SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

Esper Finance

AUDIT

SECURITY ASSESSMENT

07. September, 2023

FOR

SolidProof_io

@solidproof_io

Introduction 4
Disclaimer 4
Project Overview 5
Summary 5
Social Medias 5
Audit Summary 6
File Overview 7
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 15
Blacklist addresses 16
Fees and Tax 17
Lock User Funds 18
Components 19
Exposed Functions 19
StateVariables 19
Capabilities 20
Inheritance Graph 21
Centralization Privileges 22
Audit Results 26
Critical issues 26
High issues 26

Medium issues 27
Low issues 28
Informational issues 29

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	Esper Finance
Website	https://esper.finance/
About the project	Esper Finance is an ecosystem-focused, community-driven DEX built upon the robust, highly scalable Mantle Network. We aim to enable both builders and users on Mantle ecosystem to leverage our protocol infrastructure for a deep, adaptable, and sustainable liquidity.
Chain	Mantle
Language	Solidity
Codebase Link	https://github.com/Nava-Labs/esper-contracts/tree/main
Commit	e572ce0
Unit Tests	Not Provided

Social Medias

Telegram	N/A
Twitter	https://twitter.com/EsperFinance
Facebook	N/A
Instagram	N/A
Github	N/A
Reddit	N/A
Medium	N/A
Discord	N/A
Youtube	N/A
TikTok	N/A
LinkedIn	N/A

Audit Summary

Version	Delivery Date	Changelog
v1.0	06. September 2023	• Layout Project • Automated- /Manual-Security Testing • Summary
v1.1	07. September 2023	• Reaudit

Note - The following audit report presents a comprehensive security analysis of the smart contract utilized in the project. 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 it was not functionally tested by us.

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
contracts/esper-periphery-contracts/EsperRouter.sol	a0405bcc498e79e10cb850c70c007fc58b9db820
contracts/esper-periphery-contracts/libraries/SafeMath.sol	123c932c8701c1178d049c82339bc68cd3c61d18
contracts/esper-periphery-contracts/libraries/UniswapV2Library.sol	fc7c1e33b78359f5200f889c553bd1ca6390ca54
contracts/esper-core-contracts/tokens/ERC20Snapshot.sol	1682ad0f06b98ccd9e2fe557ae133f92d0296d34
contracts/esper-core-contracts/tokens/XEsperToken.sol	02000b7775f22b82a56d425a621dfa06644d1a5c
contracts/esper-core-contracts/tokens/EsperToken.sol	1d1c2c8d0ec772058c012edabcf2bfdcc4570b00
contracts/esper-core-contracts/launchpad/FairAuction.sol	76449181a1b4c1984bad5dec4e4c2fede493ac36
contracts/esper-core-contracts/launchpad/Launchpad.sol	05d5e67f960725258c35799a83a91e95dee2411f
contracts/esper-core-contracts/launchpad/FairAuctionFactory.sol	d574cce9100ce0cd829111b2d1955a79145d68f7
contracts/esper-core-contracts/launchpad/EsperPresale.sol	ff09b61d3eb7f9c0f7f1b47e114ba82e81d16c4a
contracts/esper-core-contracts/utils/interfaces/IEsperRouter.sol	9e5c927c81c1e88f2ed04533fcb9a93fc70af336
contracts/esper-core-contracts/utils/interfaces/INFTPool.sol	8174d7fb422f0f73f6c0789d6b18f3adcb6266c0
contracts/esper-core-contracts/utils/PositionHelper.sol	b21badf59d81d1f85d37aed4cdffefcad0b55ab8
contracts/esper-core-contracts/utils/ProtocolEarnings.sol	8046890657b7773826c40bea7c2822298c4eef45
contracts/esper-core-contracts/plugins/YieldBooster.sol	6cecb0fb3b54e1329b8e87b9b1cb9ec4b73d67c8

