



Security Assessment

# Firebird Vault

Jun 19th, 2021



# Table of Contents

## Summary

### Overview

Project Summary

Audit Summary

Vulnerability Summary

Audit Scope

### Findings

SAC-01 : Centralization Risks

SAC-02 : Lack of Event Emissions for Significant Transactions

SAL-01 : Centralization Risks

SAL-02 : Lack of Event Emissions for Significant Transactions

SSL-01 : Centralization Risks

SSL-02 : Lack of Event Emissions for Significant Transactions

VAU-01 : Lack of Event Emissions for Significant Transactions

VAU-02 : Incompatibility With Deflationary Tokens

VAU-03 : Lack of Return Value Handling

VCE-01 : Lack of Event Emissions for Significant Transactions

VCE-02 : Lack of Return Value Handling

VME-01 : Logic of Withdrawal Protection Fee

VME-02 : Lack of Event Emissions for Significant Transactions

VME-03 : Centralization Risks

### Appendix

### Disclaimer

### About

# Summary

This report has been prepared for Firebird Vault smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

# Overview

## Project Summary

Project Name	Firebird Vault
Platform	Polygon
Language	Solidity
Codebase	<a href="https://github.com/firebird-finance/firebird-vault">https://github.com/firebird-finance/firebird-vault</a>
Commit	2286c8dadff10d5f38275ddbd2756ba7722d4e6e

## Audit Summary

Delivery Date	Jun 19, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

## Vulnerability Summary

Total Issues	14
● Critical	0
● Major	3
● Medium	0
● Minor	2
● Informational	9
● Discussion	0

## Audit Scope

ID	file	SHA256 Checksum
SAC	StrategyACSV2Lp.sol	74879f749f7e6b2896fa903e72e912c771dfec8d1d48940c2c63e01deba63659
SAL	StrategyAutoLp.sol	e69b5fa4cb0439857fc902d2229584866a52c774a0161937b00eb9e71ef3c8d1
SSL	StrategySushiLp.sol	d3ea537a5672666477694acb27b52d0b7e19bb92e6a0ff9dd11c470da92ab6aa
TIM	Timelock.sol	de145a8b6db367324c306449050d3660e2994b915d8a43f8d1be24e1866dab69
VAU	Vault.sol	80b8ea0e8e2c0a8dfac8e884639c28b8a7d780be41faa1a5da797ee6e945a2e6
VCE	VaultController.sol	6c481b5b6dd8fc41a6e465bc3fee4ba1e8a427c4036ce45f7a28605aaabb7fdc
VME	VaultMaster.sol	f0241d6abf990fa1b12d1ee8aa63e85a504c94302ed6a28eb35ec41e1da62c45

There are a few depending injection contracts or addresses in the current project:

`unirouter`, `firebirdRouter`, `baseToken`, `farmingToken`, `targetCompoundToken`, `targetProfitToken`, `timelock`, `controller`, `vault` and `vaultMaster` for contract `StrategyBase`;

`acsVault`, `acsFarm`, `token0` and `token1` for contract `StrategyACSV2Lp`;

`autoFarm`, `token0` and `token1` for contract `StrategyAutoLp`;

`farmPool`, `token0` and `token1` for contract `StrategySushiLp`;

`basedToken`, `controller` and `vaultMaster` for contract `VaultBase`;

`vault` and `strategies[].strategy` for contract `VaultController`.

We assume these contracts or addresses are valid and non-vulnerable actors and implementing proper logic to collaborate with the current project.

To set up the project correctly, improve overall project quality and preserve upgradability, the following roles, are adopted in the codebase:

`governance`, is adopted to approve allowances for spenders, update configurations, and harvest rewards in contract `StrategyBase`;

`strategist`, is adopted to update configurations, harvest rewards, and earn extra yield in contract `StrategyBase`;

`strategist`, is adopted to adjust strategies in contract `StrategyACSV2Lp`, `StrategyAutoLp` and `StrategySushiLp`;

`governance`, is adopted to update configurations, add new compound, earn extra yield, and withdraw unsupported tokens in contract `VaultBase`;

