SpiritSwap

SMART CONTRACT AUDIT

ZOKYO.

May, 26th, 2022 | v. 1.0

PASS

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

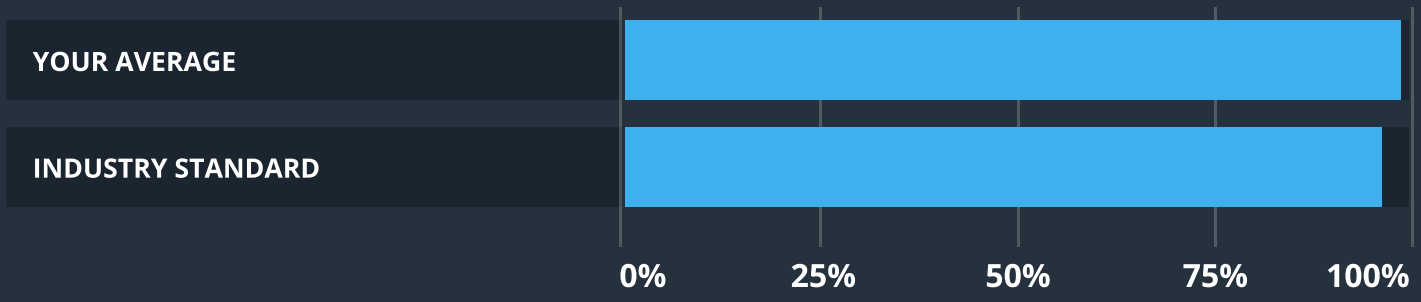
TECHNICAL SUMMARY

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

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

Contract Status

Testable Code



The testable code is 98%, 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 SpiritSwap 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

The Smart contract's source code was taken from the SpiritSwap repository:
https://github.com/acrylicfiddle/SpiritV2Contracts

Initial commit: f611bd5f8c1c7d5332d248ef1a85492f46072464
Last reviewed commit: 69032df6673779ff6e1790badaad192d1571d0d7

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

- ReentrancyGuard.sol
- Bribe.sol
- BribeFactory.sol
- Gauge.sol
- ProtocolGovernance.sol
- MasterDill.sol
- GaugeProxy.sol
- VotingFeeNotifier.sol
- AdminGaugeProxy.sol
- Owned.sol

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 SpiritSwap 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. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

1	Due diligence in assessing the overall code quality of the codebase.	3	Testing contract logic against common and uncommon attack vectors.
2	Cross-comparison with other, similar smart contracts by industry leaders.	4	Thorough, manual review of the codebase, line-by-line.

EXECUTIVE SUMMARY

SpiritSwap contracts set represent proxy contracts for staking and voting for the governance tokens together with a custom rewards system. The codebase is well-orgranized and is built around several classical solutions.

Auditors has founds several high-risk issues around the unclear functionality for the funds withdrawing and overall funds flow. Nevertheless, all issues were successfully resolved or verified.

