



SOLIDProof
Bring trust into your projects

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

MADE IN GERMANY

DegenX Diamond

AUDIT

SECURITY ASSESSMENT

11. January, 2024

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	8
Audit Information	9
Vulnerability & Risk Level	9
Auditing Strategy and Techniques Applied	10
Methodology	10
Overall Security	11
Upgradeability	11
Ownership	12
Ownership Privileges	13
Minting tokens	13
Burning tokens	14
Blacklist addresses	15
Fees and Tax	16
Lock User Funds	17
Components	18
Exposed Functions	18
StateVariables	18
Capabilities	19
Inheritance Graph	20
Centralization Privileges	21
Audit Results	23
Critical issues	23
High issues	23



Medium issues	24
Low issues	26
Informational issues	28



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	DegenX Finance
Website	https://dgnx.finance/
About the project	DegenX is multichain ecosystem that offers a suite of decentralized applications (dApps) and services to provide solutions for projects and individuals in the DeFi space. DegenX is multichain ecosystem that offers a suite of decentralized applications (dApps) and services to provide solutions for projects and individuals in the DeFi space.
Chain	Avalanche
Language	Solidity
Codebase Link	https://github.com/DEGENTOKENTTEAM/diamond/tree/main/contracts
Commit	46a477c
Unit Tests	Provided

Social Medias

Telegram	https://t.me/DegenXportal
Twitter	https://twitter.com/degenecosystem
Facebook	N/A
Instagram	https://instagram.com/degenecosystem
Github	https://github.com/DEGENTOKENTTEAM
Reddit	https://www.reddit.com/user/degentrader_sd
Medium	https://medium.com/@degentraderteam
Discord	https://discord.gg/BMaVtEVkgC
Youtube	https://youtube.com/@DGNX.FINANCE-DEGENX?sub_confirmation=1
TikTok	https://www.tiktok.com/@degen_traders
LinkedIn	N/A

Audit Summary

Version	Delivery Date	Changelog
v1.0	11. December 2023	<ul style="list-style-type: none"> • Layout Project • Automated- /Manual-Security Testing • Summary
v1.1	11. January 2024	<ul style="list-style-type: none"> • 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
contracts/DegenATM.sol	dc25c91c9ae63285b297e7d4e67daf4749bfec06
contracts/LaunchControl.sol	75101fc254350541770952c5cae2d0a883200e82
contracts/diamond/relayers/RelayerCeler.sol	fc79cfc6c65f93f96fa706f06f6e651bab3f7884
contracts/diamond/DegenX.sol	d2afa250858f3045e6eb003189d0941a87e835a8
contracts/diamond/facets/ERC20Facet.sol	fc6ca5c9350726f1357733144ea05d75b705bc88
contracts/diamond/facets/FeeDistributorFacet.sol	aa81a41e27153e08c83d2a47a9b4e3e58fc154d1
contracts/diamond/facets/FeeManagerFacet.sol	d26609e070d4cbdb8ee31b485ac294c5c0be3e00
contracts/diamond/facets/FeeStoreFacet.sol	63a26eb10f6e9dad49f106e15cc762b47685cd1a
contracts/diamond/facets/CelerFeeHubFacet.sol	05361b7d459029e139502b657ced03da04daf535

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	3
@openzeppelin/contracts/access/Ownable2Step.sol	1
@openzeppelin/contracts/security/ReentrancyGuard.sol	1
@openzeppelin/contracts/token/ERC20/IERC20.sol	3
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	2
@openzeppelin/contracts/utils/Address.sol	5
@solidstate/contracts/interfaces/IERC20.sol	2
@solidstate/contracts/security/pausable/Pausable.sol	1
@solidstate/contracts/token/ERC20/SolidStateERC20.sol	1
@solidstate/contracts/utils/SafeERC20.sol	1
celer/contracts/message/framework/MessageApp.sol	1
celer/contracts/message/libraries/MsgDataTypes.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 as possible.
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 an upgradeable

✗ Deployer can update the contract with new functionalities

Description

The deployer can replace the old contract with a new one with new features. Be aware of this, because the owner can add new features that may have a negative impact on your investments.

