SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

Arthur Exchange

AUDIT

SECURITY ASSESSMENT

27. October, 2023

FOR

ARTHUR EXCHANGE

Introduction

SolidProof.io is a brand of the officially registered company MAKE Network GmbH, based in Germany. We're mainly focused on Blockchain Security such as Smart Contract Audits and KYC verification for project teams.
Solidproof.io assess potential security issues in the smart contracts implementations, review for potential inconsistencies between the code base and the whitepaper/documentation, and provide suggestions for improvement.

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 the security or functionality of the technology we agree to analyze.

Project Overview

Summary

Project Name	Arthur Exchange
Website	https://www.arthur.exchange/
About the project	Embark on a transformative voyage with Arthur Exchange, as the groundbreaking Linea-originated launchpad seamlessly converges with a robust DEX. Awaken your trading prowess while spearheading visionary projects at the vanguard of Linea's blockchain evolution. As the inaugural launchpad on Linea, Arthur serves as the gateway to an unprecedented epoch of decentralized finance, replete with limitless prospects. Engage with us in shaping the narrative of an immersive and interwoven crypto panorama, where innovation knows no bounds.
Chain	Linea
Language	Solidity
Codebase	https://github.com/arthur-exchange-team/arthur-contracts
Forked Status	This project is 1:1 forked from CamelotLabs, The contracts can be found in the links below: core:https://github.com/CamelotLabs/core/tree/main/contracts periphery:https://github.com/CamelotLabs/periphery/tree/main/contracts Pool: https://docs.camelot.exchange/contracts/amm-v2
Commit	N/A
Unit Tests	Not Provided

Social Medias

Telegram	https://t.me/+MPnvJyKXQ982YTI1
Twitter	https://twitter.com/ArthurExchange
Facebook	N/A
Instagram	N/A
GitHub	N/A
Reddit	N/A
Medium	N/A
Discord	https://discord.com/invite/arthurexchange
YouTube	N/A
TikTok	N/A
LinkedIn	N/A

Audit Summary

Version	Delivery Date	Change Log
v1.0	25. October 2023	· Layout Project · Automated/ Manual-Security Testing · Summary
v1.1	27. October 2023	· Reaudit

Note – The following audit report presents a comprehensive security analysis of the smart contract utilized in the project that includes outside manipulation of the contract's functions in a malicious way. 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
pool/contracts/interfaces/IYieldBooster.sol	660ca7e6ab25e886babfd42608e9e0695656e8be
pool/contracts/interfaces/tokens/IArtToken.sol	137eea06c61c9d4988dfd62a2bc14cbfe88d9b73
pool/contracts/interfaces/tokens/IXArtToken.sol	55f092d156bfa2a99462967769ef67981dfa052c
pool/contracts/interfaces/IDividendsV2.sol	e1c1548ffa170a63c936d4351c5a96a79614ee94
pool/contracts/interfaces/INFTHandler.sol	bd2774fe5ec7078d2698610e5f10ad5e4895be18
pool/contracts/interfaces/IMerlinCustomReq.sol	014548c4d66b273d321cf6659300f5dc71312420
pool/contracts/interfaces/IArthurMaster.sol	ba89f47dc88800d3d6dbb097001a68d93c72d909
pool/contracts/interfaces/INFTPool.sol	a487256e5273a063adc73da9be0f5e57d61fb1b6
pool/contracts/interfaces/IMerlinPoolFactory.sol	53f88b818529885c4128fd5b69449a209ae92e45
pool/contracts/interfaces/IXArtTokenUsage.sol	6ed48f955f1b41d3250f320b291770fbc61ae31c
pool/contracts/interfaces/IArthurRouter.sol	9ead111e925ef008d3f89579bd237b381b92ae14
pool/contracts/interfaces/IUniswapV2Router01.sol	f80f64def4f799c792a1aa2fddf2128fd2fd6f69
pool/contracts/MerlinPoolFactory.sol	357859c31639af7335528c636a7c3a7f08963b35
pool/contracts/NFTPoolFactory.sol	b692fee7467941ac0bf4e2ad2d9da635e45ee6ce
pool/contracts/MerlinPool.sol	e0d8d9eb0c1197cb4ae25e712b48898bcb4bba57
pool/contracts/YieldBooster.sol	96677dc6d52e9b8858178e536b15233adace0dc6
pool/contracts/PositionHelper.sol	9b73b6a0df55e6bfd4586bf1a56750a7c9474638
pool/contracts/DividendsV2.sol	cb1fca8017d07b5a8d2659f179818f8581b286a3
pool/contracts/ArthurMaster.sol	01f4276893f75115f5eeafb3951c511aa814d1cf
pool/contracts/TokenTest.sol	5008f933a3e6abd1515114094ba4cae73f129350

