SHERLOCK

SHERLOCK SECURITY REVIEW FOR

Contest type:	Public
Prepared for:	Arcadia
Prepared by:	Sherlock
Lead Security Expert:	zzykxx
Dates Audited:	April 22 - April 25, 2024
Prepared on:	May 14, 2024

Introduction

Arcadia connects passive lenders and on-chain leverage strategists. Earn passive interest or 10x your liquidity to deploy across protocols.

Scope

Repository: arcadia-finance/accounts-v2

Branch: AF-543-aerodrome-finance-AM

Commit: c0b9c92c210aac1b759c78215dbf40d40e85bf66

For the detailed scope, see the contest details.

Findings

Each issue has an assigned severity:

- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- High issues are directly exploitable security vulnerabilities that need to be fixed.

Issues found

Medium	High
3	0

Issues not fixed or acknowledged

Medium	High
0	0

Security experts who found valid issues

zzykxx	blackhole
merlin	PUSH0

Issue M-1: Donating (and syncing) tokens to an Aerodrome allows to bypass exposure limits

Source: https://github.com/sherlock-audit/2024-04-arcadia-pricing-module-judging/issues/17

Found by

zzykxx

Summary

An attacker can bypass the exposure limits by donating tokens to an Aerodrome pool in which he owns ~100% of the liquidity, this will inflate reserves which will increase the value of the collateral and the attacker won't lose capital because he owns ~100% of the liquidity.

Vulnerability Detail

Attack scenario:

1. Create a new Aerodrome pool, let's suppose an STG/WETH pool
2. Provide liquidity to the pool and mint LPs, let's suppose we mint ~1000$ worth of liquidity
3. Deposit the LP in an Arcadia account, the collateral is worth ~1000$
4. Send STG and WETH directly to the Aerodrome pool, let's suppose we send ~500_000$ worth of tokens
5. Call sync() on the Aerodrome pool
6. The pool reserves are increased and the collateral is now worth ~501_000$

By doing this an attacker can bypass exposure limits. Note that Arcadia allows the use of any volatile Aerodrome pool LPs as collateral, as long as the underlying tokens are allowed.

Impact

An attacker can bypass the exposure limits set by a creditor.

Code Snippet

Tool used

Manual Review

Recommendation

Discussion

sherlock-admin4

1 comment(s) were left on this issue during the judging contest.

shealtielanz commented:

> dup of #018

Thomas-Smets

Valid, think it is not a duplicate of 18 tho

sherlock-admin3

The protocol team fixed this issue in the following PRs/commits: https://github.com/arcadia-finance/accounts-v2/pull/201

midori-fuse

I understand that the escalation phase is over, and I fully accept that the result cannot be changed further. However I do think this is a dupe of #18 for the following reasons.

Essentially both attacks involve opening a new Aerodrome pool, and increasing the LP position's underlying worth using fees. When you donate your assets into the pool, it has the same effect as when you swap back and forth within the pool (whatever stays in the pool gets counted as fees). Therefore the two issues are a dupe because:

- While the root causes can be considered different, the differences lie in the Aerodrome pool, not the Arcadia pricing module. What enabled this attack within the pricing module are the same: fees are always applied directly and fully without capping.
- I can't identify a fix that would fix one problem but without fixing the other.
    - Note that the sponsor making the same PR isn't a factor, there may still be a fix that separates these two issues that we failed to identify.

Am I missing anything? Is there a difference in root cause/impact/fix that makes the two issues different?

We shall still respect the current/final judgement regardless. Even if the "ideal" result is different from currently, it was fully on us to not have examined the issues diligently in the 48-hour escalation period that we were given.

sherlock-admin2

The Lead Senior Watson signed off on the fix.

Issue M-2: Swapping large amounts of assets back and forth in an Aerodrome pool allows to bypass exposure limits

Source: https://github.com/sherlock-audit/2024-04-arcadia-pricing-module-judging/issues/18

Found by

PUSH0, zzykxx

Summary

Swapping large amounts of assets back and forth in an Aerodrome pool allows to bypass exposure limits.

