



SMART CONTRACT SECURITY AUDIT OF

Beefy Finance



Geist - Eth Strategy

Summary - Preliminary Report

Audit Firm - Guardian Audits

Client Firm - Beefy Finance

Final Report Date - Preliminary Report

Audit Summary

After a line by line manual analysis and automated review, Guardian Audits has concluded that:

- Beefy Finance's smart contracts have a **LOW RISK SEVERITY**
- Beefy Finance's smart contracts have an **ACTIVE OWNERSHIP**
- Important owner privileges – `panic`, `pause`, `deleverageOnce`, `rebalance`, `addRewardToNativeRoute`, `removeRewardToNativeRoute`
- Beefy Finance's smart contract owner has multiple "write" privileges. Centralization risk correlated to the active ownership is **LOW**

Notice that the examined smart contracts are not resistant to internal exploit. For a detailed understanding of risk severity, source code vulnerability, and potential attack vectors, refer to the complete audit report below.

 Blockchain network: **Fantom Opera**

 Verify the authenticity of this report on Guardian's GitHub: <https://github.com/guardianaudits>

Table of Contents

Project Information

Overview 4

Audit Scope & Methodology 5

Smart Contract Risk Assessment

Contract Overview 6

Inheritance Graph 10

Findings & Resolutions 11

Report Summary

Auditor’s Verdict 20

Addendum

Disclaimer 21

About Guardian Audits 22

Project Overview

Project Summary

Project Name	Beefy Finance
Language	Solidity
Codebase	https://ftmscan.com/address/0x739c33c453f6449810b2997a320e5ea0b79fa438#code
Commit	

Audit Summary

Delivery Date	Preliminary Report
Audit Methodology	Static Analysis, Manual Review

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
● Critical	0	0	0	0	0	0
● High	0	0	0	0	0	0
● Medium	0	0	0	0	0	0
● Low	8	8	0	0	0	0

Audit Scope & Methodology

Scope

ID	File	SHA-1 Checksum
STR	StrategyGeist.sol	9616a5aa5eaa7abd6dbea2ea4d2296bb11dd5ede

Methodology

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.

Vulnerability Classifications








Vulnerability Level	Classification
● Critical	Easily exploitable by anyone, causing loss/manipulation of assets or data.
● High	Arduously exploitable by a subset of addresses, causing loss/manipulation of assets or data.
● Medium	Inherent risk of future exploits that may or may not impact the smart contract execution.
● Low	Minor deviation from best practices.

Contract Overview

Legend

	Symbol		Meaning	
	:-----:		-----	
			Function can modify state	
			Function is payable	

