



# SOLIDProof

*Bring trust into your projects*

**Blockchain Security | Smart Contract Audits | KYC  
Development | Marketing**

MADE IN GERMANY

## Heart's Law

# AUDIT

SECURITY ASSESSMENT

**11. December, 2024**

FOR



[SolidProof.io](https://solidproof.io)



[@solidproof\\_io](https://t.me/solidproof_io)

Introduction	4
Disclaimer	4
Project Overview	5
Summary	5
Social Medias	6
Audit Summary	7
File Overview	8
Imported packages	9
Audit Information	10
Vulnerability & Risk Level	10
Auditing Strategy and Techniques Applied	11
Methodology	11
Overall Security	12
Upgradeability	12
Ownership	13
Ownership Privileges	14
Minting tokens	14
Burning Tokens without Allowance	15
Blacklist addresses	16
Fees and Tax	17
Lock User Funds	18
Components	19
Exposed Functions	19
StateVariables	19
Capabilities	19
Inheritance Graph	21
Centralization Privileges	22
Audit Results	24
Critical issues	24
High issues	24



Medium issues	24
Low issues	26
Informational issues	27





## Introduction

[SolidProof.io](https://solidproof.io) is a brand of the officially registered company Future Visions Deutschland. We're mainly focused on Blockchain Security, such as Smart Contract Audits and KYC verification for project teams.

Solidproof.io assesses potential security issues in the smart contracts implementations, reviews for potential inconsistencies between the code base and the whitepaper/documentation, and provides suggestions for improvement.

## Disclaimer

[SolidProof.io](https://solidproof.io) reports are not, nor should they be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should they be considered, an indication of the economics or value of any “product” or “asset” created by any team. SolidProof.io does not cover testing or auditing the integration with external contracts or services (such as Unicrypt, Uniswap, PancakeSwap, etc.).

**SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analysed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.**

SolidProof.io Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. SolidProof's position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of the security or functionality of the technology we agree to analyse.

# Project Overview

## Summary

Project Name	Heart's Law
Website	<a href="http://www.heartslaw.com">www.heartslaw.com</a>
About the project	HLAW offers an alternative to native PulseX yield farming, providing users the opportunity to earn additional dividends alongside the stable yields that current DAI-WPLS farmers are accustomed to.
Chain	Pulse chain
Language	Solidity
Codebase Link	<p>HLAWToken: <a href="https://scan.pulsechainfoundation.org/#/address/0xd197D8ECCA9b5B35d5f3b42cb874E2819f315224">https://scan.pulsechainfoundation.org/#/address/0xd197D8ECCA9b5B35d5f3b42cb874E2819f315224</a></p> <p>HLAWExchange: <a href="https://scan.pulsechainfoundation.org/#/address/0xa224C402Edf26E3027416d85f25d4D7B5b43E015">https://scan.pulsechainfoundation.org/#/address/0xa224C402Edf26E3027416d85f25d4D7B5b43E015</a></p> <p>HLAWIncentiveRewardPool: <a href="https://scan.pulsechainfoundation.org/#/address/0xbF2D4eA035c1b6530f14B4cD63ef9528Ee0871A2">https://scan.pulsechainfoundation.org/#/address/0xbF2D4eA035c1b6530f14B4cD63ef9528Ee0871A2</a></p> <p>HLAWInstantRewardPool: <a href="https://scan.pulsechainfoundation.org/#/address/0xb37D077e5e6A2720503809b6fAf076D78867EC32">https://scan.pulsechainfoundation.org/#/address/0xb37D077e5e6A2720503809b6fAf076D78867EC32</a></p> <p>HLAWPrizePool: <a href="https://scan.pulsechainfoundation.org/#/address/0x873debA13845E6f79542b753f6CC4B2EbE861D80">https://scan.pulsechainfoundation.org/#/address/0x873debA13845E6f79542b753f6CC4B2EbE861D80</a></p> <p>HLAWReferral: <a href="https://scan.pulsechainfoundation.org/#/address/0xd910E9310c10664A2493670617974CF80Ca5489D">https://scan.pulsechainfoundation.org/#/address/0xd910E9310c10664A2493670617974CF80Ca5489D</a></p> <p>HLAWRewardPool: <a href="https://scan.pulsechainfoundation.org/#/address/0x1225e657844584f623C0cC6896Ca9e56Fc6fB6b5">https://scan.pulsechainfoundation.org/#/address/0x1225e657844584f623C0cC6896Ca9e56Fc6fB6b5</a></p> <p>HLAWStaking: <a href="https://scan.pulsechainfoundation.org/#/address/0x55d1C8e1F854f12F47F3FB98268fF883FA7369D6">https://scan.pulsechainfoundation.org/#/address/0x55d1C8e1F854f12F47F3FB98268fF883FA7369D6</a></p> <p>HLAWBoost: <a href="https://scan.pulsechainfoundation.org/#/address/0x0e45b05D2d32a94650124F08576Abae86f104971">https://scan.pulsechainfoundation.org/#/address/0x0e45b05D2d32a94650124F08576Abae86f104971</a></p> <p>HLAWZapper: <a href="https://scan.pulsechainfoundation.org/#/address/0x9B24A6Bf2A27ca8b4Af226A9c7e0F80a718deEba">https://scan.pulsechainfoundation.org/#/address/0x9B24A6Bf2A27ca8b4Af226A9c7e0F80a718deEba</a></p>
Commit	N/A
Unit Tests	Not Provided



