SmartDec

Opium Smart Contracts Security Analysis

This report is public.

Published: January 28, 2020.

Abstract

In this report, we consider the security of the Opium project. Our task is to find and describe security issues in the smart contracts of the platform.

Disclaimer

The audit does not give any warranties on the security of the code. One audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Besides, security audit is not an investment advice.

Summary

In this report, we considered the security of Opium smart contracts. We performed our audit according to the procedure described below.

The audit showed no critical issues. However, three medium severity and a number of low severity issues were found.

We discussed them with the developers, so in the latest version of the code they fixed most of the issues found in the audit and provided their comments on the rest of the issues.

General recommendations

The contracts code is of good code quality. Thus, we do not have any additional recommendations.

Checklist

Security

The audit showed no vulnerabilities.
Here by vulnerabilities we mean security issues that can be exploited by an external attacker. This does not include low severity issues, documentation mismatches, overpowered contract owner, and some types of bugs.

Compliance with the documentation

The audit showed no discrepancies between the code and the provided documentation.

Tests

All the tests passed successfully.

The text below is for technical use; it details the statements made in Summary and General recommendations.

Procedure

In our audit, we consider the following crucial features of the smart contract code:

1. Whether the code is secure.
2. Whether the code corresponds to the documentation (including whitepaper).
3. Whether the code meets best practices in efficient use of gas, code readability, etc.

We perform our audit according to the following procedure:

- automated analysis
  - we scan project's smart contracts with our own Solidity static code analyzer SmartCheck
  - we scan project's smart contracts with several publicly available automated Solidity analysis tools such as Ethlint and Solhint
  - we manually verify (reject or confirm) all the issues found by tools
- manual audit
  - we manually analyze smart contracts for security vulnerabilities
  - we check smart contracts logic and compare it with the one described in the documentation
  - we run tests
- report
  - we reflect all the gathered information in the report

Checked vulnerabilities

We have scanned Opium smart contracts for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered (the full list includes them but is not limited to them):

- Reentrancy
- Front running
- DoS with (unexpected) revert
- DoS with block gas limit
- Gas limit and loops
- Locked money
- Integer overflow/underflow
- Unchecked external call
- ERC20 Standard violation
- Authentication with tx.origin
- Unsafe use of timestamp
- Using blockhash for randomness
- Balance equality
- Unsafe transfer of ether
- Fallback abuse
- Using inline assembly
- Short address attack
- Private modifier
- Compiler version not fixed
- Style guide violation
- Unsafe type deduction
- Implicit visibility level
- Use delete for arrays
- Byte array
- Incorrect use of assert/require
- Using deprecated constructions

Project overview

Project description

In our analysis we consider Opium specification (”docs/index.md” in the project repository) and smart contracts’ code (private repository, version on commit 60ff6f80996b83f6ad19c35b74480fef34f7dc03). Also, the main dependency – erc721o (version on commit 60723b713e3f3f9d98d71b800673e0b4696fe108) was analysed.

The latest version of the code

After the initial audit, some fixes were applied and the code was updated to the latest version (commit 8c7d076fa0ee958a82134c32412d972452746dff).

Project architecture

For the audit, we were provided with the truffle project. The project is an npm package and includes tests.

- The project successfully compiles with `truffle compile` command (with some warnings, see Compilation output in Appendix)
- The project successfully passes all the tests, however, code coverage was not generated

The total LOC of audited Solidity sources is 1569.

Automated analysis

We used several publicly available automated Solidity analysis tools. Here are the combined results of SmartCheck, Solhint, and Ethlint scanning. All the issues found by tools were manually checked (rejected or confirmed).

True positives are constructions that were discovered by the tools as vulnerabilities and can actually be exploited by attackers or lead to incorrect contracts operation.

False positives are constructions that were discovered by the tools as vulnerabilities but do not consist a security threat.

Cases when these issues lead to actual bugs or vulnerabilities are described in the next section.

Tool	Rule	True positives	False positives
Solhint	Avoid to make time-based decisions in your business logic		7
	Avoid to use inline assembly. It is acceptable only in rare cases		3
	Compiler version 0.5.4 does not satisfy the 0.5.10 semver requirement		32
	Compiler version must be fixed	32	
	Contract name must be in CamelCase	2	
	Explicitly mark visibility of state	10	
	Variable name must be in mixedCase	1	
Total Solhint		45	42
Ethlint	Avoid using ’now’ (alias to ’block.timestamp’).		7
	Avoid using experimental features in production code	12	1
	Code contains empty block		4

	Visibility modifier ”internal” should come before other modifiers.		1
Total Ethlint		12	13
SmartCheck	Compiler version not fixed	34	
	Extra gas consumption	1	
	Implicit visibility level	10	
	Prefer external to public visibility level	42	2
	Private modifier		6
	Pure-functions should not read/change state		2
	Use of assembly		3
	Use of SafeMath		4
	View-function should not change state		1
	Costly loop	2	
Total SmartCheck		89	18
Total Overall		**146**	**73**

Manual analysis

The contracts were completely manually analyzed, their logic was checked and compared with the one described in the documentation. Besides, the results of the automated analysis were manually verified. All the confirmed issues are described below.

