WasabiPerps Security Assessment

Dec 17, 2023

Prepared for

Wasabi

Prepared by:

0xfoobar

Table of Contents

Summary

Overview

Wasabi is an NFT finance platform.

Project Scope

We reviewed the core smart contracts and test suite contained within commit hash `50c8c8ed37de5f1dcd31c585a63f524067f5f9c5` on branch `main` at https://github.com/DkodaLabs/wasabi_perps. Fixes were reviewed at commit hash `502c42e90430e93e2aa4bb0225a65da25dfd118a` on branch `main` at https://github.com/DkodaLabs/wasabi_perps.

Severity Classification

High - Assets can be stolen/lost/compromised directly (or indirectly if there is a valid attack path that does not have hand-wavy hypotheticals).
Medium - Assets not at direct risk, but the function of the protocol or its availability could be impacted, or leak value with a hypothetical attack path with stated assumptions, but external requirements.
Low - Assets are not at risk: state handling, function incorrect as to spec, issues with comments.
Informational - Code style, clarity, syntax, versioning, off-chain monitoring (events, etc)

Summary of Findings

Severity	Findings	Resolved
High	**0**	**-**
Medium	**0**	**-**
Low	**2**	**2**
Informational	**6**	**2**

Disclaimer

This security assessment should not be used as investment advice.

We do not provide any guarantees on eliminating all possible security issues. foostudio recommends proceeding with several other independent audits and a public bug bounty program to ensure smart contract security.

We did not assess the following areas that were outside the scope of this engagement:

- Website frontend components
- Offchain order management infrastructure
- Multisig or EOA private key custody
- Metadata generation

Key Findings and Recommendations

1. _msgSender() should be replaced with msg.sender

Severity: **Low**
Files: *.sol

Description

OpenZeppelin's _msgSender() is only useful in explicit cases of paymasters, and was recently involved in the Thirdweb exploit when combined with ERC2771 multicall. Since it adds unnecessary gas and potential future attack vectors without being intended to be used by a paymaster, it should be removed.

Impact

Increased gas costs, potential future exploits when combined into more complex codebase.

Recommendation

Replace with msg.sender.

Response

Fixed.

2. Make liquidation threshold dynamic

Severity: **Low**
Files: WasabiLongPool.sol, WasabiShortPool.sol,

Description

Margin of safety for leverage and debt depends on factors such as market depth, volatility, outstanding OI, etc. The liquidation threshold is currently hardcoded.

Impact

Potential protocol insolvency in future unexpected market conditions.

Recommendation

Make the liquidation threshold dynamic and adjustable - either by an algorithm, or market controllers.

Response

Contracts are upgradeable so this can be changed in the future. We want to start by experimenting with these parameters.

t

3. Use latest compiler version and associated features

Severity: **Informational**
Files: pool.sol

Description

All files currently have the solidity pragma at 0.8.17. Best practice is to target the latest 0.8.23 version. This will have three benefits: cheaper deployment costs, gas optimization and new language features. This protocol is targeting Ethereum, so there are no PUSH0 incompatibility issues. However if multichain expansion is a plan in the future, it'll be important to explicitly set the compiler's target EVM version as "paris" rather than "shanghai" to avoid PUSH0 opcode incompatibility.

Impact

Increased deployment size and gas costs.

Recommendation

Use a floating ^0.8.23 pragma everywhere.

Response

Fixed.

4. Reverting fallback function is unnecessary

Severity: **Informational**
Files: BaseWasabiPool.sol

Description

Fallback function is included which reverts always. This is unnecessary, since lack of a fallback function will already revert by default on mismatched calldata.

Impact

Increased deployment size and gas costs.

Recommendation

Remove it.