## Social Medias

<b>Telegram</b>	<a href="https://t.me/hlawchat">https://t.me/hlawchat</a>
<b>Twitter</b>	<a href="https://x.com/hlawdefi">https://x.com/hlawdefi</a>
<b>Facebook</b>	N/A
<b>Instagram</b>	N/A
<b>Github</b>	N/A
<b>Reddit</b>	N/A
<b>Medium</b>	N/A
<b>Discord</b>	N/A
<b>Youtube</b>	N/A
<b>TikTok</b>	N/A
<b>LinkedIn</b>	N/A

## Audit Summary

Version	Delivery Date	Changelog
v1.0	22. October 2024	<ul style="list-style-type: none"> <li>• Layout Project</li> <li>• Automated- /Manual-Security Testing</li> <li>• Summary</li> </ul>
v1.4	11. December 2024	<ul style="list-style-type: none"> <li>• Reaudit</li> </ul>

**Note** - The following audit report presents a comprehensive security analysis of the smart contract utilized in the project that includes malicious outside manipulation of the contract's functions. This analysis did not include functional testing (or unit testing) of the contract/s logic. We cannot guarantee 100% logical correctness of the contract as we did not functionally test it. This includes internal calculations in the formulae used in the contract.



## File Overview

The Team provided us with the files that should be tested in the security assessment. This audit covered the following files listed below with an SHA-1 Hash.

File Name	SHA-1 Hash
HLAWExchange.sol	54485a7523e714191a8f85fbca1c681d44701717
HLAWToken.sol	a7749cb4fa42a6e155ebffb2e38efe703e455166
HLAWZapper.sol	39477141592a08455b0c9c681d561cfb522f5e3d
HLAWReferral.sol	1bc0bff3a9dd3f86447ce8d6acb12dbd5c400a1d
HLAWIncentiveRewardPool.sol	a28f0fab316315e1d18e843eb515199fc31b353b
HLAWPrizePool.sol	e6e5e0d018593bea3cca914859c60eef25af48cd
HLAWBoost.sol	7fc2e2149d6537d66b727f7119460c8aad1e93a6
HLAWInstantRewardPool.sol	4dc67da41236bee72b0e14831662369522a70b53
HLAWRewardPool.sol	cd8b755b2d1e0107332729cafa45755f4d2bbbee0
HLAWStaking.sol	8734f99a226d777100c73325cc32f45cb198e229

*Please note: Files with a different hash value than in this table have been modified after the security check, either intentionally or unintentionally. A different hash value may (but need not) indicate a changed state or potential vulnerability that was not the subject of this scan. HLAWExchange and HLAWStaking contract is not verified by the project owner. Hence, we have to verify the byte code of the contract so if any changes made in these contract without any explanation the team will not be responsible and it is recommended to do research before investing.*





## Imported packages

*Used code from other Frameworks/Smart Contracts (direct imports).*

Dependency / Import Path	Count
@openzeppelin/contracts/access/Ownable.sol	10
@openzeppelin/contracts/token/ERC20/ERC20.sol	1
@openzeppelin/contracts/token/ERC20/IERC20.sol	7
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	2
@openzeppelin/contracts/utils/Address.sol	6
@openzeppelin/contracts/utils/ReentrancyGuard.sol	4
@openzeppelin/contracts/utils/math/Math.sol	1
@openzeppelin/contracts/utils/structs/EnumerableSet.sol	1

**Note for Investors:** We only audited contracts mentioned in the scope above. All contracts related to the project apart from that are not a part of the audit, and we cannot comment on its security and are not responsible for it in any way

## Audit Information

### Vulnerability & Risk Level

Risk represents the probability that a certain source threat will exploit vulnerability and the impact of that event on the organization or system. The risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
<b>Critical</b>	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
<b>High</b>	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
<b>Medium</b>	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
<b>Low</b>	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
<b>Informational</b>	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

## Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to check the repository for security-related issues, code quality, and compliance with specifications and best practices. To this end, our team of experienced pen-testers and smart contract developers reviewed the code line by line and documented any issues discovered.

We check every file manually. We use automated tools only so that they help us achieve faster and better results.

## Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
  - a. Review the specifications, sources, and instructions provided to SolidProof to ensure we understand the smart contract's size, scope, and functionality.
  - b. Manual review of the code, i.e., reading the source code line by line to identify potential vulnerabilities.
  - c. Comparison to the specification, i.e., verifying that the code does what is described in the specifications, sources, and instructions provided to SolidProof.
2. Testing and automated analysis that includes the following:
  - a. Test coverage analysis determines whether test cases cover code and how much code is executed when those test cases are executed.
  - b. Symbolic execution is analysing a program to determine what inputs cause each part of a program to execute.
3. Review best practices, i.e., smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on best practices, recommendations, and research from industry and academia.
4. Concrete, itemized and actionable recommendations to help you secure your smart contracts.



## Overall Security

### Upgradeability

**Contract is not an upgradeable**



**Deployer cannot update the contract with new functionalities**

Description

The contract is not an upgradeable contract. The deployer is not able to change or add any functionalities to the contract after deploying.

Comment

N/A



## Ownership

The ownership is not renounced		✗ The ownership is not renounced
Description	<p>The owner has not renounced the ownership that means that the owner retains control over the contract's operations, including the ability to execute functions that may impact the contract's users or stakeholders. This can lead to several potential issues, including:</p> <ul style="list-style-type: none"><li>• Centralizations</li><li>• The owner has significant control over contract's operations</li></ul>	
Example	<p>We assume that you have funds in the contract and it has been audited by any security audit firm. Now the audit has passed. After that, the deployer can upgrade the contract to allow him to transfer the funds you purchased without any approval from you. This has the consequence that your funds can be taken by the creator.</p>	
Comment	N/A	

**Note** - If the contract is not deployed then we would consider the ownership to be not renounced. Moreover, if there are no ownership functionalities then the ownership is automatically considered renounced.



## Ownership Privileges

*These functions can be dangerous. Please note that abuse can lead to financial loss. We have a guide where you can learn more about these Functions.*

### Minting tokens

*Minting tokens refers to the process of creating new tokens in a cryptocurrency or blockchain network. This process is typically performed by the project's owner or designated authority, who can add new tokens to the network's total supply.*


Contract owner cannot mint new tokens <span>✓ The owner cannot mint new tokens</span>	
Description	The owner is not able to mint new tokens once the contract is deployed.
Comment	The minting is restricted to the hlawExchange, allowing only this contract to create new tokens as needed, ensuring that token supply can be managed in line with business logic or exchange operations.