Critical issues

Critical issues seriously endanger smart contracts security. We highly recommend fixing them.

The audit showed no critical issues.

Medium severity issues

Medium issues can influence smart contracts operation in current implementation. We highly recommend addressing them.

Overpowered role

The `governor` address is responsible for whitelisting addresses in **TokenSpender** contract. That means that `governor` address is responsible for Opium Token transfers.

So, the system depends heavily on `governor` address in the current implementation. In this case, there are scenarios that may lead to undesirable consequences for investors, e.g. if the `governor`'s private keys become compromised. Thus, we recommend designing contracts in a trustless manner.

Comment from the developers: "TokenSpender contract is used to hold user's allowances on ERC20 contracts and allows only whitelisted contracts to spend tokens. For purpose of users security we introduced "time locks", which notify users when content of whitelist is about to change and doesn't allow to change it before specific moment in future, so users could verify newly proposed whitelist and decide whether they want to keep their allowance or not."

Bug

There is a bug in **SwaprateMatchBase.sol** file, line 62. `address(0)` is used instead of `_token` address to transfer tokens:

```
IERC20(address(0)).transfer(msg.sender, balance);
```

We recommend fixing this bug (i.e. calling `_token` address).

The issue has been fixed and is not present in the latest version of the code.

Deployment process

There is no deployment script in the project. However, the contracts deployment does not seem trivial. Bugs and vulnerabilities often appear in deployment scripts and severely endanger system's security.

We highly recommend developing and testing deployment scripts very carefully.

The issue has been fixed and is not present in the latest version of the code.

Moreover, there are a lot of setter-functions in **Registry.sol** (lines 44–89).

We recommend moving the complexity of the project deployment and initialization from smart contracts to deployment scripts in order to reduce gas costs, and removing `set***()` functions from ABI, as they will be used only once. Bounded smart contracts can be deployed as described in this article.

Comment from the developers: "We didn't implement bounding contracts like described in your article, but eliminated all setter to one init() function in Registry contract."

Low severity issues

Low severity issues can influence smart contracts operation in future versions of code. We recommend taking them into account.

Compiler version

Solidity source files indicate the versions of the compiler they can be compiled with.

```
Example:
pragma solidity ^0.5.4; // bad: compiles w 0.5.4 and above
pragma solidity 0.5.4; // good: compiles w 0.5.4 only
```

We recommend following the latter example, as future compiler versions may handle certain language constructions in a way the developers have not foreseen. Besides, we recommend using the latest compiler version – 0.5.16 at the moment.

The issue has been fixed and is not present in the latest version of the code.

Gas usage

Gas usage may be optimized in several places in the project:

- **WhitelistedWithGovernance.sol**, line 53
  `whitelist` variable is used as the `Committed` event argument.

  We recommend using `_whitelist` variable instead.

  The issue has been fixed and is not present in the latest version of the code.

- **WhitelistedWithGovernance.sol**, line 59
  `proposedWhitelist` variable is used as the `Proposed` event argument.

  We recommend using `_whitelist` variable instead.

  The issue has been fixed and is not present in the latest version of the code.

- **Core.sol**, line 521
  `registry.getOpiumAddress()` is called multiple times.

  We recommend saving it to the local memory after the first call.

  The same issue arises at **MatchLogic.sol**, lines 159, 166.

  The issues have been fixed and are not present in the latest version of the code.

- **MatchLogic.sol**, line 151
  `registry.getTokenSpender()` is called multiple times.

  We recommend saving it to the local memory after the first call.

  The issue has been fixed and is not present in the latest version of the code.

- **Whitelisted.sol**, line 14
  `whitelist.length` is read in a loop.

  We recommend saving it to the local memory.

  The issue has been fixed and is not present in the latest version of the code.

- **Whitelisted.sol**, line 9
  `onlyWhitelisted()` modifier consumes a lot of gas due to lots of storage read operations. Keep in mind, that the price of reading storage variables can be changed in the future. For example, it was increased 4 times in the Istanbul hardfork.

Code logic

- **WhitelistedWithGovernance.sol**, line 50
  `initialized` variable is used to check whether whitelist has been initialized or not.

  We recommend using `whitelist.length` instead.

  The issue has been fixed and is not present in the latest version of the code.

- **SyntheticAggregator.sol**, line 119
  `_derivativeHash` is a function argument. However, hash of the derivative is calculated again.

  We recommend not passing it as a function's argument and just calculating it instead.

  Comment from the developers: "In places, where this function is called internally we consider _derivativeHash as trusted and use it to retrieve Ticker state without recalculating the hash and optimise gas usage."

- **MatchSwap.sol**, lines 137, 143
  When calculating order's filled percentage, 1 is added to every calculated value.

  We recommend removing these additions.

  The issue has been fixed and is not present in the latest version of the code.

