



# SOLIDProof

*Bring trust into your projects*

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

MADE IN GERMANY

## Fyda

# AUDIT

SECURITY ASSESSMENT

**01. December, 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	23
Critical issues	23
High issues	23



Medium issues	23
Low issues	24
Informational issues	25



## 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	Fyda
Website	<a href="https://fyda.fi">https://fyda.fi</a>
About the project	Fyda Finance is a non-custodial decentralized protocol for automating trading strategies. It introduces advanced trade automation through smart contracts, seamlessly integrating sophisticated features to bridge existing gaps in decentralized trading.
Chain	Polygon, Optimism, Arbitrum, Base and Ethereum
Language	Solidity
Codebase Link	<a href="https://github.com/Fyda-Finance/protocol/tree/main">https://github.com/Fyda-Finance/protocol/tree/main</a>
Commit	9e1a4a3
Unit Tests	Provided

## Social Medias

Telegram	N/A
Twitter	<a href="https://twitter.com/fydafinance">https://twitter.com/fydafinance</a>
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	01. December 2023	<ul style="list-style-type: none"> <li>• Layout Project</li> <li>• Automated- /Manual-Security Testing</li> <li>• Summary</li> </ul>

**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
contracts/interfaces/IERC173.sol	21f0b458d570867197163ef6b6553f902af5a1b0
contracts/interfaces/IERC165.sol	c8d701553f8522ee450cade7da74729eb23384cc
contracts/interfaces/IDiamondLoupe.sol	59baa672f8f3e6f3f8a22ccbc37168784bf60d99
contracts/interfaces/IDiamondCut.sol	ee65cf7443b0650dd7f22b2a0849679fb5ef3ee8
contracts/AppStorage.sol	d4688cb8e6f7a1a74922db47dbfa2e4d20bb242e
contracts/Diamond.sol	e1630dd7868b893f1bf50a55e293e2bf0e9ea62f
contracts/utills/Modifiers.sol	5931cdbac3efac3b19b9e25049ad2f118e368c7a
contracts/utills/GenericErrors.sol	435e14f49bbd0ea9c5e4e375f1191fd29f27572e
contracts/libraries/LibPrice.sol	8ae0ed00d0f1a84ef1ea2d91424ed69f3189ee1f
contracts/libraries/LibSwap.sol	4be969c2583db5a0d9a8710ff56802a853f5fad9
contracts/libraries/LibBytes.sol	8c8e97c62f6b6b544d05b5a724154e6cacc2ef7b
contracts/libraries/LibAsset.sol	67c204f03eb70dafa7ef41b08c9d098675cd1769
contracts/libraries/LibUtil.sol	8c85cfeac35ec4d8793cd00607e28e45d65b5b73
contracts/libraries/LibTrade.sol	6180350f58ba9ec7b3477abd3d1f78680f837282
contracts/libraries/LibTime.sol	497ddae775f137e6c26bc004a58cd8c617712852
contracts/libraries/LibDiamond.sol	d34165fae3c9be140629c09a8aadf3b63d0c6ef7
contracts/libraries/LibSignature.sol	49ba205052d7db73a58b8629168e866d482e6bd1
contracts/upgradeInitializers/DiamondInit.sol	07b2e9d54fa21e3967b6e78983eb457c6682032f
contracts/facets/FloorFacet.sol	55336a609d964ddf03133b5f7a2e729ebb445967
contracts/facets/SellFacet.sol	c340e2e24e465622a32cf45cbabcd9f6cedf2261
contracts/facets/StrategyFacet.sol	5a5a394b2b5f16786b1697af80708f56e1df2e5e
contracts/facets/BuyFacet.sol	56157dce97df2a81d573f387758350495856caa8
contracts/facets/DiamondLoupeFacet.sol	e774ee313b96f6a5d3a6db37e432b0eaa00d2a2a
contracts/facets/PriceOracleFacet.sol	4669a12ff6be5ea3b30ddda0d6a10b5b8bc8ab63
contracts/facets/LensFacet.sol	54ae29472fe5a2a5013b6ddac59c0645fa32c258
contracts/facets/OwnershipFacet.sol	93b1e02f2fd96e64996197ec76d40f358e36f8fb
contracts/facets/DiamondCutFacet.sol	50ab7c5b8bdc35cddb3edba44cd73c63c182c85a



