cOinspect

You build, we defend.

flare

Smart Contract Audit

USDT Swapper

February 2025

USDT0 Swapper
Smart Contract Audit

Version: v250327 Prepared for: Flare February 2025

Security Assessment

1. Executive Summary

In **February 2025**, Flare engaged Coinspect to conduct a smart contract security audit of the USDT Swapper contract, which allows Flare chain users to redeem USDT.e tokens on a one-to-one basis for USD₮0 (usdt0.to) tokens at no extra cost.

Solved	Caution Advised	Resolution Pending
High 0	High 0	High 0
Medium 0	Medium 0	Medium 0
Low 0	Low 0	Low 0
No Risk 5	No Risk 0	No Risk 0
Total **5**	Total **0**	Total **0**

During this assessment, Coinspect identified five informational issues: potential loss of privileged access due to misconfigured deployment, absence of balance checks before executing swaps, missing events for key contract operations, allowance of zero-value swaps, and lack of an ownership transfer mechanism.

2. Summary of Findings

This section provides a concise overview of all the findings in the report grouped by remediation status and sorted by estimated total risk.

2.2 Solved issues & recommendations

These issues have been fully fixed or represent recommendations that could improve the long-term security posture of the project.

Id	Title	Risk
USWA-001	Deposited funds can be lost due to missing contract owner address	None
USWA-002	Zero-value swaps can be used to mislead third parties consuming swap events	None
USWA-003	Missing ownership transfer mechanism	None
USWA-004	Lack of contract balance check leads to failed swaps	None
USWA-005	Missing events for owner operations	None

3. Scope

The scope was set to be the `USDTSwapper.sol` contract from the https://github.com/flare-foundation/flare-smart-contracts-v2 repository at commit `85cc9a50f70fd7776ee1e669c62f7c38dae4ead4`.

Additionally, the review included the USDT contract instances below:

- USDT.e,
- USDT0

4. Assessment

The USDT Swapper was designed to facilitate a one-way 1:1 exchange of Stargate-bridged USDT (USDT.e) for USDT0 on the Flare Network. This serves as a temporary interoperability solution with the Tether ecosystem, offering retail users a seamless migration path during a transitional period, though the period is not enforced by the contract.

Coinspect's review covered both the swapper contract and the deployments of USDT and USDT0, considering the deviation from the standard ERC-20 implementation of USDT on Ethereum. Coinspect verified that both contracts adhere to the ERC-20 standard and maintain the same decimal precision (6) to prevent imbalances in swaps.

4.1 Security assumptions

Coinspect assumed that the swapper contract would only support token swaps from the contracts listed in the Scope section of this report.

4.2 Decentralization

The contract includes privileged functions that enable the owner to deposit USDT0, withdraw USDT0 and USDT.e tokens, and pause or resume swap operations.

4.3 Code quality and testing

The contract was concise and easy to understand, thanks to NatSpec documentation on each function and variable. It was accompanied by 15 tests that covered the basic functionality, ensuring 100% code coverage.

5. Detailed Findings

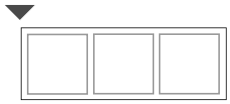
USWA-001

Deposited funds can be lost due to missing contract owner address

Status
Solved

Resolution
Acknowledged

Risk
None



Impact
Recommendation

Likelihood
–

Location

`USDTSwapper.sol`

Description

Privileged access to the USDT swapper contract, and therefore access to the deposited funds, can be lost if the `owner` is set to the zero address during deployment.

The swapper contract, as shown below, does not prevent the deployer from setting the contract `owner` or the USDT contracts to the zero address:

```
constructor(address _owner, IERC20Metadata _usdte, IERC20Metadata
_usdt0) {
    owner = _owner;
    usdte = _usdte;
    usdt0 = _usdt0;
}
```

If funds are deposited and the zero address configuration is not detected beforehand, the contract admin will be unable to withdraw the funds.

Recommendation

Implement checks to ensure the swapper contract cannot be instantiated with any zero address values.

Since USDT contract addresses are predetermined, consider hard-coding them for mainnet deployment to prevent deployment errors.

Status

Acknowledged. The Flare team noted they could add non-zero address checks in the constructor, but this would not prevent incorrect addresses or order. Since deposits will use the depositUSDT0 method, which verifies the owner, and all addresses will be checked beforehand, they believe the check is not needed.

Proof-of-Concept

The following test demonstrates that it is possible to set zero address parameters when deploying the `USDTSwapper` contract.

```
function testZeroAddress() public {
    swapper = new USDTSwapper(address(0), IERC20Metadata(address(0)),
IERC20Metadata(address(0)));
    assertEq(address(swapper.owner()),address(0));
    assertEq(address(swapper.usdt0()),address(0));
}
```

To execute it locally, place it in the `USDTSwapper.t.sol` file and run `forge test --match-contract USDTSwapperTest --match-test testZeroAddress`.