Missing input validation

There are input validation issues in several places:

- **WhitelistedWithGovernance.sol**, line 81
  In case `_governor` argument is zero address, whitelisting functionality will be lost.

- **Registry.sol**, line 86
  In case `_opiumAddress` argument is zero address, fees will be locked.

We recommend adding check that `_governor` and `_opiumAddress` arguments are non-zero.

The issues have been fixed and are not present in the latest version of the code.

Redundant code

The following lines are redundant:

- **WhitelistedWithGovernanceAndChangeableTimelock.sol**, line 13

```
uint256 timelockProposalTime = 0;
```

- **WhitelistedWithGovernanceAndChangeableTimelock.sol**, line 15

```
uint256 proposedTimelock = 0;
```

- **WhitelistedWithGovernance.sol**, line 22

```
bool public initialized = false;
```

- **WhitelistedWithGovernance.sol**, line 25

```
uint256 public proposalTime = 0;
```

At these lines, the initial values are reassigned.

We highly recommend removing redundant code in order to improve code readability and transparency and decrease cost of deployment.

The issues have been fixed and are not present in the latest version of the code.

Visibility level

`registry` variable of **usingRegistry** contract has `private` visibility level without getter function. This may affect user experience in a negative way as users cannot easily check the `registry` address.

The issue has been fixed and is not present in the latest version of the code.

Implicit visibility level

The following variables have an implicit visibility level:

1. **ExecutableByThirdParty.sol**, line 6

```
mapping (address => bool) thirdpartyExecutionAllowance;
```

2. **LibCommission.sol**, line 6

```
uint256 constant COMMISSION_BASE = 10000;
```

3. **LibCommission.sol**, line 9

```
uint256 constant OPIUM_COMMISSION_BASE = 10;
```

4. **LibCommission.sol**, line 12

```
uint256 constant OPIUM_COMMISSION_PART = 1;
```

5. **WhitelistedWithGovernanceAndChangableTimelock.sol**, line 13

```
uint256 timelockProposalTime = 0;
```

6. **WhitelistedWithGovernanceAndChangableTimelock.sol**, line 15

```
uint256 proposedTimelock = 0;
```

7. **SwaprateMatch.sol**, line 17

```
mapping (bytes32 => uint256) filled;
```

8. **SwaprateMatchBase.sol**, line 31

```
mapping (bytes32 => bool) canceled;
```

9. **SwaprateMatchBase.sol**, line 36

```
mapping (bytes32 => bool) verified;
```

10. **SwaprateMatchBase.sol**, line 44

```
mapping (address => mapping (address => uint256)) public
balances;
```

We recommend specifying visibility levels (`public`, `private`, or `internal`) explicitly and correctly in order to improve code readability.

The issues have been fixed and are not present in the latest version of the code.

Misleading name

`validateCanceled()` function of **SwaprateMatchBase** contract checks whether the order is not canceled.

We recommend renaming it to `validateNotCanceled()`.

The issue has been fixed and is not present in the latest version of the code.

Transfer of an unknown ERC20 tokens

Return value of `transfer()` function is ignored in the following places:

- **MatchLogic.sol**, line 70
- **Core.sol**, line 64
- **SwaprateMatchBase.sol**, line 62

According to ERC20 token standard:

> `The function SHOULD throw if the message caller's account balance does not have enough tokens to spend.`

Since the code of this token contract is unknown, `transfer()` function call may not revert in the case of unsuccessful token transfer. We recommend using **SafeERC20** contract from **OpenZeppelin** library.

The issues have been fixed and are not present in the latest version of the code.

Code duplications

There are code duplications in the contracts, for example at **MatchSwap.sol**, lines 133–137 and 139–143.

We recommend refactoring duplicated parts of code.

Comment from the developers: "We consider refactoring not feasible, thus we agreed to keep it as is."

Constant state variable

`commission` variable at **HasCommission.sol**, line 8 is never changed.
We recommend adding `constant` keyword and naming it in UPPER_CASE.

The issue has been fixed and is not present in the latest version of the code.

Prefer external to public visibility level

In the code, there are functions with the `public` visibility level that are not called internally.

We recommend changing visibility level of such functions to `external` in order to improve code readability. Moreover, in many cases functions with `external` visibility modifier require less gas comparing to functions with `public` visibility modifier.

Comment from the developers: "We consider refactoring not feasible, thus we agreed to keep it as is in most of the places where issue occurs."

Notes

Naming convention

- **WhitelistedWithGovernance.sol**, line 16
  `TIME_LOCK_INTERVAL` state variable is not constant, but is named in UPPER_CASE_WITH_UNDERSCORES.

  We recommend naming it in mixedCase.

- **usingRegistryErrors.sol**, line 3
  usingRegistryErrors contract is named in mixedCase.

  We recommend naming it in CapWords.

- **usingRegistry.sol**, line 8
  usingRegistry contract is named in mixedCase.

  We recommend naming it in CapWords.

The issues have been fixed and are not present in the latest version of the code.

