NAME OF THE COMPANY

SMART CONTRACT AUDIT

zokyo

June 16th 2022 | v. 1.0

Security Audit Score

PASS

Zokyo's Security Team has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges.

TECHNICAL SUMMARY

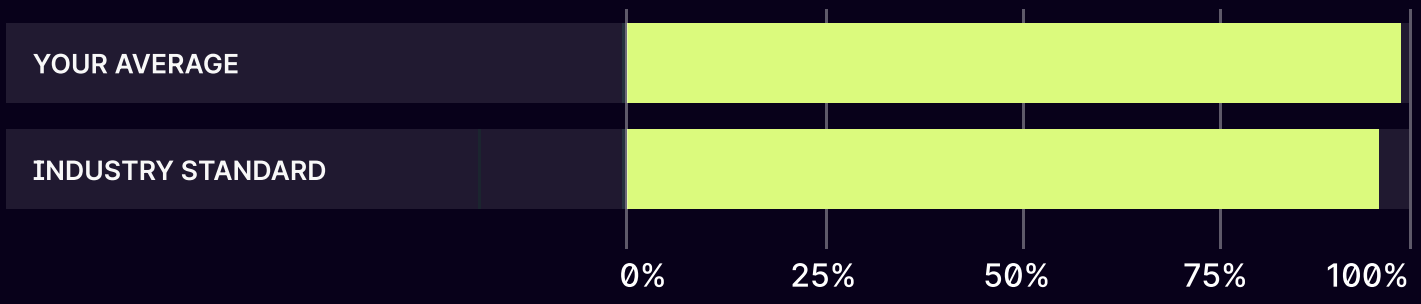
This document outlines the overall security of the Unifarm AMM smart contracts, evaluated by Zokyo's Blockchain Security team.

The scope of this audit was to analyze and document the Unifarm AMM smart contract codebase for quality, security, and correctness.

Contract Status

There was one critical issue found during the audit. (See Complete Analysis)

Testable Code



The testable code is 100%, which is above the industry standard of 95%.

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the Ethereum network's fast-paced and rapidly changing environment, we at Zokyo recommend that the Unifarm AMM team put in place a bug bounty program to encourage further and active analysis of the smart contract.

Table of Contents

Auditing Strategy and Techniques Applied 3

Executive Summary 4

Structure and Organization of Document 5

Complete Analysis 6

Code Coverage and Test Results for all files 14

AUDITING STRATEGY AND TECHNIQUES APPLIED

The Smart contract's source code was taken from the Unifarm AMM repository.

Repository: https://github.com/themohitmadan/unifarm_amm/tree/development

Last commit: a52ac3ca9f8534fbcc3a2ba908dfe29373fe9f74

Within the scope of this audit Zokyo auditors have reviewed the following contract(s):

- AMMUtility
- ReentracyGuard
- UnifarmRouter02
- BaseRelayRecipient
- MultiSigWallet
- Ownable
- UnifarmERC20
- UnifarmFactory
- GovernorBravoDelegator
- GovernorBravoDelegate
- UnifarmPair

Throughout the review process, care was taken to ensure that the contract:

- Implements and adheres to existing standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of resources, without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the latest vulnerabilities;
- Whether the code meets best practices in code readability, etc.

Zokyo's Security Team has followed best practices and industry-standard techniques to verify the implementation of Unifarm AMM smart contracts. To do so, the code is reviewed line-by-line by our smart contract developers, documenting any issues as they are discovered. Part of this work includes writing a unit test suite using the Truffle testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

01	Due diligence in assessing the overall code quality of the codebase.	02	Cross-comparison with other, similar smart contracts by industry leaders.
03	Testing contract logic against common and uncommon attack vectors.	04	Thorough, manual review of the codebase, line-by-line.

Executive Summary

There was one critical issue found during the audit. All the mentioned findings may have an effect only in case of specific conditions performed by the contract owner.

Contracts are well written and structured. The findings during the audit have no impact on contract performance or security, so it is fully production-ready.

Despite the fact, the expected logic is managing all vestings by the owner, it should be careful with parameters to avoid mistakes during the vesting process.

STRUCTURE AND ORGANIZATION OF DOCUMENT

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. Issues tagged “Verified” contain unclear or suspicious functionality that either needs explanation from the Customer’s side or it is an issue that the Customer disregards as an issue. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

Critical

The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

High

The issue affects the ability of the contract to compile or operate in a significant way.

Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

Low

The issue has minimal impact on the contract’s ability to operate.

Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

CRITICAL | RESOLVED

In contract AMMUtility, there is a dangerous external call to a user-inputted address with a user input calldata, in internal function _swap at line 86, a malicious user could carefully craft an attack here where the _swapTarget variable will have the address of a token that an user would have previously allowed the AMMUtility to spend his balance and the calldata would be the sighash of the transferFrom function with it's parameter, using this crafter input a malicious actor would be able to steal a user funds of an arbitrary token that has given AMMUtility contract the allowance to use his balance and that is an action that is happening already because the swap function from AMMUtility is already using the transferFrom function.

