0x Guard

Smart contracts security assessment

Final report

Tariff: Standard

Salvo Financial InvestorLp

August 2022

0xguard.com

hello@0xguard.com

Contents

Introduction

The report has been prepared for **Salvo Financial InvestorLp**.

Only the InvestorLp contract of the Salvo Financial project was audited. The Salvo contracts are designed to be deployed with the EIP-2535 scheme. We can't ensure the contract's interaction within the current audit.

Name	Salvo Financial InvestorLp
Audit date	2022-08-22 - 2022-08-22
Language	Solidity
Platform	Avalanche Network

Contracts checked

Name	Address
InvestorLp	`https://github.com/sebastianroa/salvo/blob/adb99611958fe0189ff672f82c410cce295e5c86/InvestorLp.sol`

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual verification (reject or confirm) all the issues found by the tools

Manual audit

- Manually analyze smart contracts for security vulnerabilities
- Smart contracts' logic check

Known vulnerabilities checked

Title	Check result
Unencrypted Private Data On-Chain	passed
Code With No Effects	passed
Message call with hardcoded gas amount	passed
Typographical Error	passed
DoS With Block Gas Limit	passed
Presence of unused variables	passed
Incorrect Inheritance Order	passed
Requirement Violation	passed
Weak Sources of Randomness from Chain Attributes	passed
Shadowing State Variables	passed
Incorrect Constructor Name	passed
Block values as a proxy for time	passed
Authorization through tx.origin	passed
DoS with Failed Call	passed
Delegatecall to Untrusted Callee	passed
Use of Deprecated Solidity Functions	passed
Assert Violation	passed
State Variable Default Visibility	passed
Reentrancy	passed

Unprotected SELFDESTRUCT Instruction	passed
Unprotected Ether Withdrawal	passed
Unchecked Call Return Value	passed
Floating Pragma	not passed
Outdated Compiler Version	passed
Integer Overflow and Underflow	passed
Function Default Visibility	passed

Classification of issue severity

High severity	High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.
Medium severity	Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.
Low severity	Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

Issues

High severity issues

1. Unclear authorization in deposit function (InvestorLp)

`routerDepositAvaxLp()` function calls for `stakingManager.deposit()` without `msg.sender` address in parameters, although `routerWithdraw()` calls for `stakingManager.poolStakers(poolId, msg.sender)`. Authorization model of StakingManager is out of the scope of this audit.

Recommendation: Use consistent and documented authorization.

2. Owner can change staking pool ID (InvestorLp)

Changing the `poolId` variable after the contract becomes operable may cause locking of the users' funds.

```
    function changePoolId(uint256 _newId) external onlyOwner {
        poolId = _newId;
    }
```

Recommendation: Remove the `changePoolId()` function.

3. Use of delegatecall to external sources (InvestorLp)

Multiple delegatecalls to `swapRouter`, `investorHelper`, and `helperRouter` external contracts out of the scope of this audit. Contracts' interaction via delegatecall must be reasonably tested to avoid storage collisions.

Recommendation: Increase the tests' coverage.

4. Owner can seize the users' funds (InvestorLp)

Changing the `stakingManager` address allows the owner to control all the deposited funds.

```
    function changeStakingManager(address _newStakingManager)
        external
        onlyOwner
    {
        stakingManager = _newStakingManager;
    }
```

The `evacuateFunds()` function can be called by the owner to directly withdraw and move deposited funds to the Treasury address, which is out of the scope of this audit.

```
    function evacuateFunds() public onlyOwner {
        require(
            paused == true,
            "Contract must be paused before performing this operation."
        );
        //Withdraw Entire Amount
        (bool successWithdraw, bytes memory dataWithdraw) = helperRouter
            .delegatecall(
                abi.encodeWithSignature(
                    "withdrawLp(uint256,address)",
                    IERC20(investmentAddress).balanceOf(address(this)),
                    investmentAddress
                )
            );
        require(
            successWithdraw,
            "Delegate Call for Evacuation Withdrawing from Lp Failed."
        );
        IERC20(liquidityPool).transfer(
            IAddressRouter(addressRouter).viewAddressDirectory("Treasury"),
            IERC20(liquidityPool).balanceOf(address(this))
        );
    }
```

Recommendation: Remove the `changeStakingManager()` and `evacuateFunds()` functions.