Example

We assume that you have funds in the contract and it has been audited by any security audit firm. Now the audit has passed. After that, the deployer can upgrade the contract to allow him to transfer the funds you purchased without any approval from you. This has the consequence that your funds can be taken by the creator.

Comment

The contracts are based on the Diamond Pattern so they are upgradeable by nature

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

N/A

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



Ownership Privileges

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

Minting tokens

Minting tokens 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

According to the functionality of the contract, only the whitelisted addresses will be able to use (deposit funds) the "DegenATM" smart contract.

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 can set fees greater than 25%

✗ The owner able to set high fee

Description

For example, a decentralized exchange (DEX) smart contract may charge a fee for each trade executed on the platform. This fee can be set by the owner of the contract and may be a percentage of the trade value or a flat fee. In other cases, the owner of the smart contract may set fees for accessing or using certain features of the contract. For instance, a subscription-based service smart contract may charge a monthly or yearly fee for access to premium features.

Overall, fees set by the owner of a smart contract can provide an additional source of revenue for the contract's owner and can help to ensure the sustainability of the contract over time.

Example

Our assumption is that the owner can adjust the fees up to 100%. If the transfer fee is set to 100%, it implies that the full amount of tokens you intend to send will be sent to the address specified as the recipient in the contract. This implies that the recipient will never have the intended amount of tokens in their wallet as it has all been used up in paying for the transfer fee.

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.

Contract owner can lock the user funds		✗ The owner is able to lock the contract
Description	Locking the contract means that the owner is able to lock any funds of addresses that they are not able to transfer bought tokens anymore.	
Example	An example of locking is by pausing the ERC20 contract or removing addresses from the whitelist of the ATM contract. Moreover, the owner can also lock the functions in the “DegenATM” cotntract	
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 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
9	0	0	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
126	16

External	Internal	Private	Pure	View
123	115	0	5	41

StateVariables

Total	 Public
39	29



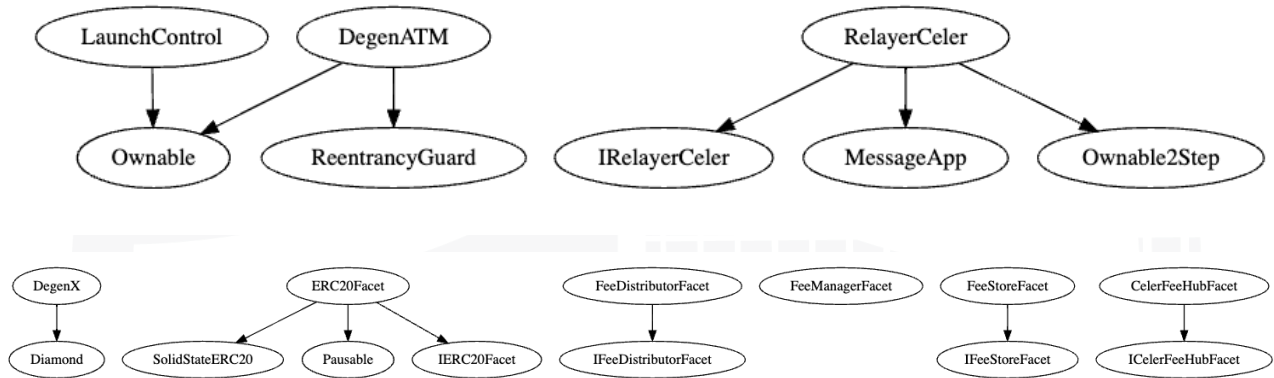
Capabilities

Solidity Versions observed	Transfers ETH	💰 Can Receive Funds	💻 Uses Assembly	💣 Has Destroyable Contracts
^0.8.19 ^0.8.17	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 a single entity controls the contract 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
CelerFeeHubFacet	<ul style="list-style-type: none"> • Add/Remove Relay address for a specific Chain • Set the threshold of the total fees that can be sent to the home chain • Set the amount of fees used to initiate the “send fees”, and “deploy fees” processes
ERC20Facet	<ul style="list-style-type: none"> • Only the bridge address can mint tokens • Enable/Disable the functionality of the contract at any time • Set LP token address • Add/Remove buy and sell fees • Update the bridge supply cap for a particular bridge address
FeeDistributorFacet	<ul style="list-style-type: none"> • Add/Remove Fee distributor receiver • Start/Stop Fee Distribution • Set Fee distributor bounty share • Enable/Disable bounty in token
FeeManagerFacet	<ul style="list-style-type: none"> • Add/Remove chains and target addresses • Add/Remove a fee configuration where the fees could be 100% or more • Manually Assign/Unassign a fee config to a chain • Manually clear the queue and remove all current jobs
FeeStoreFacet	<ul style="list-style-type: none"> • Collect Fees from fee store • Set Operator and Intermediate Asset addresses • Deposit a Fee manually