Response

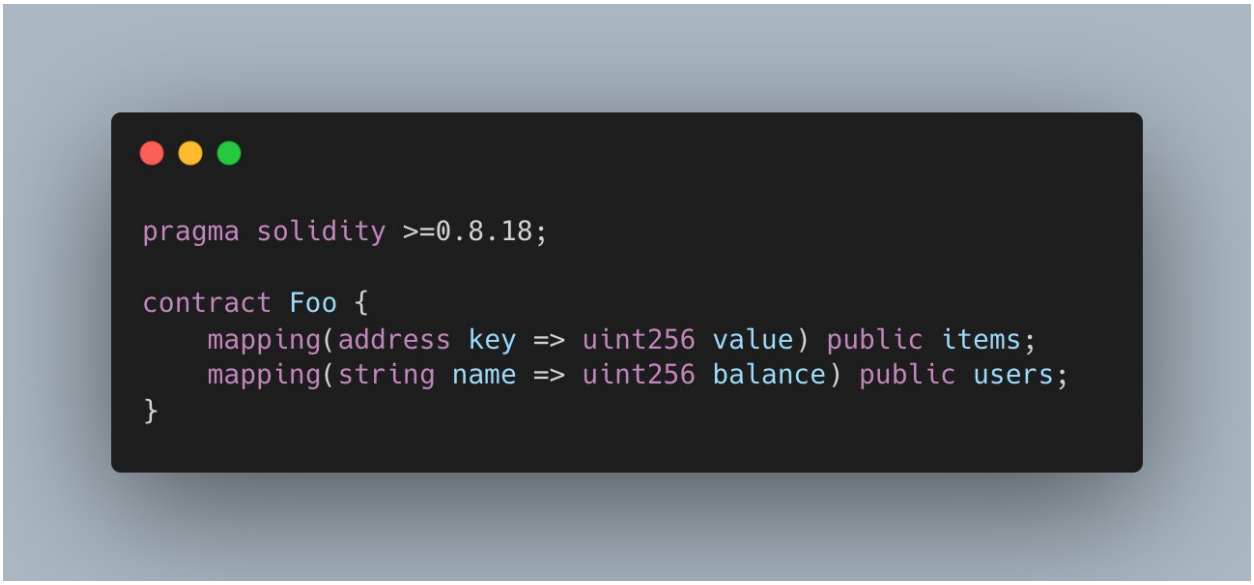
Fixed.

5. Use named mappings

Severity: **Informational**
Files: *.sol

Description

Solidity 0.8.18 and later support a language feature called named mappings, where keys and values can be named to give devs better documentation of what's expected in each.



```
pragma solidity >=0.8.18;

contract Foo {
    mapping(address key => uint256 value) public items;
    mapping(string name => uint256 balance) public users;
}
```

Impact

Extraneous comments that could be inlined.

Recommendation

Experiment with named mappings where the storage layout otherwise requires comment explanations.

Response

6. Fix Natspec

Severity: **Informational**
Files: *.sol

Description

The first word of Natspec return comments should be the name of the returned variable, instead of the first word of the description of it. Comments should be prefixed with @dev for internal/private functions, and prefixed with @notice for external/public functions. Etherscan will display @notice but not @dev on verified code read and write functions.

Impact

Cleaner autogenerated docs.

Recommendation

Update Natspec.

Response

7. Add multicall

Severity: **Informational**
Files: BaseWasabiPool.sol

Description

Currently you have single and plural methods for several actions - liquidatePosition & liquidatePositions. Users may also want to chain together multiple unrelated actions in a single transaction.

It makes sense to use a generalizable *nonpayable* multicall here instead, like so:

```
function multicall(bytes[] calldata data) external override returns (bytes[] memory results) {
        results = new bytes[](data.length);
        bool success;
        unchecked {
                for (uint256 i = 0; i < data.length; ++i) {
                        //slither-disable-next-line calls-loop,delegatecall-loop
                        (success, results[i]) = address(this).delegatecall(data[i]);
                        if (!success) revert MulticallFailed();
                }
        }
}
```

Impact

Cleaner UX for frontend batching implementation, less deployment codesize.

Response

8. Miscellaneous gas optimizations and comment fixes

Severity: **Informational**
Files: *.sol

Description

- Mark internal functions with a leading underscore
- DebtController dangling commented-out-code in line 23
- DebtController liquidationThreshold is never used
- Unchecked for-loop increments no longer beneficial in solidity 0.8.22+, should also use precrements instead of increments in for-loops

Response