Gas limit

The assign operations at **WhitelistedWithGovernance.sol**, lines 52, 58, and 72 are costly as the whole arrays are being copied. If there are too many items in these arrays, the execution of the corresponding functions will fail due to an out-of-gas exception.

Comment from the developers: "By conducting tests we consider refactoring not feasible, thus we agreed to keep it as is."

Avoid using experimental features in production code

`ABIEncoderV2` is used in the code. We do not recommend using experimental features in the code since they might contain bugs that will only be fixed in future versions of the compiler.

Comment from the developers: ”We are widely using passing structures as function arguments, thus ABIEncoderV2 is required for us and we can’t avoid it.”

Gas limit and loops

The following loops traverse through arrays of variable length:

- **Core.sol**, line 300

```
for (uint256 i; i < _tokenIds.length; i++)
```

- **Core.sol**, line 344

```
for (uint256 i; i < _tokenIds.length; i++)
```

The traversed arrays are passed as parameters of the functions. Therefore, if there are too many items in these arrays, the execution of the corresponding functions will fail due to an out-of-gas exception.

In these cases, we recommend separating the calls into several transactions.

This analysis was performed by SmartDec.

Alexander Seleznev, Chief Business Development Officer
Igor Sobolev, Security Analyst
Pavel Kondratenkov, Security Analyst

January 28, 2020

Appendix

Compilation output

```
Compiling your contracts...
===========================
> Compiling ./contracts/Core.sol
> Compiling ./contracts/Errors/CoreErrors.sol
> Compiling ./contracts/Errors/MatchingErrors.sol
> Compiling ./contracts/Errors/OracleAggregatorErrors.sol
> Compiling ./contracts/Errors/RegistryErrors.sol
> Compiling ./contracts/Errors/SyntheticAggregatorErrors.sol
> Compiling ./contracts/Errors/usingRegistryErrors.sol
> Compiling ./contracts/Helpers/ExecutableByThirdParty.sol
> Compiling ./contracts/Helpers/HasCommission.sol
> Compiling ./contracts/Interface/IDerivativeLogic.sol
> Compiling ./contracts/Interface/IOracleId.sol
> Compiling ./contracts/Lib/LibCommission.sol
> Compiling ./contracts/Lib/LibDerivative.sol
> Compiling ./contracts/Lib/LibEIP712.sol
> Compiling ./contracts/Lib/Whitelisted.sol
> Compiling ./contracts/Lib/WhitelistedWithGovernance.sol
> Compiling ./contracts/Lib/WhitelistedWithGovernanceAndChan
gableTimelock.sol
> Compiling ./contracts/Lib/usingRegistry.sol
> Compiling ./contracts/Matching/Match/LibOrder.sol
> Compiling ./contracts/Matching/Match/Match.sol
> Compiling ./contracts/Matching/Match/MatchCreate.sol
> Compiling ./contracts/Matching/Match/MatchLogic.sol
> Compiling ./contracts/Matching/Match/MatchPool.sol
> Compiling ./contracts/Matching/Match/MatchSwap.sol
> Compiling ./contracts/Matching/SwaprateMatch/LibSwaprateOr
der.sol
> Compiling ./contracts/Matching/SwaprateMatch/SwaprateMatch
.sol
> Compiling ./contracts/Matching/SwaprateMatch/SwaprateMatch
Base.sol
> Compiling ./contracts/Migrations.sol
> Compiling ./contracts/OracleAggregator.sol
> Compiling ./contracts/Registry.sol
> Compiling ./contracts/SyntheticAggregator.sol
> Compiling ./contracts/TokenMinter.sol
> Compiling ./contracts/TokenSpender.sol
```

```
> Compiling ./contracts/test/OptionCallSyntheticIdMock.sol
> Compiling ./contracts/test/OracleIdMock.sol
> Compiling ./contracts/test/TestToken.sol
> Compiling ./contracts/test/WETH.sol
> Compiling erc721o/contracts/ERC721OBackwardCompatible.sol
> Compiling erc721o/contracts/ERC721OBase.sol
> Compiling erc721o/contracts/ERC721OComposable.sol
> Compiling erc721o/contracts/ERC721OMintable.sol
> Compiling erc721o/contracts/ERC721OTransferable.sol
> Compiling erc721o/contracts/Interfaces/IERC721O.sol
> Compiling erc721o/contracts/Interfaces/IERC721OReceiver.so
l
> Compiling erc721o/contracts/Libs/LibPosition.sol
> Compiling erc721o/contracts/Libs/ObjectsLib.sol
