SB SECURITY

Diamonds

Security Review

Oct 13, 2025

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	Diamonds
Commit Hash	1151c2fc09261f7c57844cc193ba7bb0b20e64b8
Resolution	483a8127ef709fef81ed75b184494a29603c2d4b
Timeline	October 9 - October 11, 2025

Scope

CertificatesVault.sol, Diamonds.sol, DiamondsBlazeStake.sol, DiamondsBuyCrush.sol, DiamondsCertificates.sol, DiamondsExchange.sol, DiamondsLpInjector.sol, DiamondsOracle.sol, DiamondsVault.sol

Post Audit Condition

The codebase is well-structured and follows solid development practices. No Critical/High severity issues were found.

5. Findings

Issues Found

Critical Risk	0
High Risk	0
Medium Risk	0
Low/Info Risk	3

ID	Title	Severity	Resolution
[L-01]	blaze snapshot must sync Blaze/TitanX pair 1st	Low	Acknowledged
[L-02]	burn bonus cap can trick users into burning more than the bonus	Low	**Fixed**
[L-03]	LP fees not collected before injection	Low	Acknowledged

Low severity

[L-01] blaze snapshot must sync Blaze/TitanX pair 1st

Severity: Low Risk

Description: Sinceprice1CumulativeLast will be incorrect if there's a donation to the pair and no mints, burns or swaps are performed. The reserves will not be up to date and the price will be incorrect. You can read more here. - https://rareskills.io/post/twap-uniswap-v2#only-calculating-the-last-1-hour-twap-in-solidity.

Recommendation: To fix it call `Pair.sync()` in `DiamondsOracle.currentCumulativePrice()`.

Resolution: Acknowledged

[L-02] burn bonus cap can trick users into burning more than the bonus

Severity: Low Risk

Description: burn bonus cap to 3x the `stakeAmount` will unfairly make users burn more `diamonds` than they bonus will be:

```
function _calculateBurnBonus(uint256 stakeAmount, uint256 burnAmount, uint16 lengthBonus)
    internal
    pure
    returns (uint256)
{
    uint256 maxAllowed = stakeAmount * 3;
    uint256 bonusShares = burnAmount > maxAllowed ? maxAllowed : burnAmount;
    bonusShares += _applyBps(bonusShares, lengthBonus);
    return _applyBps(bonusShares, BURN_BONUS_MULTIPLIER);
}
```

This way, someone burning `maxAllowed` and someone burning 10x more will get the same `bonusShares`, despite contributing to the `DIAMONDS` inflation 10x more.

Recommendation: Cap the burn amount to the `maxAllowed`.

Resolution: Fixed

[L-03] LP fees not collected before injection

Severity: Low Risk

Description: There's no check for pending LP fees in the `DiamondsLPInjector` and new injects are allowed to happen while there are pending unclaimed fees. This will delay them in being reinvested and grow the position's liquidity.

```
function inject(uint256 minAmountOut, uint256 minAmountAddX28, uint256 minAmountAddDiamonds, uint256
deadline)
    external
{
    if (!isActive()) revert InjectorInactive();
    CallParams storage p = params[X28];
    uint64 currentT = uint64(block.timestamp);
    IERC20 x28 = IERC20(X28);
    (uint256 x28Amount, uint256 incentive, uint256 nextAvailable) = _processParams(p,
x28.balanceOf(address(this)));
    if (currentT < nextAvailable) revert Cooldown();
    if (x28Amount == 0) revert InsufficientBalance();

    p.lastPerformed = currentT;
    x28Amount -= incentive;
    uint256 swapAmount = _applyBps(x28Amount, INJECT_SWAP_BPS);
    /// Account for price impact of the swap to use 100% of Diamonds
    x28Amount -= swapAmount;

    DIAMONDS_ORACLE.x28DiamondsTwapCheck(swapAmount, minAmountOut);
    _handleV3Swap(X28, DIAMONDS, swapAmount, minAmountOut, deadline);

    IERC20 diamonds = IERC20(DIAMONDS);
    uint256 diamondsBalance = diamonds.balanceOf(address(this));
    diamonds.safeIncreaseAllowance(UNISWAP_POS_MANAGER, diamondsBalance);
    x28.safeIncreaseAllowance(UNISWAP_POS_MANAGER, x28Amount);

    uint256 token0Amount = diamondsBalance;
    uint256 token1Amount = x28Amount;
    uint256 amount0Min = minAmountAddDiamonds;
    uint256 amount1Min = minAmountAddX28;
    if (!isToken0) {
        token0Amount = x28Amount;
        token1Amount = diamondsBalance;
        amount0Min = minAmountAddX28;
        amount1Min = minAmountAddDiamonds;
    }
    INonfungiblePositionManager.IncreaseLiquidityParams memory increaseParams = INonfungiblePositionManager
        .IncreaseLiquidityParams({
        tokenId: LP_TOKEN_ID,
        amount0Desired: token0Amount,
        amount1Desired: token1Amount,
        amount0Min: amount0Min,
        amount1Min: amount1Min,
        deadline: deadline
    });
    INonfungiblePositionManager(UNISWAP_POS_MANAGER).increaseLiquidity(increaseParams);
    x28.safeTransfer(msg.sender, incentive);
    emit Injection();
}
```

Recommendation: Ensure all the fees are claimed before reinvesting in the LP.

Resolution: Acknowledged

Notes

Severity: Informational Risk

Description:

1. Don't forget to give valid `PROTOCOL_START` and `MIT_START` timestamps

2. getTotalPrice can be simplified to assigned to 50% not subtracting 50%

3. In `ethBlazeTwapCheck` `blazeDeviation` should be reversed for better readability

4. Big whales can get part of their `Diamonds` tokens lost, because of the max uint check. Better revert instead and indicate they have to open multiple stakes, instead of fitting it to the uint128.

Resolution: Fixed - 3, 4