HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Leech Protocol
Date: 17 Aug, 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 Leech Protocol
Approved By	Oleksii Zaiats | SC Audits Head at Hacken OÜ
Tags	ERC20 token; Yield Aggregator
Platform	EVM
Language	Solidity
Methodology	Link
Website	https://website.com
Changelog	31.07.2023 - Initial Review 09.08.2023 - Second Review 17.08.2023 - Third 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 Leech Protocol (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

Leech Protocol is a cross-chain yield aggregator. It allows users to deposit funds into different pools with underlying yield-generating strategies. It has the following contracts:

- *LeechRouterOptimism – a contract that users use to make deposits and withdrawals to preferred pools.*
- *LeechRouterBase* – The Router is the main protocol contract, for user interactions and protocol automatizations.
- StrategyVelodrome – FarmLeech Protocol farming strategy for Velodrome.
- BaseFarmStrategy – Base farming strategy.
- Helpers – Leech Protocol helpers and utilities.

Privileged roles

BaseFarmStrategy and *StrategyVelodromeFarm* are using OwnableUpgradeable to restrict access. Contract owner can:

- Sets paths for token swap.
- Sets fee taken by the Leech protocol.
- Sets the treasury address.
- Sets the controller address.
- Sets slippage tolerance.

LeechRouterBase and LeechRouterOptimism are using AccessControlUpgradeable to restrict access.
Addresses with *ADMIN_ROLE* can:

- Move liquidity to the new strategy.
- Move liquidity to another blockchain.
- Change strategy.
- Change address of validator, finalizer, transporter and treasury.
- Set a new pool, strategy and router.
- Set withdrawal delay.
- Unpause contract.
- Change Uniswap based router.
- Ban and unban given address.
- Set deposit token and minimal value that should be deposited.

Addresses with *PAUSER_ROLE* can pause a contract.

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 **10** out of **10**.

- Functional requirements are provided.
- Technical description is provided.

Code quality

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

- The code follows style guides and best practices.
- The development environment is configured.

Test coverage

Code coverage of the project is **100%** (branch coverage).

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.

Summary

According to the assessment, the Customer's smart contract has the following score: **10**. 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
31 July 2023	4	5	6	3
09 Aug 2023	3	0	0	0
17 Aug 2023	0	0	0	0

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

Risks

- Iterating over a dynamic array populated with custom tokenId can lead to Gas limit denial of service if the number of tokenId goes out of control.
- The multisig ADMIN role can perform migrations at any time without restrictions. This is part of the business logic of the protocol.

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.	Not Relevant	
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.	Passed	
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.	Not Relevant	
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.	Not Relevant	
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.	Passed	
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.	Passed	
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.	Passed	
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

C02. Signed Message Replay Attack

Impact	High
Likelihood	High

The *LeechRouterBase* contract is not utilizing EIP712 for signatures and deadlines. The current implementation of signed messages lacks mechanisms to prevent replay attacks.

Without a standard such as EIP712, signed messages can potentially be reused on other chains (replay attacks).

This can result in the execution of a signed transaction when it is not desired by a signer.

Path: ./router/LeechRouterBase.sol : checkWhitelist();

Recommendation: implement EIP712 for signatures in the protocol. EIP712 offers a standard way to structure data and generate signatures, which would significantly enhance the security of the contract by protecting it from replay attacks. Add nonce and chainId parameters to each signature, which will highly increase security.

Found in: f1019667115308f0945d38120d9af9c88c853501

Status: Mitigated (The customer has explained that the possibility of replaying doesn't pose a threat to their business logic, as it is by should be possible to "replay" the message within the given timeframe)

■■■ High

H01. Front-Running; Sandwich Attack

Impact	High
Likelihood	Medium

The minimum return amount is not specified, or *0* is specified as a minimum value during operations with the swap router.

This may lead to unexpected token loss and wrong swap rates during the interaction with the system.

Paths:

./strategies/VelodromeFarm/StrategyVelodromeFarm.sol : _deposit(), _withdraw();

./router/LeechRouterOptimism.sol : _swap();

./router/LeechRouterBase.sol : crosschainDeposit();

Recommendation: specify the minimum amount manually or use oracles to calculate the minimum amount or use newer versions of the DEXs.

