OKC-USDT

(CrossChain)

Audit Report

VER 1.0

3rd August 2022

No. 2022080317231

Project Summary

1. Project Introduction

OKChain issued the project of the USDT, the symbol that belongs to Tether on ERC20, deployed in OKC. The project adopts "delegateproxy" as the scalable proxy contract framework, which implements the EIP897 standard. What is more sophisticated is that the functions based on KIP20, such as the management of freezing assets are realized in the part of the token implementation, and the function of crosschain also takes effect in the contract on OKC.

2. Audit Summary

Project Name	OKC-USDT(CrossChain)	Platform	N/A
Token	USDT	Token symbol	USDT
Start date	26 May 2022	Language	Solidity
End date	27 May 2022	Website	N/A
Github	N/A	Whitepaper	N/A

3. Audit Scope

ID	ADDRESS	File	SHA-256 checksum
1	0x382bb369d343125bfb2117af9c149795c6c65c50	Proxy.sol	283dd4cc66bd5a554e504822b0c28fa9462d1491b34d9220fdefc40afcea26b8
2	0x382bb369d343125bfb2117af9c149795c6c65c50	Address.sol	1766ef0b83305645016ab40e1c262d5db1e6a9cd51f1b89d295e3e1ec6b3240c
3	0x382bb369d343125bfb2117af9c149795c6c65c50	UpgradeabilityProxy.sol	362996d399a418801a3f643016d994d7bc5386a8d361e837328a240ee75804cf
4	0x382bb369d343125bfb2117af9c149795c6c65c50	OwnedUpgradeabilityProxy.sol	ed67759d7b707b61f4ab9497aa0ed8f8e41c03e055667c1b2067f73b5ceebfd2
5	0xc49cc300e4420767f4c2103d89d80fff327e7238	SafeMath.sol	93c99439f77ffea008a4da3520ec01e286a30b371e8bdbe9349c391af74a90a2
6	0xc49cc300e4420767f4c2103d89d80fff327e7238	USDTimpl.sol	9371e72541cc81460a895c1e94da2a7c1d1a8d36fbea74c643969b74252f2675