5. Owner can lock users' funds (InvestorLp)

The audited contract can be paused by the owner. A malicious or hacked owner can completely lock the deposited funds.

Recommendation: Restrict the owner's ability to pause instantly or introduce the bypass method for withdrawal.

Medium severity issues

1. Slippage is set for a different token (InvestorLp)

Workflow of the `routerDepositAvaxLp()` function includes swap `swapRouter.exchangeExactAvaxForTokens()` for the user-provided token address `addressRouter.viewAddressDirectory(_tokenName)`, while setting slippage through the `getExchangeRate()` function with fixed `targetToken` path.

Recommendation: The `_targetToken` array should be filled locally inside the `routerDepositAvaxLp()` function. The state variable `targetToken` should be removed.

2. Improper use of swap parameters (InvestorLp)

The `amountOutMin` and `deadline` parameters for UniswapRouter-like contract calls must be acquired outside of the chain, otherwise, it would be constantly allowed or denied depending on chosen values in comparison to `getAmountOut(...)` and `block.timestamp` variables. For example, calling the swap function with a deadline `block.timestamp+1` always succeeds, while with `block.timestamp-1` always fails.

Low severity issues

1. Owner is allowed to use reentrancy (InvestorLp)

The `disengageMutex()` function can be used by the owner to re-enter the swap functions that are guarded for other users.

```
function disengageMutex() external onlyOwner {
     if (mutex == true) {
          mutex = false;
     }
 }
```

2. Gas optimisation (InvestorLp)

The contract is gas inefficient in terms of repetitive reads of the same variables from blockchain, e.g.

```
function routerWithdraw(uint256 _amount, address _spenderAddress)
    public
    payable
    verifyPool
{
    ...
    for (uint8 i = 0; i < tokensToBeRewardedAddress.length; i++) {
        rewardSnapshot[tokensToBeRewardedAddress[i]] = IERC20(
            tokensToBeRewardedAddress[i]
        ).balanceOf(address(this));
    }

    for (uint8 i = 0; i < tokensToBeRewardedAddress.length; i++) {
        if (rewardSnapshot[tokensToBeRewardedAddress[i]] != 0) {
            IERC20(tokensToBeRewardedAddress[i]).transfer(
                IAddressRouter(addressRouter).viewAddressDirectory(
                    "Treasury"
                ),
                rewardSnapshot[tokensToBeRewardedAddress[i]] /
                    IStakingManager(stakingManager).bankCut()
            );
        }
    }
    ...
}
```

List of functions with repetitive reads of state variables: `routerDepositAvaxLp()`, `routerWithdraw()`, `evacuateFunds()`.

Requirements of the `verifyPool()` modifier should be verified inside the governance functions that change the corresponding parameters, i.e. `stakingManager`, `poolId`, and `liquidityPool`. In that case, gas would be saved on ordinary contract calls by users.

`addressRouter`, `helperRouter`, `swapRouter`, `investmentAddress`, `liquidityPool`, `investorHelper` variables should be declared as immutable.

The `setSlippage()` function performs excessive operations:

```
    function setSlippage(uint256 _amount, uint256 _slippage)
        internal
        pure
        returns (uint256)
    {
        uint256 PRECISION = 10000;
        return (((_amount * PRECISION) / 100) * _slippage) / PRECISION;
    }
```

It's equivalent to:

```
    function setSlippage(uint256 _amount, uint256 _slippage)
        internal
        pure
        returns (uint256)
    {
        return _amount * _slippage / 100;
    }
```

3. Few events (InvestorLp)

There are no events emitted during the contract's lifecycle. This may significantly complicate the user's interaction with the contract as well as debugging the potential problems.

Conclusion

Salvo Financial InvestorLp InvestorLp contract was audited. 5 high, 2 medium, 3 low severity issues were found.

Only the InvestorLp contract of the Salvo Financial project was audited. The Salvo contracts are designed to be deployed with EIP-2535 scheme. The InvestorLp contract contains delegated calls to external sources that haven't been checked within the current audit.

