



Oiler LBP Controller
SMART CONTRACT AUDIT

05.05.2021

Made in Germany by Chainsulting.de



Table of contents

1. Disclaimer	3
2. About the Project and Company	4
2.1 Project Overview	5
3. Vulnerability & Risk Level.....	6
4. Auditing Strategy and Techniques Applied	7
4.1 Methodology.....	7
4.2 Used Code from other Frameworks/Smart Contracts	8
4.3 Tested Contract Files.....	9
4.4 Metrics / CallGraph	10
4.6 Metrics / Capabilities.....	12
4.7 Metrics / Source Unites in Scope.....	13
5. Scope of Work.....	15
5.1 Manual and Automated Vulnerability Test	16
5.2. SWC Attacks & Special Checks	17
6. Test Deployment & Verify claims	21
7. Executive Summary	23
8. Deployed Smart Contract.....	23



1. Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of Oiler DeFi. If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Major Versions / Date	Description
0.1 (15.03.2021)	Layout
0.5 (16.03.2021)	Verify Claims and Test Deployment
0.6 (17.03.2021)	Testing SWC Checks
0.8 (17.03.2021)	Automated Security Testing Manual Security Testing
0.9 (18.03.2021)	Summary and Recommendation
1.0 (20.03.2021)	Final document
1.1 (05.05.2021)	Added deployed contract

2. About the Project and Company

Company address:

Oiler DeFi c/o International Corporation Services Ltd
PO Box 472, Harbour Place 2nd
Floor103 South Church Street
Grand Cayman
KY1-1106, Cayman Islands

Website: <https://oiler.network>

Twitter: <https://twitter.com/OilerNetwork>

Telegram: <https://t.me/oilernetwork>

Telegram: https://t.me/oiler_official

Medium: <https://medium.com/oiler-network>

LinkedIn: <https://www.linkedin.com/company/oiler-network/>

2.1 Project Overview

Oiler is a protocol for blockchain native derivatives.

Before 2019/2020, it was practically impossible to deliver blockchain native derivatives. Without stablecoins and AMMs (automatic market makers), it was not possible to provide a reliable pricing solution. What has changed? Stablecoins introduced a non-volatile on-chain base for pricing (a USD peg) and AMMs introduced a pricing discovery mechanism; on-chain and with high volumes. It has also been proven that if the markets are efficient then the AMMs are efficient too and arbitrageurs will set the price right.

In order to settle derivatives on-chain nowadays, we need to ensure that the payout can be calculated entirely on-chain. At Oiler, they not only assume that they will not take off-chain data but also that there is no oracle hidden behind the layers of on-chain data sources that the smart contracts use. It means that the prices of the underlying instruments should not be derived from a protocol that uses off-chain oracles. Moreover, it is desirable to avoid any on-chain oracles like Uniswap since they can always be manipulated within a flash loan based attack. The last requirement is much stronger but still holds for the initial set of Oiler products.

DeFi users are most likely to use Oiler if they already have exposure to blockchain protocol parameters — their operations rely on big shifts in the network behavior, manifesting via big hashrate changes, massive gas price movements and protocol behavior changes. Who would that be?

- Exchanges that cover the highly volatile on-chain assets withdrawal costs
- Miners having exposure to shifting block reward and transaction fees
- Institutions with exposures to many blockchains and a need for hedging the protocol-level risks

Oiler will continue bringing products — both new underlying blockchain parameters and new instrument types that will be related to the category of blockchain protocol trading. They will continue working on pricing models without oracles and will look at the instruments related to both Ethereum 1.0 and Ethereum 2.0 (and their coexistence).

read more at <https://docs.oiler.network/oiler-network/>

3. Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 – 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

4. Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

4.1 Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
 - i. Review of the specifications, sources, and instructions provided to Chainsulting to make sure we understand the size, scope, and functionality of the smart contract.
 - ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
 - iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Chainsulting describe.
2. Testing and automated analysis that includes the following:
 - i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
 - ii. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