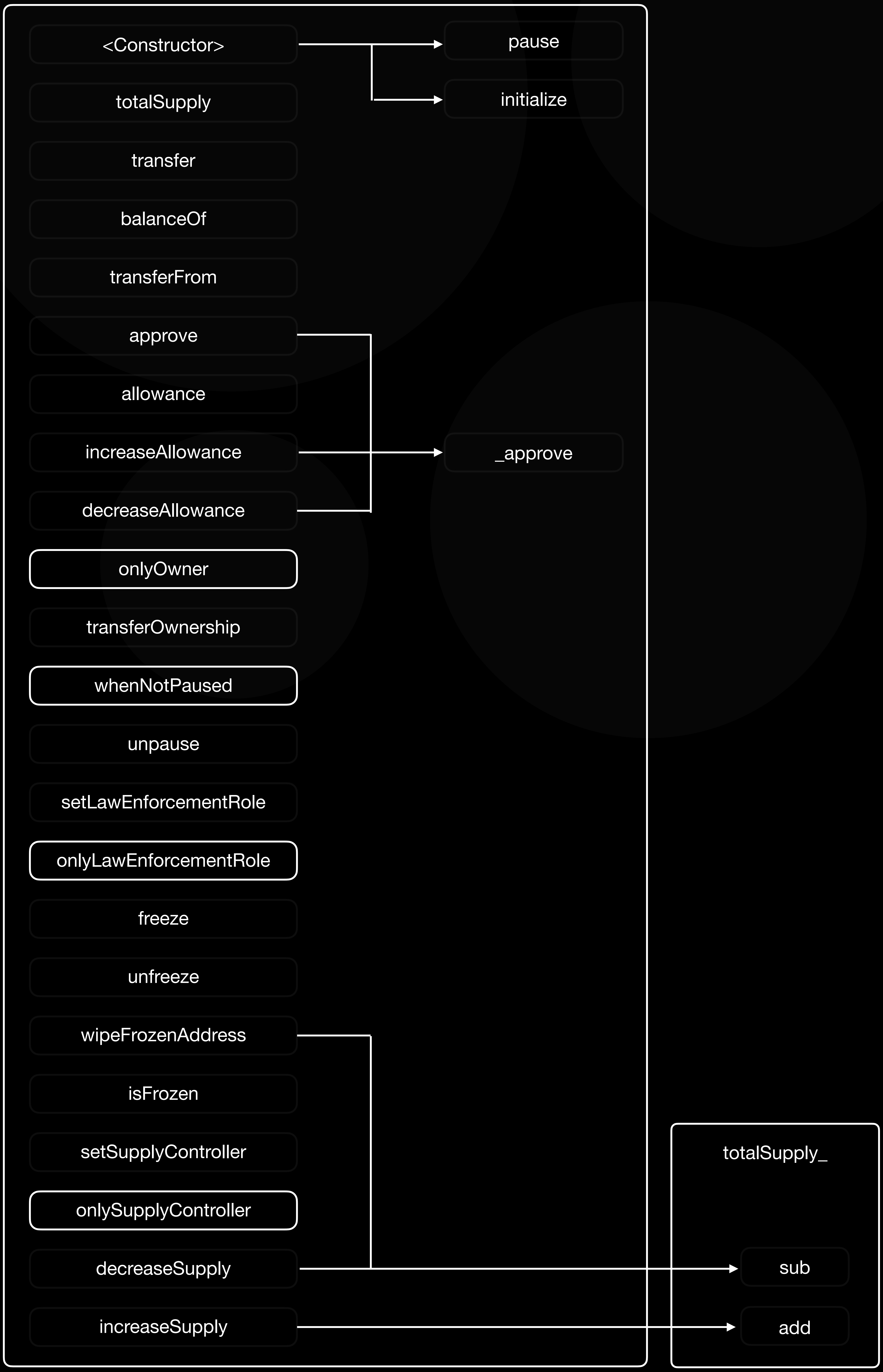
4. Code Source

ID	Link
1	https://www.oklink.com/zh-cn/okc/address/0x382bb369d343125bfb2117af9c149795c6c65c50
2	https://www.oklink.com/zh-cn/okc/address/0xc49cc300e4420767f4c2103d89d80fff327e7238

5. Storage Order Of State Variables

ID	Name	Type	Slot	Offset
1	USDTImplementation.initialized	bool	0	0
2	USDTImplementation.balances	mapping(address => uint256)	1	0
3	USDTImplementation.totalSupply_	uint256	2	0
4	USDTImplementation._allowed	mapping(address => mapping(address => uint256))	3	0
5	USDTImplementation.owner	address	4	0
6	USDTImplementation.paused	bool	4	20
7	USDTImplementation.lawEnforcementRole	address	5	0
8	USDTImplementation.frozen	mapping(address => bool)	6	0
9	USDTImplementation.supplyController	address	7	0

6. Interpretation Of Calling Contracts



3rd August 2022

Audit Report Summary

1. Audit Methods

By clearly understanding the design purpose, operation principle and implementation mode of the project, the audit team conducted in-depth research and analysis of the contract code. Based on clarifying the calling relationship between each contract and its functions, the possible loopholes in the contract are located and analyzed. Finally, a document containing the problem descriptions and corresponding modification suggestions is formed.