> Compiling erc721o/contracts/Libs/UintArray.sol
> Compiling erc721o/contracts/Libs/UintsLib.sol
> Compiling openzeppelin-solidity/contracts/introspection/ER
C165.sol
> Compiling openzeppelin-solidity/contracts/introspection/IE
RC165.sol
> Compiling openzeppelin-solidity/contracts/math/SafeMath.so
l
> Compiling openzeppelin-solidity/contracts/token/ERC20/IERC
20.sol
> Compiling openzeppelin-solidity/contracts/token/ERC20/Safe
ERC20.sol
> Compiling openzeppelin-solidity/contracts/token/ERC721/IER
C721.sol
> Compiling openzeppelin-solidity/contracts/token/ERC721/IER
C721Receiver.sol
> Compiling openzeppelin-solidity/contracts/utils/Address.so
l
> Compiling openzeppelin-solidity/contracts/utils/Reentrancy
Guard.sol

> compilation warnings encountered:

./contracts/Lib/LibDerivative.sol:2:1: Warning: Experimenta
l features are turned on. Do not use experimental features o
n live deployments.
pragma experimental ABIEncoderV2;
^-------------------------------^
,./contracts/Interface/IDerivativeLogic.sol:2:1: Warning: Ex
perimental features are turned on. Do not use experimental f
```

```
eatures on live deployments.
pragma experimental ABIEncoderV2;
^------------------------------^
,./contracts/SyntheticAggregator.sol:2:1: Warning: Experimen
tal features are turned on. Do not use experimental features
on live deployments.
pragma experimental ABIEncoderV2;
^------------------------------^
,./contracts/Core.sol:2:1: Warning: Experimental features ar
e turned on. Do not use experimental features on live deploy
ments.
pragma experimental ABIEncoderV2;
^------------------------------^
,./contracts/Matching/Match/LibOrder.sol:2:1: Warning: Exper
imental features are turned on. Do not use experimental feat
ures on live deployments.
pragma experimental ABIEncoderV2;
^------------------------------^
,./contracts/Matching/Match/MatchLogic.sol:2:1: Warning: Exp
erimental features are turned on. Do not use experimental fe
atures on live deployments.
pragma experimental ABIEncoderV2;
^------------------------------^
,./contracts/Matching/Match/MatchCreate.sol:2:1: Warning: Ex
perimental features are turned on. Do not use experimental f
eatures on live deployments.
pragma experimental ABIEncoderV2;
^------------------------------^
,./contracts/Matching/Match/MatchSwap.sol:2:1: Warning: Expe
rimental features are turned on. Do not use experimental fea
tures on live deployments.
pragma experimental ABIEncoderV2;
^------------------------------^
,./contracts/Matching/Match/Match.sol:2:1: Warning: Experime
ntal features are turned on. Do not use experimental feature
s on live deployments.
pragma experimental ABIEncoderV2;
^------------------------------^
,./contracts/Matching/Match/MatchPool.sol:2:1: Warning: Expe
rimental features are turned on. Do not use experimental fea
tures on live deployments.
pragma experimental ABIEncoderV2;
^------------------------------^
,./contracts/Matching/SwaprateMatch/LibSwaprateOrder.sol:2:1
```

```
: Warning: Experimental features are turned on. Do not use e
xperimental features on live deployments.
pragma experimental ABIEncoderV2;
^------------------------------^
,./contracts/Matching/SwaprateMatch/SwaprateMatchBase.sol:2
:1: Warning: Experimental features are turned on. Do not use
experimental features on live deployments.
pragma experimental ABIEncoderV2;
^------------------------------^
,./contracts/Matching/SwaprateMatch/SwaprateMatch.sol:2:1: W
arning: Experimental features are turned on. Do not use expe
rimental features on live deployments.
pragma experimental ABIEncoderV2;
^------------------------------^
,./contracts/test/OptionCallSyntheticIdMock.sol:2:1: Warning
: Experimental features are turned on. Do not use experiment
al features on live deployments.
pragma experimental ABIEncoderV2;
^------------------------------^

> Artifacts written to ./build/contracts
> Compiled successfully using:
- solc: 0.5.7+commit.6da8b019.Emscripten.clang
```