contracts/esper-core-contracts/plugins/Dividends.sol	02e745cbbc2b11cd0924c7c242d784236d177f4a
contracts/esper-core-contracts/nft-pool-factory/NFTPoolFactory.sol	66a25be90a5ae3a6dbf894aee51120db69c8a040
contracts/esper-core-contracts/nft-pool-factory/NFTPool.sol	5fdf031b3e7aab98018d33ea649e3f5eae1f7df5
contracts/esper-core-contracts/nft-pool-factory/EsperMaster.sol	e68187e60e0cf33d1b37d65e5953f1aa58600ef2
contracts/esper-core-contracts/nitro-pool/NitroPoolFactory.sol	c14679a4b59fca275650b49692abc12dac7e2c6c
contracts/esper-core-contracts/nitro-pool/NitroPool.sol	0eb502039ca7c19c8c5f6ca6293c04b87de010c5
contracts/esper-amm-contracts/EsperFactory.sol	72da38e01136e6ec07230265719d2bf94102e10e
contracts/esper-amm-contracts/libraries/Math.sol	e6f63d883294ea708b0ab5ecee646f9fcac6722c
contracts/esper-amm-contracts/libraries/UQ112x112.sol	5c0f96357914f9f80b6d616b79ece099d5f91ec4
contracts/esper-amm-contracts/libraries/SafeMath.sol	97a5b17b0fd90ece89930aeff76cc32fef1a6f14
contracts/esper-amm-contracts/EsperPair.sol	d95dea0981154edecf5f8b638ea752212983f48d
contracts/esper-amm-contracts/UniswapV2ERC20.sol	726171dcf4ed5ed4adba5507039c5e6f7abd8260

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 (direct imports).

Dependency / Import Path	Count
@openzeppelin/contracts/access/Ownable.sol	12
@openzeppelin/contracts/math/Math.sol	3
@openzeppelin/contracts/math/SafeMath.sol	13
@openzeppelin/contracts/token/ERC20/ERC20.sol	4
@openzeppelin/contracts/token/ERC20/IERC20.sol	5
@openzeppelin/contracts/token/ERC20/SafeERC20.sol	10
@openzeppelin/contracts/token/ERC721/ERC721.sol	1
@openzeppelin/contracts/token/ERC721/IERC721.sol	1
@openzeppelin/contracts/utils/Address.sol	1
@openzeppelin/contracts/utils/Arrays.sol	1
@openzeppelin/contracts/utils/Counters.sol	2
@openzeppelin/contracts/utils/EnumerableSet.sol	7
@openzeppelin/contracts/utils/ReentrancyGuard.sol	9
@uniswap/lib/contracts/libraries/TransferHelper.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)
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 executing the 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 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. 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 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 owner is not renounce**
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
Comment	The contracts in the project are Fokred from an Exisiting Project named "***Camelot Finance***". More information about that can be found at https://docs.camelot.exchange/

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 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 can blacklist addresses	❌ **The owner able to blacklist addresses**
Description	If the owner or developers of the smart contract abuse their power to blacklist addresses without proper justification or transparency, it can lead to a decrease in trust and credibility in the project. For example, suppose an owner or developer blacklists an address without proper explanation or communication to stakeholders. In that case, it can create speculation and uncertainty among investors, potentially causing them to sell their tokens and decreasing the token's value. Furthermore, if the owner or developers have a significant number of tokens themselves and use their power to blacklist competitors or manipulate the market, it can lead to an unfair advantage and concentration of power, potentially harming the interests of other stakeholders. Therefore, it is important for projects and platforms to be transparent about their blacklisting practices and ensure that they are justified and in the best interest of their stakeholders. Investors should carefully research the project and its blacklisting practices and exercise caution before investing in any cryptocurrency or DeFi project to avoid potential losses.
Comment	The addresses that are not whitelisted will not be able to transfer xEsper tokens.