Audit methods	Static analysis, Manual Review

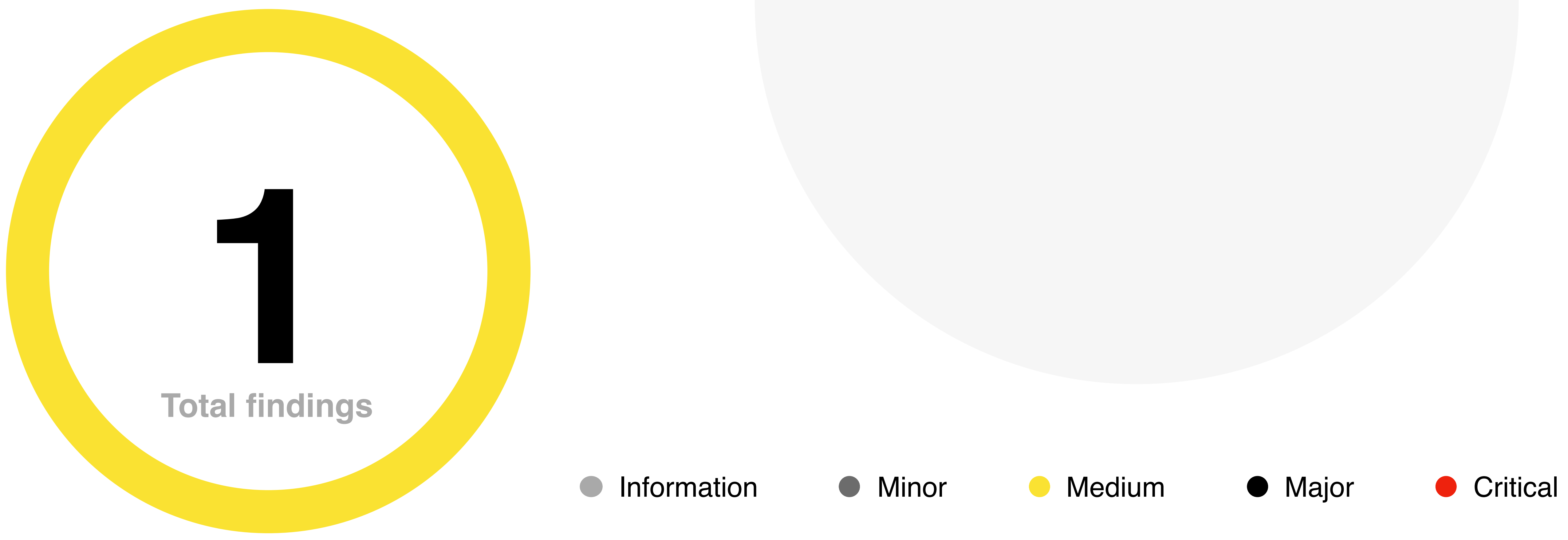
2. Audit Process

Steps	Operation	Description
1	Background	Reading the descriptions, white papers, contract source code, and other relevant information the project team provides to ensure a proper understanding of project functions.
2	Automated testing	Automated detection tools will be mainly used to scan the source code to find common potential vulnerabilities
3	Manual reveiw	The code will be thoroughly reviewed line by line by engineers to find potential vulnerabilities
4	Logical proofread	The engineer will compare the understanding of the code with the information provided by the project and check whether the code implementation is in line with the white paper information.
5	Test case	Including test case design, test scope analysis, symbolic execution, etc.
6	Optimization items	Review the project from the aspects of maintainability, security and operability according to the application scenarios, call methods and the latest research results

3. Risk Levels

Risk level	Issue description
Critical	Fatal risks and hazards that need to fixed immediately.
Major	Some high risks and hazards that will lead to related problems that must be solved
Medium	Some moderate risks and pitfalls may lead to potential risks that will eventually need to be addressed
Minor	There are low risks and hazards, mainly details of various types of mishandling or warning messages, which can be set aside for the time being
Information	Some parts can be optimized, such problems can be shelved, but it is recommended that the final solution

4. Audit Results



ID	Audit project	Risk level	Status
1	Reentrancy	None	
2	Injection	None	
3	Authentication bypass	None	
4	MEV Possibility	None	
5	Revert	None	
6	Race condition	None	
7	Insufficient Gas Griefing	None	
8	The major impact of flash loans	None	
9	Unreasonable economic model	None	
10	Predictable random numbers	None	
11	Voting rights management confusion	None	

ID	Audit project	Risk level	Status
12	Privacy leak	None	
13	Improper use of time on chain	None	
14	Improper codes in fallback function	None	
15	Improper identification	None	
16	Inappropriate opcode	None	
17	Inappropriate assembly	None	
18	Constructor irregularities	None	
19	Return value irregularity	None	
20	Event irregularity	None	
21	Keywords irregularity	None	
22	Not following ERC standards	None	
23	Irregularity of condition judgment	None	
24	Risk of liquidity drain	None	
25	Centralization Risk	Medium	**Acknowledged**
26	Logic change risk	None	
27	Integer overflow	None	
28	Improper function visiblity	None	
29	Improper initialization of variables	None	
30	Improper contract calls	None	
31	Variable irregularities	None	
32	Replay	None	
33	Write to Arbitrary Storage Location	None	
34	Honeypot logic	None	
35	Hash collision	None	
36	Decimal conflicts	None	
37	Proxy Irregularity	None	

* In the above table, if the status column is **Acknowledged**, the audit team has informed the project owner of the vulnerability. Still, the project owner has not made any changes to the vulnerability or has not announced to the audit team the progress of the changes to the vulnerability. If the status column is **Resolved**, the project owner has changed the exposure, and the audit team has confirmed the changes.