## Burning Tokens without Allowance

*Burning tokens is the process of permanently destroying a certain number of tokens, reducing the total supply of a cryptocurrency or token. This is usually done to increase the value of the remaining tokens, as the reduced supply can create scarcity and potentially drive up demand.*

**Contract owner cannot burn tokens**

 **The owner cannot burn tokens**

Description	The owner is not able to burn tokens without any allowances.
Comment	The burning is restricts the ability to destroy tokens to the hlawExchange, preventing unauthorized parties from reducing the supply of tokens in circulation.



## Blacklist addresses

*Blacklisting addresses in smart contracts is the process of adding a certain address to a blacklist, effectively preventing them from accessing or participating in certain functionalities or transactions within the contract. This can be useful in preventing fraudulent or malicious activities, such as hacking attempts or money laundering.*

**Contract owner cannot blacklist addresses**



**The owner cannot blacklist addresses**

Description

The owner is not able to blacklist addresses to lock funds.

Comment

N/A





## Fees and Tax

*In some smart contracts, the owner or creator of the contract can set fees for certain actions or operations within the contract. These fees can be used to cover the contract's cost, such as paying for gas fees or compensating the contract's owner for their time and effort in developing and maintaining the contract.*

**Contract owner cannot set fees more than 25%**

☒ **The owner cannot levy unfair taxes**

Description	The owner is not able to set the fees above 25%
-------------	---

Comment	N/A
---------	-----



## Lock User Funds

*In a smart contract, locking refers to the process of restricting access to certain tokens or assets for a specified period of time. When tokens or assets are locked in a smart contract, they cannot be transferred or used until the lock-up period has expired or certain conditions have been met.*

### Owner cannot lock the contract

 The owner cannot lock the contract

Description	The owner is not able to lock the contract by any functions or updating any variables.
-------------	--

Comment	N/A
---------	-----

## External/Public functions

External/public functions are functions that can be called from outside of a contract, i.e., they can be accessed by other contracts or external accounts on the blockchain. These functions are specified using the function declaration's external or public visibility modifier.

## State variables

State variables are variables that are stored on the blockchain as part of the contract's state. They are declared at the contract level and can be accessed and modified by any function within the contract. State variables can be defined with a visibility modifier, such as public, private, or internal, which determines the access level of the variable.

## Components

 <b>Contracts</b>	 <b>Libraries</b>	 <b>Interfaces</b>	 <b>Abstract</b>
10	1	15	0


## Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

 <b>Public</b>	 <b>Payable</b>
121	6

External	Internal	Private	Pure	View
88	145	0	16	30

## StateVariables

<b>Total</b>	 <b>Public</b>
105	58



0.8.20 ^0.8.20		Yes		
-------------------	--	-----	--	--

 <b>Transfers ETH</b>	 <b>Low-Level Calls</b>	 <b>Delegation Call</b>	 <b>Uses Hash Functions</b>	 <b>ECRe cover</b>	 <b>New/Create/Create2</b>
yes					



*An inheritance graph is a graphical representation of the inheritance hierarchy among contracts. In object-oriented programming, inheritance is a mechanism that allows one class (or contract, in the case of Solidity) to inherit properties and methods from another class. It shows the relationships between different contracts and how they are related to each other through inheritance.*



## Centralization Privileges

*Centralization can arise when one or more parties have privileged access or control over the contract's functionality, data, or decision-making. This can occur, for example, if a single entity controls the contract or if certain participants have special permissions or abilities that others do not.*

In the project, some authorities have access to the following functions:

