marchev

Inferno Buy-and-Burn V2 Security Review

Version 1.0

Reviewed by:

Martin Marchev

October 12, 2024

Table of Contents

1 Introduction **3**
1.1 About Martin Marchev 3
1.2 Disclaimer 3
1.3 Risk Classification 3
2 Executive Summary **4**
2.1 About Inferno Buy-and-Burn V2 4
2.2 Synopsis 4
2.3 Issues Found 4
3 Findings **5**

1 Introduction

1.1 About Martin Marchev

Martin Marchev is an independent security researcher specializing in web3 security. His track record includes top placements in competitive audits such as 1st place in the Immunefi Arbitration contest, 2nd place in the PartyDAO contest (as a team), 3x Top 10 and 5x Top 25 rankings. Martin has responsibly disclosed vulnerabilities in live protocols on Immunefi and has been involved in high-profile audits of projects exceeding $1B in Total Value Locked (TVL), demonstrating his ability to navigate and address complex security challenges. For bookings and security review inquiries, you can reach out on Telegram: https://t.me/martinmarchev

1.2 Disclaimer

While striving to deliver the highest quality web3 security services, it is important to understand that the complete security of any protocol cannot be guaranteed. The nature of web3 technologies is such that new vulnerabilities and threats may emerge over time. Consequently, no warranties, expressed or implied, are provided regarding the absolute security of the protocols reviewed. Clients are encouraged to maintain ongoing security practices and monitoring to address potential risks.

1.3 Risk Classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Informational

1.3.1 Severity Criteria

- **Critical** - Severe loss of funds, complete compromise of project availability, or significant violation of protocol invariants.
- **High** - High impact on funds, major disruption to project functionality or substantial violation of protocol invariants.
- **Medium** - Moderate risk affecting a noticeable portion of funds, project availability or partial violation of protocol invariants.
- **Low** - Low impact on a small portion of funds (e.g. dust amount), minor disruptions to service or minor violation of protocol invariants.
- **Informational** - Negligible risk with no direct impact on funds, availability or protocol invariants.

2 Executive Summary

2.1 About Inferno Buy-and-Burn V2

A new Buy-and-Burn contract for the Inferno token.

2.2 Synopsis

Project Name	Inferno Buy-and-Burn V2
Project Type	ERC-20
Repository	inferno-bnb-v2-contracts
Commit Hash	b97c96e4...4d4eb2e7
Review Period	11 October 2024 - 12 October 2024

2.3 Issues Found

Severity	Count
Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	2
Informational	4
Total Issues	**6**

3 Findings

Low Risk

[L-1] Misleading slippage configuration naming may lead to incorrect configuration being set

Context: SwapActions.sol#L112

Description: Slippage typically refers to *the difference* between the expected price of a trade and the actual price at which the trade executes. It denotes how much the price can "slip". Slippage is generally expressed as a percentage, e.g. if a slippage of 1% is set, this means a price up to 1% worse than the expected price is acceptable.

However, the current implementation uses the term `slippage` to describe the actual price as a percentage of the expected instead of the difference. This non-standard naming may lead to confusion and incorrect slippage configuration being set.

Recommendation: Modify the implementation so that the `slippage` configuration property is used to denote the acceptable difference (as a percentage) between the TWAP price and the actual trade price.

The acceptable price should be calculated as follows:

$$Acceptable\ Price = TWAP\ Price - Slippage \times TWAP\ Price$$

Resolution: Acknowledged

[L-2] Missing slippage validation in `changeSlippageConfig()` may lead to lack of slippage protection

Context: SwapActions.sol#L112

Description: Unlike the `_newLookBack`, the `changeSlippageConfig()` function does not have any validation for `_newSlippage`. This may lead to lack of slippage protection in the scenario where the lookback period is correct set but `_newSlippage` protection is set to 0.

In the above scenario, the contract implementation will not use the default slippage of 20% which is enforced by the following check because the lookback period would be greater than 0:

```
if (slippageConfig.twapLookback == 0 && slippageConfig.slippage == 0) {
    slippageConfig = Slippage({twapLookback: 15, slippage: WAD - 0.2e18});
}
```

This would effectively leave the contract without slippage protection in the described scenario.

Recommendation: Implement validation for the `_newSlippage` parameter in `changeSlippageConfig()`

Resolution: Fixed in 7ad43a

Informational

[I-1] Missing event emissions for state-changing functions

Context: InfernoBuyAndBurnV2.sol#L49, InfernoBuyAndBurnV2.sol#L53

Description: It is recommended for state-changing functions to emit events for off-chain monitoring purposes. The `changeIntervalBetweenBurns()` and `changeSwapCap()` in the `InfernoBuyAndBurnV2` contract modify important contract state variables but do not emit events.

Recommendation: Emit events when changing the interval between burns and the swap cap respectively in the `changeIntervalBetweenBurns()` and `changeSwapCap()` functions.

Resolution: Acknowledged

[I-2] `BuyAndBurn` may report incorrect burned Inferno amount in some cases

Context: InfernoBuyAndBurnV2.sol#L68

Description: The `InfernoBuyAndBurnV2.buyNBurn()` function emits a `BuyAndBurn` event with the Inferno amount that has been burned:

```
uint256 infernoAmount = swapExactInputV3(...);
burnInferno();
emit BuyAndBurn(infernoAmount);
```

However, the `burnInferno()` function burns the whole Inferno balance that belongs to the contract. In some rare cases the contract balance could be greater than the swapped amount, e.g. if someone directly sent Inferno to the buy-and-burn contract (either mistakenly or purposely). In the case, the burned amount would be incorrectly reported as the swapped Inferno amount instead of the contract balance.

Recommendation: Use the `InfernoBuyAndBurnV2` contract Inferno balance as an argument when emitting the `BuyAndBurn` event in `buyNBurn()`:

```
@@ -65,7 +65,7 @@ contract InfernoBuyAndBurnV2 is SwapActions {

         burnInferno();

-        emit BuyAndBurn(infernoAmount);
+        emit BuyAndBurn(erc20Bal(inferno));

         s.lastBurnTs = uint32(block.timestamp);
     }
```

Resolution: Acknowledged

[I-3] `InfernoBuyAndBurnV2` is not `Pausable`

Context: InfernoBuyAndBurnV2.sol#L16

Description: The `InfernoBuyAndBurnV2` contract is not pausable. Even if this addition will increase its centralization, it could be useful in case of an emergency. For example, if the contract becomes a target of MEV attacks the currently implemented mechanics are not able to provide adequate protection against MEV bots, pausability could be used as a security measure in such scenarios to prevent leaking value from the protocol.

Recommendation: Make the `InfernoBuyAndBurnV2` contract pausable as a mitigation against unforeseen MEV attack vectors.

Resolution: Acknowledged. Pause/delay could be achieved by configuring bigger interval between burns.

[I-4] Unused modifier `notGt` should be removed

Context: Errors.sol#L57

Description: The `notGt` modifier in `Errors.sol` is not used anywhere in the smart contract code. Although with newer versions of the Solidity compiler (0.8.19+), the optimizer removes unused code from the compiled smart contract bytecode, it is a good idea to remove any functions or modifiers that are unused to improve code readability and maintainability.

Recommendation: Remove the unused `notGt` modifier.

Resolution: Fixed in 7ad43a