5. Risk and Modification Program

The following section provides detailed information about the risk items learned after the audit, including the type of risk, risk level, location of the issue, description of the problem, recommendations for changes, and feedback from the project owner.

1. Centralization Risk

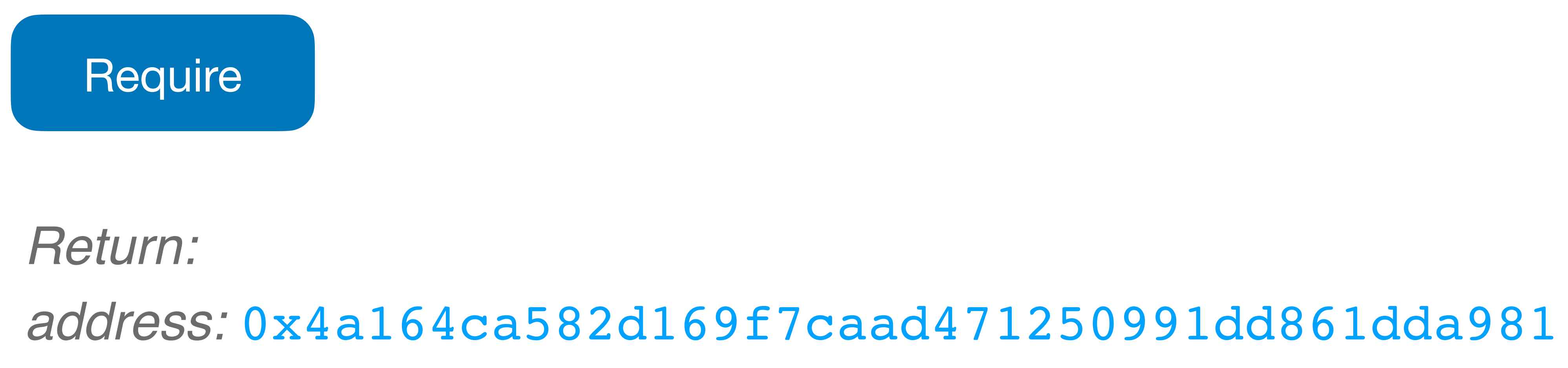
Location	Contract File	Risk Status	Risk Level
Line 260，370	USDTimpl.sol	⚠️Acknowledged	Medium

① Description

Centralization Risk refers to the problem of a single address management mechanism in the process of contract implementation. If `timelock` or multi-sign addresses are not used by the owner or management addresses of the contract in the process of setting, it is easy to encounter the risk of private key loss or disclosure in the subsequent operation and management. If the private key of a single address is lost, the contract management function will be lost, and the normal processing logic of assets will be affected. Moreover, user assets will suffer irreparable losses when the private key of a single address is leaked.

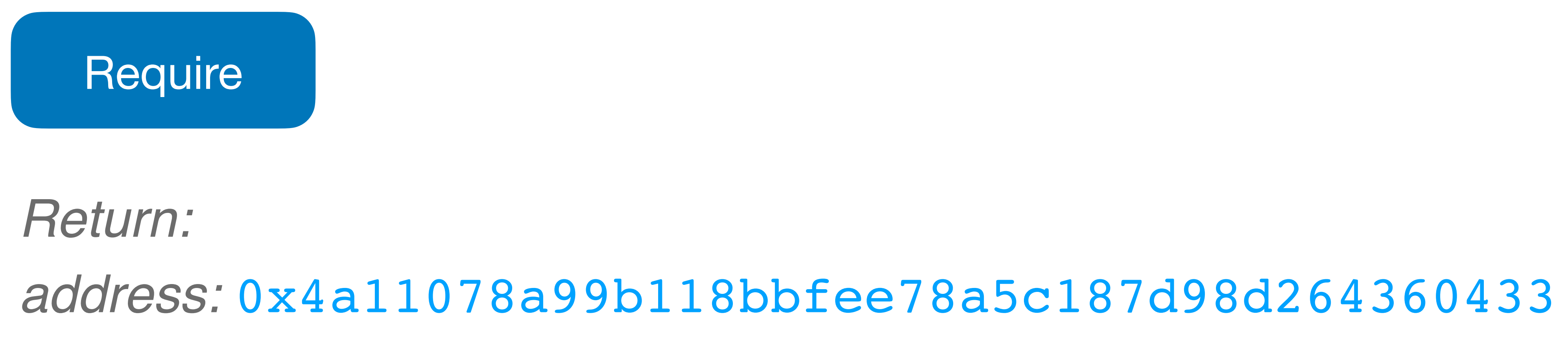
In this contract, the owner and supplycontroller are ordinary addresses, as shown in the figures below:

