Valory

Smart Contract Audit Report

AUDIT SUMMARY

Valory is releasing a series of tokens with added DAO functionality.

For this audit, we reviewed the project team's contracts at commit 673b09acd9441227e6bef2e7ba0a718a807b89ad on the team's private GitHub repository.

AUDIT FINDINGS

> *No findings were identified.*
> *Date: June 23rd, 2022.*

CONTRACTS OVERVIEW

> - *As the contracts are implemented with Solidity v0.8.14, they are safe from overflows/underflows outside of unchecked math.*
> - *The contracts leverage variable packing in structures and unchecked math where appropriate to optimize gas usage.*
>
> *OLAS Contract:*
> - *The maximum supply for the first 10 years after deployment is 1 billion OLAS tokens.*
> - *The maximum supply will increase after 10 years by 2% per year.*
> - *Any user may burn their own tokens to reduce the total supply.*
> - *The Minter address may mint any amount of tokens up to the maximum supply at any time.*
> - *The token supports EIP-2612 permits which allow the user to give an address approval to move the user's tokens in a gasless manner.*
> - *The owner may transfer ownership at any time.*
> - *The owner may change the Minter address at any time.*
> - *The contract complies with the ERC-20 token standard with solmate optimization.*
>
> *buOLAS Contract:*
> - *This contract defined the buOLAS token.*
> - *buOLAS tokens are non-transferrable.*
> - *Any user may create a lock for any address. Each address may only have one lock at a time.*
> - *Creating a lock will hold OLAS tokens for a specified number of steps, up to the maximum number of steps, in exchange for buOLAS tokens.*
> - *Tokens will be locked for one "step time" per step.*
> - *Users may withdraw unlocked tokens at any time.*
> - *Tokens will be unlocked proportionally as each step time passes from the initial lock time.*
> - *The owner may "revoke" a user's lock at any time.*
> - *Revoking an account will burn all tokens that have not already been released.*
> - *Users may still withdraw the released tokens, but all remaining OLAS tokens will be burned when the user withdraws.*
> - *The owner may transfer ownership at any time.*

veOLAS Contract:

- *veOLAS tokens are used to represent voting weight in a DAO system.*
- *veOLAS tokens are non-transferrable and non-delegatable.*
- *Users receive veOLAS tokens by creating a lock for their OLAS tokens where the minimum lock time is 1 week.*
- *Users' voting power is weighted where the longer tokens are locked the greater the voting power.*
- *The maximum amount of time tokens can be locked is 4 years.*
- *After tokens have been locked, their voting power will decrease over time.*
- *Users' votes are stored once per checkpoint. When casting a vote, the closest checkpoint to the proposal's block number will be used.*
- *Users may deposit additional tokens into and extend the unlock time of an existing lock.*
- *Users may withdraw their locked tokens once the unlock time has passed.*

Sale Contract

- *The owner may create a veOLAS and/or buOLAS "balance" for any number of addresses at any time.*
- *Creating a balance will store an amount of each type of token with corresponding parameters.*
- *buOLAS tokens will have a number of steps, up to 10, whereas veOLAS tokens will have a lock time between 1 and 4 years.*
- *This contract must have at least as many OLAS tokens as there are buOLAS and veOLAS tokens across all created balances.*
- *Users who have a balance may claim their balance at any time. Claiming a balance will create a lock for the user in the corresponding token contract.*
- *The owner may transfer ownership at any time.*

GovernorOLAS Contract:

- *This contract is used to manage the DAO voting and proposal functionality within the ecosystem.*
- *Any address with at least the proposal threshold number of votes may propose a transaction.*
- *After a proposal is submitted, votes may not be cast until the "voting delay" has passed.*
- *The proposal will remain open for voting for the entire "voting period".*
- *Users may vote for, vote against, or abstain from a proposal.*
- *Users also have the option to vote through the use of a signed message, allowing for a gasless vote for the user.*
- *Votes for and abstained from a proposal will count towards a quorum.*
- *A "quorum threshold" percent of the total token supply must be reached to pass a proposal.*
- *If a quorum is reached and the proposal is passed, the execution of the proposal is delegated to a TimelockController contract.*
- *Users may change the previously described quorum threshold, proposal threshold, voting delay, and voting period through a successful proposal.*

TimelockController Contract:

- *This contract is used in conjunction with the GovernorOLAS contract to implement a timelock on operations.*
- *This contract and the deployer are given the TimeLockAdmin role upon deployment.*
- *The TimeLockAdmin role serves as the admin for all other roles, allowing any address with this role to add and remove any other address from any role.*
- *We recommend the team transfer this role to the GovernorOLAS contract after deployment.*
- *An additional list of addresses may be granted the Proposer and Canceller role and an additional list of addresses may be given the Executor role.*
- *We recommend that the team not grant any addresses other than the GovernorOLAS contract the Proposer or Canceller roles.*
- *Any address with the Proposer role may schedule any arbitrary transaction with a corresponding delay; the delay must be greater than the minimum delay.*
- *Any address with the Executor role may execute a transaction, or batch of transactions, once the corresponding delay has passed.*
- *If the 0x0 address is given the Executor role, any address may execute a transaction.*
- *Any address with the Canceller role may cancel a pending transaction.*
- *The contract may be used to update its own minimum delay through a transaction after deployment.*

AUDIT RESULTS

Vulnerability Category	Notes	Result
Arbitrary Jump/Storage Write	N/A	PASS
Centralization of Control	The team intends to transfer ownership of all contracts to the Timelock contract.	PASS
Compiler Issues	N/A	PASS
Delegate Call to Untrusted Contract	N/A	PASS
Dependence on Predictable Variables	N/A	PASS
Ether/Token Theft	N/A	PASS
Flash Loans	N/A	PASS
Front Running	N/A	PASS
Improper Events	N/A	PASS
Improper Authorization Scheme	N/A	PASS
Integer Over/Underflow	N/A	PASS
Logical Issues	N/A	PASS
Oracle Issues	N/A	PASS
Outdated Compiler Version	N/A	PASS
Race Conditions	N/A	PASS
Reentrancy	N/A	PASS
Signature Issues	N/A	PASS
Unbounded Loops	N/A	PASS
Unused Code	N/A	PASS
Overall Contract Safety		PASS

CONTRACT SOURCE SUMMARY AND VISUALIZATIONS

Name	Address/Source Code	Visualized (Hover-Zoom Recommended)
OLAS	GitHub (Not yet deployed on mainnet)	Inheritance Chart. Function Graph.

buOLAS	GitHub (Not yet deployed on mainnet)	Inheritance Chart. Function Graph.
veOLAS	GitHub (Not yet deployed on mainnet)	Inheritance Chart. Function Graph.
Sale	GitHub (Not yet deployed on mainnet)	Inheritance Chart. Function Graph.
GovernorOLAS	GitHub (Not yet deployed on mainnet)	Inheritance Chart. Function Graph.
Timelock	GitHub (Not yet deployed on mainnet)	Inheritance Chart. Function Graph.

ABOUT SOLIDITY FINANCE

Solidity Finance was founded in 2020 and quickly grew to have one of the most experienced and well-equipped smart contract auditing teams in the industry. Our team has conducted 1300+ solidity smart contract audits covering all major project types and protocols, securing a total of over $10 billion U.S. dollars in on-chain value.
Our firm is well-reputed in the community and is trusted as a top smart contract auditing company for the review of solidity code, no matter how complex. Our team of experienced solidity smart contract auditors performs audits for tokens, NFTs, crowdsales, marketplaces, gambling games, financial protocols, and more!

Contact us today to get a free quote for a smart contract audit of your project!

WHAT IS A SOLIDITY AUDIT?

Typically, a smart contract audit is a comprehensive review process designed to discover logical errors, security vulnerabilities, and optimization opportunities within code. A *Solidity Audit* takes this a step further by verifying economic logic to ensure the stability of smart contracts and highlighting privileged functionality to create a report that is easy to understand for developers and community members alike.

HOW DO I INTERPRET THE FINDINGS?

Each of our Findings will be labeled with a Severity level. We always recommend the team resolve High, Medium, and Low severity findings prior to deploying the code to the mainnet. Here is a breakdown on what each Severity level means for the project:

- **High** severity indicates that the issue puts a large number of users' funds at risk and has a high probability of exploitation, or the smart contract contains serious logical issues which can prevent the code from operating as intended.
- **Medium** severity issues are those which place at least some users' funds at risk and has a medium to high probability of exploitation.
- **Low** severity issues have a relatively minor risk association; these issues have a low probability of occurring or may have a minimal impact.
- **Informational** issues pose no immediate risk, but inform the project team of opportunities for gas optimizations and following smart contract security best practices.