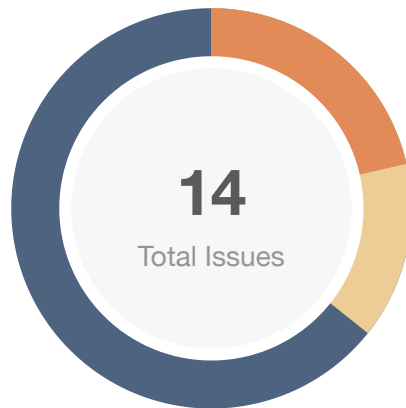
`governance`, is adopted to update configurations in contract `VaultController`;

`strategist`, is adopted to adjust strategies in contract `VaultController`;

`governance`, is adopted to update configurations, and withdraw tokens in contract `VaultMaster`.

To improve the trustworthiness of the project, dynamic runtime updates in the project should be notified to the community. Any plan to invoke the aforementioned functions should be also considered to move to the execution queue of the `Timelock` contract.

# Findings




Critical	0 (0.00%)
Major	3 (21.43%)
Medium	0 (0.00%)
Minor	2 (14.29%)
Informational	9 (64.29%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
SAC-01	Centralization Risks	Centralization / Privilege	Major	Partially Resolved
SAC-02	Lack of Event Emissions for Significant Transactions	Coding Style	Informational	Pending
SAL-01	Centralization Risks	Centralization / Privilege	Major	Partially Resolved
SAL-02	Lack of Event Emissions for Significant Transactions	Coding Style	Informational	Acknowledged
SSL-01	Centralization Risks	Centralization / Privilege	Major	Resolved
SSL-02	Lack of Event Emissions for Significant Transactions	Coding Style	Informational	Acknowledged
VAU-01	Lack of Event Emissions for Significant Transactions	Coding Style	Informational	Acknowledged
VAU-02	Incompatibility With Deflationary Tokens	Logical Issue	Minor	Acknowledged
VAU-03	Lack of Return Value Handling	Logical Issue	Informational	Acknowledged
VCE-01	Lack of Event Emissions for Significant Transactions	Coding Style	Informational	Acknowledged
VCE-02	Lack of Return Value Handling	Logical Issue	Informational	Acknowledged
VME-01	Logic of Withdrawal Protection Fee	Logical Issue	Informational	Resolved

ID	Title	Category	Severity	Status
VME-02	Lack of Event Emissions for Significant Transactions	Coding Style	● Informational	ⓘ Acknowledged
<b>VME-03</b>	Centralization Risks	<b>Centralization / Privilege</b>	● <b>Minor</b>	☑ <b>Resolved</b>



## SAC-01 | Centralization Risks

Category	Severity	Location	Status
Centralization / Privilege	● Major	StrategyACSV2Lp.sol: 1468, 1476, 1497, 1723, 1727, 1735	 Partially Resolved

### Description

The role `governance` has authority to

- grant spend allowance to any spender account by calling `StrategyBase.approveForSpender()`;
- set new `StrategyBase.unirouter`, `StrategyBase.firebirdRouter` and `StrategyBase.controller`, which are allowed to withdraw token from the contract;
- set new `StrategyBase.strategist`, which is allowed to modify strategy configurations and allowances of vaults;
- modify `StrategyBase.performanceFee`.

### Recommendation

We advise the client to handle the governance account carefully to avoid any potential hack. We also advise the client to consider the following solutions:

1. `Timelock` with reasonable latency for community awareness on privileged operations;
2. Multisig with community-voted 3rd-party independent co-signers;
3. DAO or Governance module increasing transparency and community involvement.

### Alleviation

**[Firebird Finance Team]:** This strategy does not exist at Polygon and given the current farming state at Polygon so we don't plan to use multiple strategies in a near future.