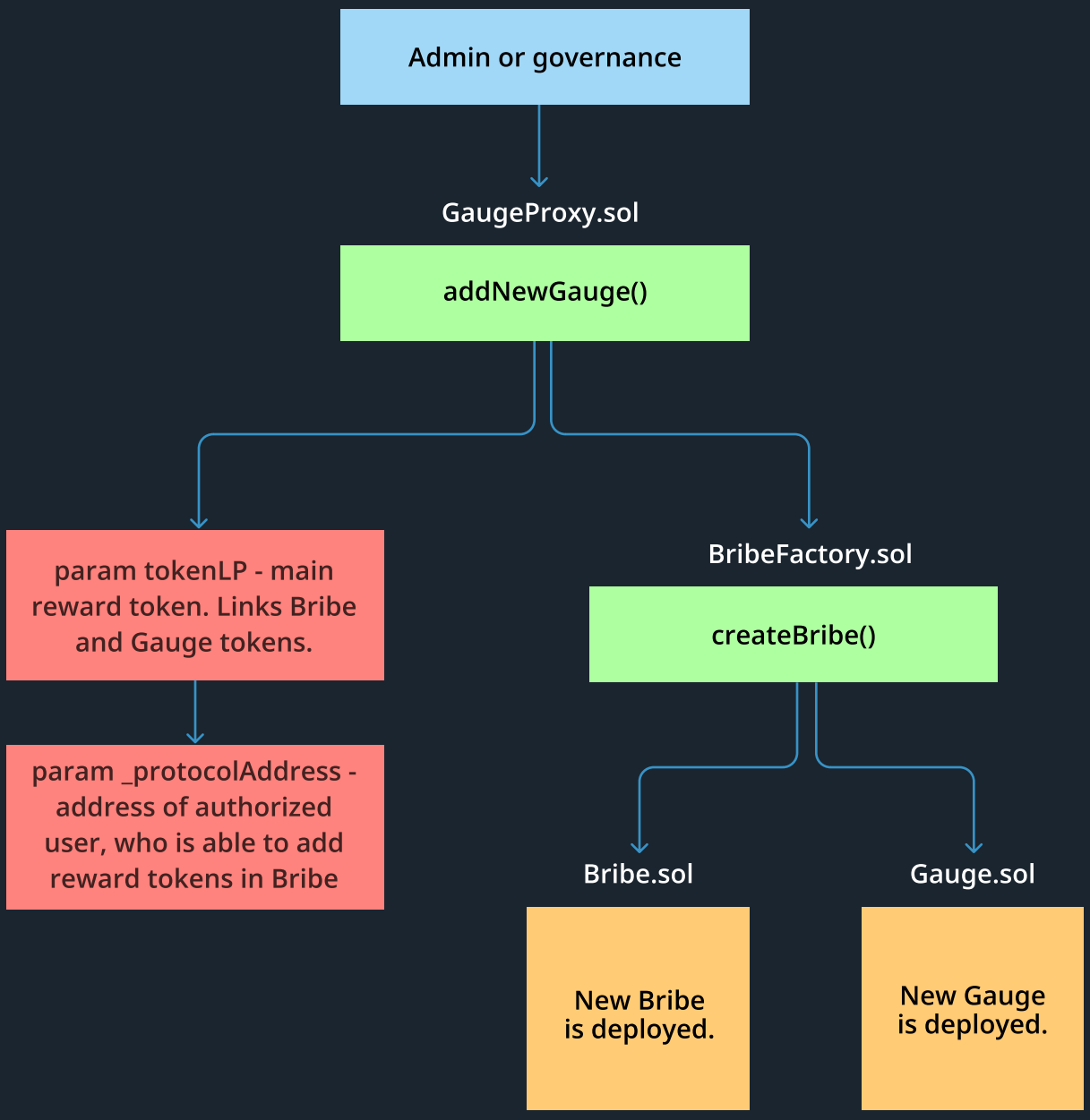
Auditors have verified native test-coverage in order to check its correctness. Also, auditor's team has provided own set of tests and testing scenarios to verify that contracts correspond to the security standard.

Auditors have reviewed the removal of the obsolete SafeMath usage and verified the correctness of the contracts with the regressional tests run.