Contracts Description Table

	Contract		Type		Bases			
	:-----:		:-----:		:-----:		:-----:	
	L		**Function Name**		**Visibility**		**Mutability**	
	Context		Implementation					
	L _msgSender		Internal					
	L _msgData		Internal					
	Ownable		Implementation		Context			
	L <Constructor>		Internal					
	L owner		Public		!		NO !	
	L renounceOwnership		Public		!			
	L transferOwnership		Public		!			
	IERC20		Interface					
	L totalSupply		External		!		NO !	
	L balanceOf		External		!		NO !	
	L transfer		External		!			
	L allowance		External		!		NO !	
	L approve		External		!			
	L transferFrom		External		!			

```

| **Address** | Library | ||| |
| L | isContract | Internal | 🔒 | | |
| L | sendValue | Internal | 🔒 | 🔴 | |
| L | functionCall | Internal | 🔒 | 🔴 | |
| L | functionCall | Internal | 🔒 | 🔴 | |
| L | functionCallWithValue | Internal | 🔒 | 🔴 | |
| L | functionCallWithValue | Internal | 🔒 | 🔴 | |
| L | functionStaticCall | Internal | 🔒 | | |
| L | functionStaticCall | Internal | 🔒 | | |
| L | functionDelegateCall | Internal | 🔒 | 🔴 | |
| L | functionDelegateCall | Internal | 🔒 | 🔴 | |
| L | _verifyCallResult | Private | 🔒 | | |
| |||||

| **SafeERC20** | Library | ||| |
| L | safeTransfer | Internal | 🔒 | 🔴 | |
| L | safeTransferFrom | Internal | 🔒 | 🔴 | |
| L | safeApprove | Internal | 🔒 | 🔴 | |
| L | safeIncreaseAllowance | Internal | 🔒 | 🔴 | |
| L | safeDecreaseAllowance | Internal | 🔒 | 🔴 | |
| L | _callOptionalReturn | Private | 🔒 | 🔴 | |
| |||||

| **Pausable** | Implementation | Context ||| |
| L | <Constructor> | Internal | 🔒 | 🔴 | |
| L | paused | Public | ! | | NO ! |
| L | _pause | Internal | 🔒 | 🔴 | whenNotPaused |
| L | _unpause | Internal | 🔒 | 🔴 | whenPaused |
| |||||

| **IDataProvider** | Interface | ||| |
| L | getReserveTokensAddresses | External | ! | | NO ! |
| L | getUserReserveData | External | ! | | NO ! |
| |||||

| **IMultiFeeDistributor** | Interface | ||| |
| L | claimableRewards | External | ! | | NO ! |
| L | exit | External | ! | 🔴 | NO ! |
| |||||

| **IIncentivesController** | Interface | ||| |
| L | claimableReward | External | ! | | NO ! |
| L | claim | External | ! | 🔴 | NO ! |
| |||||

| **ILendingPool** | Interface | ||| |
| L | deposit | External | ! | 🔴 | NO ! |
| L | borrow | External | ! | 🔴 | NO ! |
| L | repay | External | ! | 🔴 | NO ! |
| L | withdraw | External | ! | 🔴 | NO ! |
| L | getUserAccountData | External | ! | | NO ! |

```

```

| **IUniswapRouter** | Interface | |||
| L | factory | External ! | |NO ! |
| L | WBNB | External ! | |NO ! |
| L | addLiquidity | External ! | ● |NO ! |
| L | addLiquidityBNB | External ! | $ |NO ! |
| L | removeLiquidity | External ! | ● |NO ! |
| L | removeLiquidityBNB | External ! | ● |NO ! |
| L | removeLiquidityWithPermit | External ! | ● |NO ! |
| L | removeLiquidityBNBWithPermit | External ! | ● |NO ! |
| L | removeLiquidityBNBSupportingFeeOnTransferTokens | External ! | ● |NO ! |
| L | removeLiquidityBNBWithPermitSupportingFeeOnTransferTokens | External ! | ● |NO ! |
| L | swapExactTokensForTokens | External ! | ● |NO ! |
| L | swapTokensForExactTokens | External ! | ● |NO ! |
| L | swapExactTokensForTokensSupportingFeeOnTransferTokens | External ! | ● |NO ! |
| L | swapExactBNBForTokensSupportingFeeOnTransferTokens | External ! | $ |NO ! |
| L | swapExactTokensForBNBSupportingFeeOnTransferTokens | External ! | ● |NO ! |
| L | swapExactBNBForTokens | External ! | $ |NO ! |
| L | swapTokensForExactBNB | External ! | ● |NO ! |
| L | swapExactTokensForBNB | External ! | ● |NO ! |
| L | swapBNBForExactTokens | External ! | $ |NO ! |
| L | quote | External ! | |NO ! |
| L | getAmountOut | External ! | |NO ! |
| L | getAmountIn | External ! | |NO ! |
| L | getAmountsOut | External ! | |NO ! |
| L | getAmountsIn | External ! | |NO ! |
| |||||
| **StratManager** | Implementation | Ownable, Pausable |||
| L | <Constructor> | Public ! | ● |NO ! |
| L | setKeeper | External ! | ● |onlyManager |
| L | setStrategist | External ! | ● |NO ! |
| L | setUnirouter | External ! | ● |onlyOwner |
| L | setVault | External ! | ● |onlyOwner |
| L | setBeefyFeeRecipient | External ! | ● |onlyOwner |
| L | beforeDeposit | External ! | ● |NO ! |
| |||||
| **FeeManager** | Implementation | StratManager |||
| L | setCallFee | Public ! | ● |onlyManager |
| L | setWithdrawalFee | Public ! | ● |onlyManager |
| |||||

