Opium protocol audit

25 Jan 2022, Igor Gulamov

Introduction

Igor Gulamov conducted the audit of Opium protocol smart contracts.

This review was performed by an independent reviewer under fixed rate.

Scope

https://github.com/OpiumProtocol/opium-protocol-v2/tree/24b2d653bade302d522a5b588595e9728f6f9995

contracts/core/Core.sol
contracts/core/OpiumProxyFactory.sol
contracts/core/OpiumPositionToken.sol
contracts/core/OracleAggregator.sol
contracts/core/TokenSpender.sol
contracts/core/SyntheticAggregator.sol
contracts/core/registry/Registry.sol
contracts/core/base/RegistryManager.sol

Disclaimer

The audit gives no warranties for the security of the code. We recommend passing multiple audits for more safety. Also, this report is corresponding to the state of the source code at the date of publication and any future changes should be re-audited.

Issues

We found no critical and major issues.

We consider commit 3f89b8d9b6c9a3e67311fd379cfb81556a4b6bfb as a safe version from the informational security point of view.

Warnings

OracleAggregator.sol#L25

The contract could be optimized to use only one storage slot for most cases.

We will not implement it, as oracle data provision only happens once at the end of derivative maturity and doesn't bring additional gas costs to the users. The possible optimization solutions are only make the OracleAggregator more restrictive.

Core.sol#L447

Using `safeTransferFrom` directly is more gas effective.

We will keep using TokenSpender as it proved to be a good way to hold users' token approvals and ease the potential pause or upgrades of the core contracts

Core.sol#L90

Using memory allocated variable `_registry` here will be more gas efficient.

Fixed

Core.sol#L523

The description is not corresponding to the implementation.

Fixed

OpiumProxyFactory.sol#L74

Collision on `derivativeHashSlice` could be easily forged. So, an attacker could mint two different pairs of derivatives with the same name. We propose using a unique counter for the names.

We will not fix this, `derivativeHashSlice` is not and not supposed to be used in the codebase, so it doesn't bring any security threats. It's only implemented for Wallets and Explorers. Possible solution will require to spend additional gas.

Comments

SyntheticAggregator.sol#L44

SyntheticAggregator.sol#L56

Core.sol#L727

Function name looks like getter, but the state is mutable.

Fixed

Core.sol#L706-L715

Parentheses are unnecessary.

Fixed

Severity Terms

Comment

Comment issues are generally subjective in nature, or potentially deal with topics like "best practices" or "readability". Comment issues in general will not indicate an actual problem or bug in code.

The maintainers should use their own judgment as to whether addressing these issues improves the codebase.

Warning

Warning issues are generally objective in nature but do not represent actual bugs or security problems.

These issues should be addressed unless there is a clear reason not to.

Major

Major issues will be things like bugs or security vulnerabilities. These issues may not be directly exploitable, or may require a certain condition to arise in order to be exploited.

Left unaddressed these issues are highly likely to cause problems with the operation of the contract or lead to a situation which allows the system to be exploited in some way.

Critical

Critical issues are directly exploitable bugs or security vulnerabilities.

Left unaddressed these issues are highly likely or guaranteed to cause major problems or potentially a full failure in the operations of the contract.