Audit

Project Grizzly

smart contracts

Reference to contracts under study

https://www.dropbox.com/s/ueq7lp9b7i5xzee/grizzlyficontracts.zip?dl=0

List of contracts to be audited

- contracts/Grizzly.sol
- contracts/HoneyBNBFarm.sol
- contracts/HoneyToken.sol
- contracts/StakingPool.sol

- Contracts/Strategy/GrizzlyStrategy.sol
- contracts/Strategy/StableCoinStrategy.sol
- contracts/Strategy/StandardStrategy.sol

- contracts/Referral/Referral.sol
- contracts/DEX/DEX.sol
- Contracts/Config/BaseConfig.sol

Description of the investigated smart contracts

Grizzly.fi is a decentralized investment and crowdfunding platform embodied in several smart contracts.

GRIZZLY HONEY ($GHNY) is a utility and management token in the Grizzly.fi ecosystem. $GHNY tokens are minted when work happens on the platform and users receive rewards.
There are three investment strategies. Each strategy determines what happens to the generated rewards:
- Stablecoin
- Standart
- Grizzly

Functional purpose of smart contracts:
- Grizzly - The main contract serving as a single entry point for the investor. This contract combines several other contracts:
  - BaseConfig - this contract contains all external addresses and dependencies for the Grizzly contract. It also initializes all dependency contracts to handle tokens.
  - DEX - this contract is responsible for converting various tokens and native coins. To exchange these tokens PancakeSwap swap router is used. All exchanges are made on behalf of this contract.
  This means that all tokens are owned by that contract and then shared between different investors in strategic contracts.
  - StableCoinStrategy - this contract is used to keep track of the balances of users who have chosen the Stablecoin strategy. It is also responsible for reinvestments and issuing rewards.
  - StandardStrategy - this contract serves to account for the balances of users in LP tokens who chose the Standart strategy. It is also responsible for reinvestments and issuing rewards.
  - GrizzlyStrategy - this contract serves to account for the balances of users in LP tokens who have chosen the Grizzly strategy. It is also responsible for reinvestments and issuing rewards.

- LaunchSale - This contract is used as an initial token sale for first-time users.
    Users will place orders to buy tokens before the token is minted.
    Orders will be fulfilled once the token is deployed and initial liquidity is secured.
    This contract also allows the renewal program to provide initial liquidity by filling
    purchase orders in the same block. This ensures that all orders will be filled at the same time,
    preventing faster buyers from getting a better price.

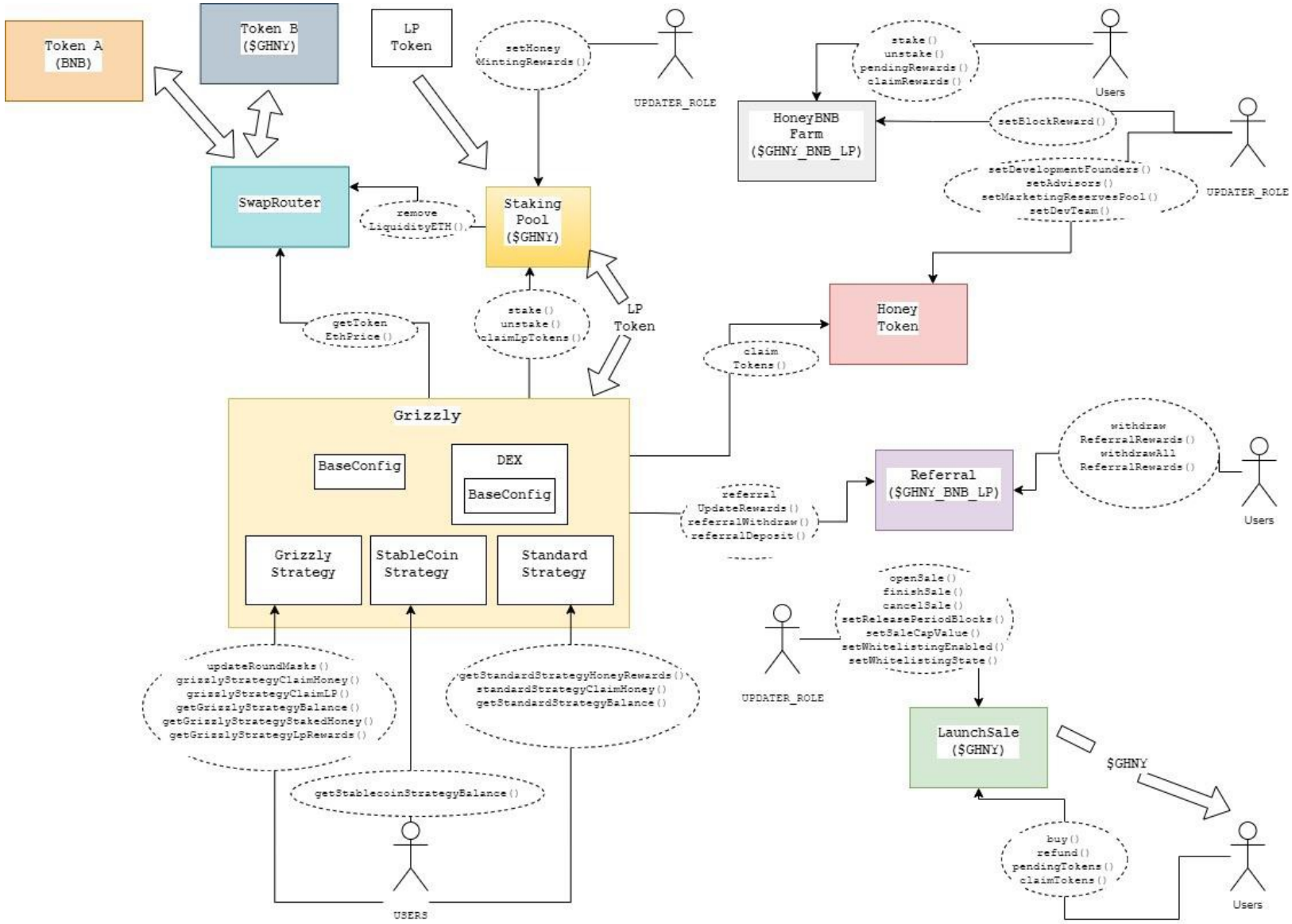
- Referral - This contract tracks referral balances and rewards.
  It uses TokenA-TokenB-LP from the referral recipient to split the rewards.

