SOLIDITY FINANCE

# RevomonVR

## Smart Contract Audit Report

### AUDIT SUMMARY

RevomonVR is building a DeFi ecosystem to support their innovative VR game which includes two ERC-20 tokens, ERC-721 NFTs, and ways for holders to earn staking rewards.

For this audit, we reviewed RevomonVR's contracts at commit 9d0da49342a1a87b6cafebd4945612c2cfb8669d on the team's GitHub repository.

### AUDIT FINDINGS

*Please ensure trust in the team prior to investing as they have substantial control in the ecosystem.*
*Date: December 29th, 2021.*
*Updated: March 8th, 2022 to support changes from commitb85a270eb11fffae745e7faa0baac3b51cf46d17 to commit 9d0da49342a1a87b6cafebd4945612c2cfb8669d.*

#### Finding #1 - RevoStaking - High (Resolved)

***Description:*** *The owner or PoolManager can set the total reward balance of each pool to any value at any time without contributing any Revo tokens.*

```
function updatePool(uint256 _index, string memory _name, uint256 _balance, uint256 _duration, uint256 _apr, uint2
 pools[_index].poolName = _name;
 pools[_index].poolIndex = _index;
 pools[_index].startTime = block.timestamp;
 pools[_index].totalReward = _balance;
 pools[_index].duration = _duration;
 pools[_index].APR = _apr;
 pools[_index].maxRevoStaking = _maxRevoStaking;
}
```

***Risk/Impact:*** *It is possible that users' staked funds will be used to fund reward tokens in the event that the project team has not deposited enough reward tokens to fund the reward system.*
***Recommendation:*** *The project team should enforce in the code that a sufficient amount of tokens are sent each time the reward balance is increased for any pool.*
***Resolution:*** *The project team has added logic to ensure that each pool is supplied with a sufficient amount of tokens when the reward balance is increased.*

---

#### Finding #2 - LockRevoTokenContract - High (Resolved)

***Description:*** *In the LockRevoToken contract, the increaseLockAmount() function incorrectly uses a transfer() on*

*line 487, when intending to use a transferFrom() in order to transfer tokens from the msg.sender to the contract. In the current state, no tokens are moved at all, as the tokens are being transferred from the contract to the contract.*

```
function increaseLockAmount(string memory _reason, uint256 _amount)
 public override onlyOwner
 returns (bool)
{
 bytes32 reason = stringToBytes32(_reason);
 require(tokensLocked(msg.sender, _reason) > 0, NOT_LOCKED);
 token.transfer(address(this), _amount);

 locked[msg.sender][reason].amount = locked[msg.sender][reason].amount.add(_amount);

 emit Locked(msg.sender, reason, locked[msg.sender][reason].amount, locked[msg.sender][reason].validity);
 return true;
}
```

***Risk/Impact:*** *This can be used to drain locked tokens from other beneficiaries in the system by artificially increasing the amount of tokens the contract believes are locked for a user without actually contributing any tokens.*
***Recommendation:*** *The project team should ensure that the intended amount of tokens are properly being transferred from the caller to the contract.*
***Resolution:*** *The project team has changed the code to use a transferFrom() instead of a transfer().*

---

## Finding #3 - RevoStaking - Low

***Description:*** *There are two instances of multiplication occurring on the result of a division.*

```
function getUserPoolReward(uint256 _poolIndex, uint256 _stakeAmount, address _wallet) public view returns(uint256
 IRevoTierContract.Tier memory userTier = revoTier.getRealTimeTier(_wallet);
 uint256 userPercentage = getPoolPercentage(_poolIndex, userTier.index);
 uint256 reward = _stakeAmount.div(100).mul(userPercentage).div(rewardPrecision);
 return reward;
}

...

function getPoolPercentage(uint256 _poolIndex, uint256 _tierIndex) public view returns(uint256){
 uint256 APR = pools[_poolIndex].APR;//.add(revoTier.getTier(_tierIndex).stakingAPRBonus);
 return APR.mul(rewardPrecision).div(SECONDS_IN_YEAR).mul(pools[_poolIndex].duration);
}
```

***Risk/Impact:*** *In Solidity, division can lead to integer truncation, so dividing and subsequently multiplying can cause your results to lose precision, thus becoming less accurate.*
***Recommendation:*** *The project team should rearrange the operations so that multiplication is performed before division wherever possible.*
***Resolution:*** *The team has not yet addressed this issue.*

---

## Finding #4 - RevoPoolManager - Informational

***Description:*** *The RevoPoolManager file is saved as a (.ts) Typescript file when it should be a .sol (Solidity) file.*
***Recommendation:*** *The team should change the file extension to .sol so that it can be compiled and deployed correctly.*