PROTOCOL OVERVIEW

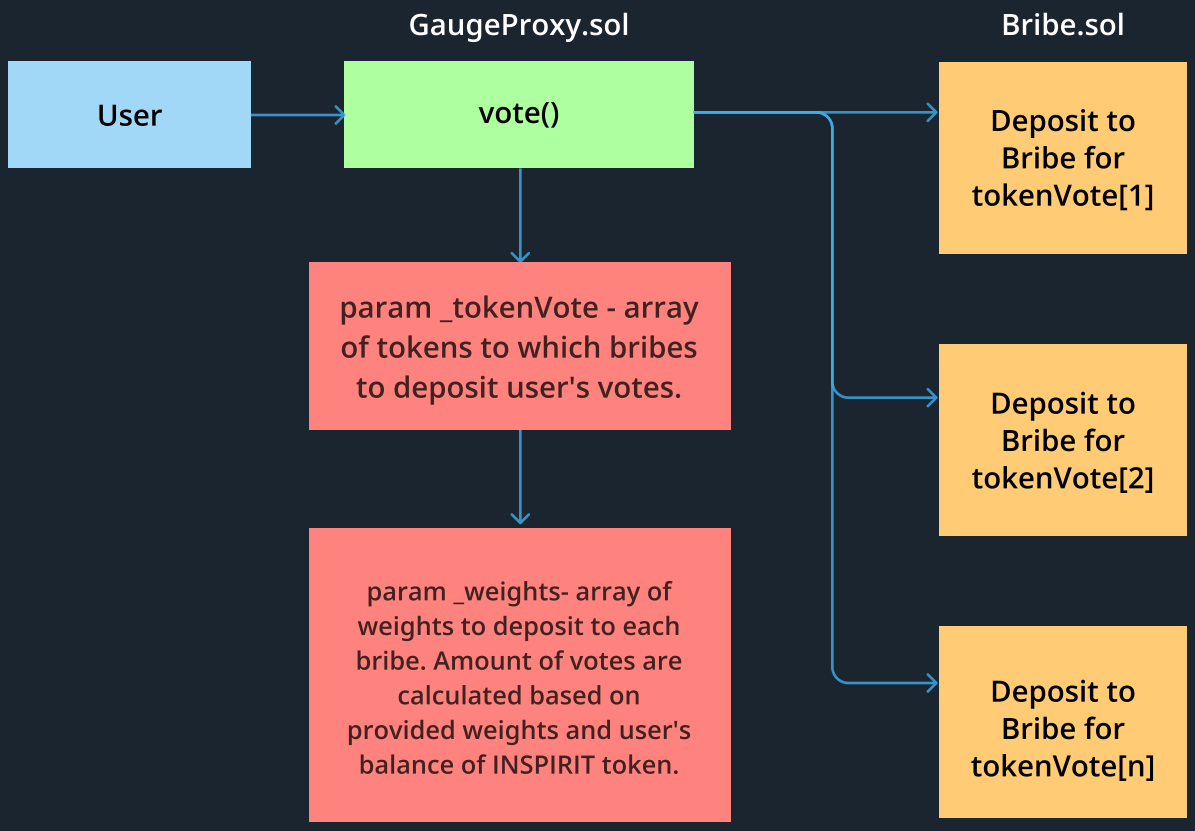
1 CREATING NEW BRIBE

Admin is able to create bribe and gauge through GaugeProxy.
Bribe is staking for votes, where users can earn rewards for voting in GaugeProxy
Gauge is staking for tokenLP, where users can stake tokenLP and earn SPIRIT token



2. VOTING IN GAUGEPROXY

Anyone with INSPIRIT balance can vote and stake their votes in Bribes. User specify weight(part of INSPIRIT balance that he has), corresponding amount of votes is generated and being staked to every Bribe contract for specified voteToken.



3. DISTRIBUTION OF REWARDS IN GAUGEPROXY.SOL

After users have voted(See previous block), anyone is able to call distribute(). Based on how much a Bribe has collected votes, corresponding Gauge receives SPIRIT tokens as a reward to be distributed in this Gauge.

4. DISTRIBUTION OF REWARDS IN ADMINGAUGEPROXY.SOL

In AdminGaugeProxy, an admin makes decision on the weights for gauges and how much rewards to be distributed to each of them.

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

HIGH | RESOLVED

Owner is able to withdraw reward tokens.

Bribes.sol: function recoverERC20().
Owner is able to withdraw any tokens from the contract, including reward tokens, which might lead to errors in reward distribution. In case, owner is set to a DAO contract, this can potentially be used by malicious actors.

Recommendation:
Validate that reward tokens cannot be withdrawn.

HIGH | VERIFIED

depositFor() transfers tokens from target account instead of msg.sender.

Gauge.sol: function depositFor().
Function is supposed to stake the amount of tokens for “account” by sender of transaction, however in function _deposit() funds are transferred from “account” instead of msg.sender which prevents function from correct execution. In case, “account” had approved tokens but never deposited, his funds will be deposited without his knowing.

Recommendation:
Transfer funds from msg.sender instead of “account”.

Post-audit.
Client assured, that they know about this functionality and that it is purposely designed to deposit the balance of account in case, account has approved his tokens.

HIGH | RESOLVED

Deprecated Eth transfer

GaugeProxy.sol: function withdrawFee().
Due to the Istanbul update there were several changes provided to the EVM, which made .transfer() and .send() methods deprecated for the ETH transfer. Thus it is highly recommended to use .call() functionality with mandatory result check, or the built-in functionality of the Address contract from OpenZeppelin library.

Recommendation:
Correct ETH sending functionality.

Post-audit.
Functionality of withdrawing ETH was removed from contract as well as ability of contract to collect ETH.

MEDIUM | RESOLVED

Unused mapping.

GaugeProxy.sol: mapping deprecated.
Values from mapping “deprecated” are never read or written.This might signalize about unfinished contract logic. Contract should not contain unused variables or unfinished logic.

Recommendation:
Either remove unused mapping or implement logic, based on it.

Post-audit.
Mapping was removed.

MEDIUM | RESOLVED

Unit-tests fail due to changes.

GaugeProxyBribeMiddleManTesting.js: before() hook.
GaugeProxyDistributeTesting.js: before() hook, it('Admin functions only, all tests to revert) test.
GaugeProxyBribeTesting.js: before() hook, it('Add a new Gauge for LP4') test.
Call to function addGauge() from GaugeProxy.sol fails due to change of parameters.

Recommendation:
Pass the correct parameters to function call.

LOW | RESOLVED

Function exit() always reverts.

Gauge.sol: function exit().
Functions exit() calls nonReentrant modifier multiple times(in functions exit(), _withdraw(), getReward()). nonReentrant is designed in such a way, that it must be called only once during a transaction otherwise it reverts. Issue is marked as low, since it is still possible to call withdraw() and getReward() separatle, however function exit() remains invalid.

Recommendation:
Call nonReentrant modifier only one time during a function execution.

Post-audit.
NonReentrant modifier was removed from function exit(), however it still remains on both functions _withdraw() and getReward(). The recommendation would be to move the logic of getReward() to private function without NonReentrant modifier and call it instead.

Post-audit.
Function was removed.

LOW | VERIFIED

Function is not restricted.

BribeFactory.sol: function createBribe().
Function createBribe() deploys a new instance of Bribe.sol, where the second parameter is the sender of the transaction. In the constructor of Bribe.sol, the second parameter is supposed to be GaugeProxy.sol, however, there is no validation that the caller of createBribe() is GaugeProxy.sol. Issue is marked as low, since it is not yet clear what impact an issue might cause the protocol and whether function is supposed to be restricted.