owner：



Return:
address: 0x4a164ca582d169f7caad471250991dd861dda981

supplyController：



Return:
address: 0x4a11078a99b118bbfee78a5c187d98d264360433

Once the private key of 0x4a11078a99b118bbfee78a5c187d98d264360433 is lost or leaked, serious harm to the assets of USDT will happen.
As for the module of crosschain, since both `increaseSupply` and `decreaseSupply` are constrained by the modifier `onlySupplyController` , if the private key of "supplycontroller" is lost, The mechanism of mint and destruction of crosschain will not be implemented. If the private key of "supplycontroller" is leaked, unexpected additional mint or destruction of assets will be caused.
It is recommended to modify "owner" and "supplycontroller" to multi contract or timelock contract address.

② Recommendation

It is recommended to modify "owner" and "supplycontroller" to multi-contract or timelock contract addresses.

③ Code

```
Solidity /**

// OWNER FUNCTIONALITY

    /**
     * @dev Throws if called by any account other than the owner.
     */
    modifier onlyOwner() {
        require(msg.sender == owner, "onlyOwner");
        _;
    }

    /**
     * @dev Allows the current owner to transfer control of the contract to a newOwner.
     * @param _newOwner The address to transfer ownership to.
     */
    //@OKlink Audit Description: The owner is a normal address, and there is a risk of
private key loss or disclosure, resulting in management failure
    //@OKlink Audit Solution: Modify the owner to timelock or multi-contract address
    function transferOwnership(address _newOwner) public onlyOwner {
        require(_newOwner != address(0), "cannot transfer ownership to address zero");
        emit OwnershipTransferred(owner, _newOwner);
        owner = _newOwner;
    }

    // PAUSABILITY FUNCTIONALITY

    /**
     * @dev Modifier to make a function callable only when the contract is not paused.
     */
    modifier whenNotPaused() {
        require(!paused, “whenNotPaused");
        _;
    }

    /**
     * @dev called by the owner to pause, triggers stopped state
     */
    //@OKlink Audit Description: The owner is a normal address, and there is a risk of
private key loss or disclosure, resulting in management failure
    //@OKlink Audit Solution: Modify the owner to timelock or multi-contract address
    function pause() public onlyOwner {
        require(!paused, "already paused");
        paused = true;
        emit Pause();
    }
```

Solidity /**

```
    /**
     * @dev called by the owner to unpause, returns to normal state
     */
```

```
    //@OKlink Audit Description: The owner is a normal address, and there is a risk of
private key loss or disclosure, resulting in management failure
    //@OKlink Audit Solution: Modify the owner to timelock or multi-contract address
```

```
    function unpause() public onlyOwner {
        require(paused, "already unpaused");
        paused = false;
        emit Unpause();
    }

    // LAW ENFORCEMENT FUNCTIONALITY

    /**
     * @dev Sets a new law enforcement role address.
     * @param _newLawEnforcementRole The new address allowed to freeze/unfreeze
addresses and seize their tokens.
     */
```

```
    //@OKlink Audit Description: The owner is a normal address, and there is a risk of
private key loss or disclosure, resulting in management failure
    //@OKlink Audit Solution: Modify the owner to timelock or multi-contract address
```

```
    function setLawEnforcementRole(address _newLawEnforcementRole) public {
        require(_newLawEnforcementRole != address(0),"lawEnforcementRole cannot
address(0)");
        require(msg.sender == lawEnforcementRole || msg.sender == owner, "only
lawEnforcementRole or Owner");
        emit LawEnforcementRoleSet(lawEnforcementRole, _newLawEnforcementRole);
        lawEnforcementRole = _newLawEnforcementRole;
    }
      modifier onlyLawEnforcementRole() {
        require(msg.sender == lawEnforcementRole, "onlyLawEnforcementRole");
        _;
    }
    /**
     * @dev Freezes an address balance from being transferred.
     * @param _addr The new address to freeze.
     */
```

```
   //@OKlink Audit Description: Lawenforcementrole is a normal address, and there is a
risk of private key loss or disclosure, resulting in management failure
    //@OKlink Audit Solution: Modify Lawenforcementrole to timelock or multi-contract
address
```

```
    function freeze(address _addr) public onlyLawEnforcementRole {
        require(!frozen[_addr], "address already frozen");
        frozen[_addr] = true;
        emit AddressFrozen(_addr);
    }
```