File	Privileges
HLAWBoost.sol	<ul style="list-style-type: none"> <li>The signer can distribute boost.</li> <li>The owner can update the signers in the contract.</li> <li>The owner can update the distribution time to any arbitrary value in the contract.</li> </ul>
HLAWExchange.sol	<ul style="list-style-type: none"> <li>The owner can initialize the contract only once.</li> <li>The owner can update the fees distribution percentage in the contract.</li> <li>The owner can update the total buy and sell fees to not more than 10%.</li> <li>The owner can update the treasury fee received prize fee receiver address.</li> <li>The owner can update any arbitrary value in the reference stake amount required.</li> </ul>
HLAWIncRewardPool.sol	<ul style="list-style-type: none"> <li>The owner can initialize the the contract only once.</li> </ul>
HLAWInstantRewardPool.sol	<ul style="list-style-type: none"> <li>The owner can initialize the the contract only once.</li> </ul>
HLAWPrizePool.sol	<ul style="list-style-type: none"> <li>The owner or signer can distribute the tokens to the winners.</li> <li>The owner can update the signer address in the contract.</li> <li>The owner can update the reward interval and max percentage value in the contract.</li> </ul>
HLAWReferral.sol	<ul style="list-style-type: none"> <li>The owner can initialize the the contract only once.</li> <li>The HLAW exchange contract can set referral in the contract.</li> <li>The HLAW exchange contract can add rewards to the contract.</li> </ul>
HLAWRewardPool.sol	<ul style="list-style-type: none"> <li>The owner can initialize the the contract only once.</li> </ul>
HLAWToken.sol	<ul style="list-style-type: none"> <li>The HLAWExchange contract can mint and burn the tokens.</li> </ul>
HLAWZapper.sol	<ul style="list-style-type: none"> <li>The owner can withdraw the balance from the contract.</li> <li>The owner can withdraw the tokens from the contract.</li> <li>The owner can add the allowed currencies in the contract.</li> <li>The owner can update the convenience fees to not more than 0.3% and update the treasury wallet address.</li> </ul>

File	Privileges
HLAWStaking.sol	<ul style="list-style-type: none"> <li>• The owner can update the boost contract address.</li> <li>• The owner can update the treasury address.</li> <li>• The owner can update the increment claim fees to not more than 10%.</li> <li>• The owner can update the fees distribution settings.</li> <li>• The owner can update the update interval to not less than 24 hours.</li> <li>• The owner can update the ratio between 0 to 100%.</li> <li>• The owner can update the update threshold between 0 to 100%.</li> <li>• The owner can update the signer address.</li> <li>• The owner can initialize the pool.</li> <li>• The owner can set the allocPoint, active/inactive status on pool.</li> <li>• The owner or HLA exchange contract can distribute the reward dividends.</li> <li>• The owner or HLA exchange contract can distribute the incentive dividends.</li> <li>• The boost contract can distribute tokens to the reward pool.</li> <li>• The HLA exchange contract can add users.</li> </ul>

## Recommendations

To avoid potential hacking risks, the client should manage the private key of the privileged account with care. Additionally, we recommend enhancing the security practices of centralized privileges or roles in the protocol through a decentralized mechanism or smart-contract-based accounts, such as multi-signature wallets.

Here are some suggestions of what the client can do:

- Consider using multi-signature wallets: Multi-signature wallets require multiple parties to sign off on a transaction before it can be executed, providing an extra layer of security, e.g. Gnosis Safe
- Use of a timelock at least with a latency of, e.g. 48-72 hours for awareness of privileged operations
- Introduce a DAO/Governance/Voting module to increase transparency and user involvement
- Consider Renouncing the ownership so that the owner can no longer modify any state variables of the contract. Make sure to set up everything before renouncing.

# Audit Results

## Critical issues

No critical issues

## High issues

### #1 | Incorrect validation logic.

File	Severity	Location	Status
HLAWStaking.sol	High	L375-383	Fixed

**Description** - The require statements for updateRatio and updateThreshold use the logical OR operator (||), This line attempts to validate that updateRatio is between 0 and 100. However, the logical condition used here is incorrect, which does not enforce the desired range. Instead, the correct validation should check that both conditions are true using the logical AND operator (&&).

## Medium issues

### #1 | Missing 'isContract' check.

File	Severity	Location	Status
HLAWIncentiveRewardPool.sol	Medium	L17-22	Fixed
HLAWInstantRewardPool.sol	Medium	L18-24	Fixed
HLAWReferral.sol	Medium	L38-43	Fixed
HLAWRewardPool.sol	Medium	L18-24	Fixed
HLAWStaking.sol	Medium	L353-356	Fixed

**Description** - The contract currently allows the owner to update any arbitrary address as the contract address, which can lead to potential failures if an incorrect or non-contract address is set. To mitigate this risk, a check should be added to ensure that the provided address is a valid contract address, preventing accidental or malicious errors that could disrupt the contract's functionality.

