Brokkr

Smart Contract Audit Report

AUDIT SUMMARY

Brokkr is releasing two ERC-20 tokens with the ability to earn them in various ways.

For this audit, we reviewed the project team's contracts folder at commit 0b91fba887fb3c380db54749178e967afe08021d on the team's private GitHub repository.

AUDIT FINDINGS

> *All findings have been resolved, though some centralized aspects are present.*
> *Date: September 16th, 2022.*
> *Updated: September 20th, 2022 with changes from commit tree v1.0.0-rc1 to commit 0b91fba887fb3c380db54749178e967afe08021d.*
>
> #### Finding #1 - StakingV1 - High (Resolved)
>
> ***Description:*** *The withdraw() function does not subtract the reward generating amount from the total BRO staked when removing a withdrawal structure.*
> ***Risk/Impact:*** *The total BRO staked value will be inflated after each withdraw leading to diminishing returns over time.*
> ***Recommendation:*** *The team must subtract the reward generating amount from the total BRO staked for each withdrawal structure that is removed.*
> ***Resolution:*** *The team has implemented the above solution.*
>
> ---
>
> #### Finding #2 - Treasury - Low (Acknowledged)
>
> ***Description:*** *The _nativeTransfer() function uses the native .transfer() function to send AVAX.*
> ***Risk/Impact:*** *A recipient contract will run out of gas if they intend to do any non-trivial logic in their receive() or fallback() functions.*
> ***Recommendation:*** *The team should consider using the call() function instead of the transfer() function.*
> ***Resolution:*** *The team has acknowledged the above limitation.*

CONTRACTS OVERVIEW

> - *As the contracts are implemented with Solidity v0.8.0, they are safe from any possible overflows/underflows.*
> - *The team must exercise caution when using bonding tokens to avoid using fee-on-transfer tokens.*
>
> *BroToken Contract:*
> - *BRO tokens serve as the main token of the Brokkr platform and can be used to earn additional BRO tokens and BBRO tokens in various ways.*
> - *The total supply of the token is set to 1 billion $BRO [1,000,000,000].*
> - *The total supply is minted to the "initial holder" address upon deployment.*
> - *No mint or burn functions are present though the circulating supply can be decreased by sending tokens to the 0x..dead address.*
> - *There are no fees associated with transferring tokens.*
> - *The owner may update the token's name and symbol at any time.*
> - *The contract complies with the ERC-20 token standard.*

BBroToken Contract:

- *There is no maximum supply of $BBRO tokens.*
- *BBRO tokens may not be transferred.*
- *Any user can burn their own tokens to reduce the total supply.*
- *Any user can burn tokens on another user's behalf if an allowance has been granted.*
- *Whitelisted addresses may mint any amount of BBRO tokens to any address at any time.*
- *The owner may add and remove any address from the whitelist at any time.*
- *The owner may update the token's name and symbol at any time.*

Treasury Contract:

- *This contract is used to store whitelisted tokens and AVAX.*
- *The owner may transfer any amount of a whitelisted token to any address at any time.*
- *The owner may transfer any amount of AVAX to any address at any time.*
- *The owner may optionally provide data when transferring that will be used to perform a function call on the destination contract.*
- *The owner may add and remove any token from the whitelist at any time.*

Airdrop Contract:

- *This contract is used to distribute BRO tokens with a Merkle Tree used as verification.*
- *The owner may add a new Merkle Root at any time. Newly added Merkle Roots will create a new "stage" so that old Merkle Roots are maintained.*
- *Users may claim tokens they have been allocated at any time.*
- *The Merkle Root associated with the specified stage will be used to verify the claim amount.*
- *The BRO tokens will then be marked as claimed and transferred to the user.*
- *Sufficient BRO tokens must be provided to the contract or not all users will be able to claim their allocated tokens.*

Vesting Contract:

- *This contract is used to distribute BRO tokens over vesting schedules.*
- *The owner may add a new vesting schedule for a specified address at any time.*
- *Vesting schedules are split into a series of amounts and end timestamps.*
- *Users may claim tokens at any time. They will receive the specified amount of BRO tokens for each end timestamp that has passed and is unclaimed.*
- *Sufficient BRO tokens must be supplied to the contract or users will be unable to claim.*
- *The owner may remove a specified address' vesting schedule at any time.*

TokenDistributor Contract:

- *This contract is used to distribute BRO tokens to handlers once per epoch.*
- *Any address may trigger a distribution if at least one epoch has passed from the last distribution.*
- *Epochs are determined from the EpochManager contract.*
- *The contract will calculate the number of passed epochs and distribute each Distribution's token amount for each epoch that has passed and subsequently triggering the handler's handleDistribution() function.*
- *Sufficient BRO tokens must be provided to the contract or distributions will fail.*
- *The owner may add a new Distribution at any time, specifying the destination handler and amount of tokens per distribution.*
- *Distribution handlers must implement the supportsDistributions() function to be accepted as a valid handler.*
- *Each handler address may only have one Distribution at a time.*
- *The owner may remove a Distribution at any time.*
- *The owner may update the amount of tokens per distribution for any Distribution at any time.*
- *The owner may pause the contract, preventing distributions, at any time.*

BondingV1 Contract:

- *This contract allows users to purchase BRO tokens at a price discounted from the current market price and also functions as a Distributor for the TokenDistributor contract.*
- *The TokenDistributor address may trigger the handleDistribution() function at any time.*

- *This function will add the specified amount of BRO token to each active bond option's balance.*
- *Any address may perform a "bond" at any time.*
- *Users must provide a specified amount of the underlying token in a bond option.*
- *They will receive in return an amount of BRO tokens based on the current market price with an additional amount of BRO based on the bond option's discount.*
- *The amount of BRO the users receives must be at least the "minimum BRO payout".*
- *The current market price for each bond option is intended to be determined by the OnePoolTWAPOracle contract.*
- *Each bond option must have a sufficient balance of BRO tokens for bonding to succeed.*
- *The user's BRO tokens will be stored in the contract if the contract is in normal mode.*
- *Users may claim their tokens once the vesting period has passed from the time the user performed the bond.*
- *The vesting period is calculated in epochs which are determined from the EpochManager contract.*
- *The BRO tokens will be staked in the Staking contract as a protocol stake with the current unstaking period if the contract is in community mode.*
- *The owner may add a new bond option, with a discount between the minimum and maximum allowed discount, for any non-BRO token that does not already have a bond option.*
- *The owner may enable a disabled bond option at any time.*
- *The owner may disable a bond option at any time, as long as there is at least one remaining bond option.*
- *The owner may remove a bond option at any time, as long as there is at least one remaining bond option. Any remaining BRO token balance will be transferred to the distributor address.*
- *The owner may update the discount for a bond option to any value between the min and max discount of 1% and 99% respectively.*
- *The owner may put the contract in normal mode and set the current vesting period at any time.*
- *The owner may put the contract in community mode and set the Staking contract and unstaking period at any time.*
- *The owner may update the minimum BRO payout to any value at any time.*
- *The owner may update the TokenDistributor address at any time.*

StakingV1 Contract:

- *This contract allows users to stake their BRO tokens for rewards in additional BRO and BBRO tokens and also functions as a Distributor for the TokenDistributor contract.*
- *All user initiated actions in the contract will update the user's pending BRO and BBRO rewards.*
- *The TokenDistributor contract may trigger the handleDistribution() function at any time.*
- *The handleDistribution() function will distribute the BRO tokens received from the TokenDistributor contract proportionally amongst all stakers relative to the amount of BRO they have staked.*
- *The BRO tokens will be returned to the TokenDistributor contract if there is no BRO currently staked.*
- *Users will earn BBRO per staked epoch based on the base BBRO rewards index, the amount of staked BRO, and the unstaking period for the staked BRO.*
- *Users may claim their pending BRO and BBRO rewards at any time.*
- *Any user may stake any amount more than the "minimum BRO stake amount" of BRO tokens.*
- *Users must specify the unstaking period between the "minimum unstaking period" and "maximum unstaking period".*
- *Of the total amount of tokens staked, only a portion will be reward generating. The reward generating amount is determined from the unstaking period, the reward generating base index, and raw amount of BRO staked.*
- *The remaining tokens will not generate rewards and will be in a "locked" state in the contract.*
- *If the user already has tokens staked for the specified unstaking period the newly staked tokens will be added to the existing staked tokens.*
- *If the user does not have tokens staked for the specified unstaking period a new unstaking period will be created.*
- *Users may only have up to the "maximum unstaking periods per staker" amount of unique unstaking periods if performing a regular stake.*
- *A protocol member may perform a protocol stake which may exceed the maximum unstaking periods per staker.*
- *Users may elect to compound their rewards for a specified unstaking period at any time.*
- *Compounding will update the users pending rewards for the unstaking period and add them to the users staked amount for the unstaking period.*
- *Users may increase the unstaking period for any of their current unstaking periods, up to the maximum unstaking period.*
- *The tokens will be moved from the current unstaking period to the increased period if one already exists for the user. The tokens will be moved to a new unstaking period if the user does not already have one for the increased unstaking period.*
- *Users may unstake a specified amount of tokens from one of their unstaking periods at any time.*
- *Unstaked tokens will be moved into a "withdrawal" state.*
- *Users may only have up to the "maximum withdrawals per unstaking period" of unique withdrawals per unstaking period.*