Recommendation:
Either confirm, that the function should not be restricted or validate, that the caller of function is verified contract GaugeProxy.sol.

Post-audit.
Client verified, that function should not be restricted.

LOW | RESOLVED

SafeMath usage can be omitted.

Contracts set utilizes Solidity 0.8.x, which has built in support of overflow and underflow warnings, thus SafeMath library can be omitted for gas savings and code simplification.

Recommendation:
Remove SafeMath library.

INFORMATIONAL | RESOLVED

Outdated Solidity version.

Currently the protocol utilizes Solidity version 0.5.17, 0.6.7, . It works well, though the general auditors security checklist recommends to utilize the newest stable version of Solidity which is 0.8.11. The newest version of Solidity includes last optimization for the compiler, last bugfixes and last features such as built-in overflow and underflow validations.

Recommendation:
Update Solidity version to the latest version 0.8.11.

INFORMATIONAL | VERIFIED

Emit event when creating a new bribe.

BribeFactory.sol: function createBribe().
In order to keep track of all created bribes, emit an event, so that an address can always be recovered.

Recommendation:
Emit event with an address of newly created bribe.

Post-audit.
Client verified that there is no need to emit an event.

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

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

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

	AdminGaugeProxy.sol
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 SpiritSwap team

As part of our work assisting SpiritSwap team in verifying the correctness of their contract code, our team has checked the complete set of unit tests prepared by the SpiritSwap team.

It needs to be mentioned, that the original code has a significant original coverage with testing scenarios provided by the SpiritSwap team. All of them were also carefully checked by the auditors' team.

GaugeProxy set by Admin System Testing

- ✓ GaugeProxy Weight Status (57ms)
- ✓ User1 deposits LP1 in all gauges (206ms)
- ✓ User1 calls distribute (89ms)
- ✓ Owner sets weight for LP1 (111ms)
- ✓ User1 calls distribute (101ms)
- ✓ Fill Gauge Proxy with SPIRIT for 10 blocks
- ✓ Owner sets weight for LP2
- ✓ User2 calls distribute (115ms)
- ✓ Forward time by 1 week then User1 claims rewards (154ms)
- ✓ Fill Gauge Proxy with SPIRIT for 10 blocks
- ✓ Owner sets weight for LP3
- ✓ User2 calls distribute (134ms)
- ✓ Forward time by 1 week then User1 claims rewards (168ms)
- ✓ Fill Gauge Proxy with SPIRIT for 10 blocks
- ✓ Owner sets weight for LP2 and LP3 (39ms)
- ✓ User2 calls distribute (130ms)
- ✓ Forward time by 1 week then User1 claims rewards (169ms)
- ✓ Fill Gauge Proxy with SPIRIT for 10 blocks
- ✓ Owner sets weight for LP3
- ✓ User2 calls distribute (118ms)
- ✓ Forward time by 1 week then User1 claims rewards (155ms)
- ✓ Fill Gauge Proxy with SPIRIT for 10 blocks
- ✓ Owner sets ve to true
- ✓ Owner sets weight for LP2 (40ms)
- ✓ User2 calls distribute (123ms)
- ✓ Forward time by 1 week then User1 claims rewards (184ms)
- ✓ Owner locks more inSPIRIT (180ms)

✓ Fill Gauge Proxy with SPIRIT for 10 blocks
✓ Owner sets weight for LP2
✓ User2 calls distribute (131ms)
✓ Forward time by 1 week then User1 claims rewards (164ms)
✓ Fill Gauge Proxy with SPIRIT for 10 blocks
✓ Owner sets weight for LP3
✓ Owner sets ve to false
✓ User2 calls distribute (129ms)
✓ Forward time by 1 week then User1 claims rewards (169ms)
✓ Fill Gauge Proxy with SPIRIT for 10 blocks
✓ Owner sets weight for LP3
✓ Owner sets ve to true
✓ User2 calls distribute (145ms)
✓ Forward time by 1 week then User1 claims rewards (164ms)
✓ Update deposit fee rate tests
✓ Set new admin addr
✓ Update FeeDistributor address
✓ Update treasury tests
✓ Admin functions only, all tests to revert (148ms)
✓ Set and accept governance changes

Fee Distributor Testing

✓ inSPIRIT Balance
✓ Owner Funds Fee Distributor with 1000 SPIRIT
✓ Forward 10 days and claim (451ms)
✓ Owner Funds Fee Distributor with 1000 SPIRIT
✓ Forward 10 days and claim (146ms)
✓ Owner Funds Fee Distributor with 1000 SPIRIT
✓ Forward 10 days and claim (67ms)
✓ Owner Funds Fee Distributor with 1000 SPIRIT
✓ Forward 10 days and claim (100ms)
✓ Owner Funds Fee Distributor with 1000 SPIRIT
✓ Owner Funds Fee Distributor with 1000 SPIRIT
✓ Forward 10 days and claim (138ms)
✓ Owner Funds Fee Distributor with 1000 SPIRIT
✓ Forward 10 days and claim
✓ Owner Funds Fee Distributor with 1000 SPIRIT
✓ Forward 10 days and claim
✓ Owner Funds Fee Distributor with 1000 SPIRIT
✓ Forward 10 days and claim (202ms)

