Smart Contract Security Assessment

November 21, 2023

Project Name:

Revert Finance: Auto-Exit, Auto-Move Range

Prepared by:

HYDN

Executive Summary

Client Name: Revert Finance: Auto-Exit, Auto-Move Range

Ecosystem: Ethereum

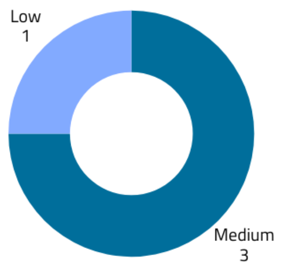
Language: Solidity

Timeline: Delivered November 21st 2023

Repository: https://github.com/revert-finance/v3utils

Method: Static Automated Analysis + Manual Review

Vulnerability Summary



4 Total Findings

3 Acknowledged

1 Resolved

Contents

Introduction

About HYDN

HYDN is an industry leader in blockchain security and smart contract audits. Founded by Warren Mercer, a world renowned cybersecurity and blockchain expert, who has previously held senior roles at NYSE, Cisco, and Alert Logic. Having been involved in cryptocurrency for over 10 years, Warren is dedicated to making the blockchain ecosystem as secure as it can be for everyone. Warren serves as the CEO for HYDN and heads up the delivery team to ensure that work is carried out to the highest standard.

The HYDN delivery team has over 10 years combined experience in blockchain, having performed smart contract audits and built security systems for a large range of protocols and companies. HYDN have performed smart contract security services for the likes of SushiSwap, Bittrex Global, Sablier, Revert Finance, Swapsicle, StakeDAO, Dancing Seahorse, Nau Finance, and more.

HYDN worked closely with Revert Finance to consider their unique business needs, objectives, and concerns. Our mission is to ensure the blockchain supports secure business operations for all and he has built the team at HYDN from the ground up to meet this very personal objective.

To keep up to date with our latest news and announcements, check out our website https://hydnsec.com/ or follow @hydnsecurity on Twitter.

About Revert Finance

Revert develops analytics and management tools for liquidity providers in AMM protocols.

Revert believes AMMs are going to become a fundamental part of financial markets in the coming years, this will create new investment opportunities for retail investors but will also require open, transparent, and accessible tools for everyone.

Methodology

When tasked with conducting a security assessment, HYDN works through multiple phases of security auditing to ensure smart contracts are audited thoroughly. To begin the process automated tests are carried out, before HYDN then moves onto carrying out a detailed manual review of the smart contracts.

HYDN uses a variety of open-source tools and analyzers as and when they are required and alongside this, HYDN primarily focuses on the following classes of security and reliability issues:

Basic Coding Mistakes

One of the most common causes of critical vulnerabilities is basic coding mistakes. Countless projects have experienced hacks and exploits due to simple, surface level mistakes that could have been flagged and addressed by a simple code review. The HYDN automated audit process which includes model checkers, fuzzers, and theorem provers analyses the smart contract for many of these basic coding mistakes. Once the automated audit has taken place, HYDN then performs a manual review of the code to gain familiarity with the contracts.

Business Logic Errors

HYDN reviews the platform or projects design documents before analysing the code to ensure that the team has a deep understanding of the business logic and goals. Following this, HYDN reviews the smart contracts to ensure that the contract logic is in line with the expected functionality.

HYDN also analyses the code for inconsistencies, flaws, or vulnerabilities which could impact business logic such as Tokenomics errors, arbitrage opportunities, and share pricing.

Complex Integration Risks

Several high-profile exploits have been the result of not any bug within the contract itself, but rather an unintended consequence of its interaction with the broader DeFi ecosystem.
We perform a meticulous review of all of the contract's possible external interactions, and summarise the associated risks; for example: flash loan attacks, oracle price manipulation, MEV/sandwich attacks, etc.

Code Maturity

HYDN reviews the smart contracts for potential improvements in the codebase. These improvements ensure that the code follows industry best practices and guidelines, or code quality standards. Alongside this, HYDN makes suggestions for code optimization items such as gas optimization, upgradeability weaknesses, centralization risks, and more.