## SAC-02 | Lack of Event Emissions for Significant Transactions

Category	Severity	Location	Status
Coding Style	● Informational	StrategyACSV2Lp.sol: 1476, 1497, 1714, 1718, 1723, 1727, 1735, 1740, 1744, 1748, 1757, 2042, 2046, 2055, 2066, 2092	ⓘ Pending

### Description

Functions that affect the status of sensitive variables should emit events as notifications to users. For example,

- `StrategyBase.setUnirouter()`
- `StrategyBase.setFirebirdRouter()`
- `StrategyBase.setGovernance()`
- `StrategyBase.setTimelock()`
- `StrategyBase.setStrategist()`
- `StrategyBase.setController()`
- `StrategyBase.setPerformanceFee()`
- `StrategyBase.setFarmingToken()`
- `StrategyBase.setFarmingToken()`
- `StrategyBase.setTargetProfitToken()`
- `StrategyBase.setSlippageFactor()`
- `StrategyACSV2Lp.setBlocksToReleaseCompound()`
- `StrategyACSV2Lp.setACSFarmContract()`
- `StrategyACSV2Lp.setACSVaultContract()`
- `StrategyACSV2Lp.setTokenLp()`
- `StrategyACSV2Lp.setRateToClaimReward()`

### Recommendation

We advise the client to consider adding events for sensitive actions and emit them in the corresponding functions.

## SAL-01 | Centralization Risks

Category	Severity	Location	Status
Centralization / Privilege	● Major	StrategyAutoLp.sol: 1468, 1476, 1497, 1723, 1727, 173 5	⌚ Partially Resolved

### Description

The role `governance` has authority to

- grant spend allowance to any spender account by calling `StrategyBase.approveForSpender()`;
- set new `StrategyBase.unirouter`, `StrategyBase.firebirdRouter` and `StrategyBase.controller`, which are allowed to withdraw token from the contract;
- set new `StrategyBase.strategist`, which is allowed to modify strategy configurations and allowances of vaults;
- modify `StrategyBase.performanceFee`.

### Recommendation

We advise the client to handle the governance account carefully to avoid any potential hack. We also advise the client to consider the following solutions:

1. `TimeLock` with reasonable latency for community awareness on privileged operations;
2. Multisig with community-voted 3rd-party independent co-signers;
3. DAO or Governance module increasing transparency and community involvement.

### Alleviation

**[Firebird Finance Team]:** This strategy does not exist at Polygon and given the current farming state at Polygon so we don't plan to use multiple strategies in a near future.

## SAL-02 | Lack of Event Emissions for Significant Transactions

Category	Severity	Location	Status
Coding Style	● Informational	StrategyAutoLp.sol: 1476, 1497, 1714, 1718, 1723, 1727, 1735, 1740, 1744, 1748, 1757, 1999, 2003, 2011, 2015	ⓘ Acknowledged

### Description

Functions that affect the status of sensitive variables should emit events as notifications to users. For example,

- `StrategyBase.setUnirouter()`
- `StrategyBase.setFirebirdRouter()`
- `StrategyBase.setGovernance()`
- `StrategyBase.setTimelock()`
- `StrategyBase.setStrategist()`
- `StrategyBase.setController()`
- `StrategyBase.setPerformanceFee()`
- `StrategyBase.setFarmingToken()`
- `StrategyBase.setFarmingToken()`
- `StrategyBase.setTargetProfitToken()`
- `StrategyBase.setSlippageFactor()`
- `StrategyAutoLp.setBlocksToReleaseCompound()`
- `StrategyAutoLp.setAutoFarmContract()`
- `StrategyAutoLp.setPoolId()`
- `StrategyAutoLp.setTokenLp()`

### Recommendation

We advise the client to consider adding events for sensitive actions and emit them in the corresponding functions.

### Alleviation

N/A

## SSL-01 | Centralization Risks

Category	Severity	Location	Status
Centralization / Privilege	● Major	StrategySushiLp.sol: 1468, 1476, 1497, 1723, 1727, 1735	✓ Resolved

### Description

The role `governance` has authority to

- grant spend allowance to any spender account by calling `StrategyBase.approveForSpender()`;
- set new `StrategyBase.uniRouter`, `StrategyBase.firebirdRouter` and `StrategyBase.controller`, which are allowed to withdraw token from the contract;
- set new `StrategyBase.strategist`, which is allowed to modify strategy configurations and allowances of vaults;
- modify `StrategyBase.performanceFee`.

