INK MARKS

墨 痕

ink! marks audit report

2022 年 7 月 x 日

Tommy

Testing and Software Engineering

PASS Have 100% branch test coverage

PASS Cover all critical edge cases with unit tests

PASS Have extensive integration tests

PASS Code Freeze Don't deploy recently written code, especially when written under a tight deadline

Resilience

We always check for code that will mitigate risk when (not if) a contract fails. When a contract doesn’t have this, it’s often a warning sign.

PASS Are there assert checks for critical values? (e.g., individual balances total to sum)

PASS Speed Bumps

PASS Does the contract have a speed bump? (e.g., delay in withdrawing funds, like the DAO)

PASS Does the contract have a circuit breaker? (preventing critical functions in an emergency mode)

Auditing

Auditing helps catch many bugs, but shouldn’t also be seen as a magic bullet. Your system still needs to handle failure gracefully.

PASS Have code audited by (preferably) multiple external parties (in series)

PASS Allocate comfortable time after the audit to address issues

PASS All functions are `internal` except where explictly required to be `public`/`external`. [[?](https://blog.zeppelin.solutions/on-the-parity-wallet-multisig-hack-405a8c12e8f7)]

PASS There are no arithmetic overflows/underflows in math operations.

PASS Using the OpenZeppelin safe math library [[?](https://github.com/OpenZeppelin/openzeppelin-solidity/tree/master/contracts/math)].

PASS Ether or tokens cannot be accidentally sent to the address `0x0`.

PASS Conditions are checked using `require` before operations and state changes.

PASS State is being set before and performing actions.

PASS Protected from reentry attacks (A calling B calling A). [[?](https://medium.com/@gus_tavo_guim/reentrancy-attack-on-smart-contracts-how-to-identify-the-exploitable-and-an-example-of-an-attack-4470a2d8dfe4)]

PASS Properly implements the ERC20 interface [[?](https://github.com/ethereum/eips/issues/20)].

PASS Only using modifier if necessary in more than one place.

PASS All types are being explicitly set (e.g. using `uint256` instead of `uint`).

PASS All methods and loops are within the maximum allowed gas limt.

PASS There are no unnecessary initalizations in the constructor (remember, default values are set).

PASS There is complete test coverage; every smart contract method and every possible type of input is being tested.

PASS Performed fuzz testing by using random inputs.

PASS Tested all the possible different states that the contract can be in.

PASS Ether and token amounts are dealt in wei units.

PASS The crowdsale end block/timestamp comes after start block/timestamp.

PASS The crowdsale token exchange/conversion rate is properly set.

PASS The crowdsale soft/hard cap is set.

PASS The crowdsale min/max contribution allowed is set and tested.

PASS The crowdsale whitelisting functionality is tested.

PASS The crowdsale refund logic is tested.

PASS Crowdsale participants are given their proportional token amounts or are allowed to claim their contribution.

PASS The length of each stage of the crowdsale is properly configured (e.g. presale, public sale).

PASS Specified which functions are intented to be controlled by the owner only (e.g. pausing crowdsale, progressing crowdsale stage, enabling distribution of tokens, etc..).

PASS The crowdsale vesting logic is tested.

PASS The crowdsale has a fail-safe mode that when enabled by owner, restricts calls to function and enables refund functionality.

PASS The crowdsale has a fallback function in place if it makes reasonable sense.

PASS The fallback function does not accept call data or only accepts prefixed data to avoid function signature collisions.

PASS Imported libraries have been previously audited and don't contain dyanmic parts that can be swapped out in future versions which can be be used maliciously. [[?](http://swende.se/blog/Devcon1-and-contract-security.html)]

PASS Token transfer statements are wrapped in a `require`.

PASS Using `require` and `assert` properly. Only use `assert` for things that should never happen, typically used to validate state after making changes.

PASS Using `keccak256` instead of the alias `sha3`.

PASS Protected from ERC20 short address attack. [[?](https://vessenes.com/the-erc20-short-address-attack-explained/)].

PASS Protected from recursive call attacks.

PASS Arbitrary string inputs have length limits.

PASS No secret data is exposed (all data on the blockchain is public).

PASS Avoided using array where possible and using mappings instead.

PASS Does not rely on block hashes for randomness (miners have influence on this).

PASS Does not use `tx.origin` anywhere. [[?](https://vessenes.com/tx-origin-and-ethereum-oh-my/)]

PASS Array items are shifted down when an item is deleted to avoid leaving a gap.

PASS Use `revert` instead of `throw`.

PASS Functions exit immediately when conditions aren't meant.

PASS Using the latest stable version of Solidity.

PASS Prefer pattern where receipient withdrawals funds instead of contract sending funds, however not always applicable.

PASS Resolved warnings from compiler.

PASS Prevent overflow and underflow