



SCALE

Security Review



Oct 16, 2024

Conducted by:
Blckhv, Security Researcher
Slavcheww, Security Researcher

Contents

| | |
|--|----------|
| 1. About SBSecurity | 3 |
| 2. Disclaimer | 3 |
| 3. Risk classification | 3 |
| 3.1. Impact..... | 3 |
| 3.2. Likelihood | 3 |
| 3.3. Action required for severity levels..... | 3 |
| 4. Executive Summary | 4 |
| 5. Findings | 5 |
| 5.1. Critical severity | 5 |
| 5.1.1. Donation to pair under 1e9 will DoS LP funding..... | 5 |
| 5.1.2. Anyone can skim excess tokens from the pair after rate update..... | 6 |
| 5.2. High severity | 7 |
| 5.2.1. TickMath library is missing unchecked statements and _twapCheck will not work as expected | 7 |
| 5.2.2. Presale is DoSed when totalMinted + bonuses > tTotal | 8 |
| 5.3. Medium severity | 9 |
| 5.3.1. No slippage in distributeReserve | 9 |
| 5.4. Low/Info severity | 10 |
| 5.4.1. Owner can finalize the presale before start and brick the contract | 10 |
| 5.4.2. Reflection fee should not be settable to 0..... | 10 |
| 5.4.3. availableMinerTypes check can be bypassed | 11 |
| 5.4.4. purchaseDragonXForBuyBurn can leave dust DragonX | 11 |
| 5.4.5. Rewards can't be claimed for more than 1 instance back..... | 11 |

1. About SBSecurity

SBSecurity is a duo of skilled smart contract security researchers. Based on the audits conducted and numerous vulnerabilities reported, we strive to provide the absolute best security service and client satisfaction. While it's understood that 100% security and bug-free code cannot be guaranteed by anyone, we are committed to giving our utmost to provide the best possible outcome for you and your product.

Book a Security Review with us at sbsecurity.net or reach out on Twitter [@Slavcheww](https://twitter.com/Slavcheww).

2. Disclaimer

A smart contract security review can only show the presence of vulnerabilities **but not their absence**. Audits are a time, resource, and expertise-bound effort where skilled technicians evaluate the codebase and their dependencies using various techniques to find as many flaws as possible and suggest security-related improvements. We as a company stand behind our brand and the level of service that is provided but also recommend subsequent security reviews, on-chain monitoring, and high whitehat incentivization.

3. Risk classification

| | Impact: High | Impact: Medium | Impact: Low |
|--------------------|--------------|----------------|-------------|
| Likelihood: High | Critical | High | Medium |
| Likelihood: Medium | High | Medium | Low |
| Likelihood: Low | Medium | Low | Low |

3.1. Impact

- **High** - leads to a significant loss of assets in the protocol or significantly harms a group of users.
- **Medium** - leads to a moderate loss of assets in the protocol or some disruption of the protocol's functionality.
- **Low** - funds are not at risk.

3.2. Likelihood

- **High** - almost **certain** to happen, easy to perform, or highly incentivized.
- **Medium** - only **conditionally possible**, but still relatively likely.
- **Low** - requires specific state or **little-to-no incentive**.

3.3. Action required for severity levels

- High - **Must** fix (before deployment if not already deployed).
- Medium - **Should** fix.
- Low - **Could** fix.



4. Executive Summary

SCALE is a token aimed to be a one stop shop for TitanX users. SCALE is an reflection erc20 token bonded with token built on DragonX through LP's. By being bonded with these eco tokens, the market value of ELMT is a leverage play of all linked ecosystem tokens combined. The protocol also reserve part of the TitanX deposited for HYDRA miners, utilizing its potential.

Overview

| | |
|-------------|--|
| Project | SCALE |
| Repository | Private |
| Commit Hash | 64783dd557569b7685bd0a0d159673e6a37d9e04 |
| Resolution | 5b8e97a1263cee2b1ae3dd2f8d506614a7589901 |
| Timeline | September 13 - September 15, 2024 |

Scope

| |
|---------------|
| SCALE.sol |
| SHED.sol |
| SHEDMiner.sol |

Issues Found

| | |
|---------------|---|
| Critical Risk | 2 |
| High Risk | 2 |
| Medium Risk | 1 |
| Low/Info Risk | 4 |

5. Findings

5.1. Critical severity

5.1.1. Donation to pair under 1e9 will DoS LP funding

Severity: Critical Risk

Description: The fix for the UniswapV2 donation-sync attack is not sufficient in this case because SCALE is 9 decimal, compared to other tokens which are 18 decimal.

