New KLAYswap Security Audit

: Ozys New KLAYswap

July 17, 2024

Revision 1.0

ChainLight@Theori

Table of Contents

Executive Summary

Starting April 23, 2024, ChainLight of Theori audited Ozys's New KLAYswap for two weeks. Its swap component is based on KLAYswap V2/V3 (KLAYswap V3 is a Uniswap V3 fork), and the "lending" component, which supersedes the KLAYswap's Single Pool, is based on AAVE. In the audit, we primarily considered the issues/impacts listed below.

- Theft of funds
- Violation of liquidity invariant
- Any modification to core logic where not expected
- Incorrect emission of KSP
- (MEV is not profoundly considered since the contracts will be deployed on an L2.)

As a result, we identified the issues listed below.

- Total: 7
- Critical: 1 (Reinitialization issue leading to theft of funds.)
- Medium: 1 (DoS through share inflation.)
- Informational: 5 (Known upstream issue, Griefing issue, Minor issues including defense-in-depth suggestions.)

In addition to the reported issues, we also discovered a potential problem with the KSP mining halving process. If the admin does not trigger the update at the exact time, the halving can be applied late. To address this, the team plans to adjust the rate promptly at the halving time and resolve any small discrepancies that may arise.

Audit Overview

Scope

Name	New KLAYswap Security Audit
Target / Version	• Git Repository (`KlaySwap/klayswap-v2-smart-contracts`): commit (after patch) `14ab800edb2f1913a329b78edc7bc53b2042327d` • Git Repository (`KlaySwap/klayswap-v3-smart-contracts`): commit (after patch) `de91941b4f73051f095b8fbdc21bd7252da85230`
Application Type	Smart contracts
Lang. / Platforms	Smart contracts [Solidity]

Code Revision

N/A

Severity Categories

Severity	Description
Critical	The attack cost is low (not requiring much time or effort to succeed in the actual attack), and the vulnerability causes a high-impact issue. (e.g., Effect on service availability, Attacker taking financial gain)
High	An attacker can succeed in an attack which clearly causes problems in the service's operation. Even when the attack cost is high, the severity of the issue is considered "high" if the impact of the attack is remarkably high.
Medium	An attacker may perform an unintended action in the service, and the action may impact service operation. However, there are some restrictions for the actual attack to succeed.
Low	An attacker can perform an unintended action in the service, but the action does not cause significant impact or the success rate of the attack is remarkably low.
Informational	Any informational findings that do not directly impact the user or the protocol.
Note	Neutral information about the target that is not directly related to the project's safety and security.

Status Categories

Status	Description
Reported	ChainLight reported the issue to the client.
WIP	The client is working on the patch.
Patched	The client fully resolved the issue by patching the root cause.
Mitigated	The client resolved the issue by reducing the risk to an acceptable level by introducing mitigations.
Acknowledged	The client acknowledged the potential risk, but they will resolve it later.
Won't Fix	The client acknowledged the potential risk, but they decided to accept the risk.

Finding Breakdown by Severity

Category	Count	Findings
Critical	1	• `NEWKLAYSWAP-003`
High	0	• N/A
Medium	1	• `NEWKLAYSWAP-002`
Low	0	• N/A
Informational	5	• `NEWKLAYSWAP-001` • `NEWKLAYSWAP-004` • `NEWKLAYSWAP-005` • `NEWKLAYSWAP-006` • `NEWKLAYSWAP-007`
Note	0	• N/A

Findings

Summary

#	ID	Title	Severity	Status
1	`NEWKLAYSWAP-001`	Using tick spacing 1 for 5-bps pools may increase gas costs	Informational	Won't Fix
2	`NEWKLAYSWAP-002`	`Exchange.impl` is vulnerable to share inflation attack	Medium	Patched
3	`NEWKLAYSWAP-003`	Anyone can reinitialize swap routers	Critical	Patched
4	`NEWKLAYSWAP-004`	`removeLiquidityWithPermit()` and `removeLiquidityETHWithPermit()` in `V2Router` is vulnerable to griefing attack	Informational	Patched
5	`NEWKLAYSWAP-005`	Minor suggestions	Informational	Patched
6	`NEWKLAYSWAP-006`	(Lending) Mitigate the known issue inherited from AAVE	Informational	Patched
7	`NEWKLAYSWAP-007`	(Lending) Minor suggestions	Informational	Patched