```

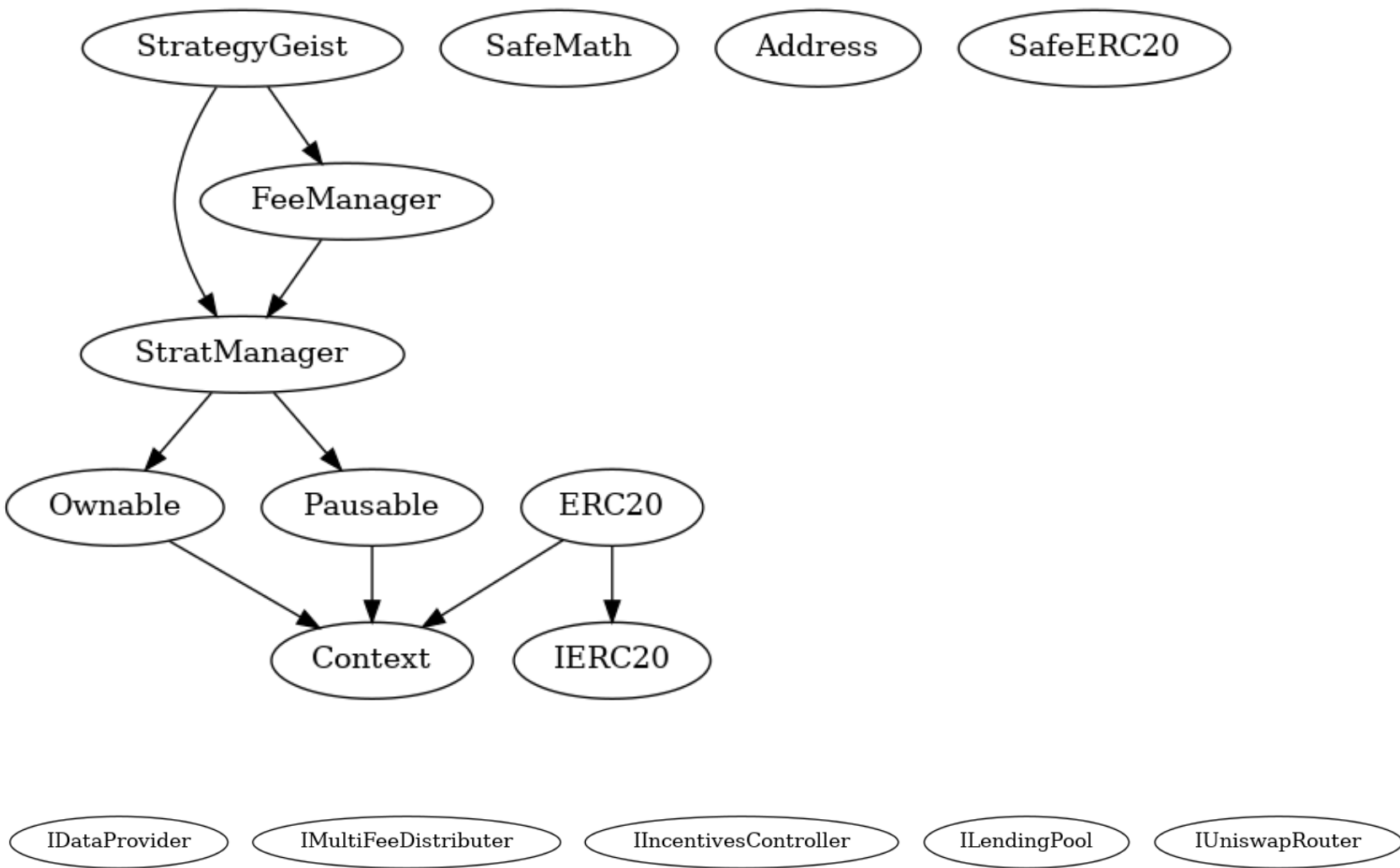


```

| **StrategyGeist** | Implementation | StratManager, FeeManager |||
| L | <Constructor> | Public ! | 🔴 | StratManager |
| L | deposit | Public ! | 🔴 | whenNotPaused |
| L | _leverage | Internal 🔒 | 🔴 | |
| L | _deleverage | Internal 🔒 | 🔴 | |
| L | deleverageOnce | External ! | 🔴 | onlyManager |
| L | rebalance | External ! | 🔴 | onlyManager |
| L | beforeDeposit | External ! | 🔴 | NO ! |
| L | harvest | External ! | 🔴 | NO ! |
| L | harvestWithCallFeeRecipient | External ! | 🔴 | NO ! |
| L | managerHarvest | External ! | 🔴 | onlyManager |
| L | _harvest | Internal 🔒 | 🔴 | whenNotPaused |
| L | chargeFees | Internal 🔒 | 🔴 | |
| L | swapRewards | Internal 🔒 | 🔴 | |
| L | withdraw | External ! | 🔴 | NO ! |
| L | availableWant | Public ! | 🔴 | NO ! |
| L | userReserves | Public ! | 🔴 | NO ! |
| L | userAccountData | Public ! | 🔴 | NO ! |
| L | balanceOf | Public ! | 🔴 | NO ! |
| L | balanceOfWant | Public ! | 🔴 | NO ! |
| L | balanceOfPool | Public ! | 🔴 | NO ! |
| L | rewardsAvailable | Public ! | 🔴 | NO ! |
| L | callReward | Public ! | 🔴 | NO ! |
| L | setHarvestOnDeposit | External ! | 🔴 | onlyManager |
| L | retireStrat | External ! | 🔴 | NO ! |
| L | panic | Public ! | 🔴 | onlyManager |
| L | pause | Public ! | 🔴 | onlyManager |
| L | unpause | External ! | 🔴 | onlyManager |
| L | _giveAllowances | Internal 🔒 | 🔴 | |
| L | _removeAllowances | Internal 🔒 | 🔴 | |
| L | addRewardToNativeRoute | External ! | 🔴 | onlyOwner |
| L | removeRewardToNativeRoute | External ! | 🔴 | onlyOwner |
| L | outputToNative | Public ! | 🔴 | NO ! |
| L | rewardToNative | Public ! | 🔴 | NO ! |
| L | nativeToWant | Public ! | 🔴 | NO ! |