Impact Rankings

Once HYDN has completed the security assessment, each issue is assigned an impact rating. This impact rating is based upon both the severity and likelihood of the issue occurring. Each security assessment is different and the business needs and goals, such as project timelines, and Revert Finance threat modelling. are taken into account when the impact rankings are created.

HYDN assigns the following impact rankings: Critical, High, Medium, Low (listed by severity).

Disclaimer

This HYDN security assessment does not provide any warranties on finding all possible issues within the scope and the evaluation results do not guarantee the absence of any issues. **HYDN cannot make any guarantees on any further code which is added or altered after the security assessment has taken place**. As a single security assessment can never be considered fully comprehensive, HYDN always recommends multiple independent assessments paired with a bug bounty program. This security assessment report should not be considered as financial or investment advice.

Summary of Findings

Executive Summary

As part of this audit, HYDN was tasked with performing a Smart Contract Audit on the Revert Finance v3utils smart contracts. HYDN found Three Medium vulnerability issues and One Low vulnerability issue. Following discussions between HYDN and Revert, the Revert team has acknowledged three of the findings but deemed no action needed to be taken and resolved one of the issues.

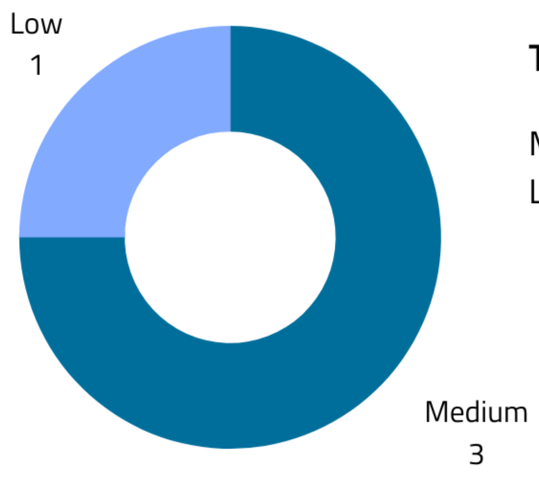
Project Summary

Platform	Ethereum
Language	Solidity
Repository	https://github.com/revert-finance/v3utils
Scope	https://github.com/revert-finance/v3utils/tree/b97e0b01e0bfa159b3d1c2fb21a37c1dd71b9309/src/automators
Initial Commit~~s~~ Hash	b97e0b01e0bfa159b3d1c2fb21a37c1dd71b9309
Latest Commit Hash	0a413f37623f6dd772f15523b0eda47b15e277f7

Audit Summary

Delivery Date	November 21st 2023

Breakdown of Findings



Total Issues - 4

Medium: 3
Low: 1

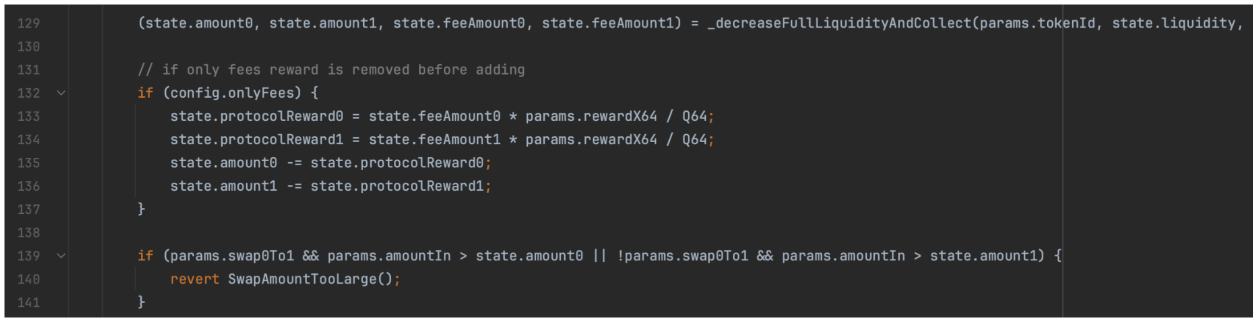
ID	Title	Severity	Status
1	User Can Bypass Operation Fees with Manual Collect	Medium	Acknowledged
2	Denial of Service - User Can Remove Position Approval/Liquidity	Medium	Acknowledged
3	Denial of Service - Updating Tick or Max Reward Config	Medium	Acknowledged
4	Gas Optimization - Move External Function Parameters to Calldata	Low	Resolved