File Name	SHA-1 Hash
pool/contracts/NFTPool.sol	a86cdcfc5b055d5af64b69adeb5266c44be72c23
pool/contracts/XArtToken.sol	f6ddd3fb082305e74c8ba654bc3fe82ef4ff63cf
pool/contracts/ArtToken.sol	1646cd71cb0ab0b75d3477766d080e3a0e8eb9bd
periphery/contracts/interfaces/IWETH.sol	6a25dd53c8494e3aef3a520f17e00608b529f061
periphery/contracts/interfaces/IArthurRouter.sol	39cf8c3aaa365d293be04ff3e09039e0f4a84b75
periphery/contracts/interfaces/IUniswapV2Router01.sol	34b1148cba9a64621026c282c370f8cb65bf88c8
periphery/contracts/libraries/SafeMath.sol	95e7522a8821aa493dcceaad8e66f192c195bcfc
periphery/contracts/libraries/UniswapV2Library.sol	56e95525e10be50d481e75dfec3fc11168f6b602
periphery/contracts/ArthurRouter.sol	23a105bab90f82a2609a356538b5d21d8ddd6575
core/contracts/interfaces/IUniswapV2Callee.sol	c7e224344966e0cfad73f086da1a105cc8f24902
core/contracts/interfaces/IArthurPair.sol	79302bd0d7d20cfb5d3832e9083d488dbb14a2cb
core/contracts/interfaces/IUniswapV2ERC20.sol	a881fff951a6284f2fa04849ebda57783de3f02a
core/contracts/interfaces/IArthurFactory.sol	4f8062e3f8d6ba591ce4ac2b3e81f6491edf1edc
core/contracts/interfaces/IERC20.sol	b7d011aaabd34898ee60760996cb702e7b2ca855
core/contracts/ProtocolEarnings.sol	cde392b59fe6452c92e1e6e2c965846097a4fdf6
core/contracts/ArthurPair.sol	8e55314b8cbaf649d470ded281b2dd9ee79c5491
core/contracts/ArthurFactory.sol	c80d7b6dbaa1d8517491532056e2326ccf91b9b1
core/contracts/libraries/Math.sol	e6f63d883294ea708b0ab5ecee646f9fcac6722c
core/contracts/libraries/UQ112x112.sol	5c0f96357914f9f80b6d616b79ece099d5f91ec4
core/contracts/libraries/SafeMath.sol	107957387539a5284287b246b955e4037d88188d
core/contracts/UniswapV2ERC20.sol	b4c635238cd6b803fdf02a054821926921668707

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) be an indication of a changed state or potential vulnerability that was not the subject of this scan.

Imported packages.

Used code from other Frameworks/Smart Contracts.

Dependency / Import Path	Count
@openzeppelin/contracts/access/Ownable.sol	8
@openzeppelin/contracts/security/ReentrancyGuard.sol	6
@openzeppelin/contracts/token/ERC20/ERC20.sol	3
@openzeppelin/contracts/token/ERC20/IERC20.sol	5
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	8
@openzeppelin/contracts/token/ERC721/ERC721.sol	1
@openzeppelin/contracts/token/ERC721/IERC721.sol	1
@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol	1
@openzeppelin/contracts/utils/Address.sol	1
@openzeppelin/contracts/utils/Counters.sol	1
@openzeppelin/contracts/utils/math/Math.sol	2
@openzeppelin/contracts/utils/math/SafeMath.sol	9
@openzeppelin/contracts/utils/structs/EnumerableSet.sol	7
@uniswap/lib/contracts/libraries/TransferHelper.sol	1
excalibur-core/contracts/interfaces/IArthurFactory.sol	1
excalibur-core/contracts/interfaces/IArthurPair.sol	2
excalibur-core/contracts/interfaces/IERC20.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.

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'sstate. They are declared at the contract level and can be accessed and modified by any function within the contract. State variables can be needed within visibility modifier, such as public, private or internal, which determines the access level of the variable.