---

## Finding #5 - RevoPoolManager - Informational

***Description:*** *Expensive looping is used when removing pool addresses which does not account for duplicates that may be in the list.*
***Risk/Impact:*** *Duplicates in the list, if any, will not be removed, and this loop will increase execution costs.*
***Recommendation:*** *The team should consider using a mapping instead of an array to track pool addresses as a more gas-efficient and less error-prone approach.*

---

### Finding #6 - RevoFarming - Informational

***Description:*** *The contract uses an outdated version of ReentrancyGuard which is very expensive in terms of gas consuption.*
***Recommendation:*** *The team should consider using the most up-to-date version that OpenZeppelin has to offer, which provides the most gas efficient implementation without compromising security. The code can be viewed here.*

---

### Finding #7 - RevomonToken - Informational

***Description:*** *The contract inherits the functionality of ERC20Capped, but this functionality is not used at all as there is no minting functionality present.*
***Recommendation:*** *We recommend removing the ERC20Capped contract to reduce contract size and in turn enjoy additional gas savings on deployment.*

---

### Finding #9 - RevoNFT - Informational

***Description:*** *In the RevoNFT contract, the local variable* `countNotEmpty` *is never used within the* `getTokensDbIdByOwnerAndCollection()` *function.*
***Recommendation:*** *We recommend removing this variable to reduce gas usage on each call to this function.*

---

### Finding #8 - RevoStaking - Informational

***Description:*** *The getUserPoolReward() function gets the user's tier data, but this is not necessary since it is not used within this function.*
***Recommendation:*** *We recommend removing this extra functionality to reduce gas usage, and additionally removing the _tierIndex parameter from the getPoolPercentage() function.*

---

### Finding #11 - RevoStaking - Informational

***Description:*** *Several functions are declared public, but are never called internally. Several state variables can never be modified, but are not declared constant.*

```
SECONDS_IN_YEAR, rewardPrecision
```

***Recommendation:*** *We recommend declaring these state variables constant for additional gas savings on each call.*

---

### Finding #10 - RevomonVR - Informational

***Description:*** *Several functions are declared public, but are never called internally.*

```
LockRevoTokenContract: lock, tokensLockedAtTime, totalBalanceOf, extendLock, increaseLockAmount, unlock, getUnloc
RevoTier: getRealTimeTier, getBatchTiers, getTier, getTiers
RevoPoolManager: getRevoStakedFromStakingPools, getLPStakedFromFarmingPools, addPoolAddress, removePoolAddress, g
RevoStaking: createPool, updateTerminated, performStake, unstake, harvest, withdrawRevo, burnEmergencyRight, getU
RevoFarming: createPool, withdraw, harvest, getUserStakes
RevoNFT: mintBatchRevo, getTokensByOwner, getTokensDbIdByOwnerAndCollection
RevoVesting: beneficiary, cliff, start, duration, released, release, getRemainingSeconds, getRemainingDays, getCu
RevoLib: getLiquidityValue, getLpTokens, calculatePercentage
```

***Recommendation:*** *We recommend declaring these functions external for additional gas savings on each call.*

## CONTRACTS OVERVIEW

*RevomonToken Contract:*

- *The total supply of the token is determined by the team and minted to the owner on deployment.*
- *There is no mint functionality present beyond deployment.*
- *Token holders can use this contract to burn their tokens, which reduces the total supply.*
- *Token holders can also transfer tokens to the 0x..dead address, which will only reduce the circulating supply; the total supply is unaffected.*
- *Users can burn tokens from other holders as long as the proper approvals have been granted.*
- *The contract implements the ERC-1363 standard functionality in order to support transfer-and-call and approve-and-call functionality; This allows for approvals or transfers and execution to occur in one call if desired.*
- *The owner can withdraw any tokens erroneously sent to the contract at any time.*
- *The contract utilities SafeMath to protect against any overflow/underflow attacks and complies with the ERC-20 standard.*

*RevomonChildTokenBSC Contract:*

- *The total supply of the token is determined by the team and minted to the owner on deployment.*
- *No mint or burn functions exist, though the circulating supply can be decreased by sending tokens to the 0x..dead address.*
- *This contract supports a Liquidity Generation Event consisting of one or many rounds as determined by the owner.*
- *Each round has a set duration, a maximum contribution amount per user, and a whitelist of addresses that are able to participate; these values are determined by the LGE Whitelister address, which is set by the owner.*
- *After the rounds are created and the Pair address is set, anyone may kick off the Liquidity Generation Event by transferring tokens to the Pair address.*
- *Only whitelisted users may participate in the Liquidity Generation Event by purchasing tokens from the Pair; there may be a different set of whitelisted users for every round.*
- *During the Liquidity Generation Event, the amount of purchased tokens are recorded to ensure it does not exceed the maximum allowed amount per user per round.*
- *The LGE Whitelister address can set the Pair address, delete the round data, modify the round duration, maximum contribution amount, and list of whitelisted users for any round at any time.*
- *The owner can set the LGE Whitelister address to any address at any time; initially, the owner is set as the LGE Whitelister.*
- *The contract utilities SafeMath to protect against any overflow/underflow attacks and complies with the ERC-20*