```

Inheritance Graph



Findings & Resolutions

ID	Title	Category	Severity	Status
<u>STR-1</u>	Centralization Risk	Centralization / Privilege	<div><div></div>Low</div>	Unresolved
<u>STR-2</u>	Unchecked Return Value	Control Flow	<div><div></div>Low</div>	Unresolved
<u>STR-3</u>	Unnecessary code	Code Cleanliness	<div><div></div>Low</div>	Unresolved
<u>STR-4</u>	Function Visibility Modifiers	Optimization	<div><div></div>Low</div>	Unresolved
<u>STR-5</u>	Contract Inheritance Structure	Code Cleanliness	<div><div></div>Low</div>	Unresolved
<u>STR-6</u>	Redundant Require Statement	Code Cleanliness	<div><div></div>Low</div>	Unresolved
<u>STR-7</u>	Incongruent Error Strings	Code Cleanliness	<div><div></div>Low</div>	Unresolved
<u>STR-8</u>	Extraneous Function	Code Cleanliness	<div><div></div>Low</div>	Unresolved

STR-1 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● Low	Global	Unresolved

Description

There are several avenues through which the contract’s owner address can remove funds from the strategy contract to another arbitrary address.

The owner can call the setVault function with an arbitrary and possibly malicious contract address, setting the strategy’s vault to that arbitrary contract. This new arbitrary contract is now free to execute retireStrat and extract the strategy’s funds.

Additionally, the owner can call the setUnirouter function with an arbitrary and possibly malicious contract address, setting the strategy’s unirouter to that arbitrary contract. The owner can then approve the maximum allowance for any token to that arbitrary address by calling addRewardToNativeRoute, potentially leading to an extraction of the strategy’s funds.

Recommendation

Add a contract level timelock for the setUnirouter and setVault functions.

To mitigate any centralization risk, it is always recommended that controlling addresses be multi-sig and timelocked. It is known that Beefy Finance’s owner address is in fact a timelocked multi-sig, therefore this centralization risk is considered low.