4.2 Used Code from other Frameworks/Smart Contracts (direct imports)

Dependency / Import Path	Source
@openzeppelin/contracts/access/Ownable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.4.0/contracts/access/Ownable.sol
@openzeppelin/contracts/math/SafeMath.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.4.0/contracts/math/SafeMath.sol
@openzeppelin/contracts/token/ERC20/ERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.4.0/token/ERC20/ERC20.sol
@openzeppelin/contracts/token/ERC20/IERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.4.0/token/ERC20/IERC20.sol
contracts/lib/BColor.sol	https://github.com/balancer-labs/balancer-core/blob/master/contracts/BColor.sol
contracts/lib/BConst.sol	https://github.com/balancer-labs/balancer-core/blob/master/contracts/BConst.sol
contracts/lib/BMath.sol	https://github.com/balancer-labs/balancer-core/blob/master/contracts/BMath.sol
contracts/lib/BNum.sol	https://github.com/balancer-labs/balancer-core/blob/master/contracts/BNum.sol
contracts/abstract/BPool.sol	https://github.com/balancer-labs/balancer-core/blob/master/contracts/BPool.sol
contracts/abstract/BFactory.sol	https://github.com/balancer-labs/balancer-core/blob/master/contracts/BFactory.sol
contracts/interfaces/IUniswapV2Pair.sol	https://github.com/Uniswap/uniswap-v2-core/blob/master/contracts/interfaces/IUniswapV2Pair.sol
contracts/interfaces/IUniswapV2Factory.sol	https://github.com/Uniswap/uniswap-v2-core/blob/master/contracts/interfaces/IUniswapV2Factory.sol

4.3 Tested Contract Files

The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review

File	Fingerprint (MD5)
contracts/interfaces/ILBPController.sol	52a65eb7e0852bd335ca387cd51e8bbe
contracts/interfaces/IUniswapV2Router02.sol	204e035b60d2c4df95c419f0c19274fa
contracts/interfaces/IUniswapV2Router01.sol	f1c22a4f35c1e1b7b5dd9b11acac2f74
contracts/lib/UniswapMath.sol	dba40d3feebae30c0092305090ad216a
contracts/lib/RightsManager.sol	f478f8fa36afa5ff51f333e301661e23
contracts/LBPController.sol	554877272e8863dae4022d76eadc6d70
contracts/abstracts/AbstractPool.sol	237701db88479d5f5dc7c3f1d5fefba5
contracts/abstracts/IERC20DecimalsExt.sol	a9dd3f78ad4f0d90a2b1903473d90dbf
contracts/abstracts/BalancerOwnable.sol	409c81d671e59b2e421fab01f2cfad38
contracts/abstracts/CRPFactory.sol	b2949ff6e487a2e3bcb6418d44e40dfd
contracts/abstracts/ConfigurableRightsPool.sol	2b005505d2a464a0f787c84b76e6fddd
contracts/abstracts/BRegistry.sol	bac287d001e1e16ec23765cca8138c99
contracts/UniswapMigrator.sol	7a5472822c6dff5bc2cd541d611e0080
contracts/Token.sol	f26b1e4f96880474c8c6c2987922c0f2

4.4 Metrics / CallGraph







Full Version: https://chainsulting.de/wp-content/uploads/2021/03/oiler_lbp_solidity-metrics.html

4.5 Metrics / Source Lines






























4.6 Metrics / Capabilities

Solidity Versions observed		 Experimental Features		 Can Receive Funds		 Uses Assembly		 Has Destroyable Contracts			
<div>0.7.6</div>				<div>yes</div>		**** (0 asm blocks)					
 Transfers ETH		 Low-Level Calls		 DelegateCall		 Uses Hash Functions		 ECRrecover		 New/Create/Create2	
<div>yes</div>											

 Public	 Payable				
7	0				
External	Internal	Private	Pure	View	
6	9	0	0	1	

4.7 Metrics / Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/interfaces/IUniswapV2Pair.sol	_____	1	53	8	5	1	55	_____
	contracts/interfaces/ILBPController.sol	_____	1	16	10	6	1	13	_____
	contracts/interfaces/IUniswapV2Factory.sol	_____	1	18	7	4	1	17	_____
	contracts/interfaces/IUniswapV2Router02.sol	_____	1	47	9	4	1	16	
	contracts/interfaces/IUniswapV2Router01.sol	_____	1	97	6	3	1	48	
	contracts/lib/UniswapMath.sol	1	_____	24	24	18	3	5	_____
	contracts/lib/BColor.sol	2	_____	29	22	7	11	9	_____
	contracts/lib/RightsManager.sol	1	_____	13	13	11	1	1	_____
	contracts/lib/BNum.sol	1	_____	164	134	88	22	48	_____
	contracts/lib/BConst.sol	1	_____	42	42	20	11	19	_____
	contracts/lib/BMath.sol	1	_____	257	188	79	91	89	_____