Vulnerability Detail

The `WrappedAerodromeAM` allows wrapping LPs of an Aerodrome pool to then deposit the wrapped position as collateral in an Arcadia account. The wrapper accounts for the trading fees accumulated by the LP when estimating the value of the collateral:

```
(, uint256 positionId) = _getAssetFromKey(assetKey);
(uint256 fee0Position, uint256 fee1Position) = feesOf(positionId);

underlyingAssetsAmounts = new uint256[](3);
underlyingAssetsAmounts[0] = positionState[positionId].amountWrapped;
underlyingAssetsAmounts[1] = fee0Position;
underlyingAssetsAmounts[2] = fee1Position;
```

It's possible to take advantage of this to bypass exposure limits. An attacker can:

1. Create a new Aerodrome pool
2. Mint LPs, the attacker owns the whole liquidity supply
3. Deposit the LP as collateral via `WrappedAerodromeAM`
4. Swap large amounts in the Aerodrome pool back and forth to increase the fees accumulated by the LP. The value of the collateral increases while the collateral is already deposited. The attacker won't lose anything besides gas fees because he controls the whole (minues 1e3) liquidity supply.

An attacker can keep swapping amounts indefinitely. Because this is done while the collateral is already deposited in the account the attacker can avoid the restrictions of exposure limits.

Impact

Exposure limits can be bypassed.

Code Snippet

Tool used

Manual Review

Recommendation

A similar attack is possible on UniswapV3 positions, however in that case the protocol already implements safety guards against this possibility. Implement a similar safety guard for Aerodrome positions as well.

Discussion

sherlock-admin4

1 comment(s) were left on this issue during the judging contest.

shealtielanz commented:

> exposure limits will truly be bypassed

sherlock-admin3

The protocol team fixed this issue in the following PRs/commits: https://github.com/arcadia-finance/accounts-v2/pull/201

sherlock-admin2

The Lead Senior Watson signed off on the fix.

Issue M-3: WrappedAerodromeAM.sol is not compatible with the Revert on Zero Value Tokens

Source: https://github.com/sherlock-audit/2024-04-arcadia-pricing-module-judging/issues/23

Found by

blackhole, merlin

Summary

From the README file: We also want to receive issues regarding minimal implementations of ERC20 Tokens, which besides a standard implementation have one of the following "weird behaviours": `Revert on Zero Value`.

Vulnerability Detail

When the owner of a position calls `WrappedAerodromeAM.decreaseLiquidity`, they should receive the transferred liquidity back and fees.

Let's consider the following scenario:

1)The owner of the position calls `decreaseLiquidity`, and they are supposed to receive fees.

```
// Claim any pending fees from the Aerodrome Pool.
(uint256 fee0Pool, uint256 fee1Pool) = _claimFees(pool);

// Calculate the new fee balances.
(poolState_, positionState_) = _getFeeBalances(poolState_, positionState_,
↪   fee0Pool, fee1Pool);
```

2)Since the fee for the `revert on zero transfer token` is zero, a DOS will occur, and the user will not be able to decrease liquidity.

```
// Pay out the fees to the position owner.
ERC20(token0[pool]).safeTransfer(msg.sender, fee0Position);  //fee0Position = 0
ERC20(token1[pool]).safeTransfer(msg.sender, fee1Position); // or fee1Position =
↪   0
emit FeesPaid(positionId, uint128(fee0Position), uint128(fee1Position));
```

Impact

A denial-of-service (DOS) can occur on the `decreaseLiquidity` and `claimFees` functions in `WrappedAerodromeAM` smart contract of the revert on zero transfer token when the fee is zero value.

Code Snippet

src/asset-modules/Aerodrome-Finance/WrappedAerodromeAM.sol#L452-L453
src/asset-modules/Aerodrome-Finance/WrappedAerodromeAM.sol#L411-L412

Tool used

Manual Review

Recommendation

Consider changing `decreaseLiquidity` and `claimFees` functions as follows:

```
+ if (fee0Position != 0)
            ERC20(token0[pool]).safeTransfer(msg.sender, fee0Position);
+ if (fee1Position != 0)
            ERC20(token1[pool]).safeTransfer(msg.sender, fee1Position);
```

Discussion

sherlock-admin4

2 comment(s) were left on this issue during the judging contest.

takarez commented:

> this should be valid considering the readMe stated to report issues with the said(weird) behavior; medium(1)

shealtielanz commented:

> dup of #033

sherlock-admin3

The protocol team fixed this issue in the following PRs/commits:
https://github.com/arcadia-finance/accounts-v2/pull/202

oxwhite

What does "besides a standard implementation have one of the following "weird behaviours": Revert on Zero Value" mean? What ı understand the protocol will not accept any vulnerability related to "Revert on Zero Value" and the followings mentioned in ReadMe, which means it is out of scope. How can this submission be valid?

sherlock-admin2

The Lead Senior Watson signed off on the fix.

Disclaimers

Sherlock does not provide guarantees nor warranties relating to the security of the project.

Usage of all smart contract software is at the respective users’ sole risk and is the users’ responsibility.