Components

📝 Contracts	📚 Libraries	🔍 Interfaces	🎨 Abstract
16	5	20	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.

🌐 Public	💰 Payable
372	10

External	Internal	Private	Pure	View
341	346	5	32	161

StateVariables

Total	🌐 Public
168	127

Capabilities

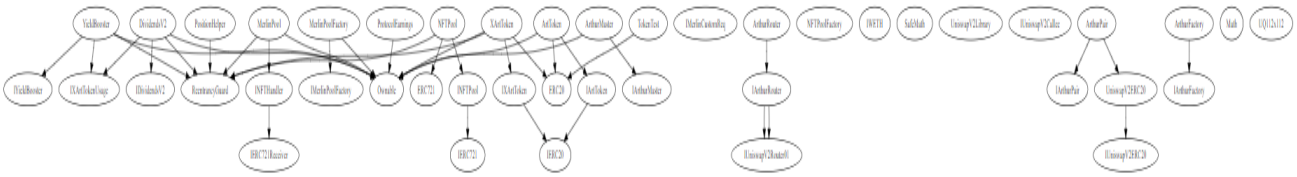
Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts

`0.8.16`
`>=0.6.2`
`>=0.5.0`
`=0.5.16`
	`----------`	`yes`	`yes`
			(3 asm blocks)
				`----------`

Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/Create/Create2
`Yes`		`yes`	`Yes`	`Yes`	`yes` ⟶ `NewContract:MerlinPool` ⟶ `AssemblyCall:Name:create2`

Inheritance Graph

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 methodsfrom another class. It shows the relationships between different contracts and how they are related to each other through inheritance.



Audit Information

Vulnerability & Risk Level

Risk represents the probability that a certain source threat will exploit the 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)
Critical	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.
High	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 aspossible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executingthe contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	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.
Informational	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 andsmart 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. Reviewing the specifications, sources, and instructions provided to SolidProof to ensure we understand the size, scope, and functionality of the smart contract.
   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, which is analysing a program to determine what inputs cause each part of a program to execute.

3. Review best practices, i.e., review 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 upgradable	✅ **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	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: • Centralizations • The owner has significant control over contract's operations.
Example	N/A
Comment	N/A

Note – *The contract cannot be considered as renounced till it is not deployed or having some functionality that can change the state of the contract.*

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 refer 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 has the ability to add new tokens to the network's total supply.

Contract owner cannot mint new tokens.	✅ **The owner cannot mint new tokens.**
Description	The owner is not able to mint new tokens once the contract is deployed.
Comment	N/A

Burning tokens

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 burn tokens without any allowances.
Comment	N/A

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 wallets.**
Description	The owner cannot blacklist addresses for transferring of tokens.
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 cost of running the contract, such as paying for gas fees or compensating the contract's owner for their time and effort indeveloping and maintaining the contract.

Contract owner cannot set fees more than 25%.	✅ **The owner cannot set fees more than 25%.**

Description	The owner cannot set fees more than 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 token 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.

Contract owner can lock functions.	✅ **The owner cannot lock the functions.**
Description	The contract contains the time lock functionality which can be updated by the owner after the initial deployment and can simply lock the functionality of the contracts for not more than 90 days.
Comment	The MAXIMUM_TIMELOCK is set to 90 days in the contract which cannot be changed. The timelock value cannot be more than 90 days.

File/Line(s): L154-163
Codebase: ArthurPair.sol

```
function setPairTimeLock(uint256 _timeLock) external lock {
  require(msg.sender == IArthurFactory(factory).owner(), "ArthurPair: only factory's owner");
  require(!pairTypeImmutable, "ArthurPair: immutable");
  require(timeLock <= MAXIMUM_TIMELOCK, "ArthurPair: timeLock mustn't exceed the maximum");

  uint256 oldValue = timeLock;
  timeLock = _timeLock;

  emit SetPairTimeLock(oldValue, timeLock);
}
```

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 the contract is controlled by a single entity or if certain participants have special permissions or abilities that others do not.