USWA-002

Zero-value swaps can be used to mislead third parties consuming swap events

Status
Solved

Resolution
Acknowledged

Risk
None

Impact
Recommendation

Likelihood
–

Location

`USDTSwapper.sol`

Description

Third parties consuming events from the swapper contract can be misled into processing zero-value swap events. This happens because the `swap` function fails to check if the requested swap amount is zero, triggering an event for an action that does not change the state of the contract.

```
function swap(uint256 _amount) external {
    require(!paused, "swapping is paused");
    usdte.safeTransferFrom(msg.sender, address(this), _amount);
    usdt0.safeTransfer(msg.sender, _amount);
    emit Swapped(msg.sender, _amount);
}
```

The risk associated with this issue is considered informational, as exploitation is unlikely.

Recommendation

Zero-value swaps should be disallowed.

Status

Acknowledged. The Flare team stated that the frontend using this contract will handle prevention of zero-value swaps. However, this does not stop third parties from interacting directly with the contract to execute such swaps.

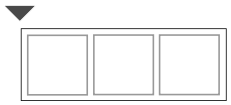
USWA-003

Missing ownership transfer mechanism

Status
Solved

Resolution
Acknowledged

Risk
None



Impact
Recommendation

Likelihood
–

Location

`USDTSwapper.sol`

Description

The `USDTSwapper` contract currently employs an immutable owner address, which cannot be modified after deployment. While the contract will only be used for a short period, this design choice limits operational flexibility and creates a single point of administration for the contract's lifetime.

```
/// Owner of the contract
address public immutable owner;

modifier onlyOwner() {
    require(msg.sender == owner, "only owner");
    _;
}
```

Recommendation

Consider implementing a two-step ownership transfer pattern and using a multisig wallet as the owner.

Status

Acknowledged. The Flare team stated that the contract owner will be a multisig account, so there should be no need to change it.

USWA-004

Lack of contract balance check leads to failed swaps

Status
Solved

Resolution
Acknowledged

Risk
None

Impact
Recommendation

Likelihood
–

Location

`USDTSwapper.sol#L49`

Description

The `swap` function in the `USDTSwapper` contract does not validate whether the contract has sufficient `USDT0` balance before attempting the transfer, potentially resulting in failed transactions and wasted gas.

```
function swap(uint256 _amount) external {
    require(!paused, "swapping is paused");
    usdte.safeTransferFrom(msg.sender, address(this), _amount);
    usdt0.safeTransfer(msg.sender, _amount); // No prior balance check
    emit Swapped(msg.sender, _amount);
}
```

Recommendation

Add balance checks to the swap function to prevent unnecessary gas consumption.

Status

Acknowledged. The Flare team stated that the frontend interacting with the swapper contract will be responsible for performing the necessary balance checks.

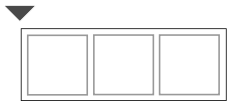
USWA-005

Missing events for owner operations

Status
Solved

Resolution
Acknowledged

Risk
None



Impact
Recommendation

Likelihood
–

Location

`contracts/utils/implementation/USDTSwapper.sol`

Description

The `USDTSwapper` contract emits an event for the swap function but lacks event emissions for other critical operations performed by the owner. Functions like `pause`, `unpause`, `depositUSDT0`, `withdrawUSDT0`, and `withdrawUSDTe` modify the contract state or move funds but do not emit corresponding events.

```
function pause() external onlyOwner {
    paused = true;
    // No event emitted
}

function unpause() external onlyOwner {
    paused = false;
    // No event emitted
}

function depositUSDT0(uint256 _amount) external onlyOwner {
    usdt0.safeTransferFrom(msg.sender, address(this), _amount);
```

```
    // No event emitted
}

function withdrawUSDT0(uint256 _amount) external onlyOwner {
    usdt0.safeTransfer(msg.sender, _amount);
    // No event emitted
}

function withdrawUSDTe(uint256 _amount) external onlyOwner {
    usdte.safeTransfer(msg.sender, _amount);
    // No event emitted
}
```

Recommendation

Add events for all state-changing owner operations to improve transparency.

Status

Acknowledged. The Flare team stated that since this contract is intended to be active only for a few months before being paused, these checks will likely not be needed.

6. Disclaimer

The contents of this report are provided "as is" without warranty of any kind. Coinspect is not responsible for any consequences of using the information contained herein.

This report represents a point-in-time and time-boxed evaluation conducted within a specific timeframe and scope agreed upon with the client. The assessment's findings and recommendations are based on the information, source code, and systems access provided by the client during the review period.

The assessment's findings should not be considered an exhaustive list of all potential security issues. This report does not cover out-of-scope components that may interact with the analyzed system, nor does it assess the operational security of the organization that developed and deployed the system.

This report does not imply ongoing security monitoring or guaranteeing the current security status of the assessed system. Due to the dynamic nature of information security threats, new vulnerabilities may emerge after the assessment period.

This report should not be considered an endorsement or disapproval of any project or team. It does not provide investment advice and should not be used to make investment decisions.