



# SOLIDProof

*Bring trust into your projects*

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

MADE IN GERMANY

# ezkalibur

# Audit

**Security Assessment**  
**05. May, 2023**

**For**



**SolidProof\_io**



**@solidproof\_io**

Disclaimer	3
Description	5
Project Engagement	5
Logo	5
Contract Links	5
Methodology	7
Used Code from other Frameworks/Smart Contracts (direct imports)	8
Tested Contract Files	9
Source Lines	11
Risk Level	11
Capabilities	12
Inheritance Graph	13
CallGraph	14
Scope of Work/Verify Claims	15
Modifiers and public functions	17
Source Units in Scope	19
Critical issues	20
High issues	20
Medium issues	20
Low issues	20
Informational issues	20
Audit Comments	21
SWC Attacks	22

# Disclaimer

SolidProof.io reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team. SolidProof.io do not cover testing or auditing the integration with external contract 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 analyzed, 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 security or functionality of the technology we agree to analyze.

Version	Date	Description
1.0	01. May 2023 - 03. May 2023	<ul style="list-style-type: none"><li>• Layout project</li><li>• Automated- /Manual-Security Testing</li><li>• Summary</li></ul>

**Note -** This Audit report consists of a security analysis of the **ezKalibur** smart contracts. This analysis did not include functional testing (or unit testing) of the contract’s logic.

## **Network**

zkSync

## **Website**

<https://ezkalibur.com>

## **Telegram**

<https://t.me/ezkalibur>

## **Twitter**

<https://twitter.com/eZKaliburDEX>

## **Discord**

<https://discord.com/invite/ypqHnKE5KF>



## Description

ZKalibur is the first ecosystem-focused and community-driven DEX built on zkSync Era.

We have built a highly efficient and customizable protocol, allowing both builders and users to leverage our custom infrastructure for deep, sustainable, and adaptable liquidity.

## Project Engagement

During the 30 of April 2023, **ezKalibur Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

## Logo



## Contract Links

**v1.0**

<https://github.com/eZKalibur/contractsV2/tree/main>

Commit: [4eef6b8](#)

# 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. 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 evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

## **Methodology**

The auditing process follows a routine series of steps:

1. Code review that includes the following:
  - i) Review of the specifications, sources, and instructions provided to SolidProof to make sure we understand the size, scope, and functionality of the smart contract.
  - ii) Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
  - iii) Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to SolidProof describe.
2. Testing and automated analysis that includes the following:
  - i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
  - ii) Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

## Used Code from other Frameworks/Smart Contracts (direct imports)

Imported packages:

ProtocolEarnings

```
./access/Ownable.sol  
./libraries/SafeERC20.sol  
./math/SafeMath.sol
```

Router

```
@uniswap/lib/contracts/libraries/TransferHelper.sol  
./interfaces/IFactory.sol  
./interfaces/IPair.sol  
./interfaces/IERC20.sol  
./interfaces/IRouter.sol  
./libraries/UniswapV2Library.sol  
./libraries/SafeMath.sol  
./interfaces/IWETH.sol
```

Pair

```
./interfaces/IPair.sol  
./UniswapV2ERC20.sol  
./libraries/Math.sol  
./interfaces/IERC20.sol  
./interfaces/IFactory.sol  
./interfaces/IUniswapV2Callee.sol
```

UniswapV2ERC20

```
./interfaces/IUniswapV2ERC20.sol  
./libraries/SafeMath.sol
```



## Tested Contract Files

This audit covered the following files listed below with a SHA-1 Hash.