GaugeProxy/Bribe System Testing

✓ User1 lock SPIRIT for inSPIRIT (54ms)

✓ User1 votes in GaugeProxy (102ms)
✓ User2 lock SPIRIT for inSPIRIT (55ms)
✓ User2 votes in GaugeProxy (108ms)
✓ Mint LP Voting Fees to VotingFeeNotifier (58ms)
✓ User3 calls notifyVotingFee (215ms)
✓ User2 claims bribes after 1 day (65ms)
✓ User1 claims bribes after 1 day (72ms)
✓ User1 and User2 claim bribes after 1 week (137ms)
✓ User3 calls notifyVotingFee (128ms)
✓ Mint LP Voting Fees to VotingFeeNotifier (38ms)
✓ User3 calls notifyVotingFee (213ms)
✓ Mint LP Voting Fees to VotingFeeNotifier (48ms)
✓ User2 claims bribes after 1 day (72ms)
✓ User1 claims bribes after 1 day (72ms)
✓ User3 calls notifyVotingFee (190ms)
✓ Mint LP Voting Fees to VotingFeeNotifier
✓ User3 calls notifyVotingFee (297ms)
✓ User1 and User2 claim bribes after 1 week (142ms)
✓ Mint LP Voting Fees to VotingFeeNotifier
✓ User3 calls notifyVotingFee (171ms)
✓ User1 and User2 claim bribes after 1 week (125ms)

GaugeProxy/Bribe System Testing

✓ Set new admin addr
✓ User1 lock SPIRIT for inSPIRIT (41ms)
✓ User1 votes in GaugeProxy (95ms)
✓ User2 locks SPIRIT for inSPIRIT (42ms)
✓ User2 votes in GaugeProxy (123ms)
✓ Testing Voting Weight and Bribe balances
✓ User1 revote
✓ Owner begins bribe on LP1Gauge with 100 WETH (47ms)
✓ User2 claims bribe after 1 day
✓ User1 claims bribe after 1 day (49ms)
✓ User1 and User2 claim bribe after forwarded 2 weeks (80ms)
✓ Owner begins bribe on LP1Gauge with 100 WETH (40ms)
✓ User2 claims bribe after 2 days
✓ Owner begins bribe on LP1Gauge with 100 WETH (38ms)
✓ User1 claims bribe after 2 days (38ms)
✓ User1 and User2 claim bribe after forwarded 2 weeks and transfer WETH (114ms)
✓ Owner begins bribe on LP1Gauge with 100 WETH (42ms)
✓ User1 claims bribe after 2 days
✓ User3 locks SPIRIT for inSPIRIT (52ms)
✓ User3 votes in GaugeProxy (101ms)

✓ User3 claims bribe after 2 days
✓ User2 claims bribe after 2 days (38ms)
✓ User1, User2, User3 claim bribe after forwarded 1 weeks (185ms)
✓ user1, user2, user3 reset votes (280ms)
✓ User1 extend SPIRIT for inSPIRIT
✓ User2 locks SPIRIT for inSPIRIT (63ms)
✓ User1 votes in GaugeProxy (66ms)
✓ User2 votes in GaugeProxy (69ms)
✓ Owner begins bribe on LP1Gauge with 100 WETH (83ms)
✓ Owner attempts griefing attack on bribes, user2 claims periodically, user1 claims in 2 weeks
✓ User2 claims bribe after 1 day (61ms)
✓ User2 claims bribe after 2 days (49ms)
✓ User2 claims bribe after 3 days
✓ Owner attempts griefing attack on bribes after 3 days
✓ User1, User2, User3 claim bribe after forwarded 1 weeks (164ms)
✓ Owner begins bribe on LP1Gauge with 100 WETH (81ms)
✓ User3 locks SPIRIT for inSPIRIT (75ms)
✓ User2 claims bribe after 2 days (42ms)
✓ User1 claims bribe after 3 days (39ms)
✓ User3 votes in GaugeProxy and pokes user1 and user2 (250ms)
✓ Owner begins bribe on LP1Gauge with 500 TK1 (48ms)
✓ Owner begins bribe on LP1Gauge with 100 WETH (43ms)
✓ User2 claims bribe after 5 days (58ms)
✓ Owner begins bribe on LP1Gauge with 500 TK1 (42ms)
✓ User2 claims user1s bribe after 7 days (57ms)
✓ User1, User2, User3 claim bribe after forwarded 3 weeks (290ms)
✓ Set LP1Bribe to new Bribe (116ms)
✓ Set LP1Bribe2 Status
✓ Poke user1, user2, user3 to reset their bribe balance (253ms)
✓ Owner begins bribe on LP1Gauge with 100 WETH (52ms)
✓ User2 claims bribe after 1 day (48ms)
✓ User1 claims bribe after 1 day (60ms)
✓ User1 and User2 and User3 claim bribe after forwarded 2 weeks (202ms)
✓ Owner begins bribe on LP1Gauge with 100 WETH on LP1Bribe2 (50ms)
✓ After 2 days set LP1Bribe2 to new Bribe (116ms)
✓ Recover funds from LP1Bribe2 contract
✓ Set LP1Bribe3 Status
✓ Owner begins bribe on LP1Gauge with 100 WETH (51ms)
✓ User2 claims bribe after 1 day
✓ User1 claims bribe after 1 day
✓ Poke user1, user2, user3 to reset their bribe balance (250ms)
✓ User1 and User2 and User3 claim bribe after forwarded 2 weeks (178ms)