- HoneyBNBFarm - Special pool for betting Honey-BNB-LP token. Allows investors to deposit Honey-BNB-LP tokens.
   For each block, the investor receives a certain reward in $GHNY tokens.

- HoneyToken is an ERC-20 contract for the $GHNY token.

- StakingPool - A special betting pool for the $GHNY token. An LP token is given in response to a steak.
  LP tokens are pulled from liquidity pools.

Scheme of interaction of smart contracts and movement of assets



Classification of found vulnerabilities

- **CRITICAL:** Errors resulting in the theft of BNB or tokens, blocking access to funds or any other loss of BNB/tokens,
  to be transferred to or received from any party (e.g., dividends).

- **MIDDLE:** Errors that can cause a contract to fail.
  Further restoration is only possible by manually changing the state of the contract or replacing it.

- **WARNINGS:** Errors that could break the intended contract logic or expose it to DoS attacks, etc.

- **COMMENTS:** Other problems and recommendations reported for the team.

Comments Found

CRITICAL

1. Incorrect transfer of LP tokens ##### Description

In contracts/Grizzly.sol on line 540 the variable "tokenPairLpShare" is passed to the function "standardStrategyRewardLP()". "tokenPairLpShare" - reflects the number of BNBs used for further conversion to LP TokenA-TokenB pair.
The "standardStrategyRewardLP()" should pass LP tokens from the given pair, not BNB.
Thus, an invalid value of the variable is passed to the LP update function for the standard strategy.

Recommendation

Instead of "tokenPairLpShare" pass "tokenPairLpAmount" to "standardStrategyRewardLP()

2. User's loss of some tokens on Grizzly contract of unused BNB and GHNY tokens from conversion

Description

In contracts/Grizzly.sol, the functions on lines 574, 638 do not process returned values of unused tokens and BNB when creating a deposit or
liquidation of the deposit. This leads firstly to the loss of part of the user's profit, and secondly to the accumulation of BNB and Honey
on the account of the Grizzly contract. The number of these tokens is not counted in any way. There is no functionality to get them either.

Recommendation

Add logic to handle unused token and BNB balances.

3. Missing shippers' funds in LaunchSale contract

Description

In contracts/LaunchSale.sol on line 42, the "receive()" function is implemented, which allows the contract to receive BNB from other addresses directly. But there is no functionality to process them. Received BNBs are permanently blocked on the contract's balance.

Recommendation

You need to add logic to process the sender's sent funds to avoid them going missing.

4. The calculations do not handle the remainder of division of integers in the LaunchSale contract

Description

In contracts/LaunchSale.sol on lines 94-95 there is a division by "totalValueSupplied".
But in Solidity, when dividing integers, the result is rounded to a whole number. In this case, the value close to "totalValueSupplied" - the sum of all obtained BNB - is not processed.
These tokens permanently remain on the balance of the contract.

Recommendation

You need to add logic to handle the remainder of the division, to avoid missing tokens.

MIDDLE

no

WARNINGS

1. No functionality to increase the number of requests ##### Description

On lines 62-72 in contracts/Config/BaseConfig.sol
for different external contracts are distributed in the constructor of this contract.
But these values, though large, are finite. With each reference to these contracts the value will decrease.
Eventually it will be reduced to and the 0work of the contract will be blocked.

Recommendation

It is necessary to add functionality to increase the number of the user.

2. No response processing when calling

Description

According to the ERC-20 standard, the token transfer must process the response. But this is not done in this project:
- in contracts/Grizzly.sol on the lines140,245,403,: 404
- in contracts/HoneyBNBFarm.sol on the lines63,82,: 127
- in contracts/LaunchSale.sol on the lines117,148,: 171
- in contracts/StakingPool.sol on the lines82,117,141,310,: 324
- in contracts/Referral.sol on the lines176,: 208
- In contracts/Strategy/GrizzlyStrategy.sol on the line 129
- In contracts/Strategy/StandardStrategy.sol.sol on the line 108

Recommendation

You need to add functionality to handle the response when transferring tokens.