*A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.*

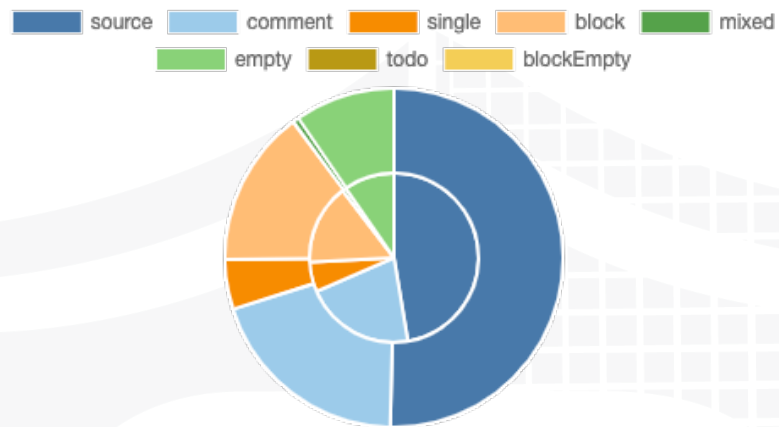
### v1.0

File Name	SHA-1 Hash
contracts/interfaces/IPair.sol	8fc7bf5881adc251370e89ed39fd9c e99a6ce404
contracts/interfaces/ IUniswapV2Callee.sol	c7e224344966e0cfad73f086da1a10 5cc8f24902
contracts/interfaces/IRouter.sol	7114df8c229216c864ca8695e1d96 5b63dc95106
contracts/interfaces/IWETH.sol	6a25dd53c8494e3aef3a520f17e006 08b529f061
contracts/interfaces/IFactory.sol	8113793dce8fa0310d1f62ffaf1d7e3 ccf8c2a0d
contracts/interfaces/ IUniswapV2ERC20.sol	a881fff951a6284f2fa04849ebda577 83de3f02a
contracts/interfaces/ IUniswapV2Router01.sol	e7a60f2a949159ba53ca018880059 9e3ba57174a
contracts/interfaces/IERC20.sol	b7d011aaabd34898ee60760996cb7 02e7b2ca855
contracts/ProtocolEarnings.sol	ea0f10adfc2a42e3d8b6695249d141 45802bae52
contracts/router.sol	35237d10acb7a31831823e2cbb920 5115bd0cb9b
contracts/math/SafeMath.sol	e8d52e78bd679fcdd64bb69f573da6 b814259f46

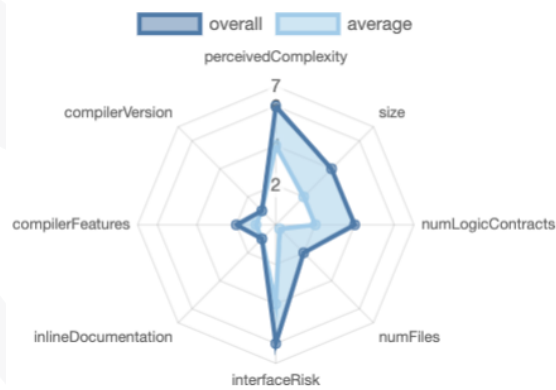
contracts/libraries/ TransferHelper.sol	b4b343678b3f894029294d85b9dc4 537f2df4a58
contracts/libraries/Math.sol	e6f63d883294ea708b0ab5ecee646f 9fcac6722c
contracts/libraries/Context.sol	02ebe0e93c5d1da25b91ba7f4cfb99 0a949263f8
contracts/libraries/Address.sol	e085bb469dc0646bfd6f9ed6a51a2e ae991b8c59
contracts/libraries/SafeMath.sol	29309c122fdd26dc3c99523f6996f6 294dfbc9cc
contracts/libraries/ UniswapV2Library.sol	fe1b38e46557c4a508d58c2f5215f4 82ac2622df
contracts/libraries/ SafeERC20.sol	364246f86ebddc2befc29388dc2319 e9bb45b1aa
contracts/libraries/Ownable.sol	32e9d02dfa308c3ef19e117c97fb3d 28523cc606
contracts/access/Ownable.sol	f86ba36d3e49e0fc30f66eaddde820 c35d11f94b
contracts/pair.sol	e34ed45a36ad60dd05c8077bbba2b a75e0a72427
contracts/factory.sol	a8d1f742446c253b4d9cfaac7d500e dab19a989f
contracts/UniswapV2ERC20.sol	7236688ab2d7f0645c7a01d625862f 2bde004f19