Codebase:

```
function updateTransferWhitelist(address account, bool add) external onlyOwner {
  require(account != address(this), "updateTransferWhitelist: Cannot remove xEsper from whitelist");

  if(add) _transferWhitelist.add(account);
  else _transferWhitelist.remove(account);

  emit SetTransferWhitelist(account, add);
}
```

XEsperToken.sol

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 in developing and maintaining the contract.

Contract owner cannot set fees more than 20%	✅ **The owner cannot levy unfair taxes**
Description	The owner is not able to set the fees above 20%
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

📝 Contracts	📚 Libraries	🔍 Interfaces	🎨 Abstract
19	5	2	1

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
276	7

External	Internal	Private	Pure	View
233	326	10	19	126

StateVariables

Total	🌐 Public
206	161

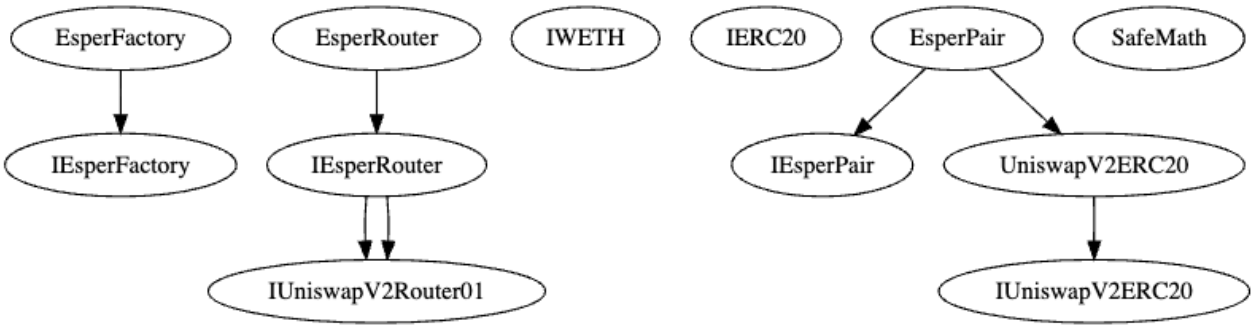
Capabilities

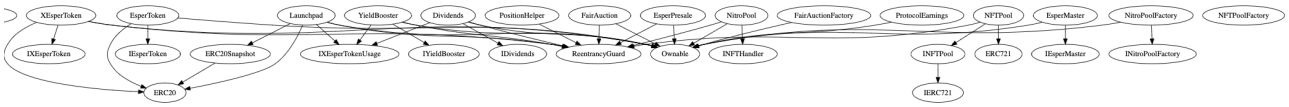
Solidity Versions observed	Transfers ETH	Can Receive Funds	Uses Assembly	ECRecover

`>=0.5.0`
`>=0.6.2`
`=0.6.6`
`^0.7.0`
`=0.7.6`
`=0.5.16`
	Yes	Yes	Yes	Yes

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 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 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

1. EsperFactory.sol	• onlyOwner
	  • Transfer Ownership
	  • Set fee receiver address
	  • Set Stable Coin owner address
	  • Set owner fee share which can be set upto 20%
	  • Set referrer fee share upto 20% maximum

2. EsperPair.sol	• The factory contract owner can set the following
	  • Set pair type as Immutable
	  • Drain any type of tokens from the contract's balance
	• The fee percent owner address can change the fee percent upto 2%
	• The stable owner of the factory contract can set the stable swap

3. EsperPresale.sol	• onlyOwner • The owner can withdraw all the tokens from the contract at anytime and the users will not get the claimable tokens

4. FairAuction.sol	• onlyOwner
	  • Pause the buying functionality
	  • Add users to the whitelist and enable/disable the whitelist, if done so then only the whitelisted users will be able to buy tokens
	  • Manually Enable Claim at anytime
	  • The owner can withdraw all the tokens from the contract at anytime and the users will not get the claimable tokens