Disclaimer

Slither's output

```
InvestorLp.routerDepositAvaxLp(string,string) (contracts/InvestorLp.sol#229-309) uses
delegatecall to a input-controlled function id
        - (successSwapAvax) = swapRouter.delegatecall(abi.encodeWithSignature(exchangeEx
actAvaxForTokens(uint256,uint256,address),setSlippage(getExchangeRate(msg.value /
2,reverseArray(_targetToken))[1],98),msg.value / 2,tokenAddress)) (contracts/
InvestorLp.sol#247-259)
InvestorLp.routerDepositAvaxLp(string,string) (contracts/InvestorLp.sol#229-309) uses
delegatecall to a input-controlled function id
        - (successLiquidity) = investorHelper.delegatecall(abi.encodeWithSignature(addLi
quidity(address,address,uint256),tokenAddress,avaxAddress,msg.value / 2)) (contracts/
InvestorLp.sol#266-273)
InvestorLp.routerDepositAvaxLp(string,string) (contracts/InvestorLp.sol#229-309) uses
delegatecall to a input-controlled function id
        - (successDeposit) = helperRouter.delegatecall(abi.encodeWithSignature(depositLP
Native(uint256,address,address,address),IERC20(liquidityPool).balanceOf(address(this)),l
iquidityPool,IAddressRouter(addressRouter).viewAddressDirectory(_spender),investmentAddr
ess)) (contracts/InvestorLp.sol#284-294)
InvestorLp.routerWithdraw(uint256,address) (contracts/InvestorLp.sol#317-450) uses
delegatecall to a input-controlled function id
        - (successWithdraw) = helperRouter.delegatecall(abi.encodeWithSignature(withdraw
Lp(uint256,address),_amount,investmentAddress)) (contracts/InvestorLp.sol#337-343)
InvestorLp.routerWithdraw(uint256,address) (contracts/InvestorLp.sol#317-450) uses
delegatecall to a input-controlled function id
        - (successRemove) = swapRouter.delegatecall(abi.encodeWithSignature(removeAvaxLi
quidity(address,address,uint256,uint256,uint256),targetToken[0],liquidityPool,differenceB
al,token0Entitlement,token1Entitlement)) (contracts/InvestorLp.sol#368-377)
InvestorLp.routerWithdraw(uint256,address) (contracts/InvestorLp.sol#317-450) uses
delegatecall to a input-controlled function id
        - (successReinvest,dataReinvest) = investorHelper.delegatecall(abi.encodeWithSig
nature(reinvestAvaxLP(address[],address[],address),targetToken,tokensToBeRewardedAddress
,_spenderAddress)) (contracts/InvestorLp.sol#416-424)
InvestorLp.getExchangeRate(uint256,address[]) (contracts/InvestorLp.sol#458-472) uses
delegatecall to a input-controlled function id
        - (success,data) = swapRouter.delegatecall(abi.encodeWithSignature(calculateExch
angeRate(uint256,address[]),_amountIn,_tokenPath)) (contracts/InvestorLp.sol#463-469)
InvestorLp.evacuateFunds() (contracts/InvestorLp.sol#567-589) uses delegatecall to a
input-controlled function id
```

```
        - (successWithdraw,dataWithdraw) = helperRouter.delegatecall(abi.encodeWithSigna
ture(withdrawLp(uint256,address),IERC20(investmentAddress).balanceOf(address(this)),inve
stmentAddress)) (contracts/InvestorLp.sol#573-580)
InvestorLp.secondWithdrawal(uint256,address) (contracts/InvestorLp.sol#591-606) uses
delegatecall to a input-controlled function id
        - (successWithdraw,dataWithdraw) = investorHelper.delegatecall(abi.encodeWithSig
nature(routerWithdraw1(uint256,address),_amount,_spenderAddress)) (contracts/
InvestorLp.sol#597-604)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#controlled-
delegatecall

InvestorLp.routerWithdraw(uint256,address) (contracts/InvestorLp.sol#317-450) ignores
return value by IERC20(tokensToBeRewardedAddress[i_scope_0]).transfer(IAddressRouter(add
ressRouter).viewAddressDirectory(Treasury),rewardSnapshot[tokensToBeRewardedAddress[i_sc
ope_0]] / IStakingManager(stakingManager).bankCut()) (contracts/InvestorLp.sol#402-408)
InvestorLp.universalTransfer(address,uint256) (contracts/InvestorLp.sol#551-560)
ignores return value by IERC20(_tokenAddress).transfer(_treasury,_amount) (contracts/
InvestorLp.sol#559)
InvestorLp.evacuateFunds() (contracts/InvestorLp.sol#567-589) ignores return value by IE
RC20(liquidityPool).transfer(IAddressRouter(addressRouter).viewAddressDirectory(Treasury
),IERC20(liquidityPool).balanceOf(address(this))) (contracts/InvestorLp.sol#585-588)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-
transfer

InvestorLp.routerWithdraw(uint256,address) (contracts/InvestorLp.sol#317-450) performs
a multiplication on the result of a division:
        -token0Entitlement = setSlippage((((differenceBal * PRECISION) /
IERC20(liquidityPool).totalSupply()) *
IERC20(targetToken[0]).balanceOf(liquidityPool)) / PRECISION,98) (contracts/
InvestorLp.sol#355-360)
InvestorLp.routerWithdraw(uint256,address) (contracts/InvestorLp.sol#317-450) performs
a multiplication on the result of a division:
        -token1Entitlement = setSlippage((((differenceBal * PRECISION) /
IERC20(liquidityPool).totalSupply()) *
IERC20(targetToken[1]).balanceOf(liquidityPool)) / PRECISION,98) (contracts/
InvestorLp.sol#361-366)
InvestorLp.setSlippage(uint256,uint256) (contracts/InvestorLp.sol#480-487) performs a
multiplication on the result of a division:
        -(((_amount * PRECISION) / 100) * _slippage) / PRECISION (contracts/
InvestorLp.sol#486)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-
multiply
```

```
InvestorLp.routerWithdraw(uint256,address) (contracts/InvestorLp.sol#317-450) uses a
dangerous strict equality:
	- require(bool,string)(successRemove == true,Delegate Call Removing Liquidity
Failed) (contracts/InvestorLp.sol#378-381)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-
strict-equalities

Reentrancy in InvestorLp.routerDepositAvaxLp(string,string) (contracts/
InvestorLp.sol#229-309):
	External calls:
	- (successSwapAvax) = swapRouter.delegatecall(abi.encodeWithSignature(exchangeEx
actAvaxForTokens(uint256,uint256,address),setSlippage(getExchangeRate(msg.value /
2,reverseArray(_targetToken))[1],98),msg.value / 2,tokenAddress)) (contracts/
InvestorLp.sol#247-259)
		- (success,data) = swapRouter.delegatecall(abi.encodeWithSignature(calcu
lateExchangeRate(uint256,address[]),_amountIn,_tokenPath)) (contracts/
InvestorLp.sol#463-469)
	- (successLiquidity) = investorHelper.delegatecall(abi.encodeWithSignature(addLi
quidity(address,address,uint256),tokenAddress,avaxAddress,msg.value / 2)) (contracts/
InvestorLp.sol#266-273)
	- (successDeposit) = helperRouter.delegatecall(abi.encodeWithSignature(depositLP
Native(uint256,address,address,address),IERC20(liquidityPool).balanceOf(address(this)),l
iquidityPool,IAddressRouter(addressRouter).viewAddressDirectory(_spender),investmentAddr
ess)) (contracts/InvestorLp.sol#284-294)
	- IStakingManager(stakingManager).deposit(poolId,IERC20(investmentAddress).balan
ceOf(address(this)) - initBal,0) (contracts/InvestorLp.sol#302-306)
	State variables written after the call(s):
	- mutex = false (contracts/InvestorLp.sol#308)
Reentrancy in InvestorLp.routerWithdraw(uint256,address) (contracts/
InvestorLp.sol#317-450):
	External calls:
	- (successWithdraw) = helperRouter.delegatecall(abi.encodeWithSignature(withdraw
Lp(uint256,address),_amount,investmentAddress)) (contracts/InvestorLp.sol#337-343)
	- (successRemove) = swapRouter.delegatecall(abi.encodeWithSignature(removeAvaxLi
quidity(address,address,uint256,uint256,uint256),targetToken[0],liquidityPool,differenceB
al,token0Entitlement,token1Entitlement)) (contracts/InvestorLp.sol#368-377)
	- (successReinvest,dataReinvest) = investorHelper.delegatecall(abi.encodeWithSig
nature(reinvestAvaxLP(address[],address[],address),targetToken,tokensToBeRewardedAddress
,_spenderAddress)) (contracts/InvestorLp.sol#416-424)
	- IStakingManager(stakingManager).withdraw(poolId,differenceBal,lpEarned)
```

```
(contracts/InvestorLp.sol#443-447)
        State variables written after the call(s):
        - mutex = false (contracts/InvestorLp.sol#449)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-1

InvestorLp.changeStakingManager(address) (contracts/InvestorLp.sol#519-524) should emit
an event for:
        - stakingManager = _newStakingManager (contracts/InvestorLp.sol#523)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
access-control

InvestorLp.changePoolId(uint256) (contracts/InvestorLp.sol#512-514) should emit an
event for:
        - poolId = _newId (contracts/InvestorLp.sol#513)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
arithmetic

InvestorLp.constructor(address,address,address[],address,address,address,address,address
[],string[],address,uint256)._helper (contracts/InvestorLp.sol#196) lacks a zero-check
on :
                - helperRouter = address(_helper) (contracts/InvestorLp.sol#208)
InvestorLp.constructor(address,address,address[],address,address,address,address,address
[],string[],address,uint256)._swap (contracts/InvestorLp.sol#197) lacks a zero-check
on :
                - swapRouter = address(_swap) (contracts/InvestorLp.sol#209)
InvestorLp.constructor(address,address,address[],address,address,address,address,address
[],string[],address,uint256)._addressRouter (contracts/InvestorLp.sol#199) lacks a zero-
check on :
                - addressRouter = _addressRouter (contracts/InvestorLp.sol#211)
InvestorLp.constructor(address,address,address[],address,address,address,address,address
[],string[],address,uint256)._stakingManager (contracts/InvestorLp.sol#200) lacks a
zero-check on :
                - stakingManager = _stakingManager (contracts/InvestorLp.sol#212)
InvestorLp.constructor(address,address,address[],address,address,address,address,address
[],string[],address,uint256)._investmentAddress (contracts/InvestorLp.sol#201) lacks a
zero-check on :
                - investmentAddress = address(_investmentAddress) (contracts/
InvestorLp.sol#213)
InvestorLp.constructor(address,address,address[],address,address,address,address,address
[],string[],address,uint256)._liquidityPool (contracts/InvestorLp.sol#202) lacks a zero-
```

```
check on :
                - liquidityPool = address(_liquidityPool) (contracts/
InvestorLp.sol#214)
InvestorLp.constructor(address,address,address[],address,address,address,address,address
[],string[],address,uint256)._investorHelper (contracts/InvestorLp.sol#205) lacks a
zero-check on :
                - investorHelper = _investorHelper (contracts/InvestorLp.sol#218)
InvestorLp.routerDepositAvaxLp(string,string).tokenAddress (contracts/
InvestorLp.sol#237-238) lacks a zero-check on :
                - (successSwapAvax) = swapRouter.delegatecall(abi.encodeWithSignature(ex
changeExactAvaxForTokens(uint256,uint256,address),setSlippage(getExchangeRate(msg.value
/ 2,reverseArray(_targetToken))[1],98),msg.value / 2,tokenAddress)) (contracts/
InvestorLp.sol#247-259)
                - (successLiquidity) = investorHelper.delegatecall(abi.encodeWithSignatu
re(addLiquidity(address,address,uint256),tokenAddress,avaxAddress,msg.value / 2))
(contracts/InvestorLp.sol#266-273)
InvestorLp.routerWithdraw(uint256,address)._spenderAddress (contracts/
InvestorLp.sol#317) lacks a zero-check on :
                - (successReinvest,dataReinvest) = investorHelper.delegatecall(abi.encod
eWithSignature(reinvestAvaxLP(address[],address[],address),targetToken,tokensToBeRewarde
dAddress,_spenderAddress)) (contracts/InvestorLp.sol#416-424)
InvestorLp.changeStakingManager(address)._newStakingManager (contracts/
InvestorLp.sol#519) lacks a zero-check on :
                - stakingManager = _newStakingManager (contracts/InvestorLp.sol#523)
InvestorLp.secondWithdrawal(uint256,address)._spenderAddress (contracts/
InvestorLp.sol#591) lacks a zero-check on :
                - (successWithdraw,dataWithdraw) = investorHelper.delegatecall(abi.encod
eWithSignature(routerWithdraw1(uint256,address),_amount,_spenderAddress)) (contracts/
InvestorLp.sol#597-604)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-
address-validation

InvestorLp.routerWithdraw(uint256,address) (contracts/InvestorLp.sol#317-450) has
external calls inside a loop: rewardSnapshot[tokensToBeRewardedAddress[i]] =
IERC20(tokensToBeRewardedAddress[i]).balanceOf(address(this)) (contracts/
InvestorLp.sol#389-391)
InvestorLp.routerWithdraw(uint256,address) (contracts/InvestorLp.sol#317-450) has
external calls inside a loop: IERC20(tokensToBeRewardedAddress[i_scope_0]).transfer(IAdd
ressRouter(addressRouter).viewAddressDirectory(Treasury),rewardSnapshot[tokensToBeReward
edAddress[i_scope_0]] / IStakingManager(stakingManager).bankCut()) (contracts/
InvestorLp.sol#402-408)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-
a-loop

Reentrancy in InvestorLp.routerWithdraw(uint256,address) (contracts/
InvestorLp.sol#317-450):
        External calls:
        - (successWithdraw) = helperRouter.delegatecall(abi.encodeWithSignature(withdraw
Lp(uint256,address),_amount,investmentAddress)) (contracts/InvestorLp.sol#337-343)
        - (successRemove) = swapRouter.delegatecall(abi.encodeWithSignature(removeAvaxLi
quidity(address,address,uint256,uint256,uint256),targetToken[0],liquidityPool,differenceB
al,token0Entitlement,token1Entitlement)) (contracts/InvestorLp.sol#368-377)
        State variables written after the call(s):
        - rewardSnapshot[tokensToBeRewardedAddress[i]] =
IERC20(tokensToBeRewardedAddress[i]).balanceOf(address(this)) (contracts/
InvestorLp.sol#389-391)
Reentrancy in InvestorLp.routerWithdraw(uint256,address) (contracts/
InvestorLp.sol#317-450):
        External calls:
        - (successWithdraw) = helperRouter.delegatecall(abi.encodeWithSignature(withdraw
Lp(uint256,address),_amount,investmentAddress)) (contracts/InvestorLp.sol#337-343)
        - (successRemove) = swapRouter.delegatecall(abi.encodeWithSignature(removeAvaxLi
quidity(address,address,uint256,uint256,uint256),targetToken[0],liquidityPool,differenceB
al,token0Entitlement,token1Entitlement)) (contracts/InvestorLp.sol#368-377)
        - (successReinvest,dataReinvest) = investorHelper.delegatecall(abi.encodeWithSig
nature(reinvestAvaxLP(address[],address[],address),targetToken,tokensToBeRewardedAddress
,_spenderAddress)) (contracts/InvestorLp.sol#416-424)
        State variables written after the call(s):
        - aprTracker = lpEarned (contracts/InvestorLp.sol#433)
        - aprTracker = aprTracker + lpEarned (contracts/InvestorLp.sol#436)
        - lastClaim = block.timestamp (contracts/InvestorLp.sol#434)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-2

InvestorLp.routerWithdraw(uint256,address) (contracts/InvestorLp.sol#317-450) uses
timestamp for comparisons
        Dangerous comparisons:
        - lastClaim + 86400 < block.timestamp (contracts/InvestorLp.sol#432)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp

InvestorLp.routerDepositAvaxLp(string,string) (contracts/InvestorLp.sol#229-309)
```

```
compares to a boolean constant:
        -require(bool,string)(mutex == false,Reentry Detected.) (contracts/
InvestorLp.sol#235)
