveraging this robust toolset, the audit aimed to identify and mitigate potential risks, ensuring the security and reliability of the smart contract system.



●	High
●	Medium
●	Low
●	Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	7
Partially Resolved Issues	0	0	0	0
Resolved Issues	1	0	5	0

PHASE 1

PROJECT - ASCENT YIELD

```
INFO:Detectors
Version constraint ^0.4.23 contains known severe issues
(https://solidity.readthedocs.io/en/latest/bugs.html)
	- DirtyBytesArrayToStorage
	- ABIDecodeTwoDimensionalArrayMemory
	- KeccakCaching
	- EmptyByteArrayCopy
	- DynamicArrayCleanup
	- ImplicitConstructorCallvalueCheck
	- TupleAssignmentMultiStackSlotComponents
	- MemoryArrayCreationOverflow
	- privateCanBeOverridden
	- SignedArrayStorageCopy
	- ABIEncoderV2StorageArrayWithMultiSlotElement
	- DynamicConstructorArgumentsClippedABIV2
	- UninitializedFunctionPointerInConstructor_0.4.x
	- IncorrectEventSignatureInLibraries_0.4.x
	- ABIEncoderV2PackedStorage_0.4.x
	- ExpExponentCleanup
	- EventStructWrongData.
It is used by:
	- ^0.4.23 (Code.sol#5)
solc-0.4.23 is an outdated solc version. Use a more recent version (at least 0.8.0), if
possible.
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity
INFO:Detectors:
Parameter BasicToken.allowAddress(address,bool)._addr (Code.sol#117) is not in
mixedCase
Parameter BasicToken.allowAddress(address,bool)._allowed (Code.sol#117) is not in
mixedCase
Parameter BasicToken.lockAddress(address,bool)._addr (Code.sol#122) is not in
mixedCase
Parameter BasicToken.lockAddress(address,bool)._locked (Code.sol#122) is not in
mixedCase
```

