SOLIDIFIED

Audit Report for Paraswap - March 29, 2021

Summary

Audit Report prepared by Solidified covering a subset of the Paraswap smart contracts, which has been added to the protocol since the previous audit (https://github.com/solidified-platform/audits/blob/master/Audit%20Report%20-%20ParaSwap%20%5B20.01.2021%5D.pdf).

Process and Delivery

Three (2) independent Solidified experts performed an unbiased and isolated audit of the code below. The final debrief took place on 29 March 2021, and the results are presented here.

Audited Files

The source code has been supplied in the form of a private GitLab repository:

https://gitlab.com/paraswap/paraswap-contracts/-/tree/audit/03

Commit number: `e5cbed367619eae60d174a5c60770f2bf305a42e`

The scope of the audit was limited to the following files:

```
original_contracts/AugustusSwapper.sol
original_contracts/UniswapProxy.sol
original_contracts/TokenTransferProxy.sol
original_contracts/lib/ReduxToken.sol
original_contracts/lib/uniswapv2/UniswapV2.sol
original_contracts/lib/Utils.sol
```

Intended Behavior

The ParaSwap smart contracts implement the on-chain component of a DEX aggregator. The audited components include single-path and multi-path token swap contracts and DEX router adapters. The contracts included in the scope of this audit are a single- and multi-path swapper contract the partnership program and Uniswap router.

Code Complexity and Test Coverage

Smart contract audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of a smart contract system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**.

Note, that high complexity or lower test coverage does equate to a higher risk. Certain bugs are more easily detected in unit testing than a security audit and vice versa. It is, therefore, more likely that undetected issues remain if the test coverage is low or non-existent.

Criteria	Status	Comment
Code complexity	Medium	-
Code readability and clarity	Medium	-
Level of Documentation	Medium-low	-
Test Coverage	Medium	-

SOLIDIFIED

Audit Report for Paraswap - March 29, 2021

Issues Found

Solidified found that the Paraswap contracts contain 1 critical issue, 2 major issues, 1 minor issues, 1 warning, in addition to 1 informational notes.

We recommend all issues are amended, while the notes are up to the team's discretion, as they refer to best practices.

Issue #	Description	Severity	Status
1	UniswapV2.sol: Functions _buy() and _swap() do not return the earned ETH back to msg.sender	Critical	Pending
2	AugustusSwapper.sol: Contract owner can arbitrarily modify _uniswapProxy while users potentially have pending transactions	Major	Pending
3	UniswapV2.sol: Missing implementation for onChainSwap() leads to loss of funds	Major	Pending
4	AugustusSwapper.sol: Implementation discrepancy for function withdrawAllWETH()	Minor	Pending
5	AugustuSwapper.sol: AugustusSwapper might misbehave with some ERC-20 tokens	Warning	-
6	Code duplication	Note	-
7	ReduxToken.sol: Function _destroyChildren() does not check if value equals zero	Note	-
8	AugustusSwapper.sol: Function approve() never used	Note	-

Critical Issues

1. UniswapV2.sol: Functions _buy() and _swap() do not return the earned ETH back to msg.sender

---

After exchanging the transferred tokens to ETH, functions `_buy()` and `_swap()` end up only withdrawing the earned ETH from the `WETH` contract, but never actually transferring it back to `msg.sender`.

Recommendation
Transfer the earned ETH back to `msg.sender` via calling `TransferHelper.safeTransferETH()`.

Major Issues

2. AugustusSwapper.sol: Contract owner can arbitrarily modify _uniswapProxy while users potentially have pending transactions

---

Function `changeUniswapProxy()` allows the contract owner to modify `_uniswapProxy` at any arbitrary time of their choice. This means that users who have pending transactions for `swapOnUniswap()` or `buyOnUniswap()` (amongst several others) will end up having their transactions executed on a different proxy contract than the one they were expecting at transaction creation.

Recommendation
Only allow the owner to change `_uniswapProxy` after a certain time-lock period has passed, which gives users the chance to drop their pending transactions in case they do not wish to have them executed using the new proxy contract.

3. UniswapV2.sol: Missing implementation for onChainSwap() leads to loss of funds

Function `onChainSwap()` is currently missing the implementation required for the actual swapping of tokens (currently commented out in code). A call to this function that sends ETH will result in loss of all the funds sent.

Recommendation
Provide appropriate implementation.

Minor Issues

4. AugustusSwapper.sol: Implementation discrepancy for function withdrawAllWETH()

Function `withdrawAllWETH()`'s documentation states that it sends the `WETH` returned during the exchange to the user, while its implementation simply converts the contract's `WETH` back to ETH.

Recommendation
Resolve discrepancy or remove function.

Warnings

5. AugustuSwapper.sol: AugustusSwapper might misbehave with some ERC-20 tokens

There are some ERC-20 implementations out there and some of them might cause unexpected consequences, such as tokens that charge fees on transfer, malicious implementations, or tokens that return false instead of reverting.

Recommendation
There's not a particular way to deal with this. One option is to add a list of allowed tokens and block execution to others. Another option is to keep this list in the user interface and warn users if they are interacting with tokens that might misbehave.

Informative Notes

6. Code duplication

Throughout the codebase there is a lot of duplicate code. Particular examples :

- duplicate implementations in `original_contracts/lib/uniswapv2/UniswapV2.sol` and `original_contracts/UniswapProxy.sol`.
- Functions `multiSwap()` and `megaSwap()` in original_contracts/AugustusSwapper.sol.

Recommendation
Consider refacturing the code to reduce code duplication.

7. ReduxToken.sol: Function _destroyChildren() does not check if value equals zero

Function `_destroyChildren()` does not check if `value` equals zero, resulting in unnecessary instructions being executed and gas wasted when this is the case. This could happen on multiple occasions, most notably when function `freeFromUpTo()` is called and `userAllowance` is zero.

Recommendation
Check that `value!=0` before executing any instructions.

8. AugustusSwapper.sol: Function approve() never used

The function `approve()` has been restricted by the `onlySelf` modifier to be only called by `AugustusSwapper` or any of its descendants/delegates, but the function never ends up being called.

Recommendation
Remove the `approve()` function to save on deployment gas fees.

Audit Report for Paraswap - March 29, 2021

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of Paraswap or its products. This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect the auditors of Solidified platform from legal and financial liability.

Solidified Technologies Inc.