# Metrics

## Source Lines v1.0



## Risk Level v1.0




# Capabilities

## Components

 Contracts	 Libraries	 Interfaces	 Abstract
5	7	8	3

### Exposed Functions

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

 Public	 Payable
137	6







External	Internal	Private	Pure	View
123	165	7	37	51

### StateVariables

Total	 Public
48	42

### Capabilities

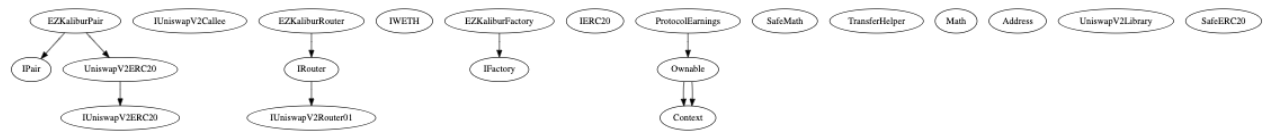
Solidity Versions observed	 Experimental Features	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts
<div><div>&gt;=0.5.0</div><div>&gt;=0.6.2</div><div>=0.7.6</div><div>=0.6.6</div><div>^0.7.0</div><div>&gt;=0.6.0</div><div>=0.5.16</div><div>&gt;=0.6.0 &lt;0.8.0</div><div>&gt;=0.5.16</div><div>^0.7.6</div></div>		<div>yes</div>	<div>yes</div> <div>(4 asm blocks)</div>	

 Transfers ETH	 Low-Level Calls	 DelegateCall	 Uses Hash Functions	 ECRRecover	 New/Create/Create2
<div>yes</div>		<div>yes</div>	<div>yes</div>	<div>yes</div>	<div>yes</div> <div>→ AssemblyCall:Name:create2</div>

 TryCatch	 Unchecked

# Inheritance Graph

## v1.0



# CallGraph v1.0



## Scope of Work/Verify Claims

The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

We will verify the following claims:

1. Overall checkup (Smart Contract Security)



## Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

### Legend

Attribute	Symbol
Verified / Checked	✓
Partly Verified	⚠
Unverified / Not checked	✗
Not available	—



# Modifiers and public functions

## v1.0

### ProtocolEarnings

```
♦ updateBuybackAndBurnWallet
Ⓜ onlyOwner
♦ updateDevelopmentFundsWallet
Ⓜ onlyOwner
♦ updateDividendsWallet
Ⓜ onlyOwner
♦ distributeShares
Ⓜ onlyOwner
♦ updateShares
Ⓜ onlyOwner
♦ safeEmergencyWithdraw
Ⓜ onlyOwner
```

### Factory

```
♦ createPair
♦ setOwner
Ⓜ onlyOwner
♦ setFeePercentOwner
Ⓜ onlyOwner
♦ setSetStableOwner
♦ setFeeTo
Ⓜ onlyOwner
♦ setOwnerFeeShare
Ⓜ onlyOwner
♦ setReferrerFeeShare
Ⓜ onlyOwner
```

### Note:

#### ❖ General fork from Camelot DEX

- Contracts inside are the same as the pancake-smart-contracts directory
  - <https://docs.camelot.exchange/contracts/amm-v2>
  - <https://docs.camelot.exchange/contracts/utilities/protocolearnings>
  - Differences between ezKalibur and Camelot contracts are the following:
    - The contracts only have name changes and the logic of all contracts remain unchanged

## Ownership Privileges

#### ❖ ProtocolEarnings.sol -

- Set/Update buyback and burn wallet
- Set/Update update development funds wallet
- Distribute shares to dividends, buybackandburning, and operating funds wallets

- Set/Update shares values but not more than 10000

❖ Factory.sol -

- Set owner
- Set fee address
- Set owner's swap fee share upto 100% which will be equivalent to 0.3 percent
- Set referrer fee share but not more than 20%