Found in: f1019667115308f0945d38120d9af9c88c853501

Status: Fixed (Revised commit: 2f79620)

Medium

M01. Missing Validation

Impact	Medium
Likelihood	Medium

It is considered that the project should be consistent and contain no self-contradictions.

According to docs the value *protocolFee* should be lower than *MAX_FEE* value (12%). However, in the *initialize()* the validation is missed.

According to docs the value *slippage* should be lower than *DENOMINATOR* value and greater than 0. However, in the *initialize()* the validation is missed.

This may lead to unexpected value processed by the contract.

Path: ./strategies/VelodromeFarm/StrategyVelodromeFarm.sol : initialize();

Recommendation: implement the validations of *protocolFee* and *slippage* in *initialize()*.

Found in: f1019667115308f0945d38120d9af9c88c853501

Status: Fixed (Revised commit: 5202e8ca796e989437d07e390b7db7aef7dce5ac)

M02. Checks-Effects-Interactions Pattern Violation

Impact	Medium
Likelihood	Medium

The Checks-Effects-Interactions pattern is violated. During the functions, some state variables are updated after the external calls.

Paths: ./strategies/BaseFarmStrategy.sol : deposit()

./router/LeechRouterBase.sol : finalizeDeposit(), finalizeWithdrawal();

Recommendation: implement the functions according to the Checks-Effects-Interactions pattern or use reentrancy locks.

Found in: f1019667115308f0945d38120d9af9c88c853501

Status: Fixed (Revised commit: eea595310c0937e654acb446fc3276350497edf4)

M03. Redundant Modifier

Impact	Medium
Likelihood	Medium

The withdrawal mechanisms, withdraw() and crosschainWithdraw(), in LeechRouterBase.sol do not need to be payable since they do not expect to receive any native coins in transactions.

Path: ./router/LeechRouterBase.sol : withdraw();

Recommendation: remove payable modifiers.

Found in: f1019667115308f0945d38120d9af9c88c853501

Status: Fixed (Revised commit: eea595310c0937e654acb446fc3276350497edf4)

M04. Uninitialized Implementation

Impact	Medium
Likelihood	High

It is considered following best practices to avoid unclear situations and prevent common attack vectors.

It is not recommended to leave an implementation contract uninitialized. An uninitialized implementation contract can be taken over by an attacker.

This may lead to the attacker taking over the contract and impacts the proxy.

Paths:

./strategies/VelodromeFarm/StrategyVelodromeFarm.sol : constructor();

./router/LeechRouterOptimism.sol : constructor();

Recommendation: follow common best practices, invoke the _disableInitializers function in the constructor to automatically lock the contract when it is deployed.

Found in: f1019667115308f0945d38120d9af9c88c853501

Status: Fixed (Revised commit: eea595310c0937e654acb446fc3276350497edf4)

M05. Highly Permissive Admin Access

Impact	High
Likelihood	Medium

ADMIN role can perform migrations anytime without notifying anyone.

If a key leak were to occur, the potential consequences could be significant, potentially leading to security breaches and undermining the overall integrity of the system.

Paths: ./router/LeechRouterBase.sol: migration(), crosschainMigration();

Recommendation: to ensure transparency and accountability, it is advised to provide a comprehensive explanation of highly-permissive access in the system's public documentation. This would help to ensure that users are fully informed of the implications of such access and can make informed decisions accordingly.

Found in: f1019667115308f0945d38120d9af9c88c853501

Status: Mitigated (The functionality and its necessity for the business logic was explained in the protocol documentation)

Low

L01. Missing Zero Address Validation

Impact	Low
Likelihood	Low

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

This can lead to unwanted external calls to 0x0.

Paths:

./strategies/VelodromeFarm/StrategyVelodromeFarm.sol : initialize();

./router/LeechRouterOptimism.sol : initialize();

Recommendation: implement zero address checks.

Found in: f1019667115308f0945d38120d9af9c88c853501

Status: Fixed (Revised commit: 21decfda4f6de62f3a0113a4ca9c810025f464da280d015b2c17f511b9bb0fa7)

L02. Redundant Import

Impact	Low
Likelihood	Medium

The contract *BaseFarmStrategy* inherits OpenZeppelin's *IERC20* but it is already part of *SafeERC20*.

The contract *StrategyVelodromeFarm* inherits OpenZeppelin's *IERC20* but it is already part of *BaseFarmStrategy*.