Tests output

```
  Contract: Core
Functionality
 should return correct getDerivativeHash
Option Call
 should revert create OptionCall derivative with SYNTHETIC_A
GGREGATOR:WRONG_MARGIN (86543 gas)
 should revert create OptionCall derivative with CORE:SYNTHE
TIC_VALIDATION_ERROR (168085 gas)
 should revert create OptionCall derivative with CORE:NOT_EN
OUGH_TOKEN_ALLOWANCE (184096 gas)
Gas used during creation = 392883
 should create OptionCall derivative (422790 gas)
Gas used during second creation = 137382
 should create second exactly the same OptionCall derivative
(167527 gas)
```

```
Contract: Core
Option Call
 should revert execution with CORE:TOKEN_IDS_AND_QUANTITIES_
LENGTH_DOES_NOT_MATCH (51140 gas)
 should revert execution with CORE:TOKEN_IDS_AND_DERIVATIVES
_LENGTH_DOES_NOT_MATCH (44342 gas)
 should revert execution before endTime with CORE:EXECUTION_
BEFORE_MATURITY_NOT_ALLOWED (382018 gas)
Gas used during longToken execution = 157625
Gas used during shortToken execution = 94271
 should execute full margin option (251896 gas)
 should revert execution before endTime with CORE:SYNTHETIC_
EXECUTION_WAS_NOT_ALLOWED (101232 gas)
Gas used during third party execution = 147136
 should allow execution for third parties (174246 gas)
 should revert execution of invalid tokenId with CORE:UNKNOW
N_POSITION_TYPE (74394 gas)
Gas used during longToken execution = 142625
Gas used during shortToken execution = 94271
 should execute over margin option (206896 gas)
Gas used during longToken execution = 142840
Gas used during shortToken execution = 125664
 should execute under margin option (238504 gas)
Gas used during longToken execution = 102584
Gas used during shortToken execution = 109784
 should execute non profit option (182368 gas)
 should revert cancellation with CORE:CANCELLATION_IS_NOT_AL
LOWED (46077 gas)
 should revert execution with ORACLE_AGGREGATOR:DATA_DOESNT_
EXIST (48870 gas)
Gas used during cancellation = 103222
Gas used during cancellation = 100398
 should successfully cancel position after 2 weeks with no d
ata (203620 gas)
 should revert execution with CORE:TICKER_WAS_CANCELLED (120
770 gas)
Gas used during withdrawal = 49367
Gas used during withdrawal = 49367
 should successfully withdraw commission (98734 gas)

Contract: Match and MatchCreate
 should revert incorrect position creation, right taker toke
n is wrong (154739 gas)
 should revert incorrect signature (90393 gas)
```

```
 should revert incorrect signature length (83772 gas)
 should revert incorrect position creation, left maker margi
n < right taker margin (275895 gas)
 should revert incorrect position creation, left maker margi
n address != right taker margin address (275211 gas)
 should revert incorrect position creation, left maker token
id is non-zero (79887 gas)
 should revert incorrect position creation, provided margin
currency is wrong (274463 gas)
 should revert incorrect position creation, provided margin
is not enough (274792 gas)
 should revert incorrect position creation, not enough allow
ed margin (326216 gas)
 should revert incorrect position creation, taker address is
different (81658 gas)
 should revert incorrect position creation, sender address i
s different (80730 gas)
 should revert incorrect position creation, order is expired
(81077 gas)
 should revert cancellation of the order by different person
(43119 gas)
Gas used during cancellation = 71927
 should successfully cancel the order (71927 gas)
 should revert repeated cancellation of the order (50957 gas
)
 should revert creation with not enough premium form buyer (
475740 gas)
Gas used during matching creation = 691310
 should partially fill right (maker) order by 1, and have 2
left (taker) (711490 gas)
Gas used during matching creation = 394960
 should partially fill right (taker) order by 1, and have 1
left (maker) (319641 gas)
Gas used during matching shared creation = 427373
 should settle shared order (no senderAddress) (531300 gas)
 should revert already filled order (115095 gas)
Gas used during matching swap = 257123
 should fully fill right order by 1 buy reselling short posi
tion of left's (567921 gas)
 should revert non-fillable swaps (96938 gas)
 should revert order without WETH fee allowance (221333 gas)
 should revert order without WETH fee balance (275771 gas)
 should successfully settle order with ETH fee to relayer (6
47654 gas)
```

```
 should successfully withdraw relayer fee (34302 gas)

Contract: MatchSwap
 should revert incorrect swap, right requires more shorts fo
r long (147409 gas)
 should revert incorrect swap, right requires more margin (1
47479 gas)
 should revert incorrect swap, left requires more margin (14
7542 gas)
 should revert incorrect swap, left requires more margin wit
hout tokens (138968 gas)
 should revert incorrect swap, right requires more tokens (1
41182 gas)
 should revert incorrect swap, wrong left tokenIds  (192676
gas)
 should revert incorrect swap, wrong left margin  (187958 ga
s)
 should revert incorrect swap, wrong right tokenIds  (196760
gas)
 should revert incorrect swap, wrong right margin  (187216 g
as)
 should revert incorrect swap, no positions to swap  (212483
gas)
Gas used during swap = 249385
 should successfully swap 1 position left -> right (656495 g
as)
Gas used during swap = 294547
 should successfully swap 1 position right -> left for margi
n left -> right (375213 gas)

Contract: OracleAggregator
Gas used during __callback = 63246