✓ Owner begins bribe on LP1Gauge with 100 WETH (88ms)
✓ Owner attempts griefing attack on bribes, user2 claims periodically, user1 claims in 2 weeks
✓ User2 claims bribe after 1 day (60ms)
✓ User2 claims bribe after 2 days (46ms)
✓ User2 claims bribe after 3 days (45ms)
✓ Owner attempts griefing attack on bribes after 3 days (38ms)
✓ User1, User2, User3 claim bribe after forwarded 1 weeks (188ms)
✓ User4 adds a new bribe token to LP1Bribe3 (40ms)
✓ User4 begins bribe on LP1Gauge with 100 TK2 (53ms)
✓ User2 claims bribe after 1 day (55ms)
✓ User1 claims bribe after 1 day (56ms)
✓ User1 and User2 and User3 claim bribe after forwarded 2 weeks (212ms)
✓ Owner sets new bribe factory (184ms)
✓ User1 votes in GaugeProxy (103ms)
✓ User2 votes in GaugeProxy (105ms)
✓ Owner begins bribe on LP2Gauge with 100 WETH (53ms)
✓ User2 claims bribe after 1 day (38ms)
✓ User1 claims bribe after 1 day (38ms)
✓ User1 and User2 and User3 claim bribe after forwarded 2 weeks (153ms)
✓ Add a new Gauge for LP4 (150ms)
✓ User1 votes in GaugeProxy (95ms)
✓ User2 votes in GaugeProxy (103ms)
✓ Owner begins bribe on LP4Gauge with 100 WETH (63ms)
✓ User2 claims bribe after 1 day (57ms)
✓ User1 claims bribe after 1 day (54ms)
✓ User1 and User2 and User3 claim bribe after forwarded 2 weeks (201ms)

GaugeProxy/Bribe System Testing

✓ User1 lock SPIRIT for inSPIRIT (43ms)
✓ User1 votes in GaugeProxy (121ms)
✓ GaugeProxy vote status
✓ Fill Gauge Proxy with SPIRIT for 10 blocks
✓ User1 deposits LP1 in all gauges (144ms)
✓ User1 calls preDistribute (132ms)
✓ User1 calls distribute to gauges (70ms)
✓ Forward time by 1 week and claim rewards (156ms)
✓ Fill Gauge Proxy with SPIRIT for 10 blocks
✓ User1 calls preDistribute (119ms)
✓ User1 tries to call preDistribute again
✓ User2 tries to call preDistribute again
✓ User1 calls distribute to LP1 gauge (39ms)
✓ Forward time by 3 days (47ms)
✓ Forward time by 3 days (38ms)

✓ User1 calls distribute to LP1 and LP2 gauge (40ms)
✓ Forward time by 3 days
✓ User1 calls preDistribute (92ms)
✓ User1 calls distribute to LP1, LP2, LP3 gauge (75ms)
✓ Forward time by 8 days and claim rewards (165ms)
✓ User2 calls preDistribute (100ms)
✓ User1 calls distribute to LP3 gauge (41ms)
✓ Forward time by 8 days (50ms)
✓ User1 calls distribute to LP1. LP2 gauge (55ms)
✓ User2 calls preDistribute (93ms)
✓ User1 calls distribute to gauges (83ms)
✓ Forward time by 8 days and claim rewards (168ms)
✓ Turn ve33 on, SPIRIT should now be getting sent to fee distributor
✓ Fill Gauge Proxy with SPIRIT for 10 blocks
✓ User1 calls preDistribute (156ms)
✓ User1 calls distribute to gauges (86ms)
✓ User2 tries to call preDistribute again
✓ Forward time by 8 days and claim rewards (197ms)
✓ Fill Gauge Proxy with SPIRIT for 10 blocks
✓ Owner increases locked SPIRIT for inSPIRIT (160ms)
✓ User1 calls preDistribute (177ms)
✓ User1 calls distribute to LP1 gauge (39ms)
✓ User1 calls distribute to LP3 gauge after 2 days (39ms)
✓ User1 calls distribute to Gauges after 2 days (45ms)
✓ Forward time by 4 days and check status (53ms)
✓ Forward time by 4 days and check status (68ms)
✓ User1 claims LP rewards (181ms)
✓ Update Fee Distributor address
✓ Fill Gauge Proxy with SPIRIT for 10 blocks
✓ User1 calls preDistribute (161ms)
✓ User1 calls distribute to gauges (83ms)
✓ User2 tries to call preDistribute again
✓ Forward time by 8 days and claim rewards (182ms)
✓ Deprecate LP3 Gauge
✓ Fill Gauge Proxy with SPIRIT for 10 blocks
✓ User1 calls preDistribute (138ms)
✓ User1 calls distribute to gauges (64ms)
✓ User2 tries to call preDistribute again
✓ Forward time by 8 days and claim rewards (180ms)
✓ User1 votes in GaugeProxy (141ms)
✓ GaugeProxy vote status (42ms)
✓ Fill Gauge Proxy with SPIRIT for 10 blocks