Resolution

STR-2 | Unchecked Return Value

Category	Severity	Location	Status
Control Flow	● Low	StrategyGeist.sol	Unresolved

Description

Both the `transfer` and `swapExactTokensForTokens` functions have return values that should be checked, otherwise some unexpected exception may occur. It is important to have some logic in the event these executions fail.

Recommendation

Check the return values, or opt for a `safeTransfer` alternative.

Resolution

STR-3 | Unnecessary Code

Category	Severity	Location	Status
Code Cleanliness	● Low	StrategyGeist.sol	Unresolved

Description

The strategy contains 8 individual `IERC20(want).balanceOf(address(this))` calls, but this code is already abstracted in the `balanceOfWant` function.

Recommendation

Reuse the `balanceOfWant` function in these 8 spots for improved code readability and maintainability.

Resolution

STR-4 | Function Visibility Modifiers

Category	Severity	Location	Status
Optimization	● Low	StrategyGeist.sol	Unresolved

Description

The functions `callReward`, `panic`, `userAccountData`, `outputToNative`, `rewardToNative`, and `nativeToWant` are marked as `public`, but are never called from inside the contract.

Recommendation

The functions `callReward`, `panic`, `userAccountData`, `outputToNative`, `rewardToNative`, and `nativeToWant` can be marked `external` for gas optimization.

Resolution

STR-5 | Contract Inheritance Structure

Category	Severity	Location	Status
Code Cleanliness	● Low	StrategyGeist.sol	Unresolved

Description

The StrategyGeist contract inherits from the StratManager and FeeManager contracts, but notice that the FeeManager contract also inherits from the StratManager contract.

Recommendation

Alter the inheritance hierarchy so that there are no redundancies, or provide a veritable reason for such an inheritance structure.

Resolution

STR-6 | Redundant Require Statement

Category	Severity	Location	Status
Code Cleanliness	● Low	StrategyGeist.sol	Unresolved

Description

The `require(msg.sender == vault, "!vault")` require statement appears 3 separate times at the beginning of `beforeDeposit`, `withdraw`, and `retireStrat`.

Recommendation

Consider making this requires statement into a reusable modifier.

Resolution

STR-7 | Incongruent Error Strings

Category	Severity	Location	Status
Code Cleanliness	● Low	StrategyGeist.sol	Unresolved

Description

The contract contains two require statements: `require(_borrowRate <= borrowRateMax, "!safe")` on line 1596 and `require(_borrowRate <= borrowRateMax, "!rate")` on line 1616 that enforce the same restrictions, but yield different error strings.

Recommendation

Consider standardizing these error strings.

Resolution

STR-8 | Extraneous Function

Category	Severity	Location	Status
Code Cleanliness	<div><div></div>Low</div>	StrategyGeist.sol	Unresolved

Description

The implementation for the `harvest` and `managerHarvest` functions is exactly the same, the only difference being that the manager can call the `managerHarvest` function.

Recommendation

Remove the `managerHarvest` function or provide a veritable reason for its existence.

Resolution

Auditor's Verdict

After a line by line manual analysis and automated review, Guardian Audits has concluded that:

- Beefy Finance's smart contracts have a **LOW RISK SEVERITY**
- Beefy Finance's smart contracts have an **ACTIVE OWNERSHIP**
- Important owner privileges – `panic`, `pause`, `deleverageOnce`, `rebalance`, `addRewardToNativeRoute`, `removeRewardToNativeRoute`
- Beefy Finance's smart contract owner has multiple "write" privileges. Centralization risk correlated to the active ownership is **LOW**

Disclaimer

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 Guardian 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. Guardian’s position is that each company and individual are responsible for their own due diligence and continuous security. Guardian’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.

The assessment services provided by Guardian is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract’s safety and security. Therefore, Guardian does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

About Guardian Audits

Founded in 2022 by DeFi experts, Guardian Audits is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian Audits upholds best-in-class security while achieving our mission to relentlessly secure DeFi.

To learn more, visit <https://guardianaudits.com>

To view our audit portfolio, visit <https://github.com/guardianaudits>

To book an audit, message <https://t.me/guardianaudits>