File	Privileges
RelayerCeler	<ul style="list-style-type: none"> • Forward Refund to receiver in the case of stuck tokens • Add/Remove actor addresses to the relayer
DegenATM	<ul style="list-style-type: none"> • Enable/Disable Claiming and Collecting • Start Lock Period • Recover any type of tokens from the contract's balance • Set Allocation Rate and Limit to any arbitrary value • Set the Payout Token Address • Add/Remove accounts from whitelist
LaunchControl	<ul style="list-style-type: none"> • Recover tokens from the contract • Set Router and Token Address • Add Liquidity • Set the Token Amount to start the pool with • Start Trading but cannot stop it once started

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 | No Upper Limit on Fee

File	Severity	Location	Status
FeeFacetManager	Medium	L82,100	ACK

Description - The contract states that the minimum value of the fee in the new configuration must be greater than zero, but there is no check for the upper limit of the fees. Hence, the fees could be set up to 100% or more, which is not recommended as it may cause the users to lose their funds.

Remediation - We recommend putting an upper limit on the fees of 25%

Alleviation - Defined Fees are not being used as relative fees only. Fees can also be used as absolute values, depending on the context of it. A limit hasn't been considered because of its future business use cases. Therefore, a fee definition needs to be without any limitations.

#2 | Owner can lock user funds

File	Severity	Location	Status
DegenATM	Medium	L267	Partially Fixed

Description - The contract controls the claiming of tokens, which means the owner can enable/disable the claiming at any time. Moreover, the owner can also stop the deposits and has the power to change the payout token address.

However, the function checks that it shouldn't be changeable mid-claim because, as we stated, the owner also controls claiming in the contract, so it is possible to change the payout token at any time at the owner's discretion, making the check redundant.

On the other hand, the owner can also manipulate the rewards to some extent by calling the "startLockPeriod" more than once, as every time this function is called, it will set the "startTimestamp" as the "block.timestamp" value.

Remediation - We recommend putting proper checks so that the owner cannot lock the contract directly. For example, claiming should only be stopped once all the users have claimed their tokens.



Alleviation - Fixed the possible extension of the startTimestamp. Everything else has been considered to stay like it is right now, manageable/changeable.



Low issues

#1 | Missing Address Validation

File	Severity	Location	Status
FeeDistributorFacet	Low	L408	Fixed

Description - Make sure to validate that the bounty receiver address must not be zero or the dead address.

#2 | Missing Events

File	Severity	Location	Status
LaunchControl	Low	L33, 44—85	ACK

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

#3 | Missing Value Check

File	Severity	Location	Status
FeeDistributorFacet	Low	L407, 430, 472, 487	Fixed

Description - The contract should check that the “_amountLeft” and “totalPoints” values should not be zero; otherwise, the call will fail.

#4 | Missing “isContract” check

File	Severity	Location	Status
RelayerCeler	Low	L69, 162, 208	ACK

Description - The contract has no checks to verify whether an EOA or contract is being passed as the executor address because if the executor of the address is passed as a contract with no functionality to receive ETH, then the call will fail