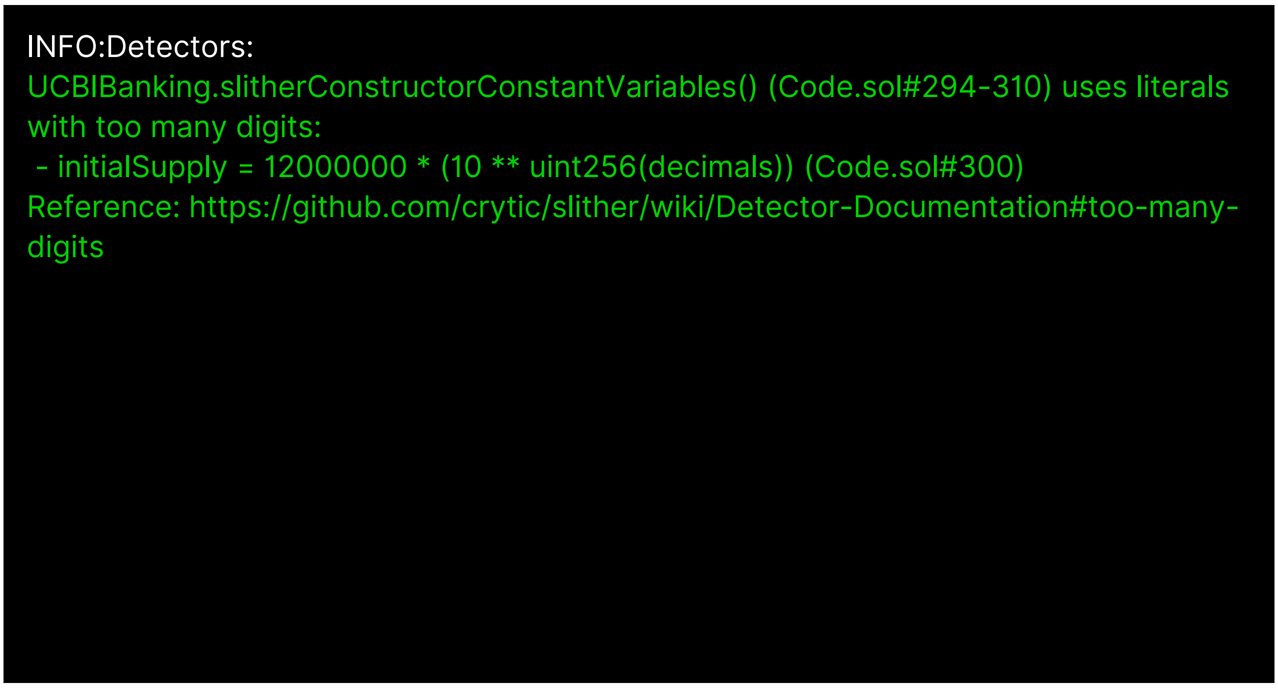
PHASE 1

PROJECT - ASCENT YIELD

```
Parameter BasicToken.setLocked(bool)._locked (Code.sol#127) is not in mixedCase
Parameter BasicToken.canTransfer(address)._addr (Code.sol#131) is not in
mixedCase
Parameter BasicToken.transfer(address,uint256)._to (Code.sol#147) is not in
mixedCase
Parameter BasicToken.transfer(address,uint256)._value (Code.sol#147) is not in
mixedCase
Parameter BasicToken.balanceOf(address)._owner (Code.sol#164) is not in
mixedCase
Parameter StandardToken.transferFrom(address,address,uint256)._from
(Code.sol#199) is not in mixedCase
Parameter StandardToken.transferFrom(address,address,uint256)._to
(Code.sol#199) is not in mixedCase
Parameter StandardToken.transferFrom(address,address,uint256)._value
(Code.sol#199) is not in mixedCase
Parameter StandardToken.approve(address,uint256)._spender (Code.sol#225) is not
in mixedCase
Parameter StandardToken.approve(address,uint256)._value (Code.sol#225) is not in
mixedCase
Parameter StandardToken.allowance(address,address)._owner (Code.sol#237) is not
in mixedCase
Parameter StandardToken.allowance(address,address)._spender (Code.sol#237) is
not in mixedCase
Parameter StandardToken.increaseApproval(address,uint256)._spender
(Code.sol#247) is not in mixedCase
Parameter StandardToken.increaseApproval(address,uint256)._addedValue
(Code.sol#247) is not in mixedCase
Parameter StandardToken.decreaseApproval(address,uint256)._spender
(Code.sol#254) is not in mixedCase
Parameter StandardToken.decreaseApproval(address,uint256)._subtractedValue
(Code.sol#254) is not in mixedCase
Parameter BurnableToken.burn(uint256)._value (Code.sol#280) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#conformance-to-solidity-naming-conventions
```

PHASE 1

PROJECT - ASCENT YIELD



```
INFO:Detectors:
UCBIBanking.slitherConstructorConstantVariables() (Code.sol#294-310) uses literals
with too many digits:
 - initialSupply = 12000000 * (10 ** uint256(decimals)) (Code.sol#300)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-
digits
```

SMART CONTRACT WEAKNESS CLASSIFICATION (SWC)

- ✓ Re-entrancy
- ✓ Timestamp Dependence
- ✓ Gas Limit and Loops
- ✓ Exception Disorder
- ✓ Gasless Send
- ✓ Use of tx.origin
- ✓ Compiler version not fixed
- ✓ Address hardcoded
- ✓ Divide before multiply
- ✓ Integer overflow/underflow
- ✓ Dangerous strict equalities
- ✓ Tautology or contradiction
- ✓ Missing Zero Address Validation
- ✓ Return values of low-level calls
- ✓ Revert/require functions
- ✓ Private modifier
- ✓ Using block.timestamp
- ✓ Multiple Sends
- ✓ Using SHA3
- ✓ Using suicide
- ✓ Using throw
- ✓ Using inline assembly

FUNCTIONAL TESTING:

Functionality Test Report for Smart Contract

- ✓ Transfer tokens between two accounts.
- ✓ Transfer From (Allowance Mechanism)
- ✓ Lock an address to prevent transfers.
- ✓ Check balance of an account.
- ✓ Approve an allowance for a spender and check the allowance.
- ✓ Attempt to transfer tokens without approval or sufficient balance.
- ✓ Measure gas consumption of common operations like transfer, approve, and burn.

CLOSING SUMMARY

The smart contract audit identified critical areas for improvement, particularly due to the use of an outdated Solidity version (^0.4.23) and several issues related to security, naming conventions, and gas optimization. Modernizing the contract by upgrading the Solidity version and adhering to best practices will significantly enhance its reliability, maintainability, and compliance with current standards. Addressing these issues will also reduce potential vulnerabilities and improve the overall user and developer experience.

Key Takeaways

- ✓ **Upgrade Solidity Version:**
  Move to ^0.8.x or later to mitigate risks associated with known vulnerabilities and improve performance.
- ✓ **Adopt Solidity Naming Conventions:**
  Refactor all function and parameter names to follow mixedCase for improved readability and adherence to standards.
- ✓ **Enhance Security Measures:**
  Implement modern safety mechanisms such as safeApprove, the Checks-Effects-Interactions pattern, and rigorous input validation.
- ✓ **Strengthen Burning and Locking Mechanisms:**
  Ensure the burning process is traceable with events and that locking mechanisms work as intended without interfering with other functions.

RECOMMENDATIONS:

Key Takeaways

✓ **1. Outdated Solidity Version**
- Upgrade to Solidity version ^0.8.x or higher.

✓ **2. Non-Compliance with Solidity Naming Conventions**
- Update all function and parameter names to follow Solidity's mixedCase naming convention for better readability and adherence to standards.

✓ **3. Literals with Too Many Digits**
- Use constants or define such values using variables with meaningful names for better readability.

✓ **4. Address Locking Mechanism**
- Add test cases to ensure:
  - Locked addresses cannot transfer tokens.
  - Unlocking an address restores its functionality.
  - Proper events (AddressLocked and AddressUnlocked) are emitted for traceability.

✓ **5. Approval and Allowance Mechanism**
- Follow the Checks-Effects-Interactions pattern in the approve and transferFrom functions to prevent reentrancy attacks.
- Implement EIP-20's best practices, including returning a boolean from the approve and transfer functions for confirmation.

✓ **6. Burning Mechanism**
- Emit a Burn event whenever tokens are burned for better traceability.
- Verify total supply updates correctly when burning tokens.

DISCLAIMER

AuditBlock does not provide security warranties, investment advice, or endorsements of any platform. This audit does not guarantee the security or correctness of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice. The authors are not liable for any decisions made based on the information in this document. Securing smart contracts is an ongoing process. A single audit is not sufficient. We recommend that the platform's development team implement a bug bounty program to encourage further analysis of the smart contract by other third parties

AUDITBLOCK
Premium

AUDITBLOCK
Premium

THANKS

AuditTime
6-December-2024

Audit Report of UCBI Banking

Contact Us

+923183506962