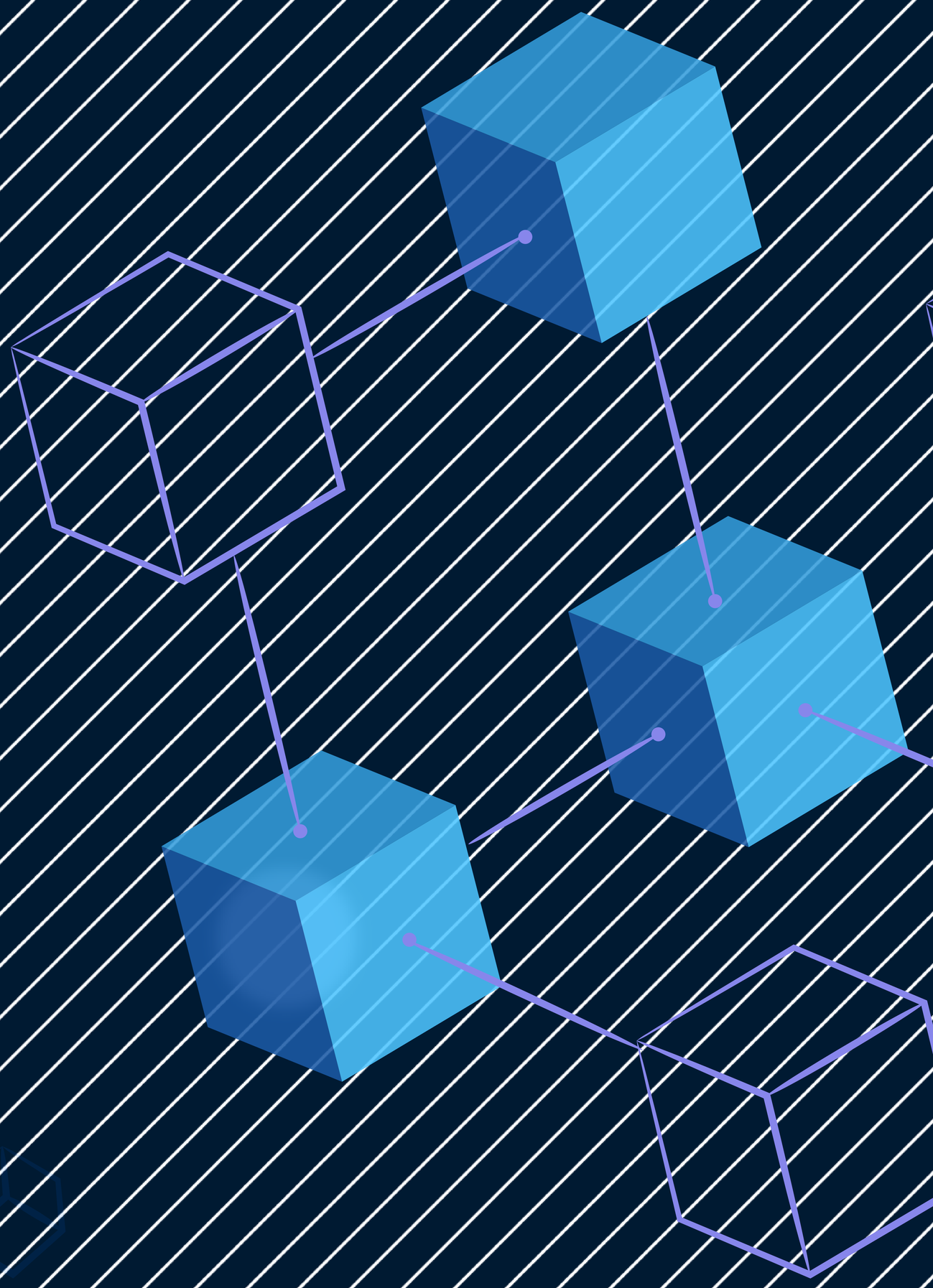
QuillAudits

Audit Report
October, 2021



For

ACKNOLEDGER

Contents

Contents

Scope of the Audit

The scope of this audit was to analyze and document the AcknoLedger smart contract codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis
Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis
Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption
In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis, Theo.