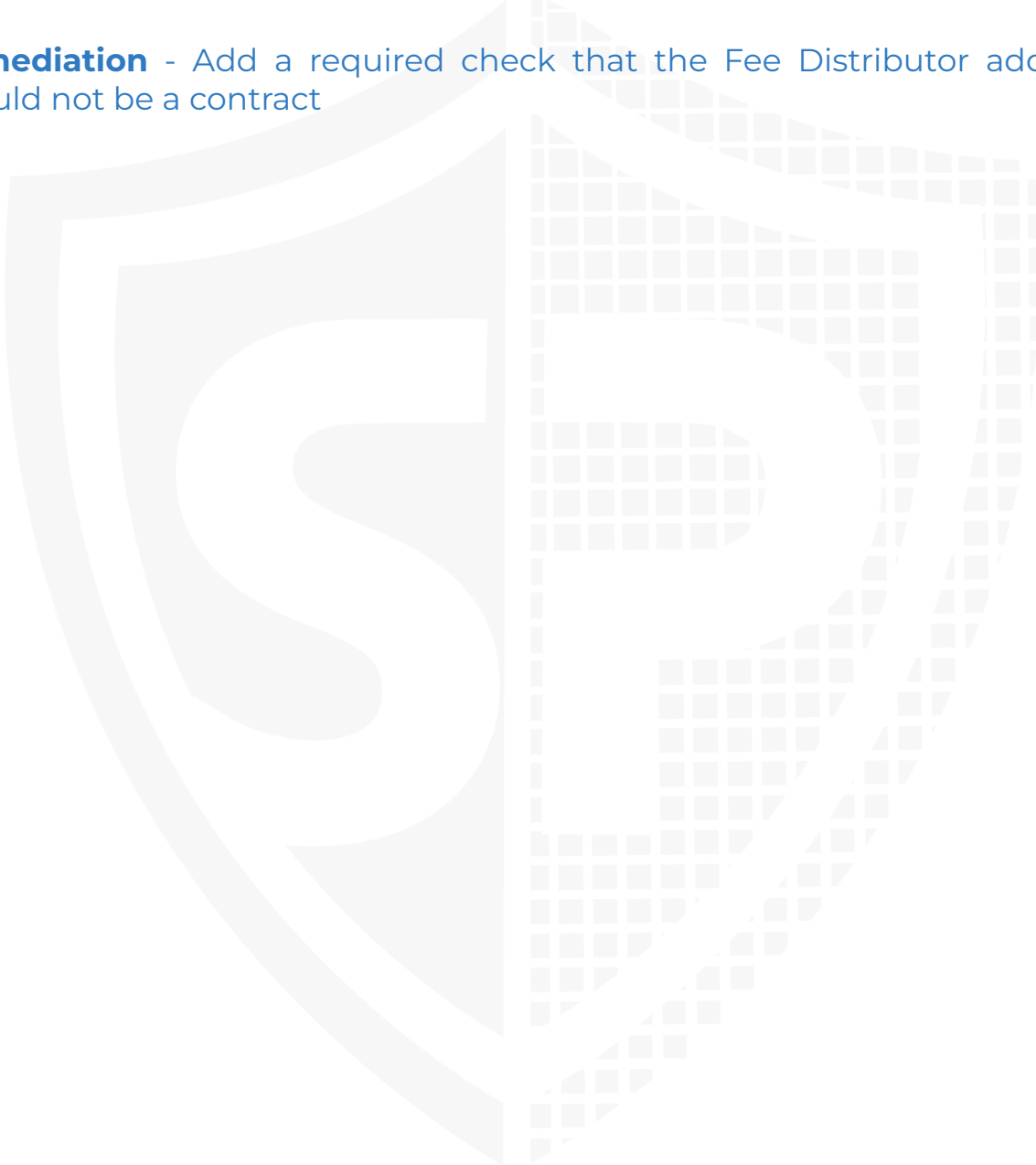


#5 | Logical Error

File	Severity	Location	Status
FeeDistributorFacet	Low	L144	Fixed

Description - The function doesn't verify whether the caller is an EOA or a contract. If "msg.sender" is a contract, then, in that case, the call will fail.

Remediation - Add a required check that the Fee Distributor address should not be a contract



Informational issues

#1 | Unemitted Events

File	Severity	Location	Status
ERC20Facet	Informational	L24, 25	Fixed

Description - The events are declared here but never emitted in the code. We recommend either removing them or implementing them.

#2 | Floating Pragma

File	Severity	Location	Status
All	Informational	N/A	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.

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

File	Severity	Location	Status
All	Informational	N/A	ACK

Description - We recommend importing all the 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.
Partially Fixed	The issue has been fixed partially.



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

MADE IN GERMANY