POC:
https://drive.google.com/file/d/1C6Zc76E4UREwlIm465yqUxzwobFWnpMG/view?usp=sharing

MEDIUM | RESOLVED

Contract UnifarmERC20 will not be able to use the relay transactions functionality, even if it's inheriting BaseRelayRecipient because the trustedForwarder address is never initialized with a value, so it will always have the default value which is the address(0), which means that when the function _msgSender() is called, the condition from line 35 in BaseRelayRecipient will never be true, we noticed that UnifarmPair is inheriting the UnifarmERC20 and there trustedForwarder it is initialized, but if you would use the UnifarmERC20 on it's own it will not work.

Recommendation:
Add a setter for trustedForwarder or initalized it in the constructor.

INFORMATIONAL | RESOLVED

In contract BaseRelayRecipient in modifier trustedForwarderOnly at line 20, there is a casting from trustedForwarder to address(trustedForwarder) but trustedForwarder is already a variable of type address so the casting is not necessarily.

Recommendation::
Eliminate the casting to address as it is not necessary

INFORMATIONAL | RESOLVED

In contract MultiSigWallet, there is the modifier called transactionExists which is checking if a transactionId is valid by checking the destination variable and is returning the ERR_INVALID_TARGET, all of this is not very consistent and it's not in term with best practices for code context consistency.

Recommendation::
Either change the modifier name to something similar as 'onlyValidTarget' or add an variable to the transaction structure of type bool called exist which will be false by default and change the error message in something similar with "ERR_INVALID_TX"

INFORMATIONAL | RESOLVED

In contract GovernorBravoDelegator, at line 44 function _setImplementation is declared with an "_" at the beginning of the name, a convention that is specific to the internal functions.

Recommendation::
Change the function name to "setImplementation" therefore removing the "_"

	AMMUtility	AMMUtility ReentracyGuard
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

	UnifarmRouter02	BaseRelayRecipient
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

	MultiSigWallet	Ownable
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

	UnifarmERC20	UnifarmFactory
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

	GovernorBravoDelegate	UnifarmPair
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

	GovernorBravoDelegator
Re-entrancy	Pass
Access Management Hierarchy	Pass
Arithmetic Over/Under Flows	Pass
Unexpected Ether	Pass
Delegatecall	Pass
Default Public Visibility	Pass
Hidden Malicious Code	Pass
Entropy Illusion (Lack of Randomness)	Pass
External Contract Referencing	Pass
Short Address/ Parameter Attack	Pass
Unchecked CALL Return Values	Pass
Race Conditions / Front Running	Pass
General Denial Of Service (DOS)	Pass
Uninitialized Storage Pointers	Pass
Floating Points and Precision	Pass
Tx.Origin Authentication	Pass
Signatures Replay	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Zokyo Secured team

As part of our work assisting Unifarm AMM in verifying the correctness of their contract code, our team was responsible for writing integration tests using the Truffle testing framework.

Tests were based on the functionality of the code, as well as a review of the Unifarm AMM contract requirements for details about issuance amounts and how the system handles these.

AMMUtility

- ✓ should deploy, all checks (93ms)
- ✓ should update fee, all checks
- ✓ should update fee to, all checks
- ✓ should whitelist swap targets
- ✓ should swap tokens, all checks (248ms)
- ✓ should withdraw tokens, all cases (38ms)
- ✓ should withdraw eth, all cases
- ✓ should receive ether, all cases (49ms)
- ✓ should renounce ownership
- ✓ should transfer ownership, all cases
- ✓ should fail re-entracy attck (75ms)

Governance

- ✓ should initialize, all checks
- ✓ should update token permission, all checks
- ✓ should check version
- ✓ should propose, all checks
- ✓ should queuq, all checks
- ✓ should execute, all checks
- ✓ should check get actions
- ✓ should check get receipt
- ✓ should check state, all cases
- ✓ should cast vote
- ✓ should cast vote with reason
- ✓ should cast vote by sig, all checks
- ✓ should set voting delay, all checks
- ✓ should set voting period, all checks
- ✓ should set proposal threshold, all checks
- ✓ should claim reward, all checks

- ✓ should check if can claim reward, all cases
- ✓ should initiate, all cases
- ✓ should set pending admin, all checks
- ✓ should accept admin, all checks
- ✓ should check math functions, all cases

MultiSigWallet

- ✓ should deploy, all checks
- ✓ should add owner, all checks
- ✓ should remove owner, all checks
- ✓ should replace owner, all checks
- ✓ should change requirement
- ✓ should submit transaction, all checks
- ✓ should revoke confirmation
- ✓ should executa transaction, all checks
- ✓ should check is tx confirmed
- ✓ should check get confirmation count
- ✓ should check get owners
- ✓ should check get confirmations
- ✓ should check get transaction ids, all cases

UnifarmERC20

- ✓ should approve, all checks
- ✓ should transfer tokens, all checks
- ✓ shoudl transfer from tokens, all checks (40ms)
- ✓ should increase allowance, all checks
- ✓ should descrease allowance, all checks
- ✓ should mint tokens, all checks
- ✓ should burn tokens, all checks
- ✓ should check version
- ✓ should permit, all checks

UnifarmFactory

- ✓ should deploy factory, all checks
- ✓ should create pair, all checks (66ms)
- ✓ should set fee to, all checks
- ✓ should update swap fee config, all checks (51ms)
- ✓ update lp fee config, all checks (51ms)
- ✓ should set trusted forwarder, all checks
- ✓ should check version
- ✓ should cehck all pairs length
- ✓ should check pair code hash

UnifarmPair

- ✓ check the deployment and the factory addrs
- ✓ check get reservers

- ✓ update trusted forwarder
- ✓ check sync the pools, all checks (49ms)
- ✓ should skim, all checks (72ms)
- ✓ initialization all cases, chechk (45ms)
- ✓ should check mint, all cases (80ms)
- ✓ should check mint, all cases, liq bigger then 0 (75ms)
- ✓ should cehck mint, fail liq 0 (70ms)
- ✓ should burn tokens, all cases (256ms)
- ✓ should burn tokens, eth fee bigger then 0 (117ms)
- ✓ should burn tokens, mintFee rootK smaller then rootKLast (136ms)
- ✓ swap 2 tokens, insuficient out amount and liq
- ✓ swap 2 tokens, fail zero address & insf inpt amnt (63ms)
- ✓ swap 2 tokens, fail invalid to
- ✓ swap 2 tokens, amounts 0 both but not simultaniously, data length bigger then 0 (49ms)
- ✓ swap 2 tokens, amount0In bigger then 0, amount1In 0, fail unifarm K, reverse fail again, both equal works and emit (634ms)
- ✓ should try re-entracy to check effect of lock mechanism

UnifarmRouter02

- ✓ should check deployment, all checks (296ms)
- ✓ should add liqudity, all checks (222ms)
- ✓ should all liquidity eth, all checks (351ms)
- ✓ should remove liqudity, all checks (93ms)
- ✓ should remove liquidity eth, all checks (58ms)
- ✓ should remove liquidity with permit, all checks (56ms)
- ✓ should remove liqudity eth with permit, all checks (62ms)
- ✓ should remove liqudity eth with fee on trasnfer, all checks (70ms)
- ✓ should remove liqudity eth with permit and fee on trasnfer, all checks (87ms)
- ✓ should swap exact tokens for tokens, all checks (137ms)
- ✓ should swap tokens for exact tokens, all checks (129ms)
- ✓ should swap exact eth for tokens, all checks (139ms)
- ✓ should swap tokens for exact eth, all checks (166ms)
- ✓ should swap exact tokens for eth, all checks (159ms)
- ✓ should swap eth for exact token, all checks (243ms)
- ✓ should swap exact tokens for tokens supporting fee on transfer tokens, all checks (199ms)
- ✓ should swap exact eth for tokens suppporting fee on transfer tokens, all checks (218ms)
- ✓ should swap exact tokens for eth supporting fee on transfe tokens (206ms)
- ✓ should check quoate
- ✓ should check get amount out
- ✓ should check get amount in
- ✓ should check get amounts out
- ✓ should check get amounts in
- ✓ should send ether to contract, all checks

105 passing (15s)

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	% UNCOVERED LINES
AMMUtility.sol	100	100	100	100	
UnifarmRouter02.so	100	100	100	100	
ReentrancyGuard.sol	100	100	100	100	
UnifarmLibrary.sol	100	100	100	100	
UQ112×112.sol	100	100	100	100	
TransferHelper.sol	100	100	100	100	
SafeMath.sol	100	100	100	100	
Math.sol	100	100	100	100	
GovernorBravoDelegate.sol	100	100	100	100	
GovernorBravoDelegator.sol	100	100	100	100	
UnifarmPair.sol	100	100	100	100	
UnifarmFactory.sol	100	100	100	100	
UnifarmERC20.sol	100	100	100	100	
Ownable.sol	100	100	100	100	
MultiSigWallet.sol	100	100	100	100	
BaseRelayRecipient.sol	100	100	100	100	
All files	**100**	**100**	**100**	**100**	

We are grateful to have been given the opportunity to work with the Unifarm AMM team.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

Zokyo's Security Team recommends that the Unifarm AMM team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

zokyo