1. The random variable in `random_uint16()` are not random, attackers can base on the hashes of current block number to choose the perfect time to join.

It cannot be solved since everything in a smart contract is publicly visible and there are no safe random methods in solidity.

2. `receiveMoney(uint16 my_former_doggy)` can be called a lot of times by the previous owner. Even the owner has already receive the money from the trade.

It can be solved by using a mapping at both `receiveMoney(uint16 my_former_doggy)` and `buyDoggy(uint16 doggy)` to keep track of every address trading income to keep track of their balance in the contract.

3. The transfer money method at `receiveMoney(uint16 my_former_doggy)` is not safe enough, it can be attacked by fallback when an address received money, to charge more than once from the contract.

Adding a counter in the function can prevent the fallback charging attack.

4. If there is a new doggy created from breed or create which has the same 16bit sequence as the doggies which already existed, the owner of that doggie will be changed.

It can be solved by checking if there exists the same doggy after the random_unit16() returns a set of 16bit sequence while creating and breeding new doggy.

5. The contract uses a loop in `createNewDoggy()` & `random_offspring(uint16 doggy1, uint16 doggy2)` to check the latest fee paid to doggy, using for loop cause a large amount of gas fee.

The for loop at `createNewDoggy()` can be replaced by an uint highest_create_fee which was used to keep track of the highest create fee.

The for loop at `random_offspring(uint16 doggy1, uint16 doggy2)` can be replaced by a binary calculations.

6. Since `reclaimFees()`can be called by anyone, anyone can transfer all the balance to the developer's address, however there are some functions in the contract that require balance of the contract to operate, those function may fail to run.

It can be protected by using hash function, developer can store a hashes in the contract, when developer wants to claim all fees, they should provide the same unsigned integer which produce the same hashes by sha256().

Besides, this function will also transfer the income of the owner from selling their doggies, the previous owner will fail to receive money from `receiveMoney(uint16 my_former_doggy)`.

Keeping track on the total balance of the owner can solve this problem, the developer can reclaim all the fees of the contract except the owners' income.