Issue Categories

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

Risk-level	Description
High	A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.
Medium	The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.
Low	Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.
Informational	These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	0	1	1	2
Closed	0	0	1	2

Introduction

During the period of **October 26, 2021, to October 27, 2021** - QuillAudits Team performed a security audit for **AcknoLedger** smart contracts.

The code for the audit was taken from following the official link: https://github.com/DefiWizard/acknoledger-contracts-antisniperbot/

Ver.	Date	Commit hash	Files
1	Oct 26	123cb6463ad2d5826e0 77c87341ce2d391715f8b	1. AcknoLedgerToken.sol 2. UsingLiquidityProtectionService
2	Oct 27	86657715555b6b834270 458437b09b6c3aa5a9ec	3. AcknoLedgerToken.sol 4. UsingLiquidityProtectionService

Issues Found

A. Contract – AcknoLedgerToken

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

No issues were found.

Informational issues

A.1 Renounce Ownership

Line	Code
5-6	import "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol"; import "@openzeppelin/contracts/utils/math/SafeMath.sol";

Description
The imported contracts are not used. It's better to remove these imports.

Remediation
Remove the unused imports.

Status: Fixed
In version 2, the team fixed the issue with the recommended changes.

B. Contract – UsingLiquidityProtectionService

High severity issues

No issues were found.

Medium severity issues

B.1 Centralization Risks

Description

The role owner has the authority to :

- Transfer funds of blocked addresses to another address
- unblock holders
- Turn off the liquidity protection

Remediation

We advise the client to handle the governance account carefully to avoid any potential hack. We also advise the client to consider the following solutions:

- Time-lock with reasonable latency for community awareness on privileged operations;
- Multisig with community-voted 3rd-party independent co-signers;
- DAO or Governance module increasing transparency and community involvement;

Status: Acknowledged by the Auditee

Low severity issues

B.2 Missing Zero Address check

Line	Code

37-39	function LiquidityProtection_setLiquidityProtectionService(IPLPS _plps) external onlyProtectionAdmin() {
	plps = _plps;
	}

	function revokeBlocked(address[] calldata _holders, address _revokeTo) external onlyProtectionAdmin() {
	require(isProtected(), 'UsingLiquidityProtectionService: protection removed');
	bool unProtectedOld = unProtected;
	unProtected = true;
	address pool = getLiquidityPool();
	for (uint i = 0; i < _holders.length; i++) {
	address holder = _holders[i];
	if (lps().isBlocked(pool, holder)) {
	token_transfer(holder, _revokeTo, token_balanceOf(holder));
	}
	}
	unProtected = unProtectedOld;
	}

Description

Certain functions lack a safety check in the address, the address-type argument should include a zero-address test, otherwise, the contract's functionality may become inaccessible or tokens may be burned in perpetuity.

Remediation

It’s recommended to undertake further validation prior to user-supplied data. The concerns can be resolved by using require() checks on the parameter.

Status: Partially Fixed

In version 2, the team fixed one of the functions with the recommended changes.

B.3 Unreadable code

Line	Code
17	contract AcknoLedgerToken is ERC20Permit, Ownable, UsingLiquidityProtectionService(0x9Ce6edF92a34ec4ee9311d9518c11Ee164b998CC){

52-55	function _beforeTokenTransfer(address _from, address _to, uint _amount) internal override {
	super._beforeTokenTransfer(_from, _to, _amount);
	LiquidityProtection_beforeTokenTransfer(_from, _to, _amount);
	}

Description

All the token transfers before **Saturday, November 27, 2021 11:59:59 PM GMT** will be processed by the contract 0x9Ce6edF92a34ec4ee9311d9518c11Ee164b998CC. The code for this contract is not verified and thus cannot be audited properly.