Let's take the **DragonX/SCALE** pool, if someone does this donation-sync attack with a lower value of 1e9 - **requiredScale** will be 0.

```
function _fixPool(address pairAddress, uint256 tokenAmount, uint256 scaleAmount, uint256 currentBalance)
internal {
    uint256 requiredScale = currentBalance * scaleAmount / tokenAmount;
    uint256 rAmount = requiredScale * _getRate();
    _rOwned[pairAddress] += rAmount;
    _rTotal += rAmount;
    _tTotal += requiredScale;
    _totalMinted += requiredScale;
    emit Transfer(address(0), pairAddress, requiredScale);
    IUniswapV2Pair(pairAddress).sync();
}
```

Recommendation: If there are tokens in the pool from **token0**, donate manually the tokens that were allocated to **addLiquidity** and mint the position, therefore the ratio may be slightly different if many tokens are deposited in the pool, but this is the only way that all tokens will be deposited. If dust amounts was deposited into the pool with the idea of **addLiquidity** to revert, they will not change the ratio because the 10 wei is nothing compared to the billions or trillions of TitanX that will be deposited.

Resolution: Fixed

5.1.2. Anyone can skim excess tokens from the pair after rate update

Severity: Critical Risk

Description: Since **SCALE** token is designed to have a positive “rebase” after **rTotal** decreases, either by anyone calling **SCALE::reflect** or taking the reflect fee as part of the transfer tax, anyone can sweep the updated balance from the **UniswapV2** pairs by calling **pair::skim**.

```
function skim(address to) external lock {
    address _token0 = token0; // gas savings
    address _token1 = token1; // gas savings
    _safeTransfer(_token0, to, IERC20(_token0).balanceOf(address(this)).sub(reserve0));
    _safeTransfer(_token1, to, IERC20(_token1).balanceOf(address(this)).sub(reserve1));
}
```

This function checks the reserves and the current balance and sends **balanceOf – reserve** to the caller. By performing that users will harvest all the positive rebases from the pairs and **SCALE** price won't be increasing.

Recommendation: Consider calling **pair::sync** when **SCALE** rate (**SCALE::reflect** and **SCALE::_update**) is being updated, this will update the reserves to match the current balance and the positive rebase will go in the pool without a way to be skimmed from anyone.

Resolution: Fixed. The team decided to add exclusions to the pairs and track their balance without reflections.

5.2. High severity

5.2.1. **TickMath** library is missing unchecked statements and **_twapCheck** will not work as expected

Severity: High Risk

Description: **_twapCheck** will be reverting in certain scenarios because **TickMath** library is missing important **unchecked** blocks, compared to the original **Uniswap** library. The problem is that ticks are large numbers and the library should be designed to allow overflows and still work as expected.

Recommendation: Make sure to copy the library from branch **0.8** of **Uniswap**, the one in main is compiled with Solidity version < 8 and there are no overflow protections. - <https://github.com/Uniswap/v3-core/blob/0.8/contracts/libraries/TickMath.sol#L27>

Resolution: Fixed

5.2.2. Presale is DoSed when `totalMinted + bonuses > tTotal`

Severity: High Risk

Description: Bonuses are not included in the `MaxSupply` checks performed in mint functions and later on this will lead to bricking the contract due to the inability to call `finalizePresale`.

This will happen because `totalMinted` (including the bonuses) will become larger than the `_tTotal` and will underflow:

```
function finalizePresale() external onlyOwner {
    if (isPresaleActive()) revert PresaleActive();
    if (shedContract == address(0)) revert ZeroAddress();
    if (presaleFinalized) revert Prohibited();

    _distributeTokens();

    // burn not minted
    //10000000000000000e9 - (74074074074074074074 + bonuses) - 25925925925925925925
    uint256 tBurn = _tTotal - _totalMinted - scaleLpPool;
    uint256 rate = _getRate();
    uint256 rBurn = tBurn * rate;
    _rOwned[address(this)] -= rBurn;
    _rTotal -= rBurn;
    _tTotal = _totalMinted + scaleLpPool;

    presaleFinalized = true;
    emit Transfer(address(0), address(this), scaleLpPool);
}
```

This doesn't even require the `maxSupply` to be minted, we only need `bonuses + totalMinted` to exceed the total supply.