✓ User1 calls preDistribute (136ms)
✓ User1 calls distribute to gauges (57ms)
✓ Forward time by 8 days and claim rewards (202ms)
✓ Resurrect LP3 Gauge
✓ User1 calls preDistribute (145ms)
✓ User1 calls distribute to gauges (61ms)
✓ Forward time by 8 days and claim rewards (207ms)
✓ User1 votes in GaugeProxy (168ms)
✓ GaugeProxy vote status (56ms)
✓ User1 calls preDistribute (292ms)
✓ User1 calls distribute to gauges (78ms)
✓ Forward time by 8 days and claim rewards (194ms)
✓ Set new admin addr
✓ Update FeeDistributor address
✓ Admin functions only, all tests to revert (67ms)
✓ Set and accept governance changes

SpiritAMMV2

✓ Test token balances (38ms)
✓ Test pair creation, lengths, reverse order, pair initialization (129ms)
✓ setAdmin change and revert
✓ setDAO changes and reverts
✓ setLocked change and reverts
✓ setBaseFeeTo tests and reverts
✓ setProtocolFeeTo tests and reverts (110ms)
✓ setVoteFeeTo tests and reverts (114ms)
✓ Check factory and weth
✓ Quote tests
✓ getAmountOut
✓ getAmountIn

SpiritMakerV2 tests

✓ Set new admin addr to auth array
✓ Set Bridge by admin/owner
✓ Convert individual tokens to Spirit

winSpiritGaugeProxy Distribute System Testing

✓ Deposit LP tokens for users so they can claim later (149ms)
✓ Distribute is called correctly (129ms)
✓ Distribute on decimal balance divisible is equally distributed (155ms)
✓ Claim rewards from all gauges (240ms)
✓ Distribute on decimal balance not divisible is distributed (151ms)
✓ Claim rewards from all gauges (235ms)
✓ Deprecate Gauge 3
✓ Distribute on 2 active gauges equally (124ms)

```
✓ Claim rewards from all gauges (236ms)
✓ Resurrect Gauge 3
✓ Turn ve33 on, SPIRIT should now be getting sent to fee distributor
✓ Distribute with ve33 tokenomics on (153ms)
✓ Owner increases locked SPIRIT for inSPIRIT (174ms)
✓ Claim rewards from all gauges (234ms)
✓ Distribute with ve33 tokenomics on (152ms)
✓ Claim rewards from all gauges (239ms)
✓ Update deposit fee rate tests
✓ Set new admin addr
✓ Update FeeDistributor address
✓ Update treasury tests
✓ Admin functions only, all tests to revert (125ms)
✓ Set and accept governance changes
286 passing (3m)
```

FILE	% STMTS	% BRANCH	% FUNCS
Owned.sol	45.45	33.33	60
ReentrancyGuard.sol	100	50	100
Bribe.sol	94.83	75	93.33
BribeFactory.sol	100	100	100
Gauge.sol	80	68.18	65
ProtocolGovernance.sol	100	75	100
MasterDill.sol	100	50	100
GaugeProxy.sol	99.21	75.93	96.15
VotingFeeNotifier.sol	100	75	100
AdminGaugeProxy.sol	100	85.29	100
All files	**91.94**	**68.77**	**91.44**

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Zokyo Security team

As part of our work assisting SpiritSwap 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 SpiritSwap contract requirements for details about issuance amounts and how the system handles these.