#1 `NEWKLAYSWAP-001` Using tick spacing 1 for 5-bps pools may increase gas costs

ID	Summary	Severity
`NEWKLAYSWAP-001`	If used in a pool of volatile assets, the configuration of `feeAmountTickSpacing[500] = 1` in `UniswapV3FactoryImpl._initialize()` would cause frequent liquidity tick boundary crossings, resulting in additional gas costs during the token swap procedure.	Informational

Description

When creating a pool through `UniswapV3Factory`, `feeAmountTickSpacing[FeeTier] = TickSpacing` can be used to set the desired tick spacing for each fee tier. Currently, selecting the 0.05% fee tier when creating a V3 pool results in a tick spacing of 1. This value is suitable for stable pools with low volatility but can also be used for trading pairs where this setting is not appropriate because anyone can create a pool by calling `createPool()`. For instance, if a highly volatile asset is traded in a pool with a tick spacing of 1, the asset price often crosses the tick boundary, resulting in significant gas consumption for the user. Frequent crossing of tick boundaries requires updating the state for many ticks, thereby consuming more gas than a trade that takes place on only one or a small number of ticks.

Impact

Informational

Users swapping in a 0.05% fee tier pool will pay more for gas than users swapping in a pool with better tick spacing if the pair is volatile.

Recommendation

It is recommended to change the tick spacing for the 0.05% fee tier pool to 10 (that is, `feeAmountTickSpacing[500] = 10`), and add an additional 0.01% fee tier pool for the stable liquidity pool (that is, `feeAmountTickSpacing[100] = 1`). Despite the potential for liquidity fragmentation due to the addition of fee tiers, liquidity providers are expected to naturally settle into the fee tier most suitable for the pair's volatility over time. Any issues with the fee for a set pool can be discussed through governance and changed through the `UniswapV3Factory.enableFeeAmount()` call.

Remediation

Won't Fix

0.05% fee tier will be used for stable pools.

#2 `NEWKLAYSWAP-002` `Exchange.impl` is vulnerable to share inflation attack

ID	Summary	Severity
`NEWKLAYSWAP-002`	`Exchange.impl` is vulnerable to an inflation attack, where an attack by the initial liquidity provider may block further liquidity provision.	Medium

Description

Users can create a pool for a token pair for which a pool does not yet exist by calling `Factory.impl.createTokenPool()`. Since the mitigation for the inflation attack (transferring a small amount of LP tokens to a dead address) is not applied to the initial liquidity provision, the initial liquidity provider can launch a denial of service (DoS) attack on the pool.

- Attack scenario

1. An attacker issues 1 wei of LP token when creating the pool.
2. The attacker transfers the two tokens traded in the pool, as many as are available, to the pool. Transactions of users who subsequently provide liquidity will always be reverted because zero LP tokens are issued unless they provide more tokens than the pool holds, resulting in their inability to provide liquidity.

Impact

Medium

An attacker can monopolize the transaction fees generated by the pool they have successfully attacked until a liquidity provider with larger funds comes along.

Recommendation

During the initial liquidity provision, a small amount of LP tokens should be transferred to the dead address.

Remediation

Patched

It is patched as recommended.

#3 `NEWKLAYSWAP-003` Anyone can reinitialize swap routers

ID	Summary	Severity
`NEWKLAYSWAP-003`	Because `_initialize()` in `V2Router.impl.sol` and `UniversalRouter.impl.sol` can be called multiple times by anyone, an attacker can easily change the `owner`, `factory`, and `WETH` addresses. This allows the attacker to steal funds from the router users.	Critical

