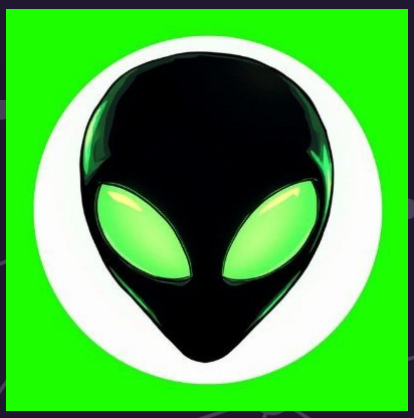
SB SECURITY

AlienX

Security Review



Oct 15, 2024

Conducted by:
Blckhv, Lead Security Researcher
Slavcheww, Lead Security Researcher

Contents

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.

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

Overview

Project	AlienX
Repository	Private
Commit Hash	ba2a35e8aa03951600c2d17c85bfff6a7883279a
Resolution	639d4a1f3b46e1dbcb8bc82363ea8501ec49249a
Timeline	Audit: October 9 - October 12, 2024 Mitigation: October 13 - October 15, 2024

Scope

src/*

Issues Found

Critical Risk	1
High Risk	1
Medium Risk	0
Low/Info Risk	9

5. Findings

5.1. Critical severity

5.1.1. AlienX pairs creation can be bricked with `pair::sync()`

Severity: Critical Risk

Description: `createAndFundLPs` will attempt to call `addLiquidity()` assuming it is the first to add liquidity to both pools (`TitanX/AlienX` and `Inferno/AlienX`).

```
function createAndFundLPs(uint32 _deadline, uint256 _amountInfernoMin) external onlyOwner {
    if (titanX.balanceOf(address(this)) < INITIAL_TITANX_X_FOR_LP * 2) revert NotEnoughBalanceForLp();
    if (addedLiquidity) revert LiquidityAlreadyAdded();

    IUniswapV2Router02 r = IUniswapV2Router02(v2Router);

    alienX.mint(address(this), INITIAL_ALIENX_FOR_LP * 2);
    alienX.approve(v2Router, INITIAL_ALIENX_FOR_LP * 2);

    {
        //@note - No one will have alienX tokens to sabotauge any of those functions,
        // if for some reason there are not enough tokens for liquidity before anyone starts to claim the
team will invest to make up for it.
        titanX.approve(address(r), INITIAL_TITANX_X_FOR_LP);
        r.addLiquidity(
            address(titanX),
            address(alienX),
            INITIAL_TITANX_X_FOR_LP,
            INITIAL_ALIENX_FOR_LP,
            0,
            0,
            address(this),
            _deadline
        );
    }
    {
        uint256 _infernoAmount = _swapTitanXForInferno(INITIAL_TITANX_X_FOR_LP, _amountInfernoMin,
_deadline);
        inferno.approve(address(r), _infernoAmount);
        r.addLiquidity(
            address(inferno), address(alienX), _infernoAmount, INITIAL_ALIENX_FOR_LP, 0, 0, address(this),
_deadline
        );
    }

    addedLiquidity = true;
    _transferOwnership(address(0));
}
```

addLiquidity will first check if this pool exists and if not, will create it. Although anyone can create a pool before the AlienX code, just creating it is not a problem, but if he creates it and adds liquidity from one of the `TitanX` or `Inferno` tokens, then `createAndFundLPs` will always revert.

This can be achieved by anyone by donating the other token to the pair and then calling UniswapV2Pair.sync(), this will update the other token's reserve. Then when `addLiquidity()` is called, since the tokens reserves are not 0, it will enter the else here and then revert in UniswapV2Library.quote().

Recommendation: Before calling `addLiquidity()`, get the token reserves and if there is TitanX, mint AlienX, donate it to the pair and call `sync()`, making sure the reserves are the same before adding liquidity.

Resolution: Fixed. Since the initial pool liquidity is desirable to be a 2:1 (TitanX-AlienX) ratio, we conclude with the team to keep the 2:1 ratio by minting 2 times less AlienX to the pair when someone has donated TitanX to the pair..

5.2. High severity

5.2.1. Wrong init code hash in the `UniswapV2Library`

Severity: High Risk

Description: Init hash used in `AlienX` to precompute the pair addresses deviates from the one used in the original `UniswapV2Library`. As a result, the addresses that will be set as taxable will be different from the actual pairs that will be deployed later on:

```
constructor(address _inferno, address _titanX, address _v2Factory) ERC20("ALIENX", "ALX")
Ownable(msg.sender) {
    address infAlienXPool = UniswapV2Library.pairFor(_v2Factory, address(this), _inferno);
    address titanXAlienXPool = UniswapV2Library.pairFor(_v2Factory, address(this), _titanX);

    isTaxable[infAlienXPool] = true;
    isTaxable[titanXAlienXPool] = true;
}
```

Due to that the designated `2.08%` sell fee won't be taken when swapping from the UniswapV2 pair.