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSL OC	Comment Lines	Complex. Score	Capabilities
	contracts/LBPCController.sol	1	_____	261	261	152	92	162	
	contracts/abstracts/BPool.sol	1	_____	30	7	4	1	49	_____
	contracts/abstracts/BFactory.sol	1	_____	12	7	4	1	11	_____
	contracts/abstracts/AbstractPool.sol	1	_____	12	8	5	1	13	_____
	contracts/abstracts/IERC20DecimalsExt.sol	_____	1	9	7	4	1	5	_____
	contracts/abstracts/BalancerOwnable.sol	1	_____	7	6	3	2	3	_____
	contracts/abstracts/CRPFactory.sol	1	_____	11	10	6	1	3	_____
	contracts/abstracts/ConfigurableRightsPool.sol	1	_____	77	49	41	1	59	_____
	contracts/abstracts/BRegistry.sol	1	_____	7	6	3	2	3	_____
	contracts/UniswapMigrator.sol	1	_____	188	188	128	37	146	_____
	contracts/Token.sol	1	_____	16	16	8	6	6	_____
	Totals	17	6	1390	1028	603	289	780	

5. Scope of Work

The Oiler Team provided us with the files that needs to be tested. The scope of the audit is the Oiler LBP Controller contract.

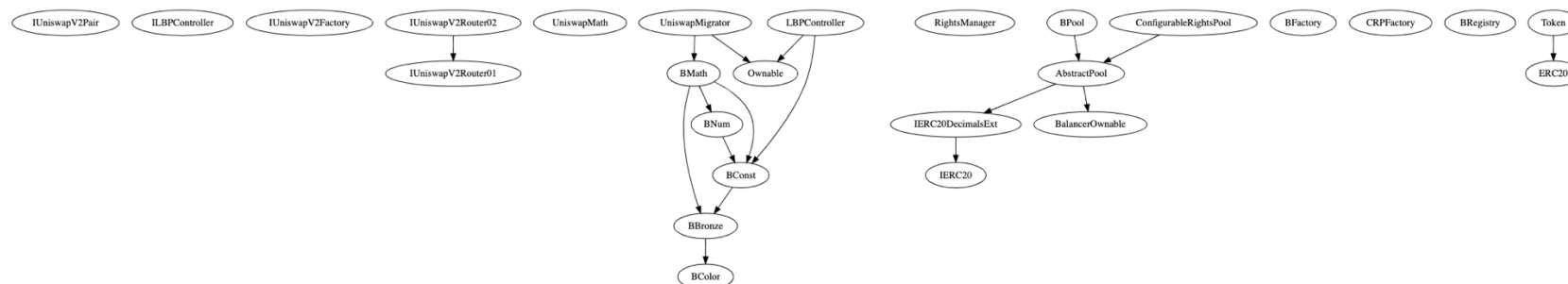
Following contracts with the direct imports has been tested:

- LBPController.sol

The team put forward the following assumptions regarding the security, usage of the contracts:

- We activate GradualUpdates parameters and lock swapping straight after CRP creation until the LBP start block. That should fully lock and prevent any user interactions with the pool before LBP officially starts. So no withdrawals, swaps, adding liquidity, tokens, etc - nothing should be possible for the users/malicious actors.
- We also protect pools from the Owner's malicious action (to a rational extent). The only thing the owner can do - is have an EscapeHatch to withdraw LBP tokens from the LBPController contract. This by its own doesn't interrupt any action of the pool (except it's ending/liquidation), but allows the Owner to withdraw ~99.999% of underlying tokens from BPool in case of any emergency. Withdrawing LBP tokens prevents pool ending/liquidation/destruction, but Owner having back 99.999% of tokens really means the pool is liquidated (in a sense).
- Pool should be actionable during the LBP period (between startBlock and endBlock). That means - no actions of users should lead to stealing or leaking funds from the pool before, during, or after the LBP, or to stalling/locking of the pool or tokens in it.
- The Owner should be able to withdraw all (~99.999% is considered as all) tokens from the pool after LBP ends.
- Overall smart contract security needs to be checked

The main goal of this audit was to verify these claims. The auditors can provide additional feedback on the code upon the client's request.



5.1 Manual and Automated Vulnerability Test

CRITICAL ISSUES

During the audit, Chainsulting's experts found **no Critical issues** in the code of the smart contract.

HIGH ISSUES

During the audit, Chainsulting's experts found **no High issues** in the code of the smart contract.

MEDIUM ISSUES

During the audit, Chainsulting's experts found **no Medium issues** in the code of the smart contract.

LOW ISSUES





During the audit, Chainsulting's experts found **no Low issues** in the code of the smart contract.