Recommendation: Since we want only ~74 trillion supply `SCALE` tokens do not allow minting more than that and include the amount with the bonus in the check.

Resolution: Fixed

5.3. Medium severity

5.3.1. No slippage in `distributeReserve`

Severity: Medium Risk

Description: Users calling `SCALE::distributeReserve` can disable the slippage and sandwich their transaction, forcing the contract into bad swap.

This happens because `minDragonXAmount` is passed as an argument and is not constrained.

As a result, `SCALE` will swap all its `reserves` for less DragonX than normal.

Recommendation: Make sure to set `minReserveDistribution` to proper value which will give incentive to be called immediately and in the same time don't allow the caller to perform sandwich attack.

Resolution: Acknowledged

5.4. Low/Info severity

5.4.1. Owner can finalize the presale before start and brick the contract

Severity: Low Risk

Description: Owner can brick the whole contract by calling `SCALE::finalizePresale` even before starting it. This will happen because `presaleEnd` is by default 0 and the `isPresaleActive` check will return `false`:

```
function finalizePresale() external onlyOwner {
    if (isPresaleActive()) revert PresaleActive();
```

```
function isPresaleActive() public view returns (bool) {
    return presaleEnd > block.timestamp;
}
```

After this check is bypassed there is another, checking whether the `SHED` contract is not `address(0)`. If `finalizePresale` is called before even starting it, `presaleFinalized` will become `true`, although there is no `TitanX` to be distributed amongst the contracts.

Recommendation:

```
function finalizePresale() external onlyOwner {
+     if (presaleEnd == 0) revert PresaleInactive();
    if (isPresaleActive()) revert PresaleActive();
```

Resolution: Fixed

5.4.2. Reflection fee should not be settable to 0

Severity: Low Risk

Description: According to the documentation, `reflectionFee` should not be able to be set to 0. Doing that will turn off the reflection functionality of `SCALE` and the rate won't be decreasing.

Recommendation:

```
function setReflectionFee(uint16 bps) external onlyOwner {
-     if (bps != 0 && bps != 150 && bps != 300 && bps != 450 && bps != 600) revert Prohibited();
+     if (bps != 150 && bps != 300 && bps != 450 && bps != 600) revert Prohibited();
    reflectionFee = bps;
}
```

Resolution: Fixed

5.4.3. **availableMinerTypes** check can be bypassed

Severity: Low Risk

Description: In `SHED::setAvailableMinerTypes` there is a missing check whether the new length is lower than the current `totalActiveMiners`.

When set to lower number it will become ineffective because in `deployMiner` you check for a strict equality.

Recommendation: Add a check in `setAvailableMinerTypes()` or restrict `AvailableMinerType` to not being able to change after launch.

Resolution: Fixed

5.4.4. **purchaseDragonXForBuyBurn** can leave dust DragonX

Severity: Low Risk

Description: Due to the divisions performed, some small amounts of `TitanX` can remain locked in the `SCALE` contract after the last `DragonX` purchase is done.

That will happen when the `bdxBuyBurnPool` is not evenly divisible to the `purchasesRequired`.

Although that doesn't pose any issues to the system, these `TitanX` tokens will be irretrievable.

Recommendation: There are 2 ways by which this can be fixed:

1. You can track the already swapped and on the last purchases you subtract `bdxBuyBurnPool - totalSwapped`.
2. You can add function which will be callable after trading is enabled and `buyBunPurchases == purchasesRequired` and it will be used to sweep all the `TitanX` left from the divisions.

Resolution: Fixed

5.4.5. **Rewards can't be claimed for more than 1 instance back**

Severity: Information Risk

Description: If the rewards for given miner aren't claimed after 2 instances (2000 new miners) they will be irrecoverable. The scenario is highly unlikely to happen because no more than 20 active miners can be deployed at once, but still possible to occur.

Resolution: Acknowledged