**Remediation** - Add a check that the address can only be set as the contract address.



## #2 | Missing 'nonReentrant' check.

File	Severity	Location	Status
HLAWZapper.sol	Medium	L124, 172	Fixed

**Description** - The use of call to transfer Pulse (plsRefund) to the user can introduce reentrancy issues. This happens because transferring Ether can invoke the recipient's fallback function, allowing them to call back into the contract before the state is updated. The zapTokens function you've provided is vulnerable to a reentrancy attack because it uses a low-level call to send Ether to the user without proper reentrancy protection. This can lead to loss of funds. Therefore, It is recommended to use the modifier in the contract to avoid these situation in the contract.

**Remediation** - The nonReentrant modifier prevents the function from being called recursively. So it is recommended to use in the contract.

## #3 | Imprecise Slippage calculation.

File	Severity	Location	Status
HLAWZapper.sol	Medium	327-389	ACK

**Description** - Slippage Manipulation occurs when market prices shift between the time \_amountOutMin1 is set and the swap executes, leading to an unexpectedly low return. Attackers can exploit this by front-running the transaction or manipulating liquidity, forcing a poor exchange rate that the function still accepts due to an outdated \_amountOutMin1.

**Remediation** - Implement a dynamic slippage tolerance that calculates minimum output based on live data, possibly using an oracle for accurate pricing. Alternatively, recheck the expected output immediately before the swap with getAmountsOut and revert if it falls below an acceptable threshold. This ensures swaps only execute under favorable rates.

**Alleviation** - The project owner confirms that the calculation of getAmountOut will be performed on the frontend by invoking the getAmountOut function. The result will then be adjusted by reducing it according to the user-specified slippage percentage before the transaction is submitted, following the standard approach used by other swap protocols.

## Low issues

### #1 | Missing Zero Address Validation

File	Severity	Location	Status
HLAWExchange.sol	Low	L348-359, L381-386	Fixed
HLAWBoost.sol	Low	L47-49	Fixed
HLAWPrizePool.sol	Low	L57-59	Fixed
HLAWStaking.sol	Low	L358-361, L385-388	Fixed
HLAWZapper.sol	Low	L118-122	Fixed

**Description** - Make sure to validate that the address passed in the function parameters is “non-zero”.

### #2 | Missing Events

File	Severity	Location	Status
HLAWExchange.sol	Low	L568-571	Fixed
HLAWBoost.sol	Low	L47-49, L55-57	Fixed
HLAWPrizePool.sol	Low	L57-59, L66-70	Fixed

**Description** - Make sure to emit events for all the critical parameter changes in the contract to ensure the transparency and trackability of all the state variable changes.

### #3 | Missing Threshold.

File	Severity	Location	Status
HLAWExchange.sol	Low	L388-390	Fixed
HLAWBoost.sol	Low	L55-57	Fixed
HLAWPrizePool.sol	Low	L66-70	Fixed

**Description** - The current contract allows the owner to arbitrarily update the value which can lead to unintended or excessive changes. To prevent misuse, it is recommended to implement minimum and maximum thresholds for the referral stake. This ensures that the value stays within a reasonable range and avoids setting the stake too low or excessively high.

#### #4 | Floating pragma solidity version.

File	Severity	Location	Status
HLAWExchange.sol	Low	L2	ACK

**Description** - Adding the constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.

## Informational issues

#### #1 | NatSpec documentation missing

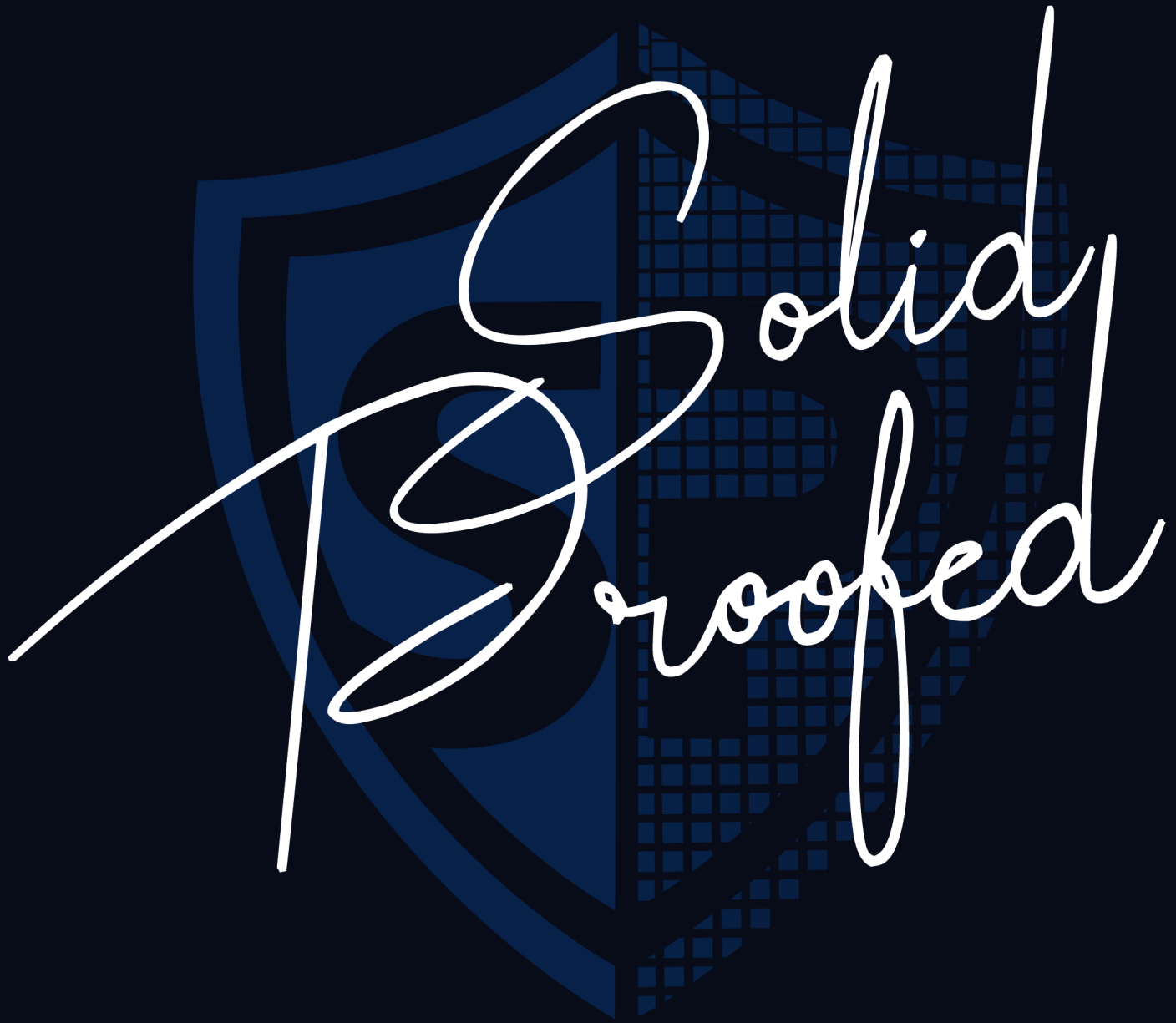
File	Severity	Location	Status
All	Informational	—	ACK

**Description** - If you started to comment on your code, comment on all other functions, variables etc.

#### #2 | State variable could be declared as constant.

File	Severity	Location	Status
HLAWExchange.sol	Informational	L62,63,64	Fixed

**Description** - It is recommended to declare variables as constant if their values are not intended to change throughout the contract's lifecycle. This practice improves code readability, gas efficiency, and helps prevent accidental modification of values that should remain fixed.



**Blockchain Security | Smart Contract Audits | KYC  
Development | Marketing**

MADE IN GERMANY



## Legend for the Issue Status

Attribute or Symbol	Meaning
<b>Open</b>	The issue is not fixed by the project team.
<b>Fixed</b>	The issue is fixed by the project team.
<b>Acknowledged(ACK)</b>	The issue has been acknowledged or declared as part of business logic.