3. Unnecessary access modifier

Description

In contracts/HoneyToken.sol on line 84 for the external function "claimTokens()" the access modifier is used
"onlyRole(MINTER_ROLE)". Next, on line 88, the internal function "claimAdditionalTokens()" is called.
The "claimAdditionalTokens()" function on line 97 also has the access modifier "onlyRole(MINTER_ROLE)".
The internal "claimAdditionalTokens()" function is called only once. In this case the access modifier "onlyRole(MINTER_ROLE)" is superfluous on line 97.

Recommendation

It is necessary to remove the unnecessary access modifier.

4. Return unused variables
Description
In contracts/Grizzly.sol on line 398 "external" function "stakeRewardsForBounty()" which goes into storage, returns two variables. But calling this function will return the transaction address, but not the values of the variables.
Recommendation
You need to remove variables that are not used and add an event to commit the values of these variables.

COMMENTS

1. No logging of important events
Description
Adding an event when different events occur makes it easier to deal with complex situations and support customers.
But contracts/Strategy/StableCoinStrategy.sol does not log important events on the following lines: 24-32, 37-56.
Contracts/Strategy/StandardStrategy.sol does not log important events on the following lines: 34-46, 51-66.
Contracts/Strategy/GrizzlyStrategy.sol does not log important events on the following lines: 41-45, 50-60, 70-92.
Contracts/Referral.sol does not log important events on the following lines: 66-102, 108-126, 153-177, 202-213.
Contracts/LaunchSale.sol does not log important events on the following lines: 56-68, 71-86, 111-118, 122-125, 134-192, 196-199, 204-209, 213-218, 222-227, 232-237.
Recommendation
It is necessary to add events for these events.

2. No check for zero when initializing address variable
Description
The contract contracts/HoneyToken.sol sets the new values of the address variables.
Such eats on the following lines137,148,159,: 170.
But during initialization there is no check for new values to zero. If any of these variables has zero values, the "claimAdditionalTokens()" and "claimTokens()" functions will be blocked.
Recommendation
You need to add a zero check before initializing variables.

3. Incorrect name for variable
Description
For a better perception of the code, names should be given to variables, functions, and their arguments that best illustrate their purpose. The wrong name is often misleading to the person studying the code.
In contracts/Grizzly.sol on line 303, the function "convertPairLpToEth()" returns the number of BNB to be withdrawn, not the number of lp tokens. #####
Recommendation
In contracts/Grizzly.sol you must change the name of the variable "lpAmount" to "bnbAmount" on lines 303 and 311.

4.
Unnecessary code
Description

On the line and61 in 99contracts/DEX/DEX.sol are initialized unnecessary

variable "lpAmount", although you could just return "lpValue".
Recommendation
Remove lines 61 and 99 in contracts/DEX/DEX.sol, and replace "lpValue" with "lpAmount" on lines 49 and 89.

5. Documentation inconsistency
Description
In contracts/HoneyToken.sol on lines 97-128 the additional tokens in "claimAdditionalTokens()" are not distributed as in the project documentation,
located at: https://blog.grizzly.fi/tokenomics/.
In the contract: 5 - development founders; 3 - advisors; 2 - marketing and reserves pool; 12 - dev team.
On the site: 5,3 - strategic partners; 3,7 - advisors; 6 - marketing and reserves pool; 7 - dev team.
This can mislead users, causing distrust of the project.
Recommendation
You should update the documentation on the site or refactor the code in the contract according to the documentation.

6. Unused variable
Description
The "claimedHoneyMint" variable is defined in the ccontracts/StakingPool.sol on line 24.
But it is not used anywhere else. A variable that is not used uses extra gas when accessing the structure.
Recommendation
It is necessary to delete a variable that is not needed.

7.### #"TODO" in the code
Description
In the ccontracts/StakingPool.sol on line 321 there is a comment with "TODO" on it.
There should be no such comments in the contract to be installed in the Main Network
Recommendation
A comment with "TODO" must be deleted.

8. Unnecessary parameters in functions ##### Description
The contracts/Grizzly.sol functions "deposit()", "depositFromToken()", "withdraw()", "withdrawAll()", "withdrawToToken()", "changeStrategy()", "stakeRewardsForBounty()", "stakeRewards()" pass parameters:
"address[] memory fromToken, address[] memory toToken, uint256[] memory amountIn, uint256[] memory amountOut".
But they may differ from the actual values of the number of tokens.
That is, these parameters can be "detached from reality" because they are not related to real values.
It makes more sense to use the "checkSlippage()" function with real values.
Recommendation
It is recommended to delete the use of these parameters.

Conclusion

Grizzly.fi is an interesting multifunctional DeFi project.
But the smart contracts submitted for audit contain serious problems and need to be improved.
In this form, contracts cannot be uploaded to the main network.