*standard.*

*RevoNFT Contract:*

- *The owner address, the minter address, and the RevoNFTMinter contract address can use this contract to mint any RevomonNFT to any address at any time.*
- *Each RevomonNFT belongs to a specific Collection and has its own DB ID; when minting, the DB ID must be unique for each NFT in a Collection.*
- *The owner address, minter address, and the RevoNFTMinter contract address can reassign any NFT to any Collection and can change the NFT's DB ID at any time.*
- *As there are no restrictions on what the Collection or DB ID can be set to, it would be possible to create duplicate NFTs in the system with the same data. We recommend the project team confirm this is intended functionality.*
- *Anyone can burn their own NFTs, or NFTs they are approved to use, at any time.*
- *The owner can set the base URI to any value at any time.*
- *The owner can set the minter address at any time.*
- *The owner can set the RevoNFTMinter contract address at any time.*
- *The RevoNFTMinter contract was not provided in the scope of this audit, so we are unable to provide an assessment of this contract with regards to security.*
- *As the contract is implemented with Solidity 0.8.X, it is protected against any underflow/overflow attacks.*
- *The contract complies with the ERC-721 token standard.*

*LockRevoToken Contract:*

- *This contract allows the owner to lock any amount of Revo tokens for any amount of time; the owner can name the beneficiary address who will receive the tokens once they are unlocked.*
- *The owner can extend the lock time to any specified time for any existing token lock at will.*
- *The owner can increase the amount of tokens locked in an existing token lock at any time.*
- *Once the lock time has elapsed, the owner can release the token lock, and the tokens will be transferred to the beneficiary address.*
- *The contract utilizes SafeMath to protect against any overflow/underflow attacks.*

*RevoVesting Contract:*

- *This contract is intended to be used by the project team to facilitate a token vesting structure.*
- *The Revo tokens deposited in this contract are all subject to a single vesting schedule, no matter when they were deposited.*
- *On deployment, the project team specifies the token address, the start time, the cliff time, and the vesting duration which determines the end time; the contract contains proper checks to ensure that the start time, cliff time, and the end time all occur in sequence.*
- *The owner can start releasing the tokens once the cliff time has passed.*
- *The owner can withdraw an amount of tokens from the contract proportional to the amount of time elapsed in the vesting period.*
- *The contract utilizes SafeMath to protect against any overflow/underflow attacks.*

*RevoPoolManager Contract:*

- *This contract serves as a control panel that allows anyone to interact with the RevoStaking contracts and the RevoFarming contracts.*
- *Users can stake, unstake, and harvest rewards on any contract address specified by the user, as there is no*

*validation to ensure that only approved addresses are called.*

- *The owner can add or remove Staking and Farming contract addresses to the list of pools maintained by the contract at any time so that users can view a list of active Staking and Farming contracts.*

*RevoFarming Contract:*

- *This contract allows the owner and the PoolManager contract address to create and manage a set of yield farming pools in which users can stake a single designated LP token asset via the RevoPoolManager contract in order to earn rewards.*
- *All functions in this contract are restricted for use by anyone except for the owner and the PoolManager contract.*
- *Upon creating a pool, the project team must transfer an amount of Revo tokens to the contract, which will determine the reward rate based on the reward duration specified by the project team.*
- *Current reward amounts are calculated and updated for the user before any deposit, withdraw, or harvest operations take place.*
- *After rewards are calculated, they need to be manually harvested via the RevoPoolManager contract.*
- *The owner or PoolManager can exit any user's position at any time.*
- *The owner or PoolManager can pause the staking functionality at any time.*
- *The owner or PoolManager can change the PoolManager address at any time.*
- *The owner or PoolManager can renounce or transfer ownership on behalf of the owner at any time.*
- *The owner or PoolManager can set the RevoLib contract address, the Revo token contract address, and the designated LP token address to any address at any time.*
- *The owner or PoolManager can withdraw any tokens (including the Revo tokens and designated LP tokens) from the contract at any time, but can forfeit the ability to do so.*