Recommendation: Use the original `UniswapV2Library::pairFor`, which uses the right init code hash, in order to compute the right pair addresses.

Resolution: Fixed

5.3. Low/Info severity

5.3.1. Initial liquidity is 30 million instead of the desired 30 billion

Severity: Low Risk

Description: The docs mention that the starting liquidity for both `TitanX/AlienX` and `Inferno/AlienX` pools is 30 billion, but in the code it's 30 million.

Recommendation: `INITIAL_ALIENX_FOR_LP` should be changed to `15_000_000_000e18`.

Resolution: Fixed

5.3.2. wrong initial titanX liquidity for AlienX/TitanX pool

Severity: Low Risk

Description: In the docs is mentioned that the `TitanX/AlienX` LP will be 20 billions and the `Inferno/AlienX` will be 10 billions. But now both are initialized equally with 15 million (Billions after L-01).

Recommendation: Change `createAndFundLPs` to create `TitanX/AlienX` LP with 20 billion and `Inferno/AlienX` LP with 10 billion.

Resolution: Fixed. At the end changed to 5 billion each.

5.3.3. dailyTitanXAllocation is wrong in TitanXBuyAndBurn

Severity: Low Risk

Description: `dailyTokenAllocation` for `TItanX/AlienX` BnB is wrong compared to docs. In the code, 0 will represent Sunday, but the implementation assumes it will be represented by 1.

```
function getDailyTokenAllocation(uint32 timestamp) public pure override returns (uint64 dailyWadAllocation)
{
    uint256 weekDay = weekDayByT(timestamp);

    if (weekDay == 1 || weekDay == 2) {
        dailyWadAllocation = 0.008e18; // 0.8%
    } else if (weekDay == 3) {
        dailyWadAllocation = 0.017e18; // 1.7%
    } else if (weekDay == 4 || weekDay == 5) {
        dailyWadAllocation = 0.026e18; // 2.6%
    } else {
        dailyWadAllocation = 0.028e18; // 2.8%
    }
}
```

Recommendation: Change the if checks to start with `if (weekDay == 0 || weekDay == 1)` and so on.

Resolution: Fixed

5.3.4. If 30bn TitanX are not collected in the first cycle, anyone can brick the LP pools

Severity: Low Risk

Description: According to docs, it's expected 30bn TitanX to be accumulated in the first minting cycle in order to create and fund the LPs. If the target is not achieved during the cycle, anyone will be able to claim his `AlienX` tokens and create pairs before the admin. As a result, funding the pairs from the `Minting` contract won't be possible due to the hardcoded amounts.

Recommendation: This can serve as a note since it's almost impossible target to not be achieved on the first day but still big harm can be done if that happens.

Resolution: Acknowledged

5.3.5. Dust TitanX will be leftover after _distribute

Severity: Low Risk

Description: 100% of the `TitanX` used to mint `AlienX` is distributed across other project, but the problem is that it will be accumulating dust in the `Minting` contract since there can be some sort of a reminder in either one of the divisions performed.

Recommendation: Consider distributing to the leftovers to the most important vault.

Resolution: Acknowledged

5.3.6. BnB contracts will emit wrong event for the AlienX swaps

Severity: Low Risk

Description: Because `AlienX` has a fee on transfer when it comes from LP or is deposited there. `_swapTitanXForAlienX` and `_swapInfernoForAlienX` return the amount before the fee because that is the amount calculated in `UniswapV2Library.getAmountsOut()` which is before the fees.

But since `burnAlienX` always burn `balanceOf(address(this))`, this will just give the wrong event data.

Recommendation: Change the `BuyAndBurn` event to emit `alienX.balanceOf(address(this))` instead of `alienXAmount` and use `UniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens()` for the swaps.

Resolution: Fixed

5.3.7. missing tx.origin check

Severity: Low Risk

Description: `swapTitanXForInfernoAndSendToVault()` has no `tx.origin` check compared to all other TitanX projects and their BnB functions.

Recommendation: Consider adding one to protect from contract to call the functions with the idea of MEV operations.

Resolution: Fixed

5.3.8. Add function to change twapLookback

Severity: Low Risk

Description: The `twapLookback` used in `getInfernoQuoteForTitanX` is hardcoded to 15 minutes.

Recommendation: Add a function to allow the admin to change it.

Resolution: Acknowledged

5.3.9. `AlienX` added as initial liquidity will be taxed

Severity: Low Risk

Description: `Inferno/AlienX` and `TitanX/AlienX` pairs are precomputed and added as taxable, the problem is that the `Minting` contract will also be taxed when providing the initial liquidity. As a result, instead of the desired 15bn `AlienX` supplied to each of the pairs, we will have ~14.688bn (**2.08% sell tax**).

Recommendation: Either set the pair addresses as taxable after they are created and funded from the `Minting` contract or mint `30bn + 4.16%` in `Minting::createAndFundLPs`.

Resolution: Fixed