Remediation

It's recommended to verify the contract on the explorer.

Status: Acknowledged by the Auditee

Informational issues

B.4 Floating pragma

Line	Code
2	pragma solidity ^0.8.0;

Description

The contract makes use of the floating-point pragma ^0.8.0 Contracts should be deployed using the same compiler version and flags that were used during the testing process. Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version that may introduce issues in the contract system.

Remediation
Consider locking the pragma version.

Status: Fixed
In version 2, the team fixed the issue with the recommended changes.

B.5 External Calls Inside a Loop

Line	Code

75-80	for (uint i = 0; i < _holders.length; i++) {
	address holder = _holders[i];
	if (lps().isBlocked(pool, holder)) {
	token_transfer(holder, _revokeTo, token_balanceOf(holder));
	}
	}

Description
The loop inside the revokeBlocked() function loops over the array _holders and for every element, an external call to the LPS contract is made. This can lead to out-of-gas errors or call failures depending on the size of the array and logic of the external contract.

Remediation
We recommend having an upper bound on the size of the _holders array passed to the function, and checking the logic of the external calls.

Status: Acknowledged by the Auditee

B.6 Redundant variable initialization

Line	Code
9	bool private unProtected = false;

Description

If a variable is not set/initialized, it is assumed to have the default value (0, false, 0x0 etc depending on the data type). If you explicitly initialize it with its default value, you are just wasting gas.
Here the boolean **unProtected** will have the default value of false.

Remediation

We recommend not to initialize the variable with false.

Status: Acknowledged by the Auditee

Functional test

Function Names	Testing results
name()	Passed
symbol()	Passed
decimals()	Passed
totalSupply()	Passed
balanceOf()	Passed
transfer()	Passed
allowance()	Passed
approve()	Passed
transferFrom()	Passed
increaseAllowance()	Passed
decreaseAllowance()	Passed
setGovernance()	Passed
recoverToken()	Passed
permit()	Passed
nonces()	Passed
LiquidityProtection_setLiquidityProtectionService()	Passed
revokeBlocked()	Passed
LiquidityProtection_unblock()	Passed

Function Names	Testing results
disableProtection()	Passed
isProtected()	Passed
getLiquidityPool()	Passed

Automated Tests

Unit Tests:

```
  ACKTokenWithProtection
LP:  0x7A470674C2D976878C76E6188a890adFBd98A6A9
    Deployment
      ✓ name should be AcknoLedger
      ✓ symbol should be ACK
      ✓ should have 18 decimals
      ✓ total supply should be 117718487000000000000000000 tokens
    allowance
      ✓ allowance works as expected (326ms)
    approve
      ✓ cannot approve the zero address to move your tokens (86ms)
    transferFrom
      ✓ allows you transfer an address' tokens to another address (1718ms)
    Liquidity Protection Service
      ✓ Should have valid initial distribution (1964ms)
      ✓ Should trap all buyers in the first block (4633ms)
      ✓ Should trap all buyers who bought above amount limit (3035ms)
      ✓ Should trap all buyers who bought above percent limit (3457ms)
      ✓ Should trap all traders if more than 8 trades in the second block (3342ms)
      ✓ Should trap all traders if more than 8 trades in the third block, but do not trap second block traders (4788ms)
      ✓ Should not trap traders after activity blocks passed (3489ms)
      ✓ Should disable protection (1868ms)


  15 passing (40s)
```

Slither:

```
Reentrancy in AcknoLedgerToken.recoverToken(address,address,uint256) (AcknoLedgerToken.sol#88-97):
        External calls:
        - require(bool,string)(IERC20(token).transfer(destination,amount),Retrieve failed) (AcknoLedgerToken.sol#95)
        Event emitted after the call(s):
        - RecoverToken(token,destination,amount) (AcknoLedgerToken.sol#96)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

ERC20Permit.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (ERC20Permit.sol#40-61) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= deadline,Permit: expired deadline) (ERC20Permit.sol#49)
UsingLiquidityProtectionService.passed(uint256) (UsingLiquidityProtectionService.sol#114-116) uses timestamp for comparisons
        Dangerous comparisons:
        - _timestamp < block.timestamp (UsingLiquidityProtectionService.sol#115)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

ERC20Permit.constructor() (ERC20Permit.sol#19-34) uses assembly
        - INLINE ASM (ERC20Permit.sol#21-23)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.8.9', '>=0.4.22<0.9.0', '^0.8.0']
        - 0.8.9 (AcknoLedgerToken.sol#3)
        - ^0.8.0 (@openzeppelin\contracts\utils\Context.sol#3)
        - ^0.8.0 (@openzeppelin\contracts\utils\Counters.sol#3)
        - ^0.8.0 (@openzeppelin\contracts\token\ERC20\ERC20.sol#3)
        - 0.8.9 (ERC20Permit.sol#3)
        - ^0.8.0 (@openzeppelin\contracts\token\ERC20\IERC20.sol#3)
        - ^0.8.0 (@openzeppelin\contracts\token\ERC20\extensions\IERC20Metadata.sol#3)
        - 0.8.9 (IERC2612Permit.sol#3)
        - ^0.8.0 (IPLPS.sol#2)
        - >=0.4.22<0.9.0 (Migrations.sol#2)
        - ^0.8.0 (@openzeppelin\contracts\access\Ownable.sol#3)
        - ^0.8.0 (external\UniswapV2Library.sol#2)
        - ^0.8.0 (external\UniswapV3Library.sol#2)
        - ^0.8.0 (UsingLiquidityProtectionService.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

UsingLiquidityProtectionService.ProtectionSwitch_block(uint256) (UsingLiquidityProtectionService.sol#106-108) is never used and should
be removed
UsingLiquidityProtectionService.ProtectionSwitch_manual() (UsingLiquidityProtectionService.sol#98-100) is never used and should be remo
ved
UsingLiquidityProtectionService.blockPassed(uint256) (UsingLiquidityProtectionService.sol#110-112) is never used and should be removed
UsingLiquidityProtectionService.counterToken() (UsingLiquidityProtectionService.sol#47-49) is never used and should be removed
UsingLiquidityProtectionService.protectionAdminCheck() (UsingLiquidityProtectionService.sol#43) is never used and should be removed
UsingLiquidityProtectionService.protectionChecker() (UsingLiquidityProtectionService.sol#53-55) is never used and should be removed
UsingLiquidityProtectionService.token_balanceOf(address) (UsingLiquidityProtectionService.sol#42) is never used and should be removed
UsingLiquidityProtectionService.token_transfer(address,address,uint256) (UsingLiquidityProtectionService.sol#41) is never used and shou
ld be removed
UsingLiquidityProtectionService.uniswapFactory() (UsingLiquidityProtectionService.sol#46) is never used and should be removed
UsingLiquidityProtectionService.uniswapVariety() (UsingLiquidityProtectionService.sol#44) is never used and should be removed
UsingLiquidityProtectionService.uniswapVersion() (UsingLiquidityProtectionService.sol#45) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