The contract *StrategyVelodromeFarm* inherits OpenZeppelin's *SafeERC20* but it is already part of *BaseFarmStrategy*.

The contract *LeechRouterBase* inherits OpenZeppelin's *IERC20* but it is already part of *SafeERC20*.

The contract *LeechRouterBase* inherits OpenZeppelin's *Initializable* but it is already part of *AccessControlUpgradeable*.

The contract *LeechRouterOptimism* inherits OpenZeppelin's *IERC20* but it is already part of *LeechRouterBase*.

The contract *Helpers* inherits OpenZeppelin's *IERC20* but it is already part of *SafeERC20*.

The redundancy in inheritance and import can lead to unnecessary gas consumption during deployment and potentially impact code quality.

Paths:

./strategies/VelodromeFarm/StrategyVelodromeFarm.sol : *;

./strategies/BaseFarmStrategy.sol : *;

./router/LeechRouterOptimism.sol : *;

./router/LeechRouterBase.sol : *;

./libraries/Helpers.sol : *;

Recommendation: remove redundant import and inheritance to save Gas on deployment and increase the code quality.

Found in: f1019667115308f0945d38120d9af9c88c853501

Status: Fixed (Revised commit: 21decfda4f6de62f3a0113a4ca9c810025f464da280d015b2c17f511b9bb0fa7)

L03. Missing Events

Impact	Low
Likelihood	Medium

Events for critical state changes should be emitted for tracking things off-chain.

Paths:

./strategies/VelodromeFarm/StrategyVelodromeFarm.sol : setRoutes(), _withdraw(), _deposit();

./strategies/BaseFarmStrategy.sol : setFee(), setTreasury(), setController(), setSlippage(), migrate();

./router/LeechRouterOptimism.sol : setRoutes();

./router/LeechRouterBase.sol : migration(), crosschainMigration(), migrateAllocations(), finalizeCrosschainMigration(), setValidator(), setFinalizer(), setStrategy(), setPool(), setRouter(), setTransporter(), setWithdrawDelay(), switchWhitelistStatus(), setUniV2(), setTreasury(), setBanned(), setDepositToken();

Recommendation: create and emit related events.

Found in: f1019667115308f0945d38120d9af9c88c853501

Status: Fixed (Revised commit: 21decfda4f6de62f3a0113a4ca9c810025f464da280d015b2c17f511b9bb0fa7)

L04. Funds Lock

Impact	Low
Likelihood	Medium

The contract accepts token deposits but lacks a withdrawal mechanism, which can result in funds being locked in the contract. The deposit() function in the LeechRouterBase.sol contract accepts native tokens however, the function is not designed to process the native tokens this will lead the sent amount to be stuck in the contract.

This may lead to imbalances and miscalculations in the project.

Path: ./router/LeechRouterBase.sol: deposit();

Recommendation: remove the payable modifier from this function so that it will not accept native tokens.

Found in: f1019667115308f0945d38120d9af9c88c853501

Status: Fixed (Revised commit: 2f79620)

Informational

I01. Natspec Mismatch

setDepositToken() and *pause()* functions' NatSpec descriptions are different than the implementation

Path: ./router/LeechRouterBase.sol : setDepositToken(), pause();

Recommendation: update the NatSpecs.

Found in: f1019667115308f0945d38120d9af9c88c853501

Status: Fixed (Revised commit: 21decfda4f6de62f3a0113a4ca9c810025f464da280d015b2c17f511b9bb0fa7)

I02. Public Functions That Should Be External

Functions that are only called from outside the contract should be defined as external. External functions are much more gas efficient compared to public functions.

Paths:

./strategies/BaseFarmStrategy.sol : deposit(), withdraw(), quotePotentialWithdraw();

./strategies/VelodromeFarm/StrategyVelodromeFarm.sol : quotePotentialWithdraw();

./router/LeechRouterOptimism.sol : initialize();

Recommendation: make these functions external.

Found in: f1019667115308f0945d38120d9af9c88c853501

Status: Fixed (Revised commit: 21decfda4f6de62f3a0113a4ca9c810025f464da280d015b2c17f511b9bb0fa7)

I03. Public Functions That Should Be Internal; Best Practice Violation

Functions that are only called inside the contract should be defined as internal. Internal functions are much more Gas efficient compared to public functions.

