SURFI

Smart Contract

Audit Report

07/13/2023

contact@scalebit.xyz https://twitter.com/scalebit_

ScaleBit

SURFI Audit Report

1 Executive Summary

1.1 Project Information

Description	The first native DEX Aggregator on Linea
Type	DEX Aggregator
Auditors	ScaleBit
Timeline	July 6, 2023 – July 11, 2023
Languages	Solidity
Platform	Linea
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/HelloSurFi/surfi-contract
Commits	bf71f08a1ef2bd8ea8c5e81c4245d27496ebbbc1

1.2 Files in Scope

The following are the SHA1 hashes of the last reviewed files.

ID	Files	SHA-1 Hash
SPY	contracts/general/SwapProxy.sol	6ae5d825b262a42848b4c04a1f1dce1d1d61afe3
ISF	contracts/iZiSwap/core/interfaces/IiZiSwapFactory.sol	a711b662c3bf52572c784e8b9014477a189a1943

ISP	contracts/iZiSwap/core/interfaces/IiZiSwapPool.sol	9a89e288f92d7d43a39b5f9946c7e8bd933045e1
ISC	contracts/iZiSwap/core/interfaces/IiZiSwapCallback.sol	aff3f363aecfda04958869d7080eee59b63571ac
IZS	contracts/iZiSwap/Swap.sol	1e8e27a0056ac0fbcacc1bcf8dabeaf8293fb4d0
BTL	contracts/iZiSwap/libraries/BytesLib.sol	ef6864a579355f2a588892ac909ce3540257f3d8
PTH	contracts/iZiSwap/libraries/Path.sol	4fc86bbb5e08f56def5ce97fabc24b5595783b22
BAS	contracts/iZiSwap/base/base.sol	e7b34a37f0039a653713b0d3c41636f148dfe834
MTC	contracts/multicall.sol	92571e25d055b7304d2d2cc598539d4ec6b0f88c

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	4		4
Informational	2		2
Minor			
Medium	2		2
Major			
Critical			

1.4 ScaleBit Audit BreakDown

ScaleBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow
- Number of rounding errors
- Unchecked External Call
- Unchecked CALL Return Values
- Functionality Checks
- Reentrancy
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic issues
- Gas usage
- Fallback function usage
- tx.origin authentication
- Replay attacks
- Coding style issues

1.5 Methodology

The security team adopted the "**Testing and Automated Analysis**", "**Code Review**" and "**Formal Verification**" strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section **1.2**.

(3) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by **SURFI** to identify any potential issues and vulnerabilities in the source code of the **SURFI** smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we have identified **4** issues of varying severity, listed below.

ID	Title	Severity	Status
SPY–1	Did not Approve to Zero First	Medium	Acknowledged
SPY–2	Revert on Large Approvals & Transfers	Medium	Acknowledged
SPY–3	Excessive Gas and Memory Consumption	Informational	Acknowledged
BSE–1	Redundant Variable	Informational	Acknowledged

3 Participant Process

Here are the relevant actors with their respective abilities within the `SURFI` SmartContract:

User

- User can execute business operations by calling the `multicall()` function.
- User can swap tokenY for tokenX, specifying the maximum amount of tokenY they are willing to pay, and ensuring the acquired tokenX exceeds a minimum threshold through the `swapY2X()` function.
- User can swap tokenX for tokenY, specifying the maximum amount of tokenX they are willing to pay, and ensuring the acquired tokenY exceeds a minimum threshold through the `swapX2Y()` function.
- User can perform multi-hop token swaps and execute business operations by calling the `swapDesire()` function, specifying the desired amount of tokens and recipient.
- User can swap tokenY for a desired amount of tokenX by calling the `swapY2XDesireX()` function, specifying the desired amount of tokenX and other parameters such as recipient and fee.
- User can swap tokenX for a desired amount of tokenY by calling the `swapX2YDesireY()` function, specifying the desired amount of tokenY and other parameters such as recipient and fee.

4 Findings

SPY-01 Did not Approve to Zero First

Severity: Medium

Status: Acknowledged

Code Location: contracts/general/SwapProxy.sol#L80

Descriptions: Some ERC20 tokens like USDT require resetting the approval to 0 first before being able to reset it to another value.

USDT:

```
function approve(address _spender, uint _value) public onlyPayloadSize(2 * 32) {

// To change the approve amount you first have to reduce the addresses`
//  allowance to zero by calling `approve(_spender, 0)` if it is not
//  already 0 to mitigate the race condition described here:
//  https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
  require(!((_value != 0) && (allowed[msg.sender][_spender] != 0)));

  allowed[msg.sender][_spender] = _value;
  Approval(msg.sender, _spender, _value);
}
```

The `SwapProxy.approveToken()` function does not do this – unlike OpenZeppelin's safeApprove implementation.

Some token contracts revert the transaction when the allowance is not zero.

```
function approveToken(address token, address spender) external {
  bool ok = IERC20(token).approve(spender, type(uint256).max);
  require(ok, 'approve fail');
}
```

Suggestion: It is recommended to set the allowance to zero before increasing the allowance and use `safeApprove/safeIncreaseAllowance`.

```
function approveToken(address token, address spender) external {
  IERC20(token).safeApprove(spender, 0);
  IERC20(token).safeApprove(spender, type(uint256).max);
}

```

SPY-02 Revert on Large Approvals & Transfers

Severity: Medium

Status: Acknowledged

Code Location: contracts/general/SwapProxy.sol#L80

Descriptions: Some tokens (e.g. UNI, COMP) revert if the value passed to approve or transfer is larger than uint96.

Both of the above tokens have special case logic in approval that sets allowance to type(uint96).max if the approval amount is type(uint256).max, which may cause issues with systems that expect the value passed to approve to be reflected in the allowances mapping.

UNI:

```
function approve(address spender, uint rawAmount) external returns (bool) {
  uint96 amount;
  if (rawAmount == uint(-1)) {
  amount = uint96(-1);
  } else {
  amount = safe96(rawAmount, "Uni::approve: amount exceeds 96 bits");
  }

  allowances[msg.sender][spender] = amount;

  emit Approval(msg.sender, spender, amount);
  return true;
}
```

SwapProxy:

```
function approveToken(address token, address spender) external {
  bool ok = IERC20(token).approve(spender, type(uint256).max);
  require(ok, 'approve fail');
}
```

Suggestion: Add validation checks: Before performing an approval or transfer, validate the amount against the maximum limit. If the amount exceeds the limit, revert the transaction with an appropriate error message indicating that the amount is too large.

SPY-03 Excessive Gas and Memory Consumption

Severity: Informational

Status: Acknowledged

Code Location: contracts/iZiSwap/base/base.sol#L109-L112, contracts/general/SwapProxy.sol#L35-L39

Descriptions: Now `(bool success, )` is actually the same as writing `(bool success, bytes memory data)` which basically means that even though the `data` is omitted it doesn't mean that the contract does not handle it. Actually, the way it works is the `data` that was returned from the receiver will be copied to memory.

```
function safeTransferETH(address to, uint256 value) internal {
        (bool success, ) = to.call{value: value}(new bytes(0));
        require(success, 'STE');
    }
```

Suggestion: Use a low-level assembly call since it does not automatically copy return data to memory, here is an example:

```
bool success;
assembly {
  success := call(gas(), receiver, amount, 0, 0, 0, 0)
}
```

BSE-01 Redundant Variable

Severity: Informational

Status: Acknowledged

Code Location: contracts/general/SwapProxy.sol#L84

Descriptions: The returned `res` is not used in the function.

Suggestion: It is recommended to delete the redundant variable or use it in the function.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed**: The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.