Description

In `V2Router.impl.sol`, an attacker can change the `owner`, `factory`, and `WETH` addresses by calling `_initialize()`. After changing the `factory` and `owner`, an arbitrary `forwarder` can be specified by calling `setTrustedForwarder()`. Because the `forwarder` can specify an arbitrary address for the `_msgSender()` when calling `swapExactTokensForTokens()`, all tokens that users granted allowance through `approve()` to the `V2Router` can be transferred into the router. Thus, a malicious `factory` contract could allow an attacker to take tokens on the router by calling `V2Router.impl.approvePair(attacker, token0, token1)`. This allows the attacker to steal all tokens that users granted allowance to the router. In the case of `UniversalRouter.impl.sol`, since it does not have the EIP-2771 Forwarder feature, it is impossible to steal all tokens approved for the router with the same exploit. However, when a user performs a swap, an attacker can change the `factory`, `v2Router`, and `v3Router` addresses by front-running to steal funds from the user. The number of tokens the attacker can steal may be limited depending on the values of `amountOutMin` and `amountInMax` used by the user when calling `swapExactTokensForTokens()` or `swapTokensForExactTokens()`.

Impact

Critical

An attacker can steal tokens that users have granted allowance to `V2Router.impl.sol`. For `UniversalRouter.impl.sol`, an attacker could use front-running to steal some of the funds of a user trying to perform a swap. However, the amount the attacker can steal is limited by the user's slippage tolerance value.

Recommendation

It is recommended to add a check such as `require(_factory == address(0))` to these functions to prevent `_initialize()` in `V2Router.impl.sol` and `UniversalRouter.impl.sol` from being called again. In addition, the same issue exists with `_initialize()` in `v7/periphery/V3TreasuryView.sol` and `v5/treasury/Treasury.impl.sol`.

Remediation

Patched

It is patched as recommended.

#4 `NEWKLAYSWAP-004` `removeLiquidityWithPermit()` and `removeLiquidityETHWithPermit()` in `V2Router` is vulnerable to griefing attack

ID	Summary	Severity
`NEWKLAYSWAP-004`	`removeLiquidityWithPermit()` and `removeLiquidityETHWithPermit()` in the `V2Router` contract are vulnerable to griefing by front-running.	Informational

Description

Liquidity providers can remove liquidity through the `V2Router` contract. Liquidity providers should first grant the `V2Router` contract an LP token `allowance` for the pair from which they want to remove liquidity. If the liquidity provider address is an externally owned account (EOA), two transactions must be submitted: one to grant the allowance and one to remove the liquidity. To improve the UX, the `Exchange.impl` (LP Token) contract provides a `permit` function to grant `allowance` with an off-chain signature, and the `V2Router` contract uses the `permit` function. However, `removeLiquidityWithPermit()` and `removeLiquidityETHWithPermit()` in the `V2Router` contract are vulnerable to front-running, which may result in a revert. When a liquidity remover calls the function with an off-chain signature, the attacker may find the `permit()` argument by observing the transaction, and the transaction of the liquidity remover will fail if the attacker calls `IExchange.permit()` first with the same argument. This issue can lead to a bad UX.

Impact

Informational

When a liquidity remover calls a `permit` function with an off-chain signature, a front-running attacker can cause a revert, leading to a loss of gas fees and a bad UX. Since the contract will be deployed to an L2, the likelihood of attack is very low since the front-running can only be performed by the sequencer or the operator of the RPC service.

Recommendation

One of the below can be applied:

1. Use a try-catch statement when calling `IExchange.permit()` to ignore the error and attempt to remove liquidity.
2. If sufficient `allowance` has already been granted prior to calling `IExchange.permit()`, skip the call to `permit()` and remove liquidity.

Remediation

Patched

It is patched as recommended. (2)

#5 `NEWKLAYSWAP-005` Minor suggestions

ID	Summary	Severity
`NEWKLAYSWAP-005`	This issue includes recommendations for preventing misconfigurations, mitigating potential risks, and improving code maturity, readability, and other concerns that may arise from operational mistakes.	Informational

Description

Operational risk

- When calculating the `DOMAIN_SEPARATOR` of `initPool()` in `v5/v2/Exchange.impl`, the value of `version` should contain the return value of `version()` rather than 1.
- The `emit SetEpochMining(mining, lastMined, amount)` event should be added to enable indexing of mining settings results per epoch in `v5/RewardTreasury.setEpochMining()`.
- A view function should be added to track the value of `v5/RewardTreasury.claimedAmount`.
- `Claim(amount)` event should be changed to `Claim(amount, claimedAmount)`, enabling the indexing of the results for `v5/RewardTreasury.giveReward()`.
- In `v5/AirdropOperator`, the event containing information about the owner before and after the change should be emitted in `changeNextOwner()` and `changeOwner()`.