 should accept data from oracle (63246 gas)
 should reject attempt to push data twice (23219 gas)
 should correctly return fetchPrice from oracle
 should revert fetchData with ORACLE_AGGREGATOR:NOT_ENOUGH_E
THER with 0 ether (31245 gas)
 should revert fetchData with ORACLE_AGGREGATOR:NOT_ENOUGH_E
THER with less ether (31245 gas)
 should revert recursivelyFetchData with ORACLE_AGGREGATOR:N
OT_ENOUGH_ETHER with 0 ether (31722 gas)
 should revert recursivelyFetchData with ORACLE_AGGREGATOR:N
OT_ENOUGH_ETHER with less ether (31722 gas)
Gas used during fetching = 60755
```

```
 should query and receive data using fetchData (128062 gas)
 should revert fetchData with ORACLE_AGGREGATOR:QUERY_WAS_AL
READY_MADE for already existing data (31742 gas)
 should revert recursivelyFetchData with ORACLE_AGGREGATOR:Q
UERY_WAS_ALREADY_MADE for already existing data (53581 gas)
Gas used during fetching recursively = 103999
 should query and receive data using recursivelyFetchData (3
06048 gas)

Contract: Registry
 should revert for non initializer address (23634 gas)
 should correctly set core address (43828 gas)
 should revert secondary call try (23967 gas)
 should correctly set token minter address (43912 gas)
 should correctly set oracle aggregator address (43868 gas)
 should correctly set synthetic aggregator address (43784 ga
s)
 should correctly set opium address (43870 gas)

Contract: SyntheticAggregator
 should successfully return isPool
 should successfully return getMargin
 should successfully return authorAddress
 should successfully return authorCommission

Contract: TokenMinter
Minting / Burning
 should revert minting with USING_REGISTRY:ONLY_CORE_ALLOWED
(27535 gas)
 should revert burning with USING_REGISTRY:ONLY_CORE_ALLOWED
(26138 gas)
Gas used during minting = 190380
 should correctly mint token positions for hashOne (190380 g
as)
Gas used during minting = 175380
 should correctly mint token positions for hashTwo (175380 g
as)
Gas used during more minting = 54816
 should correctly mint more token positions for hashOne (548
16 gas)
Gas used during burning longs = 39010
Gas used during burning shorts = 39010
 should correctly burn token positions for hashOne (48020 ga
s)
```

```
Gas used during burning longs = 38946
Gas used during burning shorts = 38946
 should correctly burn token positions for hashTwo (47892 ga
s)
 should revert burning token positions with TOKEN_MINTER:NOT
_ENOUGH_POSITIONS (57200 gas)
Composition
 should revert composing with TOKEN_MINTER:TOKEN_IDS_AND_RAT
IO_LENGTH_DOES_NOT_MATCH (25402 gas)
 should revert composing with TOKEN_MINTER:WRONG_QUANTITY (2
5628 gas)
 should revert composing with TOKEN_MINTER:WRONG_QUANTITY (2
3210 gas)
 should revert composing with TOKEN_MINTER:NOT_ENOUGH_POSITI
ONS (26943 gas)
Gas used during composing = 86941
Gas used during composing = 86941
 should successfully compose one token to position (228634 g
as)
Gas used during composing = 101859
Gas used during composing = 101859
 should successfully compose two tokens to position (313222
gas)
Decomposition
 should revert decomposing with TOKEN_MINTER:TOKEN_IDS_AND_R
ATIO_LENGTH_DOES_NOT_MATCH (25634 gas)
 should revert decomposing with TOKEN_MINTER:WRONG_QUANTITY
(25860 gas)
 should revert decomposing with TOKEN_MINTER:WRONG_QUANTITY
(23442 gas)
 should revert decomposing with TOKEN_MINTER:WRONG_PORTFOLIO
_ID (28765 gas)
 should revert decomposing with TOKEN_MINTER:NOT_ENOUGH_POSI
TIONS (29492 gas)
Gas used during decomposing = 65702
Gas used during decomposing = 65702
 should successfully decompose position to one token (101404
gas)
Gas used during decomposing = 79470
Gas used during decomposing = 79534
 should successfully decompose two tokens to position (12900
4 gas)
Recomposition
 should revert recomposing with TOKEN_MINTER:INITIAL_TOKEN_I
```

```
DS_AND_RATIO_LENGTH_DOES_NOT_MATCH (29511 gas)
 should revert recomposing with TOKEN_MINTER:FINAL_TOKEN_IDS
_AND_RATIO_LENGTH_DOES_NOT_MATCH (29540 gas)
 should revert recomposing with TOKEN_MINTER:WRONG_QUANTITY
(29766 gas)
 should revert recomposing with TOKEN_MINTER:WRONG_QUANTITY
(27284 gas)
 should revert recomposing with TOKEN_MINTER:WRONG_QUANTITY
(27374 gas)
 should revert recomposing with TOKEN_MINTER:WRONG_PORTFOLIO
_ID (36009 gas)
 should revert recomposing with TOKEN_MINTER:NOT_ENOUGH_POSI
TIONS (36736 gas)
Gas used during recomposing = 90298
Gas used during recomposing = 90298
 should successfully recompose from position with one token
to two tokens (120596 gas)
Gas used during recomposing = 135143
Gas used during recomposing = 135207
 should successfully recompose from position with two tokens
to three tokens and different ratio (355730 gas)
Founded bugs
 should restrict composition with not unique tokenIds (26068
6 gas)

Contract: TokenSpender
 should revert proposing by non governor address (25422 gas)
 should be successfully propose initial whitelist by governo
r address (94841 gas)
 should revert spending by non whitelisted address (28256 ga
s)
 should successfully spend by core (94387 gas)
 should revert commitment before proposal (22005 gas)
 should revert proposal for empty list (22517 gas)
 should be successfully propose new addresses by governor ad
dress, but keep old till time lock (109606 gas)
 should revert commitment by non governor address (21821 gas
)
 should revert commitment before time lock (22447 gas)
 should successfully commit new whitelist after time lock (3
1496 gas)

122 passing (2m)
```

Ethlint output

```
contracts/Core.sol
127:4     warning    Line exceeds the limit of 145 character
s
201:8     warning    Line exceeds the limit of 145 character
s
207:8     warning    Line exceeds the limit of 145 character
s
264:8     warning    Line exceeds the limit of 145 character
s
302:20    warning    Avoid using 'now' (alias to 'block.time
stamp').
350:73    warning    Avoid using 'now' (alias to 'block.time
stamp').
401:4     warning    Line exceeds the limit of 145 character
s
448:84    warning    There should be no whitespace or commen
ts between argument and the comma following it.
496:4     warning    Line exceeds the limit of 145 character
s
521:8     warning    Line exceeds the limit of 145 character
s

contracts/Lib/LibDerivative.sol
26:35    error    Only use indent of 12 spaces.

contracts/Lib/LibEIP712.sol
16:64    error    Only use indent of 8 spaces.
29:37    error    Only use indent of 12 spaces.
43:8     error    Avoid using Inline Assembly.

contracts/Lib/WhitelistedWithGovernance.sol
57:27    warning    Avoid using 'now' (alias to 'block.times
tamp').
69:54    warning    Avoid using 'now' (alias to 'block.times
tamp').
70:8     warning    Line contains trailing whitespace

contracts/Lib/WhitelistedWithGovernanceAndChangableTimelock.
sol
20:31    warning    Avoid using 'now' (alias to 'block.times
tamp').
30:62    warning    Avoid using 'now' (alias to 'block.times
tamp').
```

```
31:8     warning    Line contains trailing whitespace

contracts/Matching/Match/LibOrder.sol
73:64    error      Only use indent of 8 spaces.
94:8     warning    Line contains trailing whitespace

contracts/Matching/Match/Match.sol
12:68    warning    Code contains empty block

contracts/Matching/Match/MatchCreate.sol
14:4      warning    Line exceeds the limit of 145 character
s
88:8      warning    Line contains trailing whitespace
91:8      warning    Line contains trailing whitespace
103:4     warning    Line exceeds the limit of 145 character
s
121:8     warning    Line contains trailing whitespace
124:80    warning    There should be no whitespace or commen
ts between argument and the comma following it.
128:81    warning    There should be no whitespace or commen
ts between argument and the comma following it.
146:4     warning    Line exceeds the limit of 145 character
s
174:1     warning    Line contains trailing whitespace
176:1     warning    Line contains trailing whitespace
179:4     warning    Line exceeds the limit of 145 character
s
190:1     warning    Line contains trailing whitespace
201:8     warning    Line exceeds the limit of 145 character
s
202:8     warning    Line exceeds the limit of 145 character
s
238:8     warning    Line contains trailing whitespace

contracts/Matching/Match/MatchLogic.sol
95:31    warning    Avoid using 'now' (alias to 'block.times
tamp').

contracts/Matching/Match/MatchPool.sol
12:68    warning    Code contains empty block
13:4     warning    Line contains trailing whitespace
74:8     warning    Line contains trailing whitespace
77:76    warning    There should be no whitespace or comment
s between argument and the comma following it.
```

```
contracts/Matching/Match/MatchSwap.sol
54:8      warning    Line exceeds the limit of 145 character
s
71:4      warning    Line exceeds the limit of 145 character
s
125:29    warning    There should be no whitespace or commen
ts between argument and the comma following it.
134:8     warning    Line exceeds the limit of 145 character
s
135:8     warning    Line exceeds the limit of 145 character
s
140:8     warning    Line exceeds the limit of 145 character
s
141:8     warning    Line exceeds the limit of 145 character
s
151:4     warning    Line exceeds the limit of 145 character
s

contracts/Matching/SwaprateMatch/LibSwaprateOrder.sol
85:64    error    Only use indent of 8 spaces.

contracts/Matching/SwaprateMatch/SwaprateMatch.sol
21:68     warning    Code contains empty block
51:8      warning    Line exceeds the limit of 145 character
s
85:4      warning    Line exceeds the limit of 145 character
s
101:4     warning    Line exceeds the limit of 145 character
s
106:52    warning    Single space should be either on both s
ides of '!=' or not at all.
194:4     warning    Line exceeds the limit of 145 character
s

contracts/Matching/SwaprateMatch/SwaprateMatchBase.sol
74:106    warning    Visibility modifier "internal" should c
ome before other modifiers.

contracts/OracleAggregator.sol
58:4    warning    Line exceeds the limit of 145 characters
96:8    warning    Line exceeds the limit of 145 characters

contracts/SyntheticAggregator.sol
72:4    warning    Line exceeds the limit of 145 characters
```

```
75:1    warning    Line contains trailing whitespace

contracts/TokenSpender.sol
19:99    warning    Code contains empty block

 6 errors, 58 warnings found.
```