Path: ./router/LeechRouterBase.sol : initialize();

Recommendation: make this function internal. Change name of *LeechRouterBase* *initialize()* function to *__LeechRouterBase_init().*

Found in: f1019667115308f0945d38120d9af9c88c853501

Status: Reported

I04. Redundant Variable Value Assignment

The *withdrawDelay* value assignment of *0* is redundant in *initialize()*. Default value of *uint* is *0*.

Path: ./router/LeechRouterBase.sol : initialize();

Recommendation: remove redundant variable value assignment.

Found in: f1019667115308f0945d38120d9af9c88c853501

Status: Fixed (Revised commit: 21decfda4f6de62f3a0113a4ca9c810025f464da280d015b2c17f511b9bb0fa7)

I05. Typos

There is a typo in NatSpec's description of *setTreasury()* in *BaseFarmStrategy*. It says tresury, however, it should be treasury.

There is a typo in NatSpec's description of *protocolFee* in *BaseFarmStrategy*. It says comission, however, it should be commission.

There is a typo in comments of *deposit()* in *BaseFarmStrategy*. It says allcation, however, it should be allocation.

There is a typo in NatSpec's description of *routes* in *StrategyVelodromeFarm*. It says exchnage, however, it should be exchange.

There is a typo in variable names used in *_deposit()* in *StrategyVelodromeFarm*. It says swapedAmounts, however, it should be swappedAmounts.

There is a typo in variable names used in *_swap()* in *LeechRouterOptimism*. It says swapedAmounts, however, it should be swappedAmounts.

There is a typo in NatSpec's description of *enabled()* in *LeechRouterBase*. It says exlude, however, it should be exclude.

There is a typo in comments in *crosschainDeposit()* in *LeechRouterBase*. It says transtorder, however, it should be transporter.

There is a typo in NatSpec's description of *withdraw()* , *finalizeWithdrawal()* and *crosschainWithdraw()* in *LeechRouterBase*. It says withdrwalas, however, it should be withdrawals.

There is a typo in NatSpec's description of *setBanned()* in *LeechRouterBase*. It says addess, however, it should be address.

There is a typo in NatSpec's description and variable names used in *_swap()* in *LeechRouterBase*. It says swapedAmounts, however, it should be swappedAmounts.

There is a typo in NatSpec's description of *withdraw()* , *finalizeWithdrawal()* and *crosschainWithdraw()* in *ILeechRouter*. It says withdrwalas, however, it should be withdrawals.

There is a typo in NatSpec's description of *WithdrawCompleted* and *CrosschainWithdrawCompleted* in *ILeechRouter*. It says completeing, however, it should be completing.

There is a typo in the *PercentExeedsMaximalValue* event name in *Helpers*. It says PercentExeedsMaximalValue, however, it should be PercentExceedsMaximalValue.

Paths:

./strategies/BaseFarmStrategy.sol : setTreasury(), protocolFee, deposit();

./strategies/VelodromeFarm/StrategyVelodromeFarm.sol : routes, _deposit();

./router/LeechRouterOptimism.sol : _swap();

./router/LeechRouterBase.sol : enabled(), crosschainDeposit(), withdraw(), finalizeWithdrawal(), crosschainWithdraw(), setBanned(), _swap();

./router/ILeechRouter.sol : withdraw(), finalizeWithdrawal(), crosschainWithdraw(), WithdrawCompleted, CrosschainWithdrawCompleted;

./libraries/Helpers.sol : PercentExeedsMaximalValue;

Recommendation: fix typos.

Found in: f1019667115308f0945d38120d9af9c88c853501

Status: Fixed (Revised commit: 21decfda4f6de62f3a0113a4ca9c810025f464da280d015b2c17f511b9bb0fa7)

I06. Redundant Code

The contract inherits OpenZeppelin's AccessControlUpgradeable but there is a custom role based mechanism implemented to restrict access.

The redundancy in code can lead to unnecessary gas consumption during deployment and potentially impact code quality.

Path: ./router/LeechRouterBase.sol : onlyFinalizer();

Recommendation: create FINALIZER_ROLE using AccessControlUpgradeable.

Found in: f1019667115308f0945d38120d9af9c88c853501

Status: Reported

I07. Inconsistency in Errors

There is inconsistency in returned error types.

Inside *setSlippage()*, if *_slippage* parameter is bigger than *DENOMINATOR* value, then transaction is reverted with *SlippageTooHigh* error.

Inside *initialize()*, if the *slippage* parameter is bigger than the *DENOMINATOR* value, then the transaction is reverted with *BadAmount* error.

Paths:

./strategies/BaseFarmStrategy.sol : setSlippage();

./strategies/VelodromeFarm/StrategyVelodromeFarm.sol : initialize();

Recommendation: fix the mismatch between errors used in the same check.

Found in: 5202e8ca796e989437d07e390b7db7aef7dce5ac

Status: Fixed (Revised commit: 21decfda4f6de62f3a0113a4ca9c810025f464da280d015b2c17f511b9bb0fa7)

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.

Appendix 2. Scope

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

Initial review scope

Repository	https://github.com/Leech-Protocol/leech-contracts/tree/feature/dev-audit
Commit	f1019667115308f0945d38120d9af9c88c853501
Whitepaper	https://drive.google.com/file/d/1P4bDeogfeWANMeTuXz1EeqgtLUwnMEUi/view?usp=sharing
Requirements	https://github.com/Leech-Protocol/leech-contracts/blob/feature/dev-audit/docs/SC_documentation.md
Technical Requirements	https://github.com/Leech-Protocol/leech-contracts/blob/feature/dev-audit/docs/SC_documentation.md
Contracts	File: contracts/interfaces/ILeechRouter.sol SHA3: 1578cfa57394754f617c492b293eed42bfb29c11a47928ce00de216c72b795fb File: contracts/interfaces/ILeechTransporter.sol SHA3: f7f9952d85bca717533ef93bf533572586710d09e81ae8a179d8a95b01c3f3b3 File: contracts/interfaces/IRouterVelodrome.sol SHA3: 1b82704c8ffe94f9a1b734002f9d5deb3068dccf37e75ae440906af9a6a880c7 File: contracts/interfaces/IStrategyMasterchefFarmV2.sol SHA3: 396e11e844f7b8ba8a555a8d107fbfc1806e73b72fb382b509489cb5bf57ee0a File: contracts/libraries/Helpers.sol SHA3: 84bba73e8e3687fdac5c86a301b63fc8fc34d66b0fb2c02bb26c83df94c41581 File: contracts/router/ILeechRouter.sol SHA3: 729d05390ef98c39faabca9793923af4b6999a45e330783ff6e023c61b0f6e23 File: contracts/router/LeechRouterBase.sol SHA3: 33e827f8a9be224cd988f492f89d89e07437b7aef0956325883e70551da7bc0f File: contracts/router/LeechRouterOptimism.sol SHA3: 41b8df5da0462f259c5285503900b1b86fcce84a0a2ccc822d59a2507d614813 File: contracts/strategies/BaseFarmStrategy.sol SHA3: 883405390d27d2d8d1a367da03daa0c099b335ada020ed26c6162020fb8cc692 File: contracts/strategies/VelodromeFarm/IGauge.sol SHA3: 233d8f9e595f4f0ffd16226111e4c215fa386eebdb79ed0e531d8537987e235d File: strategies/VelodromeFarm/IVelodromePair.sol SHA3: ecc5c6733d8c43eaa9d2c7896c70fe4e6f425601be3307b27c02d1b491d4e17f File: contracts/strategies/VelodromeFarm/StrategyVelodromeFarm.sol SHA3: cdb3b2755d3201c93f5d1ced7668d3fa127cf183866bf11391bb77685fab9fa7

Second review scope

Repository	File: leech-contracts.zip SHA3: 2f796208115b3d7eb49c18a6229122823172c895e241a8efc7de13585e9af683
Commit	-
Whitepaper	https://drive.google.com/file/d/1P4bDeogfeWANMeTuXz1EeqgtLUwnMEUi/view?usp=sharing
Requirements	https://github.com/Leech-Protocol/leech-contracts/blob/feature/dev-audit/docs/SC_documentation.md
Technical Requirements	https://github.com/Leech-Protocol/leech-contracts/blob/feature/dev-audit/docs/SC_documentation.md
Contracts	File: interfaces/ILeechRouter.sol SHA3: 1578cfa57394754f617c492b293eed42bfb29c11a47928ce00de216c72b795fb File: interfaces/ILeechTransporter.sol SHA3: f7f9952d85bca717533ef93bf533572586710d09e81ae8a179d8a95b01c3f3b3 File: interfaces/IRouterVelodrome.sol SHA3: 0eadbc5ae1627ff67add07e4e7be6653130263df34ae75bdb344e1d78f6e75e0 File: interfaces/IStrategyMasterchefFarmV2.sol SHA3: 396e11e844f7b8ba8a555a8d107fbfc1806e73b72fb382b509489cb5bf57ee0a File: libraries/Helpers.sol SHA3: 84bba73e8e3687fdac5c86a301b63fc8fc34d66b0fb2c02bb26c83df94c41581 File: router/ILeechRouter.sol SHA3: 3f565388e39fd4da65b7cd9e50d3c90c9a1e6ccb066296b9fc5310c990e2707a File: router/LeechRouterBase.sol SHA3: dc4c462607030e4076dba30c0c59692abe249b12ed41acb03fae380eaf3c56dc File: router/LeechRouterOptimism.sol SHA3: e79f20701c9a960ed66b0af02f0fb324bd3af800b2268be4425edaaf746c40de File: strategies/VelodromeFarm/IGauge.sol SHA3: 233d8f9e595f4f0ffd16226111e4c215fa386eebdb79ed0e531d8537987e235d File: strategies/BaseFarmStrategy.sol SHA3: 883405390d27d2d8d1a367da03daa0c099b335ada020ed26c6162020fb8cc692 File: strategies/VelodromeFarm/IVelodromePair.sol SHA3: ecc5c6733d8c43eaa9d2c7896c70fe4e6f425601be3307b27c02d1b491d4e17f File: strategies/VelodromeFarm/StrategyVelodromeFarm.sol SHA3: bcd05698b964917e9ad9981847436a789eb08f4320dceb9f1fcaeeb9c323f9e6

Third review scope

Repository	File: leech-contracts.zip SHA3: 21decfda4f6de62f3a0113a4ca9c810025f464da280d015b2c17f511b9bb0fa7
Commit	-

Whitepaper	https://drive.google.com/file/d/1P4bDeogfeWANMeTuXz1EeqgtLUwnMEUi/view?usp=sharing
Requirements	https://github.com/Leech-Protocol/leech-contracts/blob/feature/dev-audit/docs/SC_documentation.md
Technical Requirements	https://github.com/Leech-Protocol/leech-contracts/blob/feature/dev-audit/docs/SC_documentation.md
Contracts	File: interfaces/ILeechRouter.sol SHA3: 181a3b51376b0e0f0cc091d8b10076133b1e4c91e8fb01a0b8167d880873f247 File: interfaces/ILeechTransporter.sol SHA3: 8896d8f2389d284a9a34ecda221830874360ab9dcf342ca6eee91bd7b2811fe7 File: interfaces/IRouterVelodrome.sol SHA3: 6d4689ce40729b74616100aabf560f060520cd79812f5feffe0dc681f9ee16ae File: interfaces/IStrategyMasterchefFarmV2.sol SHA3: 542d0e929ec3496b8ab5358ed8b21c30632bd5f47b51bbe81e81495eb4b72f1d File:libraries/Helpers.sol SHA3: e764e5d9eb18fb415d8f793b0cab62d1f63a293d7d822964af4bdf9b5e1bd460 File: /router/ILeechRouter.sol SHA3: afa38549352ac00f277697d5232a63cf90ec319d6e548390ccdd36cae2cf060a File: router/LeechRouterBase.sol SHA3: af8d4ceaf73ad0e97b35e69e0b764215535860273c54d1faee22eea3b9621a39 File: router/LeechRouterOptimism.sol SHA3: e78a6750b02475649e163fd9e9d1cc7f8ec73f39294ff10138384e30b222c6f8 File: strategies/VelodromeFarm/IGauge.sol SHA3: 20f4c8826086592814a9181381968dbd1d09fad93508ba0fa385e36558939e81 File: strategies/BaseFarmStrategy.sol SHA3: a15de9a52cfb0b2ae96f0426975ed451ad5dd4fcb39c482ffd7ad86517c076aa File: strategies/VelodromeFarm/IVelodromePair.sol SHA3: 14e483b333b865f8e766b03c6bad7b5acdba2174b84edca209cdcfb549e4b07a File: strategies/VelodromeFarm/StrategyVelodromeFarm.sol SHA3: a068d5c6dd46d7b6ea0d82e017d7d645d7faa3ca495b5cb911a8fe58c077ccff