*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
@chainlink/contracts/src/v0.8/Denominations.sol	1
@chainlink/contracts/src/v0.8/interfaces/AggregatorV2V3Interface.sol	2
@openzeppelin/contracts/token/ERC20/IERC20.sol	2
@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol	6
@openzeppelin/contracts/token/ERC20/extensions/IERC20Permit.sol	1
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.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. 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

There are no major ownership privileges in the contracts

**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 cannot blacklist addresses**



**The owner cannot blacklist addresses**

Description

The owner is not able 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 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 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>
11	9	4	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.

 <b>Public</b>	 <b>Payable</b>
48	3



External	Internal	Private	Pure	View
40	66	0	12	36

## StateVariables

<b>Total</b>	 <b>Public</b>
13	1



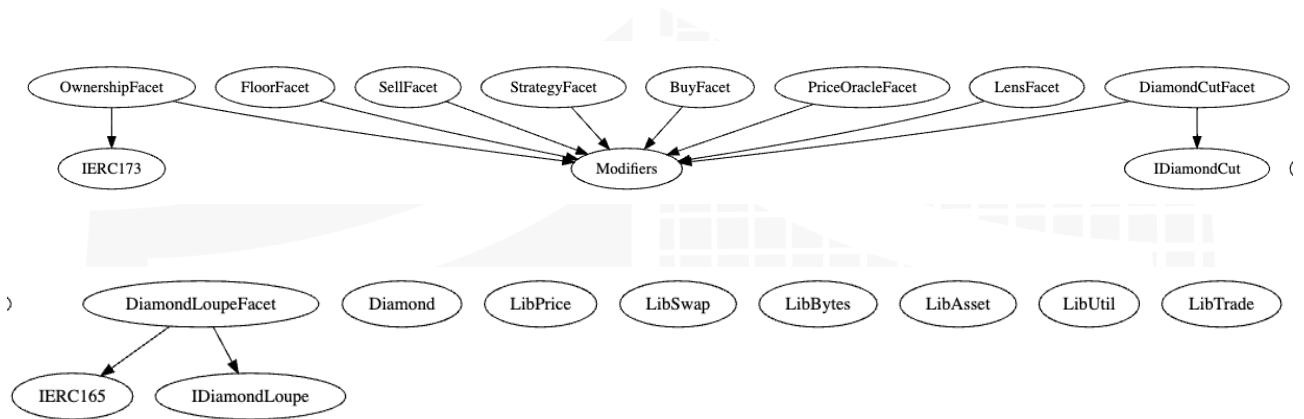
## Capabilities

Solidity Versions observed	Transfers ETH	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts
<code>^0.8.20</code> <code>^0.8.0</code> <code>0.8.20</code>	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
<b>DiamondCutFaucet</b>	<ul style="list-style-type: none"> <li>The owner can Add/Remove/Replace any number of functions</li> </ul>
<b>PriceOracleFacet</b>	<ul style="list-style-type: none"> <li>The owner can change the price feed address for the assets</li> </ul>

## 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

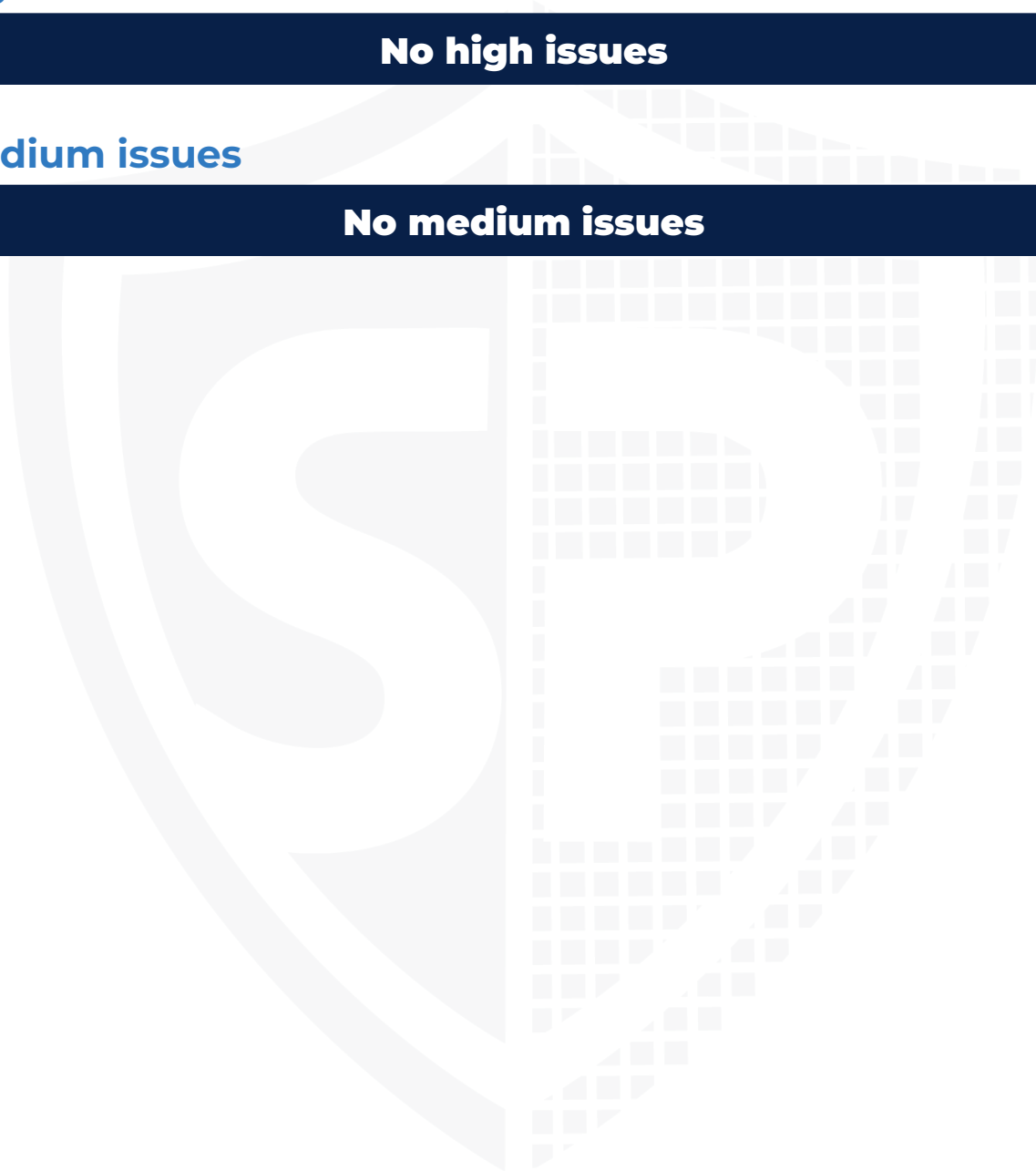
**No critical issues**

### High issues

**No high issues**

### Medium issues

**No medium issues**



## Low issues

### #1 | Missing Events

File	Severity	Location	Status
PriceOracleFacet	Low	L22	Open

**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.

### #2 | Missng valid address check

File	Severity	Location	Status
PriceOracleFacet	Low	L22	Open

**Description** - The contract has no checks to verify that the asset feed address is a valid address. i.e., it should not be an EOA and also not be the zero/dead address.

**Remediation** - We recommend putting a check to verify that the asset feed address must be valid.



## Informational issues

### #1 | Floating Pragma

File	Severity	Location	Status
All	Informational	N/A	Open

**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.

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

File	Severity	Location	Status
All	Informational	N/A	Open

**Description** - We recommend importing all packages from npm directly without flattening the contract. 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