InvestorLp.routerDepositAvaxLp(string,string) (contracts/InvestorLp.sol#229-309)
compares to a boolean constant:
        -require(bool,string)(paused == false,Contract is Paused.) (contracts/
InvestorLp.sol#234)
InvestorLp.routerWithdraw(uint256,address) (contracts/InvestorLp.sol#317-450) compares
to a boolean constant:
        -require(bool,string)(successRemove == true,Delegate Call Removing Liquidity
Failed) (contracts/InvestorLp.sol#378-381)
InvestorLp.routerWithdraw(uint256,address) (contracts/InvestorLp.sol#317-450) compares
to a boolean constant:
        -require(bool,string)(paused == false,Contract is Paused.) (contracts/
InvestorLp.sol#322)
InvestorLp.routerWithdraw(uint256,address) (contracts/InvestorLp.sol#317-450) compares
to a boolean constant:
        -require(bool,string)(mutex == false,Reentry Detected) (contracts/
InvestorLp.sol#323)
InvestorLp.routerWithdraw(uint256,address) (contracts/InvestorLp.sol#317-450) compares
to a boolean constant:
        -require(bool,string)(successReinvest == true,Delegate Call Reinvest to LP
Failed) (contracts/InvestorLp.sol#425)
InvestorLp.disengageMutex() (contracts/InvestorLp.sol#492-496) compares to a boolean
constant:
        -mutex == true (contracts/InvestorLp.sol#493)
InvestorLp.togglePause() (contracts/InvestorLp.sol#501-507) compares to a boolean
constant:
        -paused == true (contracts/InvestorLp.sol#502)
InvestorLp.evacuateFunds() (contracts/InvestorLp.sol#567-589) compares to a boolean
constant:
        -require(bool,string)(paused == true,Contract must be paused before performing
this operation.) (contracts/InvestorLp.sol#568-571)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-
equality

Pragma version^0.8.0 (contracts/InvestorLp.sol#3) allows old versions
solc-0.8.15 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity
```

```
Low level call in InvestorLp.routerDepositAvaxLp(string,string) (contracts/
InvestorLp.sol#229-309):
        - (successSwapAvax) = swapRouter.delegatecall(abi.encodeWithSignature(exchangeEx
actAvaxForTokens(uint256,uint256,address),setSlippage(getExchangeRate(msg.value /
2,reverseArray(_targetToken))[1],98),msg.value / 2,tokenAddress)) (contracts/
InvestorLp.sol#247-259)
        - (successLiquidity) = investorHelper.delegatecall(abi.encodeWithSignature(addLi
quidity(address,address,uint256),tokenAddress,avaxAddress,msg.value / 2)) (contracts/
InvestorLp.sol#266-273)
        - (successDeposit) = helperRouter.delegatecall(abi.encodeWithSignature(depositLP
Native(uint256,address,address,address),IERC20(liquidityPool).balanceOf(address(this)),l
iquidityPool,IAddressRouter(addressRouter).viewAddressDirectory(_spender),investmentAddr
ess)) (contracts/InvestorLp.sol#284-294)
Low level call in InvestorLp.routerWithdraw(uint256,address) (contracts/
InvestorLp.sol#317-450):
        - (successWithdraw) = helperRouter.delegatecall(abi.encodeWithSignature(withdraw
Lp(uint256,address),_amount,investmentAddress)) (contracts/InvestorLp.sol#337-343)
        - (successRemove) = swapRouter.delegatecall(abi.encodeWithSignature(removeAvaxLi
quidity(address,address,uint256,uint256,uint256),targetToken[0],liquidityPool,differenceB
al,token0Entitlement,token1Entitlement)) (contracts/InvestorLp.sol#368-377)
        - (successReinvest,dataReinvest) = investorHelper.delegatecall(abi.encodeWithSig
nature(reinvestAvaxLP(address[],address[],address),targetToken,tokensToBeRewardedAddress
,_spenderAddress)) (contracts/InvestorLp.sol#416-424)
Low level call in InvestorLp.getExchangeRate(uint256,address[]) (contracts/
InvestorLp.sol#458-472):
        - (success,data) = swapRouter.delegatecall(abi.encodeWithSignature(calculateExch