5.2. SWC Attacks & Special Checks

ID	Title	Relationships	Test Result
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	
SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	

ID	Title	Relationships	Test Result
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	✓
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	✓
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	✓
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	✓
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	✓
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	✓
SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✓
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	✓
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	✓

ID	Title	Relationships	Test Result
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	✓
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✓
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	✓
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	✓
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	✓
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	✓
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	✓
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	✓
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	✓
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	✓

ID	Title	Relationships	Test Result
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	

6. Test Deployment & Verify claims

6.1 Deploy Token

Tx: <https://kovan.etherscan.io/tx/0xed4376b8c3bd3b6ff714300273e6033ac595cc3a421690431f198dace168633c>

Contract: <https://kovan.etherscan.io/address/0x165892ac09286bcd9ab007818629fc63f5ceb8>

6.2 Deploy USDC

Tx: <https://kovan.etherscan.io/tx/0x0aaee274c989dff9f3ec3c2c7f75896daefeb9f0baf04fd4b9bc543a8e4f2ea>

Contract: <https://kovan.etherscan.io/address/0x4eddd18800db47779526fad141a87e5d05f65473>

6.3 Deploy LBPController

CONTRACT

LBPController - browser/contracts/LBPController.sol

DEPLOY

_CRPFACTORYADDRESS:

0x53265f0e014995363AE54DAd7059c0188aDbcD74

_BREGISTRYADDRESS:

0xC5570FC7C828A8400605e9843106aBD675006093

_BFACTORYADDRESS:

0x8f7F78080219d4066A8036ccD30D588B416a40DB

_TOKENADDRESS:

0x165892AC09286bcd9ab007818629fc63F5cEb8

_COLLATERALADDRESS:

0x4edDd18800db47779526faD141A87e5D05F65473

_STARTBLOCK:

23978872

_ENDBLOCK:

23984572

_OWNER:

0x2F1602FD37228b32Ad8D13137b1B620862771909

transact

Tx: <https://kovan.etherscan.io/tx/0xa68c6aa20613ff5521cf3ec82aaf6873760bb3104a372b2b98d87dede63f6950>

Contract: <https://kovan.etherscan.io/address/0x560a9da9ea346252a9cc32692209b4a174ca6a2c>

Status: tested and verified 

6.3 Actions before start block

As expected, all actions for LBPController are failing before the startblock is reached. The funds are safely locked and nobody can interact with the contract. Some examples listed below:

Tx: <https://kovan.etherscan.io/tx/0x3dde1ea043a23a63bc2bf46c18ab710d55b3ab5b14908e4f49ee412bfcf70d14>

Tx: <https://kovan.etherscan.io/tx/0xd94255aea398bd11e43d15939a80885778f6cc9974115455c573e6dd73bf04b2>

Tx: <https://kovan.etherscan.io/tx/0x62487943dfa435a594aedf552d3520aea76750b21e1fe549af0d49cf2847aec0>

Tx: <https://kovan.etherscan.io/tx/0x68b94704393547f367918cf57fae65e6e6c3cba0548459a70f656f743c319811>

Status: tested and verified 

6.4 Create Smart Pool

As expected, only the owner can create the smart pool in the period between start and end block.

Tx: <https://kovan.etherscan.io/tx/0x2a766959dcdaaf881de93bbf8f781b9d655bf09efeccd9d683fc1a042dbb9232>

Status: tested and verified 

6.5 Actions after smart pool creation

Register Pool

Tx: <https://kovan.etherscan.io/tx/0xa37b5fcc7315f7afb55b013e0c8b330534abf65cefc6d218ca57527bd48a41eb>


Start Pool

Tx: <https://kovan.etherscan.io/tx/0xbd7a6abcbf6370e2f03843b9a86f1c3d665d4315903c44ba79e7723f78246210>

Owner can withdraw all LBP Tokens after active period

LBP Tokens are sent directly to the owner.

Tx: <https://kovan.etherscan.io/tx/0x9c9fe5a2c4196b86cb30229143ec62d5cf9205ac0f2c26a7d1a4c48ca144d937>

Status: tested and verified 

7. Executive Summary

Two (2) independent Chainsulting experts performed an unbiased and isolated audit of the smart contract codebase. The overall code quality of the project is very good, and the simplicity greatly benefits the overall security. It implemented widely-used and reviewed contracts from OpenZeppelin and for safe mathematical operations. Our auditors highly enjoyed the good documentation and unit tests, one of the few contracts that have been reviewed at chainsulting, with no issues and consistently good code quality.

The main goal of the audit was to verify the claims regarding the security of the smart contract and the functions. During the audit, no issues were found after the manual and automated security testing.

8. Deployed Smart Contract

VERIFIED

Smart Contract is deployed here:

<https://etherscan.io/address/0xd0CA3F9B0E22a45536991889B8881642411eB7F8#code>