Detailed Findings

1 User Can Bypass Operation Fees with Manual Collect [Medium]

1.1 Description

A malicious user with a valid position and configuration, can opt in for a Revert only fees plan, but as the owner still owns the position NFT, he can manually call the collection of the Uniswap fees on his own.



```
(state.amount0, state.amount1, state.feeAmount0, state.feeAmount1) = _decreaseFullLiquidityAndCollect(params.tokenId, state.liquidity,

// if only fees reward is removed before adding
if (config.onlyFees) {
    state.protocolReward0 = state.feeAmount0 * params.rewardX64 / Q64;
    state.protocolReward1 = state.feeAmount1 * params.rewardX64 / Q64;
    state.amount0 -= state.protocolReward0;
    state.amount1 -= state.protocolReward1;
}

if (params.swap0To1 && params.amountIn > state.amount0 || !params.swap0To1 && params.amountIn > state.amount1) {
    revert SwapAmountTooLarge();
}
```

Extract of AutoRange.sol

1.2 Impact

This will reduce (up to 0) the fees collected by the AutoRange/AutoExit Revert operator execution call. This would negatively affect Revert revenue, as the Uniswap fees would have been collected by the user independently, meaning that the Revert share would be reduced.

The likelihood of this occurring is low because the position owner has authorised Revert to manage the position on their behalf, meaning that the chance that they are actively monitoring their own position to collect recurrently is low. The economic rationale to take this action also does not exceed the fact that users would avoid Revert fees.

1.3 Recommendations

HYDN recommends monitoring positions with a configuration valid that tracks when fees are not collected by the Revert operator execution. The likelihood of this issue occurring is low, so taking action is not strictly recommended, but we do suggest monitoring this activity so you can be alerted if this is happening and which proportion of users are doing it/what fees Revert is missing out on.

1.4 Remediation

Revert have acknowledged this finding and are aware of the possibility of it happening but have chosen to take no action at this time. Revert understands that the user always has the option to claim the fees manually and they are fine with it, as if the user collects too much and there are not enough fees to pay for the gas costs of the next move range, then it won't be executed.

Revert is also monitoring all the transactions of the contract. HYDN accepts Revert's response.

2 Denial of Service - Remove Position Approval/Liquidity [Medium]

2.1 Description

A malicious position owner can watch and/or predict the Revert operator execution transaction call, then front-run the transaction to remove position NFT authorization to the AutoRange/AutoExit contract, or decrease liquidity by calling position manager.

2.2 Impact

The likelihood of this occurring is low as there is no economic rationale for an attacker to perform this action and either way this action would trigger a failed transaction from the Revert operator address.

2.3 Recommendations

Every chain which is using a public mempool with gas auctions can be affected by this issue. It is recommended to closely monitor failed transaction execution, and it may be safer to not automatically retry failed transactions in the first phase of the production release in order to inspect it.

2.4 Remediation

Revert have acknowledged this finding but taken no action due to the fact that they are constantly monitoring the behaviour of the bot and it is programmed to be defensive. Revert also uses private transactions wherever possible to avoid this issue. HYDN accepts Revert's response.

3 Denial of Service - Updating Tick or Max Reward Config [Medium]

3.1 Description

A malicious position owner can watch and/or predict the Revert operator execution transaction calls and then front-run the transaction to update his position configuration.

3.2 Impact

The same as in Issue 2.

3.3 Recommendations

The same as in Issue 2.