Contract: **AdminGaugeProxy**
- ✓ Should not add gauge if treasury is zero address (83ms)
- ✓ Should not deposit if pid not initalized
- ✓ Should not transfer SPIRIT tokens to fee distributor if reward is zero (305ms)
- ✓ Should not distribute SPIRIT tokens to gauge if weight not set (583ms)

Contract: **custom Owned**
- ✓ constructor: correct owner set (200ms)
- ✓ constructor: crash if owner address = AddressZero (112ms)
- ✓ nominateNewOwner: owner should nominate new owner (373ms)
- ✓ nominateNewOwner: crash if not owner send request (353ms)
- ✓ acceptOwnership: nominated owner should accept ownership (346ms)
- ✓ acceptOwnership: not nominated owner should not accept ownership (270ms)

Contract: **custom ReentrancyGuard**
- ✓ constructor: should create ReentrancyGuard contract (114ms)
- ✓ modifier nonReentrant: should be passed (392ms)
- ✓ modifier nonReentrant: crash if reentrant calling (187ms)

Contract: **Bribe**
- ✓ constructor: should create new bribe (217ms)
- ✓ constructor: crash if bribeFactory = AddressZero (185ms)
- ✓ constructor: crash if gaugeProxy = AddressZero (188ms)
- ✓ constructor: crash if owner = AddressZero (75ms)
- ✓ addReward: should add new reward (349ms)
- ✓ addReward: crash if sender is not owner (330ms)
- ✓ addReward: crash if reward token already exists (414ms)
- ✓ _deposit: crash if amount <= 0 (308ms)
- ✓ _deposit: crash if sender is not gaugeProxy (287ms)
- ✓ _withdraw: crash if amount <= 0 (336ms)
- ✓ _withdraw: crash if sender is not gaugeProxy (331ms)

✓ notifyRewardAmount: crash if reward <= 0 (309ms)
✓ notifyRewardAmount: crash if reward token not verified (283ms)
✓ changeAuthorizedUser: owner should authorize user (322ms)
✓ changeAuthorizedUser: not owner should not authorize user (205ms)
✓ recoverERC20: owner shold recover token for himself (268ms)

Contract: BribeFactory

✓ createBribe: should create new bribe (177ms)

Contract: Gauge

✓ Should return total supply (178ms)
✓ Should return zero derived balance if in spirit supply is zero (175ms)
✓ Should return reward for duration
✓ Should deposit all (163ms)
✓ Should deposit for (178ms)
✓ Should not let deposit 0 (81ms)
✓ Should withdraw (283ms)
✓ Should let withdraw all (297ms)
✓ Should not let withdraw 0 (240ms)
✓ Should not let withdraw more than deposited (399ms)
✓ Should revert if called not by distribution account (43ms)

Contract: GaugeProxy

✓ Only admin or governance can add gauge (46ms)
✓ Should not add gauge for already existing token
✓ Should not deprecate gauge if gauge is not active (69ms)
✓ Should not deposit to master chef if pid not initialized
✓ Should return amount of tokens
✓ Only admin or governance can set bribe factory or new bribe (152ms)
✓ Should not vote if tokens and weights length mismatch (43ms)
✓ Should not count vote if gauge doesn't exist (186ms)
✓ Should not predistribute rewards if reward amount = 0 (434ms)
✓ Should not distribute reward if end < start (50ms)
✓ Should not distribute reward end > amount of tokens (67ms)
✓ Should not distribute if locked balance is zero (53ms)

Contract: ProtocolGovernance

✓ constructor: GaugeProxy creating is correct (287ms)
✓ setGovernance: current governance should pending new governance (381ms)
✓ setGovernance: not current governance should not pending new governance (462ms)
✓ acceptGovernance: pended governance should accept governance (533ms)
✓ acceptGovernance: not pended governance should not accept governance (864ms)

Contract: VotingFeeNotifier

✓ constructor: correct initializating if gaugeProxy != AddressZero (232ms)
✓ constructor: crash if gaugeProxy == AddressZero (127ms)

Use-case test

```
✓ Should create gauge for lp token 1 (266ms)
✓ Should create gauge for lp token 2 (264ms)
✓ Should create gauge for lp token 3 (280ms)
✓ Add rewards in bribes
✓ Users vote for gauges (1831ms)
✓ Collect rewards in bribes by users (1127ms)
✓ Distribute rewards to gauges (963ms)
✓ Users deposit to gauges (1297ms)
✓ Get rewards and withdraw from gauges (2314ms)
✓ Distribute rewards to gauges again (1211ms)
70 passing (1m)
```

FILE	% STMTS	% BRANCH	% FUNCS
Owned.sol	100	100	100
ReentrancyGuard.sol	100	100	100
Bribe.sol	100	100	100
BribeFactory.sol	100	100	100
Gauge.sol	98.46	90.91	100
ProtocolGovernance.sol	100	100	100
MasterDill.sol	100	50	100
GaugeProxy.sol	100	98.15	100
VotingFeeNotifier.sol	100	100	100
AdminGaugeProxy.sol	100	94.12	100
All files	**99.84**	**93.31**	**100**

We are grateful to have been given the opportunity to work with the SpiritSwap 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 SpiritSwap team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.