Code maturity

- In `v5/v2/Exchange.impl`, the token parameters of `ExchangePos` and `ExchangeNeg` can be renamed to `tokenIn`, `amountIn`, `tokenOut`, and `amountOut` to improve code readability.
- In `v5/v2/Exchange.impl`, `skim()` has not been implemented yet. It should be implemented in the future or removed now if it is not planned.
- `ExchangePos` and `ExchangeNeg` in `v7/periphery/UniversalRouter.impl` should be removed because they are unused events.

Gas Optimization

- Because the variables `treasury`, `factory`, and `rewardToken` declared as constants in `v5/AirdropOperator` are not intended to be updated, they should be declared as immutable and modified by initializing them in the constructor. This is embedded directly in the contract's bytecode without using storage, which saves gas fees.

Other

- It is recommended to replace the initial LP token issuance amount in `v5/v2/Exchange.impl` with the geometric mean of `amount0` and `amount1` instead of `amount0`. Using `amount0` leads to the LP token units being directly affected by the first liquidity provider.
- In `v5/RewardTreasury`, the SafeMath library should be applied to operators that do not use it.
- In `createPool()` of `v2/Factory.impl`, the order of `token0` and `token1` must be sorted.
- In `createTokenPool()` of `v2/Factory.impl`, the `require(token1 != WETH)` condition should be removed, or it should be verified that `token0` is also not WETH.

Impact

Informational

Remediation

Patched

Most suggestions are applied as recommended.

#6 `NEWKLAYSWAP-006` (Lending) Mitigate the known issue inherited from AAVE

ID	Summary	Severity
`NEWKLAYSWAP-006`	The known vulnerabilities inherited from the AAVE V2/V3 must be mitigated.	Informational

Description

AAVE disclosed a security issue that may occur when converting a variable rate position to a stable rate position on the AAVE v2/v3. The exact details have not been disclosed to prevent forked projects from being affected, but its impact has been identified as potentially leading to fund theft. Thus, it is recommended that a variable rate position be prevented from being converted to a stable rate position.

Impact

Informational

Recommendation

Remove `BorrowLogic.executeSwapBorrowRateMode()` to prevent converting from a variable rate position to a stable rate position.

Remediation

Patched

It is patched as recommended.

#7 `NEWKLAYSWAP-007` (Lending) Minor suggestions

ID	Summary	Severity
`NEWKLAYSWAP-007`	This issue includes recommendations for preventing misconfigurations, mitigating potential risks, and improving code maturity, readability, and other concerns that may arise from operational mistakes.	Informational

Description

Operational Risk

- If `LendingRewardTokenManager.update()` is called when `manager()` was never called, a divide-by-zero error occurs when calculating the value of the `RewardConfigInput.emissionPerSecond` field because the value of `totalWeight` is not set. Thus, the `require(totalWeight > 0)` check should be added at the beginning of `update()`, or the function should be changed to an internal function to allow it to be called only from inside `manager()`.
- `LendingRewardTokenManager` should use `Ownable2Step` instead of `Ownable` to prevent incorrect owner settings.
- To maintain semantics in `LendingRewardTokenManager`, an oracle contract must be created to return the KSP price based on Chainlink ABI so that users can see the value of the reward they should receive in USD.
- In some Testnet Helper functions, owner validation has been deleted. Operational caution is required to ensure that the tokens used here are not accidentally used as collateral in the production environment.

Impact

Informational

Recommendation

It is recommended to apply the suggestion.

Remediation

Patched

Most suggestions are applied as recommended.

Revision History

Version	Date	Description
1.0	July 17, 2024	Initial version

ChainLight