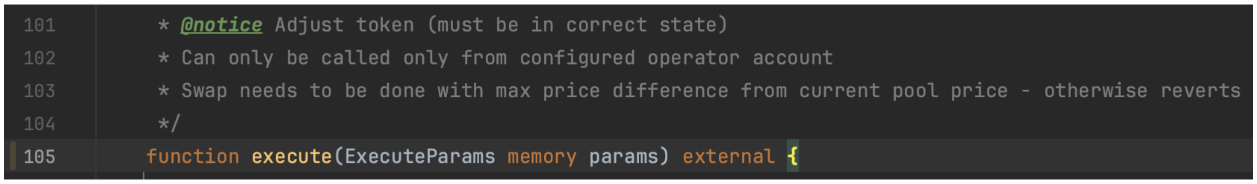
3.4 Remediation

Revert have acknowledged this finding but taken no action due to the fact that they are constantly monitoring the behaviour of the bot and it is programmed to be defensive. Revert also uses private transactions wherever possible to avoid this issue. HYDN accepts Revert's response.

4 Gas Optimization - Move External Function Parameters to Calldata [Low]

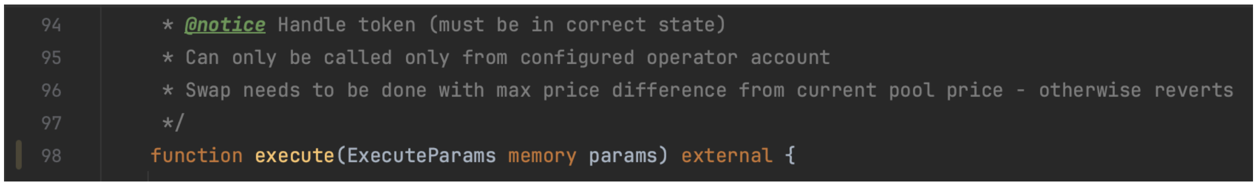
4.1 Description

A function with external visibility can be exclusively called from external contract, then its parameter storage must be *calldata* in place of *memory* in order to avoid a copy to memory.



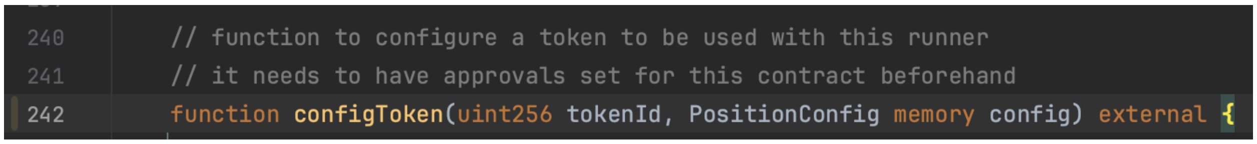
```
     * @notice Adjust token (must be in correct state)
     * Can only be called only from configured operator account
     * Swap needs to be done with max price difference from current pool price - otherwise reverts
     */
    function execute(ExecuteParams memory params) external {
```

Extract of AutoRange.sol



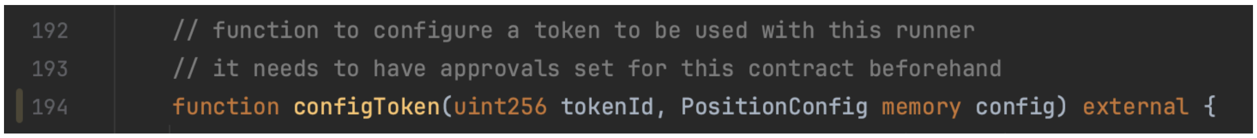
```
     * @notice Handle token (must be in correct state)
     * Can only be called only from configured operator account
     * Swap needs to be done with max price difference from current pool price - otherwise reverts
     */
    function execute(ExecuteParams memory params) external {
```

Extract of AutoExit.sol



```
    // function to configure a token to be used with this runner
    // it needs to have approvals set for this contract beforehand
    function configToken(uint256 tokenId, PositionConfig memory config) external {
```

Extract of AutoRange.sol



```
    // function to configure a token to be used with this runner
    // it needs to have approvals set for this contract beforehand
    function configToken(uint256 tokenId, PositionConfig memory config) external {
```

Extract of AutoExit.sol

4.2 Impact

The current configuration will increase deployment and execution gas costs.

4.3 Recommendations

It is recommended to move external function parameters to calldata.

4.4 Remediation

Revert has updated the code to move the recommended functions to calldata.