```
Pragma version0.8.9 (AcknoLedgerToken.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin\contracts\utils\Context.sol#3) necessitates a version too recent to be trusted. Consider deploying
with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin\contracts\utils\Counters.sol#3) necessitates a version too recent to be trusted. Consider deploying
 with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin\contracts\token\ERC20\ERC20.sol#3) necessitates a version too recent to be trusted. Consider deploy
ing with 0.6.12/0.7.6
Pragma version0.8.9 (ERC20Permit.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin\contracts\token\ERC20\IERC20.sol#3) necessitates a version too recent to be trusted. Consider deplo
ying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin\contracts\token\ERC20\extensions\IERC20Metadata.sol#3) necessitates a version too recent to be trus
ted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.9 (IERC2612Permit.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (IPLPS.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version>=0.4.22<0.9.0 (Migrations.sol#2) is too complex
Pragma version^0.8.0 (@openzeppelin\contracts\access\Ownable.sol#3) necessitates a version too recent to be trusted. Consider deploying
 with 0.6.12/0.7.6
Pragma version^0.8.0 (external\UniswapV2Library.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/
0.7.6
Pragma version^0.8.0 (external\UniswapV3Library.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/
0.7.6
Pragma version^0.8.0 (UsingLiquidityProtectionService.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0
.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function AcknoLedgerToken.token_transfer(address,address,uint256) (AcknoLedgerToken.sol#36-38) is not in mixedCase
Parameter AcknoLedgerToken.token_transfer(address,address,uint256)._from (AcknoLedgerToken.sol#36) is not in mixedCase
Parameter AcknoLedgerToken.token_transfer(address,address,uint256)._to (AcknoLedgerToken.sol#36) is not in mixedCase
Parameter AcknoLedgerToken.token_transfer(address,address,uint256)._amount (AcknoLedgerToken.sol#36) is not in mixedCase
Function AcknoLedgerToken.token_balanceOf(address) (AcknoLedgerToken.sol#39-41) is not in mixedCase
Parameter AcknoLedgerToken.token_balanceOf(address)._holder (AcknoLedgerToken.sol#39) is not in mixedCase
Parameter AcknoLedgerToken.setGovernance(address)._governance (AcknoLedgerToken.sol#75) is not in mixedCase
Variable ERC20Permit.DOMAIN_SEPARATOR (ERC20Permit.sol#17) is not in mixedCase
Function IPLPS.LiquidityProtection_beforeTokenTransfer(address,address,address,uint256) (IPLPS.sol#5-6) is not in mixedCase
Variable Migrations.last_completed_migration (Migrations.sol#6) is not in mixedCase
Function UsingLiquidityProtectionService.LiquidityProtection_setLiquidityProtectionService(IPLPS) (UsingLiquidityProtectionService.sol#
37-39) is not in mixedCase
Parameter UsingLiquidityProtectionService.LiquidityProtection_setLiquidityProtectionService(IPLPS)._plps (UsingLiquidityProtectionServi
ce.sol#37) is not in mixedCase
Function UsingLiquidityProtectionService.token_transfer(address,address,uint256) (UsingLiquidityProtectionService.sol#41) is not in mix
edCase
Function UsingLiquidityProtectionService.token_balanceOf(address) (UsingLiquidityProtectionService.sol#42) is not in mixedCase
Function UsingLiquidityProtectionService.LiquidityProtection_beforeTokenTransfer(address,address,uint256) (UsingLiquidityProtectionServ
ice.sol#61-68) is not in mixedCase
Parameter UsingLiquidityProtectionService.LiquidityProtection_beforeTokenTransfer(address,address,uint256)._from (UsingLiquidityProtect
ionService.sol#61) is not in mixedCase
Parameter UsingLiquidityProtectionService.LiquidityProtection_beforeTokenTransfer(address,address,uint256)._to (UsingLiquidityProtectio
nService.sol#61) is not in mixedCase
Parameter UsingLiquidityProtectionService.LiquidityProtection_beforeTokenTransfer(address,address,uint256)._amount (UsingLiquidityProte
ctionService.sol#61) is not in mixedCase
```