**Please check if an OnlyOwner or similar restrictive modifier has been forgotten.**



# Source Units in Scope

## v1.0

File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score
contracts/interfaces/IPair.sol	_____	1	53	7	5	_____	57
contracts/interfaces/IUniswapV2Callee.sol	_____	1	5	4	3	_____	3
contracts/interfaces/IRouter.sol	_____	1	52	6	4	_____	16
contracts/interfaces/IWETH.sol	_____	1	7	4	3	_____	10
contracts/interfaces/IFactory.sol	_____	1	22	6	4	_____	25
contracts/interfaces/IUniswapV2ERC20.sol	_____	1	23	7	5	_____	27
contracts/interfaces/IUniswapV2Router01.sol	_____	1	70	4	3	_____	22
contracts/interfaces/IERC20.sol	_____	1	17	7	5	_____	19
contracts/ProtocolEarnings.sol	1	_____	66	66	50	1	53
contracts/router.sol	1	_____	323	224	185	14	203
contracts/math/SafeMath.sol	1	_____	214	214	61	139	16
contracts/libraries/TransferHelper.sol	1	_____	51	38	28	5	26
contracts/libraries/Math.sol	1	_____	23	23	18	2	5
contracts/libraries/Context.sol	1	_____	24	24	10	12	1
contracts/libraries/Address.sol	1	_____	189	169	78	113	47
contracts/libraries/SafeMath.sol	1	_____	17	17	12	1	4
contracts/libraries/UniswapV2Library.sol	1	_____	48	48	35	5	36
contracts/libraries/SafeERC20.sol	1	_____	75	74	33	32	25
contracts/libraries/Ownable.sol	1	_____	68	68	27	33	24
contracts/access/Ownable.sol	1	_____	83	83	31	41	24
contracts/pair.sol	1	_____	426	426	336	50	330
contracts/factory.sol	1	_____	126	126	93	20	101
contracts/UniswapV2ERC20.sol	1	_____	96	96	81	1	62
<b>Totals</b>	<b>15</b>	<b>8</b>	<b>2078</b>	<b>1741</b>	<b>1110</b>	<b>469</b>	<b>1136</b>

## Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalised lines of the source unit (e.g. normalises functions spanning multiple lines)
nSLOC	normalised source lines of code (only source-code lines; no comments, no blank lines)
Comment Lines	lines containing single or block comments
Complexity Score	a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

# Audit Results

## Critical issues

**No critical issues**

## High issues

**No high issues**

## Medium issues

**No medium issues**

## Low issues

Issue	File	Type	Line	Description
#1	All	Multiple pragma is set	—	Some of the contracts contain different pragma versions which is not recommended for deployment. We recommend to have the same pragma in all contracts and also to update the old pragma versions to the new ones.
#2	Protocol Earning s.sol	Missing Events Arithmetic	28-28, 57	Emit an event for critical parameter changes
#3	All	Old Compiler Version	—	The contracts use a very old compiler version which is not recommended for deployment as it is susceptible to known vulnerabilities

## Informational issues

Issue	File	Type	Line	Description
-------	------	------	------	-------------

#1	All	Contract doesn't import npm packages from source (like OpenZeppelin etc.)	—	We recommend importing all packages from npm directly without flattening the contract. Functions could be modified or can be susceptible to vulnerabilities
----	-----	---------------------------------------------------------------------------	---	-------------------------------------------------------------------------------------------------------------------------------------------------------------

## Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information <https://docs.soliditylang.org/en/latest/natspec-format.html>) for your contracts to provide rich documentation for functions, return variables and more. This helps investors to make clear what that variables, functions etc. do.

### 05. May 2023:

- This project consists of the following forks
  - Camelot DEX
- Unit tests with 100% code coverage was not provided to SolidProof so we cannot ensure complete functional correctness of the code's logic.
- We recommend ezKalibur team to conduct unit and fuzz tests thoroughly to rule out possibilities of an unwanted logical and calculation errors.
- Read whole report and modifiers section for more information
- The low issues that remain unfixed in the CamelotDEX codebase still exist in the forked code.
- We recommend using a multisig wallet for the owner address to prevent any risk of the loss of private key
- Do your own research here

## SWC Attacks

ID	Title	Relationships	Status
<a href="#">SW C-1 36</a>	Unencrypted Private Data On-Chain	<a href="#">CWE-767: Access to Critical Private Variable via Public Method</a>	PASSED
<a href="#">SW C-1 35</a>	Code With No Effects	<a href="#">CWE-1164: Irrelevant Code</a>	PASSED
<a href="#">SW C-1 34</a>	Message call with hardcoded gas amount	<a href="#">CWE-655: Improper Initialization</a>	PASSED
<a href="#">SW C-1 33</a>	Hash Collisions With Multiple Variable Length Arguments	<a href="#">CWE-294: Authentication Bypass by Capture-replay</a>	PASSED
<a href="#">SW C-1 32</a>	Unexpected Ether balance	<a href="#">CWE-667: Improper Locking</a>	PASSED
<a href="#">SW C-1 31</a>	Presence of unused variables	<a href="#">CWE-1164: Irrelevant Code</a>	PASSED
<a href="#">SW C-1 30</a>	Right-To-Left-Override control character (U+202E)	<a href="#">CWE-451: User Interface (UI) Misrepresentation of Critical Information</a>	PASSED
<a href="#">SW C-1 29</a>	Typographical Error	<a href="#">CWE-480: Use of Incorrect Operator</a>	PASSED
<a href="#">SW C-1 28</a>	DoS With Block Gas Limit	<a href="#">CWE-400: Uncontrolled Resource Consumption</a>	PASSED