- *Users may only unstake the total amount in the unstaking period if they were to reach the maximum number of withdrawals with the current withdrawal.*
- *BRO tokens in withdrawal status will earn "withdrawal amount reduce percentage" less rewards.*
- *Additionally, tokens in the withdrawal state will not earn additional BBRO rewards once the withdrawal's expiration time has passed.*
- *Users may withdraw tokens in the withdrawal state once the withdrawal's expiration time has passed.*
- *Tokens in withdrawal status will count towards the user's maximum unstaking periods limit until all tokens for the unstaking period have been fully withdrawn.*
- *Users will receive both the reward generating tokens and the locked tokens.*
- *The expiration time is determined by the unstaking period the tokens were originally staked in.*
- *Users may cancel any withdrawals for a specified unstaking period at any time.*
- *The reward generating amount and locked amount will be moved from withdrawal status back into the staked amount for the unstaking period.*
- *The owner may pause the contract, preventing all user initiated actions, at any time.*
- *The owner may update the TokenDistributor address at any time.*
- *The owner may add and remove an address as a protocol member at any time.*
- *The owner may set the minimum BRO stake amount, minimum and maximum staking period, max unstaking periods per staker, and max withdrawals per unstaking period to any values at any time.*
- *The owner may adjust the base percentage of staked BRO rewards that contribute to rewards at anytime.*
- *The owner may adjust the withdrawal rewards reduction for BRO and BBRO at any time.*
- *The owner may adjust the BBRO base rewards index and rewards multiplier at any time.*

EpochManager Contract:

- *This contract is used to manage the duration of an epoch.*
- *Other contracts used throughout the platform may query this contract for the current epoch duration, enforcing a standard epoch across all contracts.*
- *An epoch defaults to 1 day.*
- *The owner may update the epoch duration at any time.*

ProtocolMigrator Contract:

- *This contract is used to migrate users from a previous implementation of the protocol to the new deployment.*
- *The owner may migrate any amount of users at any time.*
- *A migration will transfer the user the specified amount of BRO tokens, mint them the specified amount of BBRO tokens, and protocol stake an amount of BRO tokens in the Staking contract.*
- *The contract must have sufficient BRO tokens to transfer and stake for all users being migrated.*

OnePoolTWAPOracle Contract:

- *This contract is used to maintain the time weighted average price (TWAP) of a liquidity pool.*
- *Any address may trigger an update to the Oracle's price.*
- *The price may be updated once per the price update interval which is set upon deployment.*
- *Any address may consult the Oracle to get the current equivalent amount of output tokens for the provided amount of input tokens.*

TwoPoolTWAPOracle Contract:

- *This contract is used to maintain the time weighted average price (TWAP) of two tokens across two different liquidity pools.*
- *The liquidity pools must share one token with the other tokens being unique.*
- *Any address may trigger an update to the Oracle's price.*
- *The price may be updated once per the price update interval which is set upon deployment.*
- *Any address may consult the Oracle to get the current equivalent amount of output tokens for the provided amount of input tokens.*

AUDIT RESULTS

Vulnerability Category	Notes	Result
Arbitrary Jump/Storage Write	N/A	PASS
Centralization of Control	• Whitelisted addresses may mint any amount of BBRO tokens. • The owner of the Vesting contract can remove any address' vesting schedule. • The owner of the TokenDistributor contract can pause distributions. • The owner of the StakingV1 contract can pause all user initiated actions including claims and withdrawals. • Several contracts rely on receiving sufficient BRO tokens to function properly.	WARNING
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/Q	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)
BroToken	GitHub (Not yet deployed on mainnet)	Inheritance Chart. Function Graph.
BBroToken	GitHub (Not yet deployed on mainnet)	Inheritance Chart. Function Graph.
Treasury	GitHub (Not yet deployed on mainnet)	Inheritance Chart. Function Graph.
Airdrop	GitHub (Not yet deployed on mainnet)	Inheritance Chart. Function Graph.
Vesting	GitHub (Not yet deployed on mainnet)	Inheritance Chart. Function Graph.
TokenDistributor	GitHub (Not yet deployed on mainnet)	Inheritance Chart. Function Graph.
BondingV1	GitHub (Not yet deployed on mainnet)	Inheritance Chart. Function Graph.
StakingV1	GitHub (Not yet deployed on mainnet)	Inheritance Chart. Function Graph.
EpochManager	GitHub (Not yet deployed on mainnet)	Inheritance Chart. Function Graph.
ProtocolMigrator	GitHub (Not yet deployed on mainnet)	Inheritance Chart. Function Graph.
OnePoolTWAPOracle	GitHub (Not yet deployed on mainnet)	Inheritance Chart. Function Graph.
TwoPoolTWAPOracle	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.

GO HOME