```
Parameter UsingLiquidityProtectionService.revokeBlocked(address[],address)._holders (UsingLiquidityProtectionService.sol#70) is not in
mixedCase
Parameter UsingLiquidityProtectionService.revokeBlocked(address[],address)._revokeTo (UsingLiquidityProtectionService.sol#70) is not in
 mixedCase
Function UsingLiquidityProtectionService.LiquidityProtection_unblock(address[]) (UsingLiquidityProtectionService.sol#84-88) is not in m
ixedCase
Parameter UsingLiquidityProtectionService.LiquidityProtection_unblock(address[])._holders (UsingLiquidityProtectionService.sol#84) is n
ot in mixedCase
Function UsingLiquidityProtectionService.ProtectionSwitch_manual() (UsingLiquidityProtectionService.sol#98-100) is not in mixedCase
Function UsingLiquidityProtectionService.ProtectionSwitch_timestamp(uint256) (UsingLiquidityProtectionService.sol#102-104) is not in mi
xedCase
Parameter UsingLiquidityProtectionService.ProtectionSwitch_timestamp(uint256)._timestamp (UsingLiquidityProtectionService.sol#102) is n
ot in mixedCase
Function UsingLiquidityProtectionService.ProtectionSwitch_block(uint256) (UsingLiquidityProtectionService.sol#106-108) is not in mixedC
ase
Parameter UsingLiquidityProtectionService.ProtectionSwitch_block(uint256)._block (UsingLiquidityProtectionService.sol#106) is not in mi
xedCase
Parameter UsingLiquidityProtectionService.blockPassed(uint256)._block (UsingLiquidityProtectionService.sol#110) is not in mixedCase
Parameter UsingLiquidityProtectionService.passed(uint256)._timestamp (UsingLiquidityProtectionService.sol#114) is not in mixedCase
Parameter UsingLiquidityProtectionService.not(bool)._condition (UsingLiquidityProtectionService.sol#118) is not in mixedCase
Parameter UsingLiquidityProtectionService.feeToUint24(UsingLiquidityProtectionService.UniswapV3Fees)._fee (UsingLiquidityProtectionServ
ice.sol#122) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

UsingLiquidityProtectionService.HUNDRED_PERCENT (UsingLiquidityProtectionService.sol#11) is never used in AcknoLedgerToken (AcknoLedger
Token.sol#17-98)

UsingLiquidityProtectionService.QUICKSWAP (UsingLiquidityProtectionService.sol#14) is never used in AcknoLedgerToken (AcknoLedgerToken.
sol#17-98)
UsingLiquidityProtectionService.SUSHISWAP (UsingLiquidityProtectionService.sol#15) is never used in AcknoLedgerToken (AcknoLedgerToken.
sol#17-98)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

symbol() should be declared external:
        - ERC20.symbol() (@openzeppelin\contracts\token\ERC20\ERC20.sol#68-70)
decimals() should be declared external:
        - ERC20.decimals() (@openzeppelin\contracts\token\ERC20\ERC20.sol#85-87)
totalSupply() should be declared external:
        - ERC20.totalSupply() (@openzeppelin\contracts\token\ERC20\ERC20.sol#92-94)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#111-114)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (@openzeppelin\contracts\token\ERC20\ERC20.sol#119-121)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#130-133)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#148-162)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#176-179)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#195-203)
permit(address,address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - ERC20Permit.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (ERC20Permit.sol#40-61)

nonces(address) should be declared external:
        - ERC20Permit.nonces(address) (ERC20Permit.sol#66-68)
setCompleted(uint256) should be declared external:
        - Migrations.setCompleted(uint256) (Migrations.sol#16-18)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (@openzeppelin\contracts\access\Ownable.sol#53-55)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (@openzeppelin\contracts\access\Ownable.sol#61-64)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
. analyzed (14 contracts with 75 detectors), 79 result(s) found
```

Results

No major issues were found. Some false positive errors were reported by the tool. All the other issues have been categorized above according to their level of severity.

Closing Summary

Overall, smart contracts are very well written and adhere to guidelines.

No instances of Integer Overflow and Underflow vulnerabilities or Back-Door Entry were found in the contract.

Disclaimer

Quillhash audit is not a security warranty, investment advice, or endorsement of the **AcknoLedger** platform. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the **AcknoLedger** Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Audit Report
October, 2021

For

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com