### Recommendation

We advise the client to handle the governance account carefully to avoid any potential hack. We also advise the client to consider the following solutions:

1. `TimeLock` with reasonable latency for community awareness on privileged operations;
2. Multisig with community-voted 3rd-party independent co-signers;
3. DAO or Governance module increasing transparency and community involvement.

### Alleviation

**[Firebird Finance Team]:** The governance role of the contract `StrategySushiLp` at Polygon has been set to the `TimeLock` in the transaction

`0xe62a14940f14ba96b5951fd13f57fe2c51047fea7b22cf327b0774cfbd6c6019`.

## SSL-02 | Lack of Event Emissions for Significant Transactions

Category	Severity	Location	Status
Coding Style	● Informational	StrategySushiLp.sol: 1476, 1497, 1714, 1718, 1723, 1727, 1735, 1740, 1744, 1748, 1757, 1992, 1996, 2004, 2008	📄 Acknowledged

### Description

Functions that affect the status of sensitive variables should emit events as notifications to users. For example,

- `StrategyBase.setUnirouter()`
- `StrategyBase.setFirebirdRouter()`
- `StrategyBase.setGovernance()`
- `StrategyBase.setTimelock()`
- `StrategyBase.setStrategist()`
- `StrategyBase.setController()`
- `StrategyBase.setPerformanceFee()`
- `StrategyBase.setFarmingToken()`
- `StrategyBase.setFarmingToken()`
- `StrategyBase.setTargetProfitToken()`
- `StrategyBase.setSlippageFactor()`
- `StrategySushiLp.setBlocksToReleaseCompound()`
- `StrategySushiLp.setAutoFarmContract()`
- `StrategySushiLp.setPoolId()`
- `StrategySushiLp.setTokenLp()`

### Recommendation

We advise the client to consider adding events for sensitive actions and emit them in the corresponding functions.

### Alleviation

N/A

## VAU-01 | Lack of Event Emissions for Significant Transactions

Category	Severity	Location	Status
Coding Style	● Informational	Vault.sol: 1009, 1013, 1017, 1021, 1025, 1041, 1063, 1067, 1072, 1076, 1080, 1084, 1088, 1252	ⓘ Acknowledged

### Description

Functions that affect the status of sensitive variables should emit events as notifications to users. For example,

- `VaultBase.setAcceptContractDepositor()`
- `VaultBase.whitelistContract()`
- `VaultBase.unwhitelistContract()`
- `VaultBase.setPauseDeposit()`
- `VaultBase.setPauseWithdraw()`
- `VaultBase.addNewCompound()`
- `VaultBase.setGovernance()`
- `VaultBase.setController()`
- `VaultBase.setConverterMap()`
- `VaultBase.setVaultMaster()`
- `VaultBase.setEarnLowerLimit()`
- `VaultBase.setCap()`
- `VaultBase.setDepositLimit()`
- `VaultBase.setOpenHarvest()`

### Recommendation

We advise the client to consider adding events for sensitive actions and emit them in the corresponding functions.

### Alleviation

N/A

## VAU-02 | Incompatibility With Deflationary Tokens

Category	Severity	Location	Status
Logical Issue	● Minor	Vault.sol: 1108~1109	ⓘ Acknowledged

### Description

While `basedToken.safeTransfer(controller, _bal)` makes a transfer with amount `_bal`, it is not guaranteed that the balance of `controller` would be greater than or equal to `_bal` because `basedToken` could be a deflationary token. In this case, `_ctrl.earn(address(basedToken), _bal)` would fail due to insufficient balance.

### Recommendation

We advise the client to only allow non-deflationary tokens to be `basedToken`, or use real balance when calling `_ctrl.earn` in `VSafeVaultBase.earn()`.

### Alleviation

N/A



## VAU-03 | Lack of Return Value Handling

Category	Severity	Location	Status
Logical Issue	● Informational	Vault.sol: 1123	ⓘ Acknowledged

### Description

The function `Converter().convert()` at the aforementioned line is not a void-returning function. Ignoring its return value might cause unexpected exceptions, especially if the callee function does not revert when the action fails.

### Recommendation

We advise the client to check return value of function `Converter().convert()` at the aforementioned line before continuing processing.

### Alleviation

N/A

## VCE-01 | Lack of Event Emissions for Significant Transactions

Category	Severity	Location	Status
Coding Style	● Informational	VaultController.sol: 729, 735, 739, 743, 747, 751, 755, 760, 765, 77, 781	ⓘ Acknowledged

### Description

Functions that affect the status of sensitive variables should emit events as notifications to users. For example,

- `VaultController.setVault()`
- `VaultController.setName()`
- `VaultController.setGovernance()`
- `VaultController.setStrategist()`
- `VaultController.approveStrategy()`
- `VaultController.revokeStrategy()`
- `VaultController.setWithdrawalFee()`
- `VaultController.setStrategyLength()`
- `VaultController.setStrategyInfo()`
- `VaultController.setInvestDisabled()`
- `VaultController.setLazySelectedBestStrategy()`

### Recommendation

We advise the client to consider adding events for sensitive actions and emit them in the corresponding functions.

### Alleviation

N/A

## VCE-02 | Lack of Return Value Handling

Category	Severity	Location	Status
Logical Issue	● Informational	VaultController.sol: 847, 855	ⓘ Acknowledged

### Description

Functions at the aforementioned lines are not void-returning functions. Ignoring the return value might cause unexpected exceptions, especially if the callee function does not revert when the actions fail.

### Recommendation

We advise the client to check return values of functions at the aforementioned lines before continuing processing.

### Alleviation

N/A

## VME-01 | Logic of Withdrawal Protection Fee

Category	Severity	Location	Status
Logical Issue	● Informational	VaultMaster.sol: 434	🕒 Resolved

### Description

The variable `withdrawalProtectionFee` comments show that the state is a percentage of withdrawal go back to the vault (for auto-compounding) to protect withdrawals. However, the remaining amount in the `withdrawalProtectionFee`, can still split and withdraw at any time to get the full remaining. What would be the purpose of designing `withdrawalProtectionFee`?

### Alleviation

**[Firebird Finance Team]:** This fee is different from the withdrawal fee that this charge user but not withdraw the amount charge from the farm contract. This will help with the farm have deposit/withdraw fee. Currently, this fee is turned off.

## VME-02 | Lack of Event Emissions for Significant Transactions

Category	Severity	Location	Status
Coding Style	● Informational	VaultMaster.sol: 453, 539, 543, 547, 552, 557, 562	📄 Acknowledged

### Description

Functions that affect the status of sensitive variables should emit events as notifications to users. For example,

- `VaultMaster.setGovernance()`
- `VaultMaster.setReserveFund()`
- `VaultMaster.setPerformanceReward()`
- `VaultMaster.setPerformanceFee()`
- `VaultMaster.setGasFee()`
- `VaultMaster.setWithdrawalProtectionFee()`
- `VaultMaster.setSlippage()`

### Recommendation

We advise the client to consider adding events for sensitive actions and emit them in the corresponding functions.

### Alleviation

N/A

## VME-03 | Centralization Risks

Category	Severity	Location	Status
Centralization / Privilege	● Minor	VaultMaster.sol: 577	🟢 Resolved

### Description

The role `governance` is allowed to withdraw tokens from contract `VaultMaster` by calling the function `VaultMaster.governanceRecoverUnsupported()`.

### Recommendation

We advise the client to handle the governance account carefully to avoid any potential hack. We also advise the client to consider the following solutions:

1. Excluding all supported tokens from being withdrawn by calling function `governanceRecoverUnsupported()`;
2. `Timelock` with reasonable latency for community awareness on privileged operations;
3. Multisig with community-voted 3rd-party independent co-signers;
4. DAO or Governance module increasing transparency and community involvement.

### Alleviation

[Firebird Finance Team]: VaultMaster is being used to set some global parameters like performance fee, reserve fund, it can't touch any underlying asset at vaults.

# Appendix

## Finding Categories

### Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

### Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how `block.timestamp` works.

### Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

## Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux `"sha256sum"` command against the target file.

# Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.



## About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