<a href="#">SW C-1 27</a>	Arbitrary Jump with Function Type Variable	<a href="#">CWE-695: Use of Low-Level Functionality</a>	<b>PASSED</b>
<a href="#">SW C-1 25</a>	Incorrect Inheritance Order	<a href="#">CWE-696: Incorrect Behavior Order</a>	<b>PASSED</b>
<a href="#">SW C-1 24</a>	Write to Arbitrary Storage Location	<a href="#">CWE-123: Write-what-where Condition</a>	<b>PASSED</b>
<a href="#">SW C-1 23</a>	Requirement Violation	<a href="#">CWE-573: Improper Following of Specification by Caller</a>	<b>PASSED</b>
<a href="#">SW C-1 22</a>	Lack of Proper Signature Verification	<a href="#">CWE-345: Insufficient Verification of Data Authenticity</a>	<b>PASSED</b>
<a href="#">SW C-1 21</a>	Missing Protection against Signature Replay Attacks	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	<b>PASSED</b>
<a href="#">SW C-1 20</a>	Weak Sources of Randomness from Chain Attributes	<a href="#">CWE-330: Use of Insufficiently Random Values</a>	<b>PASSED</b>
<a href="#">SW C-11 9</a>	Shadowing State Variables	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	<b>PASSED</b>
<a href="#">SW C-11 8</a>	Incorrect Constructor Name	<a href="#">CWE-665: Improper Initialization</a>	<b>PASSED</b>
<a href="#">SW C-11 7</a>	Signature Malleability	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	<b>PASSED</b>

<a href="#">SW C-11 6</a>	Timestamp Dependence	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	<b>PASSED</b>
<a href="#">SW C-11 5</a>	Authorization through tx.origin	<a href="#">CWE-477: Use of Obsolete Function</a>	<b>PASSED</b>
<a href="#">SW C-11 4</a>	Transaction Order Dependence	<a href="#">CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')</a>	<b>PASSED</b>
<a href="#">SW C-11 3</a>	DoS with Failed Call	<a href="#">CWE-703: Improper Check or Handling of Exceptional Conditions</a>	<b>PASSED</b>
<a href="#">SW C-11 2</a>	Delegatecall to Untrusted Callee	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	<b>PASSED</b>
<a href="#">SW C-11 1</a>	Use of Deprecated Solidity Functions	<a href="#">CWE-477: Use of Obsolete Function</a>	<b>PASSED</b>
<a href="#">SW C-11 0</a>	Assert Violation	<a href="#">CWE-670: Always-Incorrect Control Flow Implementation</a>	<b>PASSED</b>
<a href="#">SW C-1 09</a>	Uninitialized Storage Pointer	<a href="#">CWE-824: Access of Uninitialized Pointer</a>	<b>PASSED</b>
<a href="#">SW C-1 08</a>	State Variable Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	<b>PASSED</b>
<a href="#">SW C-1 07</a>	Reentrancy	<a href="#">CWE-841: Improper Enforcement of Behavioral Workflow</a>	<b>PASSED</b>
<a href="#">SW C-1 06</a>	Unprotected SELFDESTRUCT Instruction	<a href="#">CWE-284: Improper Access Control</a>	<b>PASSED</b>



<a href="#">SW</a> <a href="#">C-1</a> <a href="#">05</a>	Unprotected Ether Withdrawal	<a href="#">CWE-284: Improper Access Control</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">04</a>	Unchecked Call Return Value	<a href="#">CWE-252: Unchecked Return Value</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">03</a>	Floating Pragma	<a href="#">CWE-664: Improper Control of a Resource Through its Lifetime</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">02</a>	Outdated Compiler Version	<a href="#">CWE-937: Using Components with Known Vulnerabilities</a>	<b>NOT PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">01</a>	Integer Overflow and Underflow	<a href="#">CWE-682: Incorrect Calculation</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">00</a>	Function Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	<b>PASSED</b>



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

 MADE IN GERMANY