Solidity /**

```
    /**
     * @dev Unfreezes an address balance allowing transfer.
     * @param _addr The new address to unfreeze.
     */
```

```
   //@OKlink Audit Description: Lawenforcementrole is a normal address, and there is a
risk of private key loss or disclosure, resulting in management failure
    //@OKlink Audit Solution: Modify Lawenforcementrole to timelock or multi-contract
address
```

```
    function unfreeze(address _addr) public onlyLawEnforcementRole {
        require(frozen[_addr], "address already unfrozen");
        frozen[_addr] = false;
        emit AddressUnfrozen(_addr);
    }

    /**
     * @dev Wipes the balance of a frozen address, burning the tokens
     * @param _addr The new frozen address to wipe.
     */
```

```
   //@OKlink Audit Description: Lawenforcementrole is a normal address, and there is a
risk of private key loss or disclosure, resulting in management failure
    //@OKlink Audit Solution: Modify Lawenforcementrole to timelock or multi-contract
address
```

```
    function wipeFrozenAddress(address _addr) public onlyLawEnforcementRole {
        require(frozen[_addr], "address is not frozen");
        uint256 _balance = balances[_addr];
        balances[_addr] = 0;
        totalSupply_ = totalSupply_.sub(_balance);
        emit FrozenAddressWiped(_addr);
        emit SupplyDecreased(_addr, _balance);
        emit Transfer(_addr, address(0), _balance);
    }

    /**
    * @dev Gets the balance of the specified address.
    * @param _addr The address to check if frozen.
    * @return A bool representing whether the given address is frozen.
    */
    function isFrozen(address _addr) public view returns (bool) {
        return frozen[_addr];
    }
```

```
Solidity /**
    // SUPPLY CONTROL FUNCTIONALITY

    /**
     * @dev Sets a new supply controller address.
     * @param _newSupplyController The address allowed to burn/mint tokens to control
supply.
     */
    //@OKlink Audit Description: The owner is a normal address, and there is a risk of
private key loss or disclosure, resulting in management failure
    //@OKlink Audit Solution: Modify the owner to timelock or multi-contract address
    function setSupplyController(address _newSupplyController) public {
        require(msg.sender == supplyController || msg.sender == owner, "only
SupplyController or Owner");
        require(_newSupplyController != address(0), "cannot set supply controller to
address zero");
        emit SupplyControllerSet(supplyController, _newSupplyController);
        supplyController = _newSupplyController;
    }

    modifier onlySupplyController() {
        require(msg.sender == supplyController, "onlySupplyController");
        _;
    }

    /**
     * @dev Increases the total supply by minting the specified number of tokens to the
supply controller account.
     * @param _value The number of tokens to add.
     * @return A boolean that indicates if the operation was successful.
     */
    //@OKlink Audit Description: SupplyController is a normal address, and there is a
risk of private key loss or disclosure, resulting in management failure
    //@OKlink Audit Solution: Modify SupplyController to timelock or multi-contract
address
    function increaseSupply(uint256 _value) public onlySupplyController returns (bool
success) {
        totalSupply_ = totalSupply_.add(_value);
        balances[supplyController] = balances[supplyController].add(_value);
        emit SupplyIncreased(supplyController, _value);
        emit Transfer(address(0), supplyController, _value);
        return true;
    }
```