File	Privileges
5. Launchpad.sol	• onlyOwner • Update deallocation cooldown period to any arbitrary value. If it is set to a very high value then deallocation will not be possible

6. EsperMaster.sol	• onlyOwner
	  • Set Yield Booster Address
	  • Manually Unlock all pools at any given arbitrary time
	  • Add a new pool
	  • Update configuration of an existing pool

7. NFTPool.sol	• onlyOwner or Operator
	  • Set lock multiplier settings
	  • Set boost multiplier settings
	  • Set Rewards share
	• onlyOwner
	  • Add/remove operators
	  • Enable/Disable Emergency Unlock feature

8. NitroPool.sol	• onlyOwner
	  • Withdraw rewards from the contract if the pool is not published
	  • Set rewards token (only once)
	  • Set requirements for staking on the Nitro pool like - Lock Duration, Deposit Amount, Whitelist
	  • Set Date Settings and Pool's Description
	  • Set whitelisted users
	  • Publish the Nitro Pool
	  • The owner can withdraw all the tokens from the contract to the recovery address at anytime and the users will not get the staked tokens

9. NitroPoolFactory.sol	• onlyOwner
	  • Set default fee upto 1%
	  • Set fee address
	  • Set exempted address
	  • Set emergency recovery address

File	Privileges

10. Dividends.sol	• onlyOwner
	  • The owner can withdraw all the dividend token balance from the contract at anytime.
	  • Enable/Disable the tokens that will be distributed as dividends.
	  • Update the epercentage of Pending Dividends

11. YieldBooster.sol	• onlyOwner
	  • Update the total allocation floor value
	  • Enable/Disable forced deallocation status
	  • Withdraw the complete balance of the contract

12. EsperToken.sol	• onlyOwner
	  • Initialize Master address and Emission Start Time
	  • Update allocations and Emissions Rate
	  • Update Max supply but not more than the Limit of 200,000 ETHER
	  • Update Treasury Address

13. XEsperToken.sol	• onlyOwner
	  • Update redeem ratios and durations
	  • Update Dividend contract address
	  • Update Deallocation fee
	  • Add/Remove wallets from whitelist

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 Results

Critical issues

No critical issues

High issues

No high issues

Medium issues

#1 | Owner can drain tokens

File	Severity	Location	Status
EsperPresale	Medium	L249	ACK
FairAuction	Medium	L326	ACK
NitroPool	Medium	L593	ACK
Dividends	Medium	L266, 275	ACK

Description - The owner of the contract is able to drain the complete balance of the contract by calling this function. Moreover, in the Nitro Pool contract, the owner can set any address as the recovery address to which the complete funds of the contract will be transferred.

Remediation - Make sure that it is not possible to take out native tokens. In case of an emergency, first refund the deposited sale token amount and then make the withdrawal.

Low issues

#1 | Missing Zero Address Validation

File	Severity	Location	Status
FairAuction.sol	Low	L63—66	ACK

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

#2 | Access Control

File	Severity	Location	Status
EsperMaster	Low	L278	ACK

Description - The comment specifies that the function is only callable by the NFT pool contract but it can be called by any caller which is a listed pool. Moreover, the pools can be added by the owner.

#3 | Missng "isContract" check

File	Severity	Location	Status
XEsperToken	Low	L210	ACK

Description - The contract has no checks to verify whether the dividend address is a contract or not
Remediation - We recommend putting a check to verify that the address passed in the function must be a contract

#4 | Old Compiler version

File	Severity	Location	Status
All	Low	—	ACK

Description - The contracts use outdated compiler versions, which are not recommended for deployment as they may be susceptible to known vulnerabilities.

Remediation - Use a newer pragma version. At least use the 0.8.18 version.

Informational issues

#1 | Floating Pragma

File	Severity	Location	Status
ERC20Snapshot	Informational	—	ACK

Description - The contracts should be deployed with the same compiler version and flag that they have been tested thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using other versions.

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.

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY