HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Velocore
Date: October 25, 2023

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Document

Name	Smart Contract Code Review and Security Analysis Report for Velocore
Approved By	Oleksii Zaiats | SC Audits Head at Hacken OÜ
Tags	DEX
Platform	EVM (zkSyncEra)
Language	Solidity
Methodology	Link
Website	https://velocore.xyz
Changelog	08.08.2023 - Initial Review 29.08.2023 - Second Review 13.10.2023 - Third Review 25.10.2023 - Fourth Review

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Table of contents

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Introduction

Hacken OÜ (Consultant) was contracted by Velocore (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contracts.

System Overview

Velocore is a DEX, the pool might handle up to 4 tokens, the vault system inspired by the Balancer protocol with a composable pool. Constant-product pool is meant to accept multi-in-multi-out and accepting generic tokens(ERC20,721,1155) in the pool.

- *ConstantProductPool* – pool for constant-product automated market maker.
- *ConstantProductPoolFactory* – factory contract *ConstantProductPool* smart contract pool.
- *ConstantProductLibrary* – a singleton contract for precise but expensive calculation of swap operations. Used as a fallback method for extreme cases. Deployed separately to make the contract size smaller than 24kb.
- *SingleTokenGauge* – inherits from Pool. It is a straightforward gauge with a single staking token.
- Satellite – an abstract contract with common methods for Vault interaction.
- RPow – a library for squaring exponentiation from MakerDAO DSS.
- PoolBalanceLib – a library manipulating PoolBalance, a wrapped bytes32 holding two uint128 values. This is used for pool balances.
- Token – a library abstracting tokens, defining Token as a wrapped bytes32 with token specs, id, and address. It defines functions like transfer, balance over them.
- UncheckedMemory – functions to bypass boundedness checks. Defines array.u(index) and array.u(index, value) as getter and setter. Used extensively.

Privileged roles

- *Admin* - the role which might set fees and `decayRate`.
- *Vault* - the users might interact with the vault and notify new emissions, gauge user stakes, bribe, and set fees to zero.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Executive Summary

The score measurement details can be found in the corresponding section of the scoring methodology.

Documentation quality

The total Documentation Quality score is **5** out of **10**.

- Functional requirements have some gaps:
  - Project overview is detailed.
  - All roles in the system are described.
  - Use cases are described and detailed.
  - Not all interactions are described.
- Technical description is limited:
  - Technical specification is provided.
  - No run instructions.
  - No NatSpec with technical information like:
    - param description.
    - return values description.
    - validation rules.

Code quality

The total Code Quality score is **8** out of **10**.

- Solidity Style Guide violations:
  - unclear variable naming.
- Development environment is not properly configured to be able to run the tests.

Test coverage

Code coverage of the project is **80%** (approximate branch coverage).

- Impossible to run tests due to compilation errors within the test contract "out of the box".
- Lacking coverage within functions that might change key system parameters:
  - ConstantProductPoolFactory: setDecay
  - ConstantProductPool: setParam

Security score

As a result of the audit, the code contains **no** issues. The security score is **10** out of **10**.

All found issues are displayed in the “Findings” section.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Summary

According to the assessment, the Customer's smart contract has the following score: **8.29**. The system users should acknowledge all the risks summed up in the risks section of the report.

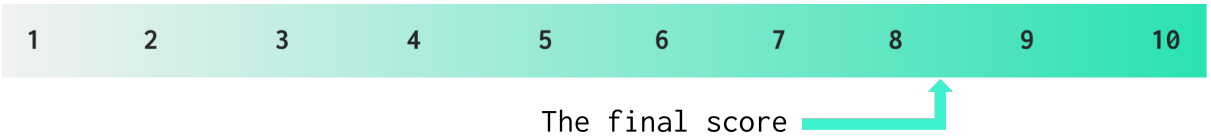

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
8 August 2023	4	1	1	0
29 August 2023	4	1	1	0
12 October 2023	0	0	1	0
25 October 2023	0	0	0	0

Risks

- **Significant part of the system is out of the audit scope.**
- The Admin of the system may update fees.
- Token transfer logic after exchange swap is out of scope.
- The project uses the library for fixed-point math computations, which is out of scope.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Checked Items

We have audited the Customers' smart contracts for commonly known and specific vulnerabilities. Here are some items considered:

Item	Description	Status	Related Issues
Default Visibility	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	Passed	
Integer Overflow and Underflow	If unchecked math is used, all math operations should be safe from overflows and underflows.	Passed	
Outdated Compiler Version	It is recommended to use a recent version of the Solidity compiler.	Passed	
Floating Pragma	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	Failed	I01
Unchecked Call Return Value	The return value of a message call should be checked.	Passed	
Access Control & Authorization	Ownership takeover should not be possible. All crucial functions should be protected. Users could not affect data that belongs to other users.	Passed	
SELFDESTRUCT Instruction	The contract should not be self-destructible while it has funds belonging to users.	Not Relevant	
Check-Effect-Interaction	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	Passed	
Assert Violation	Properly functioning code should never reach a failing assert statement.	Passed	
Deprecated Solidity Functions	Deprecated built-in functions should never be used.	Passed	
Delegatecall to Untrusted Callee	Delegatecalls should only be allowed to trusted addresses.	Passed	
DoS (Denial of Service)	Execution of the code should never be blocked by a specific contract state unless required.	Passed	

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Race Conditions	Race Conditions and Transactions Order Dependency should not be possible.	Passed	
Authorization through tx.origin	tx.origin should not be used for authorization.	Passed	
Block values as a proxy for time	Block numbers should not be used for time calculations.	Passed	
Signature Unique Id	Signed messages should always have a unique id. A transaction hash should not be used as a unique id. Chain identifiers should always be used. All parameters from the signature should be used in signer recovery. EIP-712 should be followed during a signer verification.	Passed	
Shadowing State Variable	State variables should not be shadowed.	Passed	
Weak Sources of Randomness	Random values should never be generated from Chain Attributes or be predictable.	Not Relevant	
Incorrect Inheritance Order	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order.	Passed	
Calls Only to Trusted Addresses	All external calls should be performed only to trusted addresses.	Passed	
Presence of Unused Variables	The code should not contain unused variables if this is not justified by design.	Passed	
EIP Standards Violation	EIP standards should not be violated.	Passed	
Assets Integrity	Funds are protected and cannot be withdrawn without proper permissions or be locked on the contract.	Passed	
User Balances Manipulation	Contract owners or any other third party should not be able to access funds belonging to users.	Passed	
Data Consistency	Smart contract data should be consistent all over the data flow.	Passed	

Flashloan Attack	When working with exchange rates, they should be received from a trusted source and not be vulnerable to short-term rate changes that can be achieved by using flash loans. Oracles should be used. Contracts shouldn't rely on values that can be changed in the same transaction.	Passed	
Token Supply Manipulation	Tokens can be minted only according to rules specified in a whitepaper or any other documentation provided by the Customer.	Passed	
Gas Limit and Loops	Transaction execution costs should not depend dramatically on the amount of data stored on the contract. There should not be any cases when execution fails due to the block Gas limit.	Passed	
Style Guide Violation	Style guides and best practices should be followed.	Failed	I03
Requirements Compliance	The code should be compliant with the requirements provided by the Customer.	Passed	
Environment Consistency	The project should contain a configured development environment with a comprehensive description of how to compile, build and deploy the code.	Failed	I06
Secure Oracles Usage	The code should have the ability to pause specific data feeds that it relies on. This should be done to protect a contract from compromised oracles.	Not Relevant	
Tests Coverage	The code should be covered with unit tests. Test coverage should be sufficient, with both negative and positive cases covered. Usage of contracts by multiple users should be tested.	Failed	I06
Stable Imports	The code should not reference draft contracts, which may be changed in the future.	Passed	

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Findings

Critical

No critical severity issues were found.

High

H01. Unbounded Parameter Values

Impact	High
Likelihood	Medium

The system lacks limits on key parameters such as fees in its initial configuration and setter function, allowing for unbounded values.

In the system, no constraints are defined for parameters such as *fee1e9*, *decayRate* in the relevant functions.

The absence of constraints can lead to parameter values being set excessively high or low, which can destabilize the system, leading to losses, discouraging user interaction and potentially undermining trust in the system or even denial of service.

Paths:

./src/pools/constant-product/ConstantProductPool.sol : constructor(), setParam();

./src/pools/constant-product/ConstantProductPoolFactory.sol : setFee().

Recommendation: Incorporate lower and upper boundaries for these parameters to maintain their values within reasonable ranges.

Found in: 94fc760

Status: Fixed (Revised commit: 7b62de5d)

Medium

M01. Missing Events on Critical State Updates

Impact	Low
Likelihood	High

Critical state changes should emit events for tracking things off-chain.

This can lead to inability for users to subscribe events and check what is going on with the project.

Paths:

./src/pools/constant-product/ConstantProductPool.sol : setParam(), setFeeToZero();

./src/pools/constant-product/ConstantProductPoolFactory.sol : setFee(), deploy().

Recommendation: Emit events on critical state changes.

Found in: 94fc760

Status: Fixed (Revised commit: 2176c96)

Low

L01. Missing Validation

Impact	Low
Likelihood	Medium

The system has a contract which is responsible for handling computations within the pool, the pool might handle from 2 to 4 tokens, but there is no validation to verify that the data for at least two tokens is specified.

This might lead to the Gas expense due to the pool deployment.

Path: ./src/pools/constant-product/ConstantProductPool.sol : constructor().

Recommendation: Add an assert or conditional statement to verify the constructor input data.

Found in: 94fc760

Status: Fixed (Revised commit: 2176c96)

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

L02. Missing Require Error Messages

Impact	Low
Likelihood	Medium

Require statements should have their error message described consciously in order to advise the user about the transaction failure.

Not having error messages implies a bad user/developer experience.

Paths:

./src/pools/constant-product/ConstantProductPool.sol : notifyWithdraw(), velocore__execute(), velocore__bribe(), underlyingTokens(), setParam(), constructor();

./src/pools/constant-product/ConstantProductPoolFactory.sol : deploy(), setFee();

./src/lib/Token.sol : balanceOf(), totalSupply(), symbol(), decimals(), transferFrom(), safeTransferFrom().

Recommendation: Add reasonable error messages to require statements.

Found in: 94fc760

Status: Fixed (Revised commit: 2176c96)

L03. Missing Zero Address Validation

Impact	Medium
Likelihood	Low

Address parameters are being used without checking against the possibility of 0x0.

This can lead to unwanted external calls to 0x0.

Paths: ./src/pools/constant-product/ConstantProductPool.sol : constructor()->lib, vault_;

./src/pools/constant-product/ConstantProductPoolFactory.sol : constructor()->lib, deploy()->quoteToken, baseToken;

./src/pools/SingleTokenGauge.sol : constructor()->vault, bribe;

./src/pools/Satellite.sol : constructor()->factory.

Recommendation: Add zero address validation.

Found in: 94fc760

Status: Fixed (Revised commit: 2176c96)

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

L04. Contradiction

Impact	Low
Likelihood	Medium

The contract has a constant string which states that the native token of the blockchain is “ETH”, but it depends on the chain on which the contract is deployed.

This might lead to confusions on the frontend.

Path: ./src/lib/Token.sol : symbol().

Recommendation: Set the value within the constructor.

Found in: 94fc760

Status: Fixed (Revised commit: 2176c96)

Informational

I01. Floating Pragma

The project uses floating pragmas `^0.8.0` or `^0.8.19`.

This may result in the contracts being deployed using the wrong pragma version, which is different from the one they were tested with. For example, they might be deployed using an outdated pragma version, which may include bugs that affect the system negatively.

Path: ./: *

Recommendation: Consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment. Consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen.

Found in: 94fc760

Status: Reported

I02. Missing Variable Explicit Visibility

Some variables do not have their visibility written explicitly in the code.

That leads to readability issues.

Paths:

./src/pools/SingleTokenGauge.sol : emissionPerStake1e9, stakerInformation;

./src/pools/constant-product/ConstantProductPool.sol : lib;

./src/pools/constant-product/ConstantProductPoolFactory.sol : lib, fee1e9, decay.

Recommendation: Describe the variables' visibilities explicitly in the code.

Found in: 94fc760

Status: Fixed (Revised commit: 7b62de5)

I03. Solidity Style Guide Violation: Order Of Layout

Inside each contract, library or interface, use the following order:

1. Type declarations
2. State variables
3. Events
4. Errors
5. Modifiers
6. Functions
    a. constructor
    b. initializer (if exists)
    c. receive function (if exists)
    d. fallback function (if exists)
    e. external
    f. public
    g. internal
    h. private

Paths:

./src/pools/constant-product/ConstantProductPool.sol;

./src/pools/constant-product/ConstantProductPoolFactory.sol;

./src/pools/Satellite.sol.

Recommendation: Change order of layout to fit Official Style Guide.

Found in: 94fc760

Status: Reported

I04. Solidity Style Guide Violation: Naming Convention

Functions, local and state variable names should be mixedCase: capitalize all the letters of the initialisms, except keep the first one lower case if it is the beginning of the name.

Paths:

./src/pools/constant-product/ConstantProductPool.sol : _return_logarithmic_swap(), velocore__execute(), pow_reciprocal(), velocore__bribe();

./src/pools/SingleTokenGauge.sol : velocore__emission(), velocore__gauge();

./src/pools/constant-product/ConstantProductLibrary.sol : velocore__execute().

Recommendation: Follow the official Solidity guidelines.

Found in: 94fc760

Status: Mitigated (These functions are callback functions, and intentionally violate the naming convention to prevent accidental behavior).

I05. Unused Function Parameter

The parameter *gauge* from the function *bribeTokens()* is not being used.

This results in higher Gas consumption, bad developer experience and readability issues.

Path: ./src/pools/constant-product/ConstantProductPool.sol : bribeTokens()->gauge.

Recommendation: Remove unused variables from the code.

Found in: 94fc760

Status: Fixed (Revised commit: 7b62de5)

I06. Test Compilation Error

It is impossible to build the test contracts “out of the box” or measure coverage due to the compilation error.

Path: ./: *

Recommendation: Fix compilation error.

Found in: 94fc760

Status: Reported

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the Consultant cannot guarantee the explicit security of the audited smart contracts.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Appendix 1. Severity Definitions

When auditing smart contracts Hacken is using a risk-based approach that considers the potential impact of any vulnerabilities and the likelihood of them being exploited. The matrix of impact and likelihood is a commonly used tool in risk management to help assess and prioritize risks.

The impact of a vulnerability refers to the potential harm that could result if it were to be exploited. For smart contracts, this could include the loss of funds or assets, unauthorized access or control, or reputational damage.

The likelihood of a vulnerability being exploited is determined by considering the likelihood of an attack occurring, the level of skill or resources required to exploit the vulnerability, and the presence of any mitigating controls that could reduce the likelihood of exploitation.

Risk Level	High Impact	Medium Impact	Low Impact
High Likelihood	**Critical**	**High**	**Medium**
Medium Likelihood	**High**	**Medium**	**Low**
Low Likelihood	**Medium**	**Low**	**Low**

Risk Levels

Critical: Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation.

High: High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation.

Medium: Medium vulnerabilities are usually limited to state manipulations and, in most cases, cannot lead to asset loss. Contradictions and requirements violations. Major deviations from best practices are also in this category.

Low: Major deviations from best practices or major Gas inefficiency. These issues won't have a significant impact on code execution, don’t affect security score but can affect code quality score.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Impact Levels

High Impact: Risks that have a high impact are associated with financial losses, reputational damage, or major alterations to contract state. High impact issues typically involve invalid calculations, denial of service, token supply manipulation, and data consistency, but are not limited to those categories.

Medium Impact: Risks that have a medium impact could result in financial losses, reputational damage, or minor contract state manipulation. These risks can also be associated with undocumented behavior or violations of requirements.

Low Impact: Risks that have a low impact cannot lead to financial losses or state manipulation. These risks are typically related to unscalable functionality, contradictions, inconsistent data, or major violations of best practices.

Likelihood Levels

High Likelihood: Risks that have a high likelihood are those that are expected to occur frequently or are very likely to occur. These risks could be the result of known vulnerabilities or weaknesses in the contract, or could be the result of external factors such as attacks or exploits targeting similar contracts.

Medium Likelihood: Risks that have a medium likelihood are those that are possible but not as likely to occur as those in the high likelihood category. These risks could be the result of less severe vulnerabilities or weaknesses in the contract, or could be the result of less targeted attacks or exploits.

Low Likelihood: Risks that have a low likelihood are those that are unlikely to occur, but still possible. These risks could be the result of very specific or complex vulnerabilities or weaknesses in the contract, or could be the result of highly targeted attacks or exploits.

Informational

Informational issues are mostly connected to violations of best practices, typos in code, violations of code style, and dead or redundant code.

Informational issues are not affecting the score, but addressing them will be beneficial for the project.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Appendix 2. Scope

The scope of the project includes the following smart contracts from the provided repository:

Initial review scope

Repository	https://github.com/velocore/audit
Commit	94fc760
Whitepaper	N/A
Requirements	Link
Technical Requirements	Link
Contracts	File: ./src/lib/PoolBalanceLib.sol SHA3: c5f6083cb2371670ba6b4cfe7af8be7913d09e9f339f2c1fc24a6e391a851d9b File: ./src/lib/UncheckedMemory.sol SHA3: 82cdb5d7ce598139160180803a6acbc6377d50b6564568b89e69fc3084bfffa2 File: ./src/lib/RPow.sol SHA3: 58d7a1a8de10288d7e54a82f7e8d4b35cfbbf16c8ca45c912152cc6cf7c99baf File: ./src/lib/Token.sol SHA3: f1f5ba9bf2be173c897ffe1ceed1f630f05a6633e4273f6d363098f02e2aa0f8 File: ./src/interfaces/IGauge.sol SHA3: 6ecb1ffc69b5fa0ebecf4a786c0764ddacf512ba81bc79749dcde39258fbbdde File: ./src/interfaces/IVault.sol SHA3: 05c137fdb2ac4c6d9ae32a8b91b5429af73b08edaa5119c11c34078a2705bb3d File: ./src/interfaces/IConverter.sol SHA3: f667395dcc2b53cc3d82acf56bca0353c4907e086e538fbbf2b4e4e4fce80d92 File: ./src/pools/constant-product/ConstantProductLibrary.sol SHA3: 3670feab9119f868a067b8cc8b7d3f541152876f3eeedd2af39a40f8935f7a48 File: ./src/pools/constant-product/ConstantProductPool.sol SHA3: 187b03efbbffd5f2121be67a7fec4178ff45808cacd585d0dcb30a7df780fd26 File: ./src/pools/constant-product/ConstantProductPoolFactory.sol SHA3: 171c74bb2018d48fa06767d92601889c12201804d813abd5fc021a06ff7dc924 File: ./src/pools/SingleTokenGauge.sol SHA3: befb14f712720ceef66bf65670aab2e4a53e2c8a42d9d713a83a153d21a54e3b File: ./src/pools/Satellite.sol SHA3: abc1c5734627dff3c6c23ee4a226e42ba1ee159a7cc71944a5ac14423129cb27

Second review scope

Repository	https://github.com/velocore/audit
Commit	bec69c9
Whitepaper	N/A
Requirements	Link
Technical Requirements	Link
Contracts	File: ./src/interfaces/IConverter.sol SHA3: f667395dcc2b53cc3d82acf56bca0353c4907e086e538fbbf2b4e4e4fce80d92 File: ./src/interfaces/IGauge.sol SHA3: 6ecb1ffc69b5fa0ebecf4a786c0764ddacf512ba81bc79749dcde39258fbbdde File: ./src/interfaces/IVault.sol SHA3: 5652e0a3efbcb9b8cef0fadf8cde9d5df70c2904462d321128763fc3e1bf678f File: ./src/lib/PoolBalanceLib.sol SHA3: 41ca7e01c2ccd1177871eb7b7ce2243739d38da43a735712c79e7a45ccaeb6bd File: ./src/lib/RPow.sol SHA3: 13e64fc1af5ec9f57f515f32334dcb85be888289539da689f35afcc615dcd32c File: ./src/lib/Token.sol SHA3: 9db21aa6c384219db1dc303adea660ed9e5ebf06f433bfdce4d155f8257dc4fe File: ./src/lib/UncheckedMemory.sol SHA3: 63ab3ca5b03dcbbe78493e92a7e3dcaff23638aaa98dfb5984262bff1b5a94f3 File: ./src/pools/Satellite.sol SHA3: bb1b62e487881ad65ab8b7f825eca56863cfb6d4266cbaa775ada5ba191189e9 File: ./src/pools/SingleTokenGauge.sol SHA3: c54368ff197eb208e984e7346678163387934891693504 54b96f43f3c81c09d5 File: ./src/pools/constant-product/ConstantProductLibrary.sol SHA3: 230fed8a8fd1659583ddbaabe1b58877a4735a9f8eabd4f2c700300c1092e56b File: ./src/pools/constant-product/ConstantProductPool.sol SHA3: eb6c8da78d9dac31da93bedf016e2814d2c60179cb9e6a891629d82e1e3a0849 File: ./src/pools/constant-product/ConstantProductPoolFactory.sol SHA3: 91cc2e437f3675ca63db67d432ca4430104a73f5a488bfa699c47e5180d0d777

Third review scope

Repository	https://github.com/velocore/audit
Commit	2176c96
Whitepaper	N/A
Requirements	Link
Technical Requirements	Link
Contracts	File: ./src/interfaces/IConverter.sol SHA3: f667395dcc2b53cc3d82acf56bca0353c4907e086e538fbbf2b4e4e4fce80d92 File: ./src/interfaces/IGauge.sol SHA3: 6ecb1ffc69b5fa0ebecf4a786c0764ddacf512ba81bc79749dcde39258fbbdde File: ./src/interfaces/IVault.sol SHA3: 5652e0a3efbcb9b8cef0fadf8cde9d5df70c2904462d321128763fc3e1bf678f File: ./src/lib/PoolBalanceLib.sol SHA3: 41ca7e01c2ccd1177871eb7b7ce2243739d38da43a735712c79e7a45ccaeb6bd File: ./src/lib/RPow.sol SHA3: 13e64fc1af5ec9f57f515f32334dcb85be888289539da689f35afcc615dcd32c File: ./src/lib/Token.sol SHA3: 551e277e2ffa356303e780745a9932b0f98c6dca2e8b6c9b74351b3456e64848 File: ./src/lib/UncheckedMemory.sol SHA3: 63ab3ca5b03dcbbe78493e92a7e3dcaff23638aaa98dfb5984262bff1b5a94f3 File: ./src/pools/Satellite.sol SHA3: 388a2f5f09eb49a3d2fc16d5606f2940e659742e49977bb0612fd24f9c46cceb File: ./src/pools/SingleTokenGauge.sol SHA3: bc47b8b792d935ad43ddd04a137b0b6eec3f0e1815c053a0a20fd9629445cd01 File: ./src/pools/constant-product/ConstantProductLibrary.sol SHA3: 230fed8a8fd1659583ddbaabe1b58877a4735a9f8eabd4f2c700300c1092e56b File: ./src/pools/constant-product/ConstantProductPool.sol SHA3: d7dc175d1820377a7d9b5fe66e94f80036fecc8e7e59cb8c75061dee9eacdb72 File: ./src/pools/constant-product/ConstantProductPoolFactory.sol SHA3: 514fee3c0242155f71c57ac3cc3df33be2d426081c6607a67e95a9f275200f08

Fourth review scope

Repository	https://github.com/velocore/audit
Commit	7b62de5d
Whitepaper	N/A
Requirements	Link
Technical Requirements	Link
Contracts	File: ./src/interfaces/IConverter.sol SHA3: f667395dcc2b53cc3d82acf56bca0353c4907e086e538fbbf2b4e4e4fce80d92 File: ./src/interfaces/IGauge.sol SHA3: 6ecb1ffc69b5fa0ebecf4a786c0764ddacf512ba81bc79749dcde39258fbbdde File: ./src/interfaces/IVault.sol SHA3: 5652e0a3efbcb9b8cef0fadf8cde9d5df70c2904462d321128763fc3e1bf678f File: ./src/lib/PoolBalanceLib.sol SHA3: 229587216ba63edb0bfe31cce1aa4b7780035c75b0b2cf89f24467b7935d1dc7 File: ./src/lib/RPow.sol SHA3: 07be639438868d7d6e1fe7f43bcf5ff1088ea3896216fa61a1c1b8965385b109 File: ./src/lib/Token.sol SHA3: 3fae0ac7bd139f630bd95fd0506556fc13863a4dba4f4b0b4fc1527f7034fee8 File: ./src/lib/UncheckedMemory.sol SHA3: 3a6665a67711f1d744480e92d84480b9905d656a929b2891083f57cd563ca632 File: ./src/pools/Satellite.sol SHA3: bb1b62e487881ad65ab8b7f825eca56863cfb6d4266cbaa775ada5ba191189e9 File: ./src/pools/SingleTokenGauge.sol SHA3: d86d4ef2be84d9bf36e1f7dc6f855a3040e5f92cee7c4e92554ed8615a8cd3d2 File: ./src/pools/constant-product/ConstantProductLibrary.sol SHA3: 230fed8a8fd1659583ddbaabe1b58877a4735a9f8eabd4f2c700300c1092e56b File: ./src/pools/constant-product/ConstantProductPool.sol SHA3: 92bd697fe0b562d0d792497c09c8ea9131b3c90bdd4efdd486d96745baf196cb File: ./src/pools/constant-product/ConstantProductPoolFactory.sol SHA3: 34bb264aa24efac693f271c1bdd30fbd4778ba4291aaa21854535c474a00b10b