Solidity /**

```solidity
    /**
     * @dev Decreases the total supply by burning the specified number of tokens from
the supply controller account.
     * @param _value The number of tokens to remove.
     * @return A boolean that indicates if the operation was successful.
     */
```

```solidity
    //@OKlink Audit Description: SupplyController is a normal address, and there is a
risk of private key loss or disclosure, resulting in management failure
    //@OKlink Audit Solution: Modify SupplyController to timelock or multi-contract
address
```

```solidity
    function decreaseSupply(uint256 _value) public onlySupplyController returns (bool
success) {
        require(_value <= balances[supplyController], "not enough supply");
        balances[supplyController] = balances[supplyController].sub(_value);
        totalSupply_ = totalSupply_.sub(_value);
        emit SupplyDecreased(supplyController, _value);
        emit Transfer(supplyController, address(0), _value);
        return true;
    }
```

Disclaimer

i. This audit report focuses only on the types of audits identified in the final report issued. Other unknown security vulnerabilities are not part of this audit, and we do not accept responsibility for them.

ii. We shall only issue an audit report based on an attack or vulnerability that existed or occurred before the issuance of the audit report. We cannot determine the likely impact on the security posture of our projects for new attacks or vulnerabilities that may exist or occur in the future, and we are not responsible for them.

iii. The security audit analysis and other elements of our published audit report shall be based solely on documents and materials (including, but not limited to, contract codes) provided to us by the Project Party before the release of the audit report. Such documents and materials shall not be untrue, inaccurate, uninformative, altered, deleted, or concealed, and if the documents and materials provided by the Project Party are false, inaccurate, uninformative, changed, deleted or hidden, or if the documents and materials provided by the Project Party are untrue, inaccurate, uninformative, altered, deleted or concealed, or if the documents and materials provided by the Project Party are uninformative, uninformative, altered, deleted or hidden. If the records and information provided by the Project Party are untrue, inaccurate, uninformative, altered, deleted, or concealed, or if changes to such documents and information are made after the issuance of the audit report, we shall not be liable for any loss or adverse effect arising from any inconsistency between the reflected and actual conditions.

iv. The Project Parties are aware that our audit report is based on documents and information provided by the Project Parties and relies on the technology currently available. However, due to the technical limitations of any organization, there is a possibility that our audit report may not fully detect all risks. Our audit team encourages the project development team and any interested parties to conduct subsequent testing and audits of the project.

v. The project owner warrants that the project for which we are engaged to provide audit or testing services is legal, compliant, and does not violate applicable laws. The audit report is for the project owner's reference only, and the contents, manner of obtaining, use of, and any services or resources involved in the audit report shall not be relied upon for investment, tax, legal, regulatory, or advisory purposes of any kind, and we shall not be liable therefor. The Project Party shall not refer to, quote, display, or send the Audit Report in whole or in part to any third party without our prior written consent. The Project Party shall bear any loss or liability arising from that place. We assume no responsibility for any reliance on or use of the audit report for any purpose.

vi. This audit report does not cover the compiler of the contract or any areas beyond the programming language of the Smart Contract. The risk and liability of the audited Smart Contract arising from references to off-chain information or resources is the sole responsibility of the project party.

vii. Force Majeure. Force majeure means an unforeseen event whose occurrence and consequences cannot be avoided and cannot be overcome by the parties at the time of entering into the contract, including but not limited to natural disasters such as war, typhoon, flood, fire, earthquake, tidal wave, lightning, natural disaster, strike, nuclear explosion, epidemic and other unforeseen events such as changes in laws, regulations and policies and governmental acts, whose occurrence and consequences cannot be prevented or avoided, and which contains, affects or delays the performance by either party of all or part of its obligations under the contract.

viii. Suppose either party believes that the occurrence of force majeure affects the performance of its obligations under this Agreement. In that case, it shall promptly notify the other party and, depending on the extent of the effect of the event on the performance of the Agreement; the parties shall consult to determine whether to terminate the Agreement or partially relieve itself of its obligations to perform the Agreement, or to extend the performance of the Agreement.

ix. In force majeure, neither party shall be deemed in breach or non-performance of its obligations under this Agreement. Any financial commitments existing before the event shall not be affected, and the project party shall make payment for work performed by us.

Passed.

Date	3rd August 2022
Audit Team	歐科雲鏈

OKChain created a contract based on the official USDT of Tether and realized the function of crosschain. The purpose of this audit is to review the ERC20 contract, study the design and architecture of its cross-chain mechanism, find potential security risks, and try to find possible vulnerabilities.