angeRate(uint256,address[]),_amountIn,_tokenPath)) (contracts/InvestorLp.sol#463-469)
Low level call in InvestorLp.evacuateFunds() (contracts/InvestorLp.sol#567-589):
        - (successWithdraw,dataWithdraw) = helperRouter.delegatecall(abi.encodeWithSigna
ture(withdrawLp(uint256,address),IERC20(investmentAddress).balanceOf(address(this)),inve
stmentAddress)) (contracts/InvestorLp.sol#573-580)
Low level call in InvestorLp.secondWithdrawal(uint256,address) (contracts/
InvestorLp.sol#591-606):
        - (successWithdraw,dataWithdraw) = investorHelper.delegatecall(abi.encodeWithSig
nature(routerWithdraw1(uint256,address),_amount,_spenderAddress)) (contracts/
InvestorLp.sol#597-604)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls

Parameter InvestorLp.routerDepositAvaxLp(string,string)._tokenName (contracts/
InvestorLp.sol#230) is not in mixedCase
```

```
Parameter InvestorLp.routerDepositAvaxLp(string,string)._spender (contracts/
InvestorLp.sol#231) is not in mixedCase
Parameter InvestorLp.routerWithdraw(uint256,address)._amount (contracts/
InvestorLp.sol#317) is not in mixedCase
Parameter InvestorLp.routerWithdraw(uint256,address)._spenderAddress (contracts/
InvestorLp.sol#317) is not in mixedCase
Parameter InvestorLp.getExchangeRate(uint256,address[])._amountIn (contracts/
InvestorLp.sol#458) is not in mixedCase
Parameter InvestorLp.getExchangeRate(uint256,address[])._tokenPath (contracts/
InvestorLp.sol#458) is not in mixedCase
Parameter InvestorLp.setSlippage(uint256,uint256)._amount (contracts/
InvestorLp.sol#480) is not in mixedCase
Parameter InvestorLp.setSlippage(uint256,uint256)._slippage (contracts/
InvestorLp.sol#480) is not in mixedCase
Parameter InvestorLp.changePoolId(uint256)._newId (contracts/InvestorLp.sol#512) is not
in mixedCase
Parameter InvestorLp.changeStakingManager(address)._newStakingManager (contracts/
InvestorLp.sol#519) is not in mixedCase
Parameter InvestorLp.reverseArray(address[])._array (contracts/InvestorLp.sol#530) is
not in mixedCase
Parameter InvestorLp.universalTransfer(address,uint256)._tokenAddress (contracts/
InvestorLp.sol#551) is not in mixedCase
Parameter InvestorLp.universalTransfer(address,uint256)._amount (contracts/
InvestorLp.sol#551) is not in mixedCase
Parameter InvestorLp.secondWithdrawal(uint256,address)._amount (contracts/
InvestorLp.sol#591) is not in mixedCase
Parameter InvestorLp.secondWithdrawal(uint256,address)._spenderAddress (contracts/
InvestorLp.sol#591) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-
solidity-naming-conventions

Variable InvestorLp.routerWithdraw(uint256,address).token0Entitlement (contracts/
InvestorLp.sol#355-360) is too similar to
InvestorLp.routerWithdraw(uint256,address).token1Entitlement (contracts/
InvestorLp.sol#361-366)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-
are-too-similar

routerDepositAvaxLp(string,string) should be declared external:
        - InvestorLp.routerDepositAvaxLp(string,string) (contracts/
InvestorLp.sol#229-309)
```

```
routerWithdraw(uint256,address) should be declared external:
        - InvestorLp.routerWithdraw(uint256,address) (contracts/InvestorLp.sol#317-450)
universalTransfer(address,uint256) should be declared external:
        - InvestorLp.universalTransfer(address,uint256) (contracts/
InvestorLp.sol#551-560)
evacuateFunds() should be declared external:
        - InvestorLp.evacuateFunds() (contracts/InvestorLp.sol#567-589)
secondWithdrawal(uint256,address) should be declared external:
        - InvestorLp.secondWithdrawal(uint256,address) (contracts/
InvestorLp.sol#591-606)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external
. analyzed (6 contracts with 77 detectors), 73 result(s) found
```

Guard