*RevoStaking Contract:*

- *This contract allows the owner and the PoolManager contract address to create and manage a set of staking pools in which users can stake Revo tokens via the RevoPoolManager contract in order to earn rewards in Revo tokens.*
- *All functions in this contract are restricted for use by anyone except for the owner and the PoolManager contract.*
- *Each staking pool has a it's own name, start time, duration, total reward balance, reward distribution rate, and maximum stake amount.*
- *In order to stake in a pool, the wallet address must belong to a tier within the RevoTier system.*
- *Users can only stake in each pool a single time.*
- *Upon staking, the user's total reward amount is set based on the percentage of Revo tokens the user is contributing to the pool balance.*
- *Rewards can be harvested at any time, but the user will only receive a portion of the rewards based on the amount of time the user has staked in the pool in relation to the pool's stake duration.*
- *In order to unstake, the pool's end time must have passed.*
- *Users can only unstake in each pool a single time.*
- *Upon unstaking, the staked amount and all remaining rewards are calculated and transferred to the user.*
- *The owner or PoolManager can create new pools at any time.*
- *The owner or PoolManager can set the total reward balance of each pool to any value at any time without contributing any Revo tokens. As such, it is possible that users' staked funds will be used to fund reward tokens in the event that the project team has not deposited enough reward tokens to fund the reward system. We recommend that the project team enforces in the code that enough tokens are sent each time the reward balance is increased for any pool.*
- *The owner or PoolManager can set the name, duration, total reward balance, reward distribution rate, and*

*maximum stake amount for any pool at any time.*

- *The owner or PoolManager can mark any pool as terminated at any time, halting deposits into the pool.*
- *The owner or PoolManager can change the Revo token address used as the designated stake token at any time.*
- *The owner or PoolManager can change the RevoTier contract address and the RevoLib contract address at any time.*
- *The owner or PoolManager can withdraw any tokens (including the staked Revo tokens and reward Revo tokens) from the contract at any time, but can forfeit the ability to do so.*

*RevoTier Contract:*

- *This contract is used to determine the tier that any user belongs to based on the amount of Revo tokens in their possession.*
- *Initially, there are 6 tiers supported by the platform, all with varying Revo token requirements.*
- *In order to belong to a tier, the user must be already staking Revo tokens in a RevoStaking pool, holding Revo tokens in their wallet, staking Revo-BNB LP tokens in a RevoFarming pool, or holding Revo-BNB LP tokens in their wallet.*
- *The owner can add new tier or change criteria for existing tiers at any time.*
- *The owner can set the Revo token contract address, the RevoLib contract address, the RevoPoolManager contract address, and the RevoNFT contract address to any addresses at any time.*
- *The owner can enable or disable the NFT boost in tier calculations at any time.*
- *The owner can include or exclude liquidity balances and staking balances from tier calculations at any time.*
- *The owner can set the max tier index boost value to any value at any time.*
- *The owner can set the diamond hands minimum and maximum ID values at any time.*

*RevoLib Contract:*

- *This contract is used by the RevoTier contract to retrieve the underlying amounts of Revo tokens and BNB in a given amount of LP tokens.*
- *As the value returned from this calculation is based purely on the amounts in the reserves, it is not resistant to manipulation from flash loans. The project team must exercise caution when utilizing this return value to ensure that users cannot benefit from manipulating this value. In its current state, there would be no impact to the platform in the event that these reserves were to be manipulated.*

## AUDIT RESULTS

| Vulnerability Category | Notes | Result |
| --- | --- | --- |
| Arbitrary Jump/Storage Write | N/A | PASS |
| Centralization of Control | The project team can withdraw staked tokens and reward tokens from both the RevoStaking and RevoFarming contracts, but can forfeit the right to do so. | 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/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) |
| --- | --- | --- |
| **RevomonToken** | GitHub (Not yet deployed on mainnet) | Function Graph. Inheritance Chart. |
| **RevomonChildTokenBSC** | GitHub (Not yet deployed on mainnet) | Function Graph. Inheritance Chart. |
| **RevoNFT** | GitHub (Not yet deployed on mainnet) | Function Graph. Inheritance Chart. |
| **LockRevoToken** | GitHub (Not yet deployed on mainnet) | Function Graph. Inheritance Chart. |
| **RevoVesting** | GitHub (Not yet deployed on mainnet) | Function Graph. Inheritance Chart. |
| **RevoPoolManager** | GitHub (Not yet deployed on mainnet) | Function Graph. Inheritance Chart. |
| **RevoFarming** | GitHub (Not yet deployed on mainnet) | Function Graph. Inheritance Chart. |
| **RevoStaking** | GitHub (Not yet deployed on mainnet) | Function Graph. Inheritance Chart. |
| **RevoTier** | GitHub (Not yet deployed on mainnet) | Function Graph. Inheritance Chart. |
| **RevoLib** | GitHub (Not yet deployed on mainnet) | Function Graph. Inheritance Chart. |

## *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 1000+ 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