In the project, there are authorities that have access to the following functions:

File	Privileges
ArthurFactory.sol	➢ The owner can set the fee percentage owner. ➢ Set stable owner can update the stable owner in the contract. ➢ The owner can set up a wallet to receive fees. ➢ The owner can set referrer fees of not more than 20%.
ArthurPair.sol	➢ The fee percent owner of the factory contract can set the tokenn0, token1 fee percent of not more than 2%. ➢ The stable owner of the factory contract can set the stable swap. ➢ The owner of the factory contract can set the pair type as immutable. ➢ The owner of the factory contract can set the pair time lock value in the contract only before the pair type is set to false which cannot be more than 90 days. ➢ The owner of the Arthur factory contract can set the pair start time only before the pair type is set to false. ➢ The owner of the factory contract can withdraw the foreign tokens from the contract's balance.
ProtocolEarnings.sol	➢ The owner can set the buyback and burn wallet address. ➢ The owner can set the development fund wallet address. ➢ The owner can update the dividend wallet address. ➢ The owner can distribute tokens manually between dividend, buyback, and burn, operating funds wallet address. ➢ The owner can update the quantity of dividend and

	buyback and burn shares which cannot be more than sharePrecision amount. ➢ The owner can withdraw stuck tokens from the contract.
MerlinPoolFactory.sol	➢ The owner can set the default fees of not more than 1%. ➢ The owner can update the fee address. ➢ The owner can add/remove the address from the exempted address list. ➢ The owner can update the emergency recovery address.
MerlinPool.sol	➢ The owner can withdraw rewards from the pool before the pool is published. ➢ The owner can set the reward token address. ➢ The owner can set the custom req contract address. ➢ The owner can update the requirements that the position must meet to be staked on this merlin pool. ➢ The owner can update the date settings in the contract. ➢ The owner can set the description in the contract. ➢ The owner can add accounts in the whitelist addresses list. ➢ The owner can reset the whitelist. ➢ The owner can publish the Merlin pool. ➢ The owner can send the rewards to the recovery address if there is any case of emergency.
YieldBooster.sol	➢ The owner can update the allocation floor which cannot be more than 1000 ether. ➢ The owner can enable/disable forced deallocation status in the contract. ➢ The owner can withdraw the balance from the contract.
XArtToken.sol	➢ The owner can update the redeem settings. ➢ The owner can update the dividend address. ➢ The owner can update the deallocation fee of not more than 2%. ➢ The owner can whitelist/unwhitelist wallets from transferwhitelist list.
DividendsV2.sol	➢ The owner can withdraw contract balance manually.

ArtToken.sol	➢ The owner can withdraw all tokens from the contract's balance. ➢ The owner can enable the distribution token of the particular contract address. ➢ The owner can disable the distribution token of the particular contract address. ➢ The owner can update the dividend cycle percent between 10 to 100%. ➢ The owner can remove the token from the dividend token to disable once the token is distributed. ➢ The master contract can claim the master rewards. ➢ The owner can initialize the master address. ➢ The owner can initialize the start time only once. ➢ The owner can update the allocation between the farming incentives. ➢ The owner can update the emission rate which cannot be more than 0.1 ether. ➢ The owner can update the max supply which cannot be more than 10_000_000 ether. ➢ The owner can update the treasury wallet address.
ArthurMaster.sol	➢ The owner can update the yieldBooster contract address in the contract. ➢ The owner can unlock/lock all the pools after the deployment. ➢ The owner can add a new pool. ➢ The owner can update the configuration on the existing pools.

Recommendations

To avoid potential hacking risks, it is advisable for the client to 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 cannot modify any state variables of the contract anymore. Make sure to set up everything before renouncing.

Audit Result

Critical Issues

No critical issues

High Issues

No high issues

Medium Issue

No medium issues

Low Issue

#1 | Missing events arithmetic.

File	Severity	Location	Status
All	Low	--	ACK

Description – Emit all the critical parameter changes.

#2 | Remove safemath library.

File	Severity	Location	Status
All	Low	--	ACK

Description – The compiler version above 0.8.0 has the ability to control arithmetic overflow/underflow, it is recommended to remove the unwanted code in order to avoid high gas fees.

#3 | Missing zero or dead address check.

File	Severity	Location	Status
All	Low	--	ACK

Description – Add a 'require' check that the address cannot be set to zero or dead address.

Informational Issue

#1 | NatSpec Documentation missing.

File	Severity	Location	Status
All	Informational	--	ACK

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

#2 | Contract doesn't import npm packages from source (like OpenZeppelin etc.)

File	Severity	Location	Status
All	Informational	--	ACK

Description – We recommend importing all packages from npm directly without flattening the contracts. Functions could be modified or can be susceptible to vulnerabilities.

Legend for the Issue Status

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

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY