SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC
Development | Marketing

MADE IN GERMANY

Prisma DeFi

Audit

Security Assessment
15. May, 2023

For

SolidProof_io

@solidproof_io

Disclaimer

SolidProof.io reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc’...)

SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug- free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

SolidProof.io Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. SolidProof’s position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of security or functionality of the technology we agree to analyze.

Version	Date	Description
1.0	13. May 2023	· Layout project · Automated- /Manual-Security Testing · Summary
1.1	15. May 2023	· Reaudit

Network

Binance Smart Chain (BEP20)

Website

https://prisma.fund/

Telegram

Twitter

https://twitter.com/Prisma_DeFi

Github

https://github.com/PrismaDeFi/prisma-v1

Discord

https://discord.gg/xhhknEUgXb

Youtube

https://www.youtube.com/@PrismaFinance

Description

Prisma Finance is making massive strides to bring real yield to the crypto space. We are doing this with our Investment Trading Fund or ITF. Our ITF is how we diversify farming strategies such as Liquidity Farming, Staking, Lending, Bot Trading and Dollar Cost Average. We then convert those rewards earned into Stablecoins and pay those out to the holders without them having to stake or do anything more than just hold Prisma.

All of the vaults in our system are optional and do not have any lock up periods as we want to keep things simple and flexible for the holders. Our primary goal is to make as much of a user friendly experience as we can while taking the burden away from the holder. Please feel free to read our white paper to find out more!

Project Engagement

During the 10th of May 2023, **Prisma DeFi Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

Logo

Contract Link

v1.0

- Github
    - https://github.com/PrismaDeFi/prisma-v1/tree/main/contracts
    - Commit: https://github.com/PrismaDeFi/prisma-v1/commit/a3c8ae381460ddf3fb13bc1d514eacafb28572f1

v1.1

- Github
  - https://github.com/PrismaDeFi/prisma-v1/tree/main/contracts
  - Commit: 48b3c7e

Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

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

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
    - i) Review of the specifications, sources, and instructions provided to SolidProof to make sure we understand the size, scope, and functionality of the smart contract.
    - ii) Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
    - iii) Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to SolidProof describe.

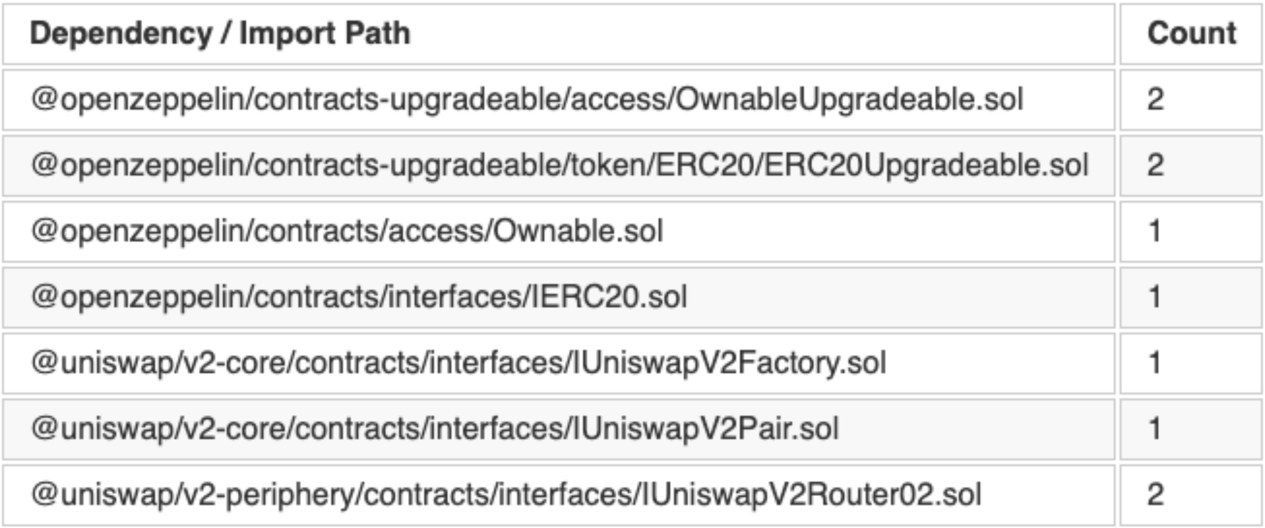
2. Testing and automated analysis that includes the following:
    - i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    - ii) Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.

3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

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

Imported packages:



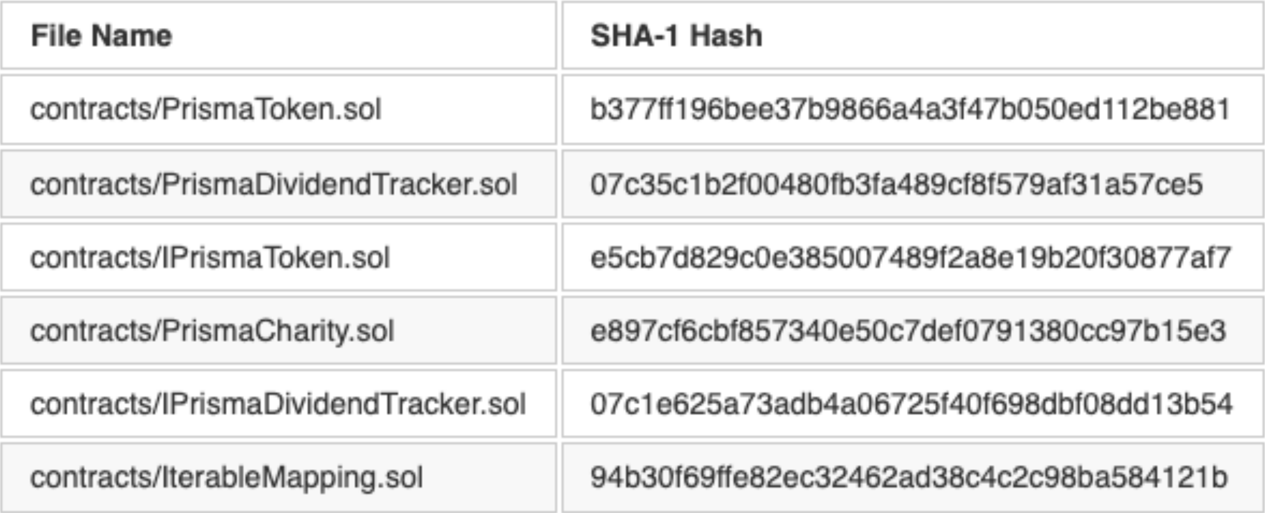
Dependency / Import Path	Count
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	2
@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol	2
@openzeppelin/contracts/access/Ownable.sol	1
@openzeppelin/contracts/interfaces/IERC20.sol	1
@uniswap/v2-core/contracts/interfaces/IUniswapV2Factory.sol	1
@uniswap/v2-core/contracts/interfaces/IUniswapV2Pair.sol	1
@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router02.sol	2

Tested Contract Files

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

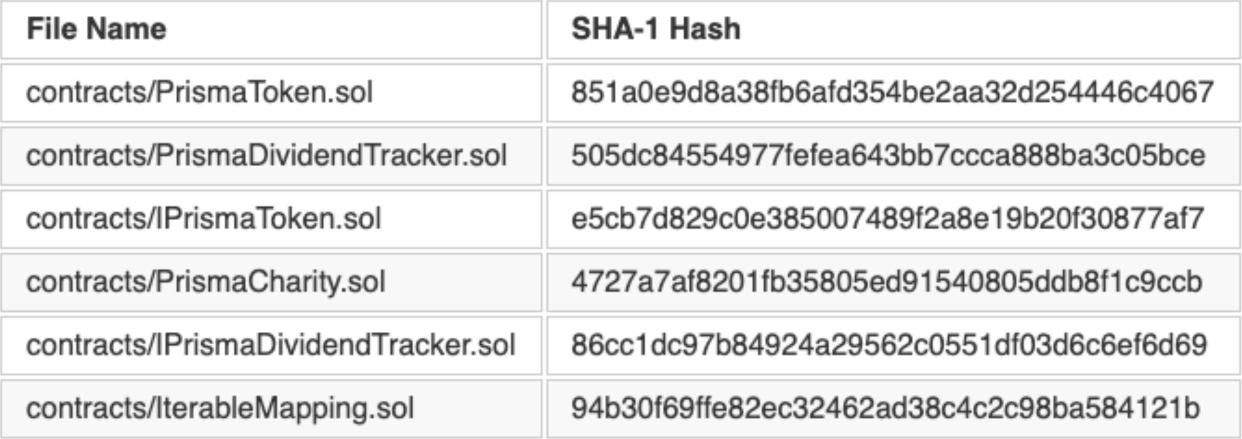
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0



File Name	SHA-1 Hash
contracts/PrismaToken.sol	b377ff196bee37b9866a4a3f47b050ed112be881
contracts/PrismaDividendTracker.sol	07c35c1b2f00480fb3fa489cf8f579af31a57ce5
contracts/IPrismaToken.sol	e5cb7d829c0e385007489f2a8e19b20f30877af7
contracts/PrismaCharity.sol	e897cf6cbf857340e50c7def0791380cc97b15e3
contracts/IPrismaDividendTracker.sol	07c1e625a73adb4a06725f40f698dbf08dd13b54
contracts/IterableMapping.sol	94b30f69ffe82ec32462ad38c4c2c98ba584121b

v1.1

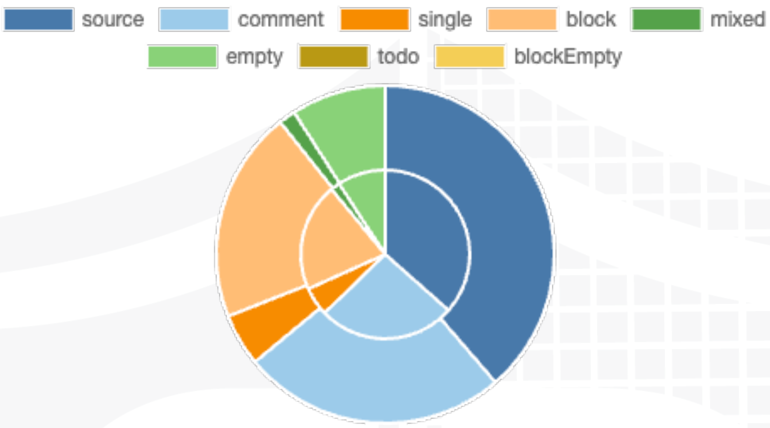


File Name	SHA-1 Hash
contracts/PrismaToken.sol	851a0e9d8a38fb6afd354be2aa32d254446c4067
contracts/PrismaDividendTracker.sol	505dc84554977fefea643bb7ccca888ba3c05bce
contracts/IPrismaToken.sol	e5cb7d829c0e385007489f2a8e19b20f30877af7
contracts/PrismaCharity.sol	4727a7af8201fb35805ed91540805ddb8f1c9ccb
contracts/IPrismaDividendTracker.sol	86cc1dc97b84924a29562c0551df03d6c6ef6d69
contracts/IterableMapping.sol	94b30f69ffe82ec32462ad38c4c2c98ba584121b

Metrics

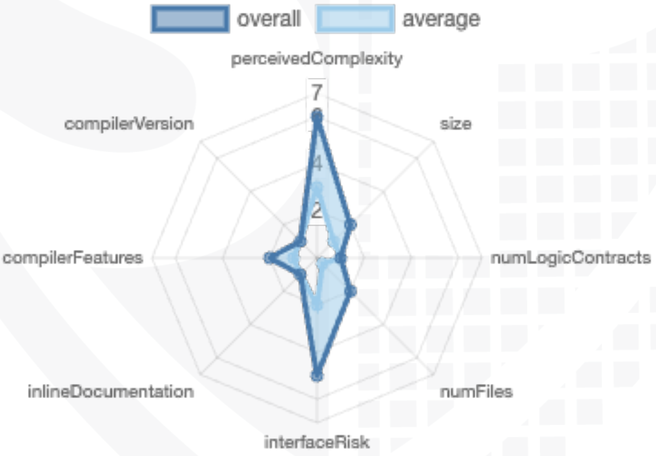
Source Lines

v1.1



Risk Level

v1.1



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.1	3	1	2	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.

Version	Public	Payable
1.1	99	0

Version	External	Internal	Private	Pure	View
1.1	77	72	0	1	60

State Variables

Version	Total	Public
1.1	40	0

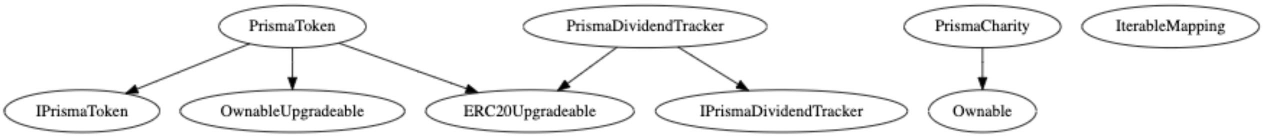
Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.1	0.8.18				

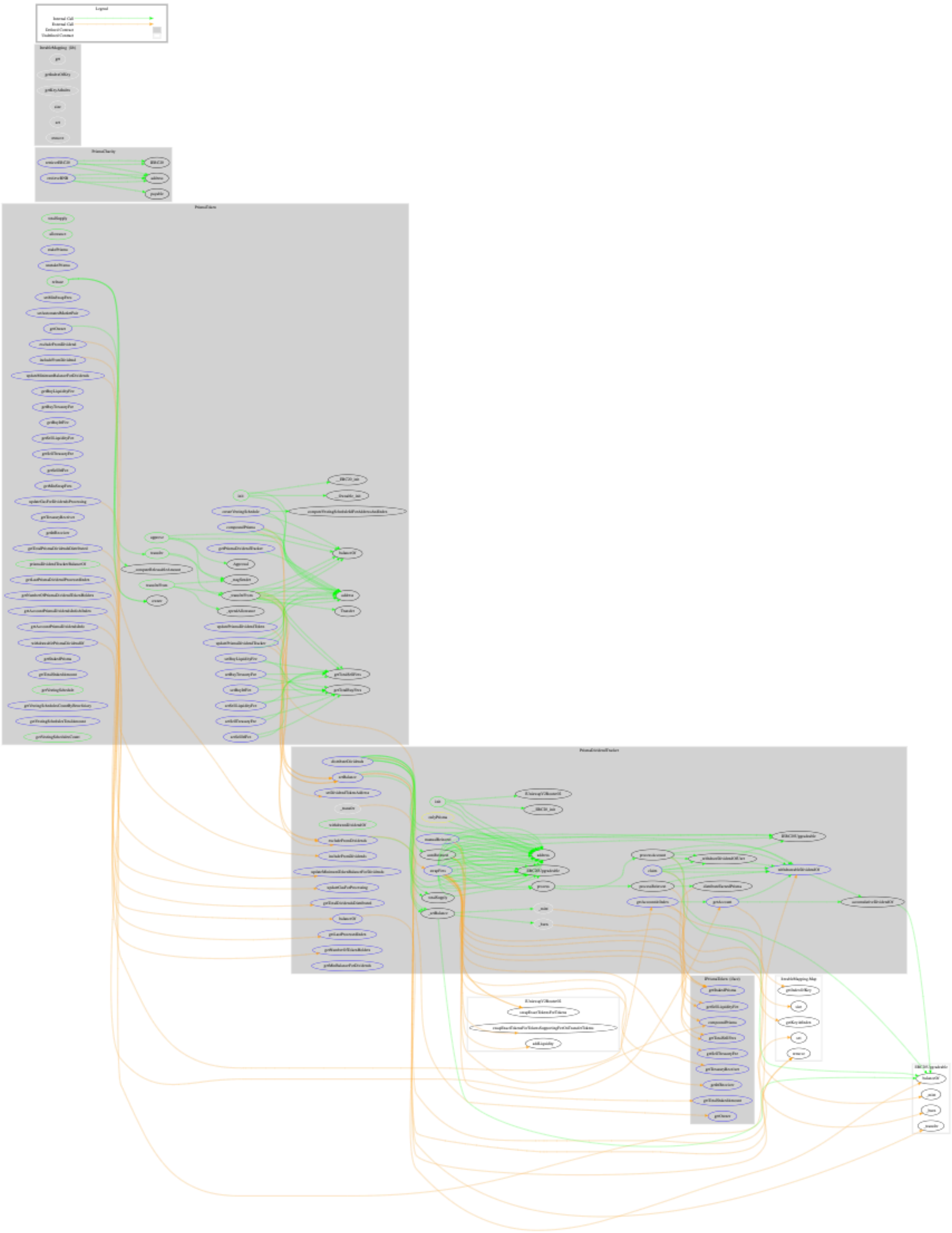
Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/Create/Create2
1.1	`yes`			`yes`		

Inheritance Graph

v1.1



CallGraph
v1.1



Scope of Work/Verify Claims

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

We will verify the following claims:

1. Is contract an upgradeable
2. Correct implementation of Token standard
3. Deployer cannot mint any new tokens
4. Deployer cannot burn or lock user funds
5. Deployer cannot pause the contract
6. Deployer cannot set fees
7. Deployer cannot blacklist/antisnipe addresses
8. Overall checkup (Smart Contract Security)

Is contract an upgradeable

Name	
Is contract an upgradeable?	Yes

Comments:

v1.0

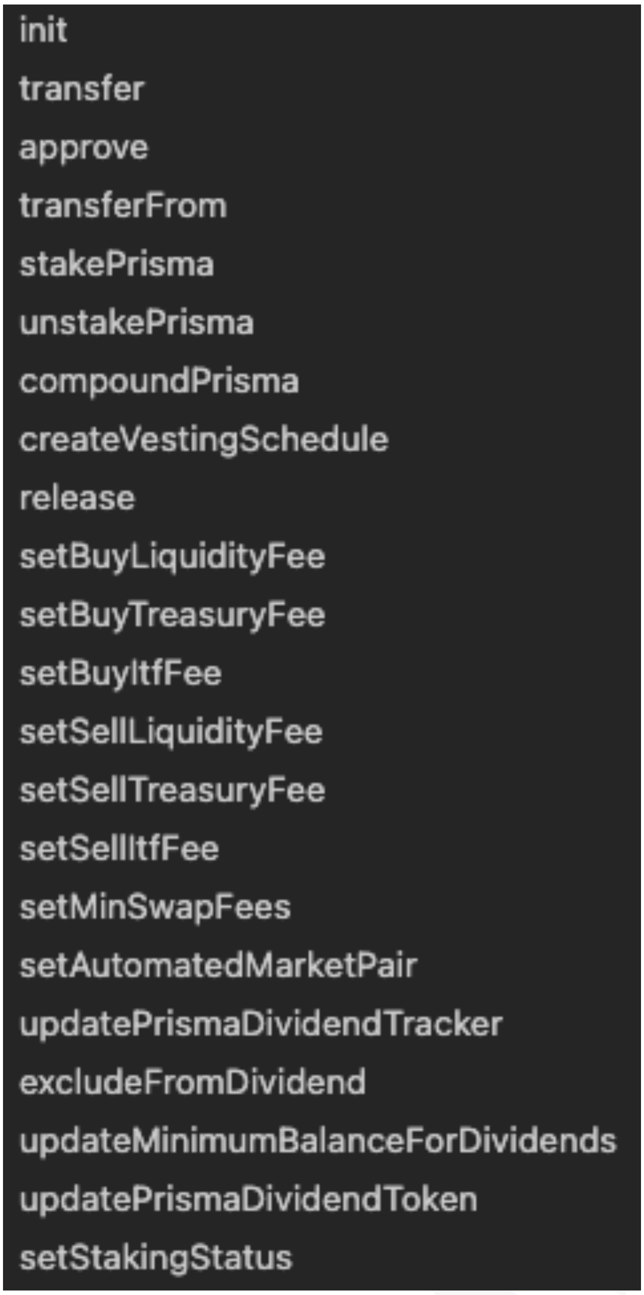
- Owner can deploy a new version of the contract which can change any limit and give owner new privileges
  - Be aware of this and do your own research for the contract which is the contract pointing to

Correct implementation of Token standard

ERC20				
Function	**Description**	**Exist**	**Tested**	**Verified**
TotalSupply	Provides information about the total token supply	✓	✓	✓
BalanceOf	Provides account balance of the owner's account	✓	✓	✓
Transfer	Executes transfers of a specified number of tokens to a specified address	✓	✓	✓
TransferFrom	Executes transfers of a specified number of tokens from a specified address	✓	✓	✓
Approve	Allow a spender to withdraw a set number of tokens from a specified account	✓	✓	✓
Allowance	Returns a set number of tokens from a spender to the owner	✓	✓	✓

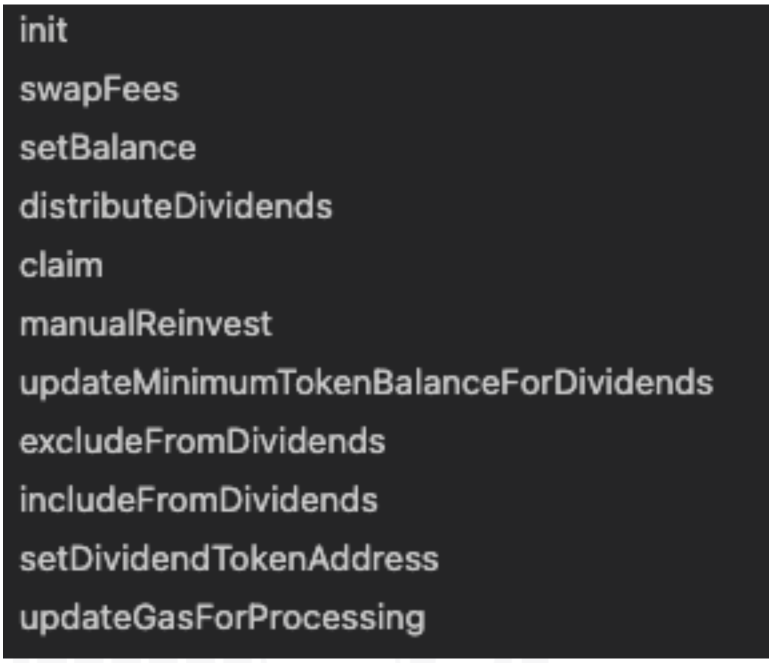
Write functions of contract v1.0

PrismaToken



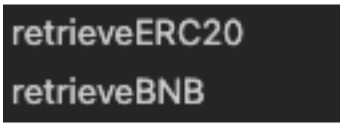
```
init
transfer
approve
transferFrom
stakePrisma
unstakePrisma
compoundPrisma
createVestingSchedule
release
setBuyLiquidityFee
setBuyTreasuryFee
setBuyItfFee
setSellLiquidityFee
setSellTreasuryFee
setSellItfFee
setMinSwapFees
setAutomatedMarketPair
updatePrismaDividendTracker
excludeFromDividend
updateMinimumBalanceForDividends
updatePrismaDividendToken
setStakingStatus
```

PrismaDividendTracker



```
init
swapFees
setBalance
distributeDividends
claim
manualReinvest
updateMinimumTokenBalanceForDividends
excludeFromDividends
includeFromDividends
setDividendTokenAddress
updateGasForProcessing
```

PrismaCharity

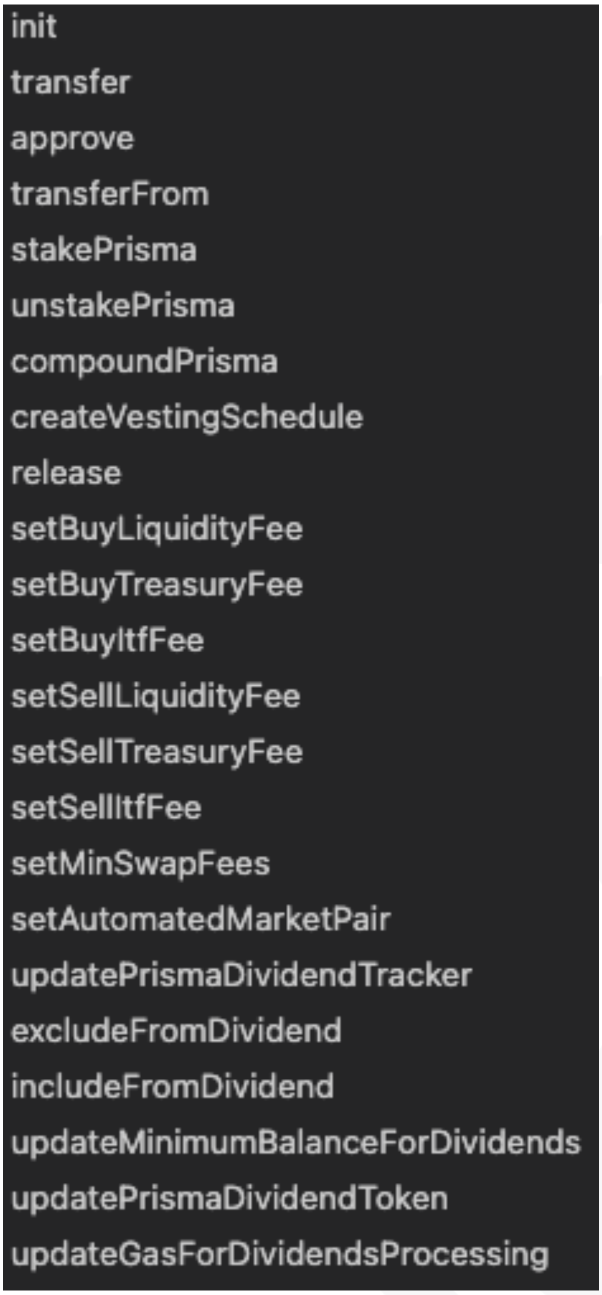


```
retrieveERC20
retrieveBNB
```

Note: Functions imported from official libraries haven’t been listed here

v1.1

PrismaToken



```
init
transfer
approve
transferFrom
stakePrisma
unstakePrisma
compoundPrisma
createVestingSchedule
release
setBuyLiquidityFee
setBuyTreasuryFee
setBuyItfFee
setSellLiquidityFee
setSellTreasuryFee
setSellItfFee
setMinSwapFees
setAutomatedMarketPair
updatePrismaDividendTracker
excludeFromDividend
includeFromDividend
updateMinimumBalanceForDividends
updatePrismaDividendToken
updateGasForDividendsProcessing
```

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✓
Max / Total Supply	10_000_000		

Comments:

v1.0

- Dividendtracker is minting new tokens with “setBalance” function but the owner cannot mint new tokens directly. The “setBalance” will only be called in the “_transferFrom” and “compundPrisma” function

Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer cannot lock	✓	✓	✓
Deployer cannot burn	–	–	–

Comments:

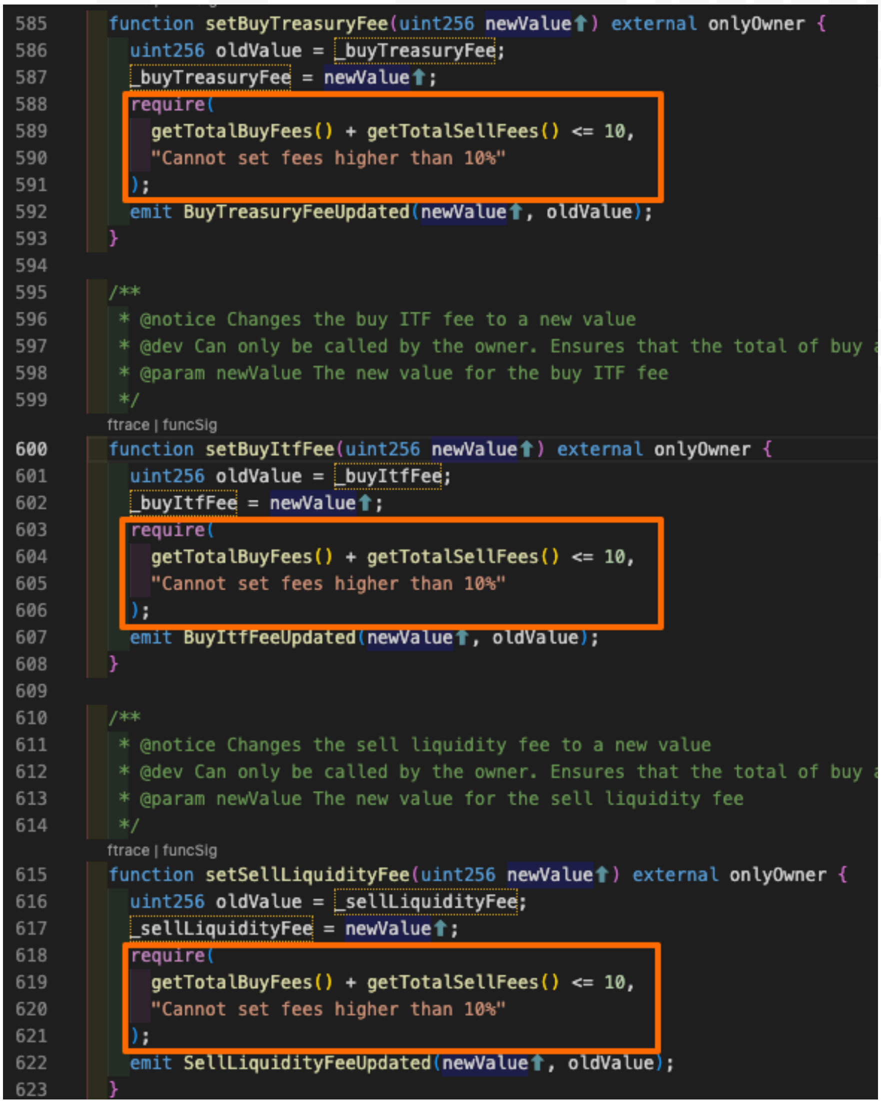
v1.0

- Owner can lock user funds by
  - Setting fees to above 100%. For more information please read the “deployer cannot set fees” section down below on page 21.

v1.1

Resolved

The PrismaDeFi team capped the fees to a max of 10%.



```
function setBuyTreasuryFee(uint256 newValue) external onlyOwner {
  uint256 oldValue = _buyTreasuryFee;
  _buyTreasuryFee = newValue;
  require(
    getTotalBuyFees() + getTotalSellFees() <= 10,
    "Cannot set fees higher than 10%"
  );
  emit BuyTreasuryFeeUpdated(newValue, oldValue);
}

/**
 * @notice Changes the buy ITF fee to a new value
 * @dev Can only be called by the owner. Ensures that the total of buy
 * @param newValue The new value for the buy ITF fee
 */
ftrace | funcSig
function setBuyItfFee(uint256 newValue) external onlyOwner {
  uint256 oldValue = _buyItfFee;
  _buyItfFee = newValue;
  require(
    getTotalBuyFees() + getTotalSellFees() <= 10,
    "Cannot set fees higher than 10%"
  );
  emit BuyItfFeeUpdated(newValue, oldValue);
}

/**
 * @notice Changes the sell liquidity fee to a new value
 * @dev Can only be called by the owner. Ensures that the total of buy
 * @param newValue The new value for the sell liquidity fee
 */
ftrace | funcSig
function setSellLiquidityFee(uint256 newValue) external onlyOwner {
  uint256 oldValue = _sellLiquidityFee;
  _sellLiquidityFee = newValue;
  require(
    getTotalBuyFees() + getTotalSellFees() <= 10,
    "Cannot set fees higher than 10%"
  );
  emit SellLiquidityFeeUpdated(newValue, oldValue);
}
```

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	✓	✓	✓

Comments:

v1.0

- Owner can pause the staking for buyers.

v1.1

Resolved

The function “setStakingStatus” has been removed by the team. The contract cannot be paused anymore.

Deployer cannot set fees

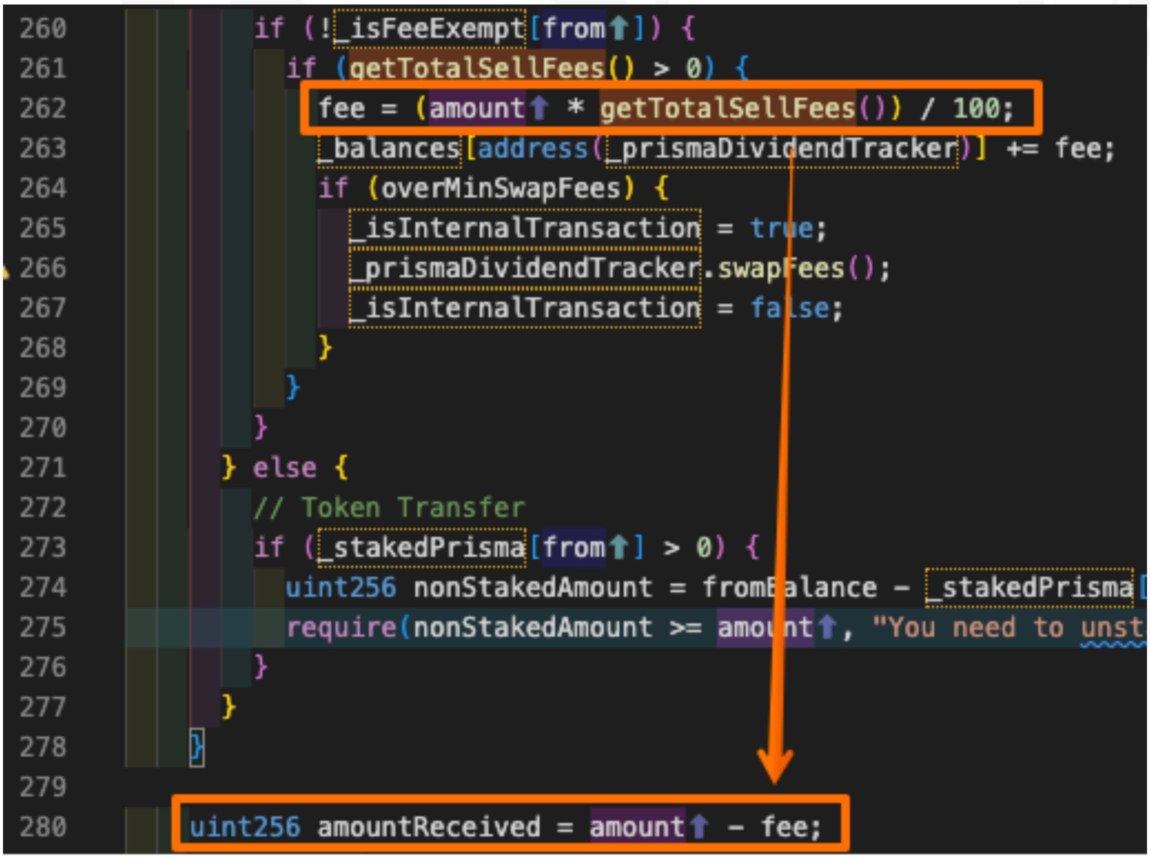
Name	Exist	Tested	Status
Deployer cannot set fees over 25%	✓	✓	✓
Deployer cannot set fees to nearly 100% or to 100%	✓	✓	✓

Comments:

v1.0

- The owner is able to set fees to an arbitrary value. That means that the owner can set the fees up to 100%. It is recommended to prevent setting fees above 25% with require statements.

  In the contract the sellFees for example (function “getTotalSellFees”) when the sum of the fees are above 100% the transfer will be reverted because the amount will be substracte by the fees. This causes that the amount is lesser than the fee and the TX reverted with an underflow issue which is handled by solidity pragma version above 0.8.x by default. Additionally when the buy fees are 100% the buyer will not get any tokens.



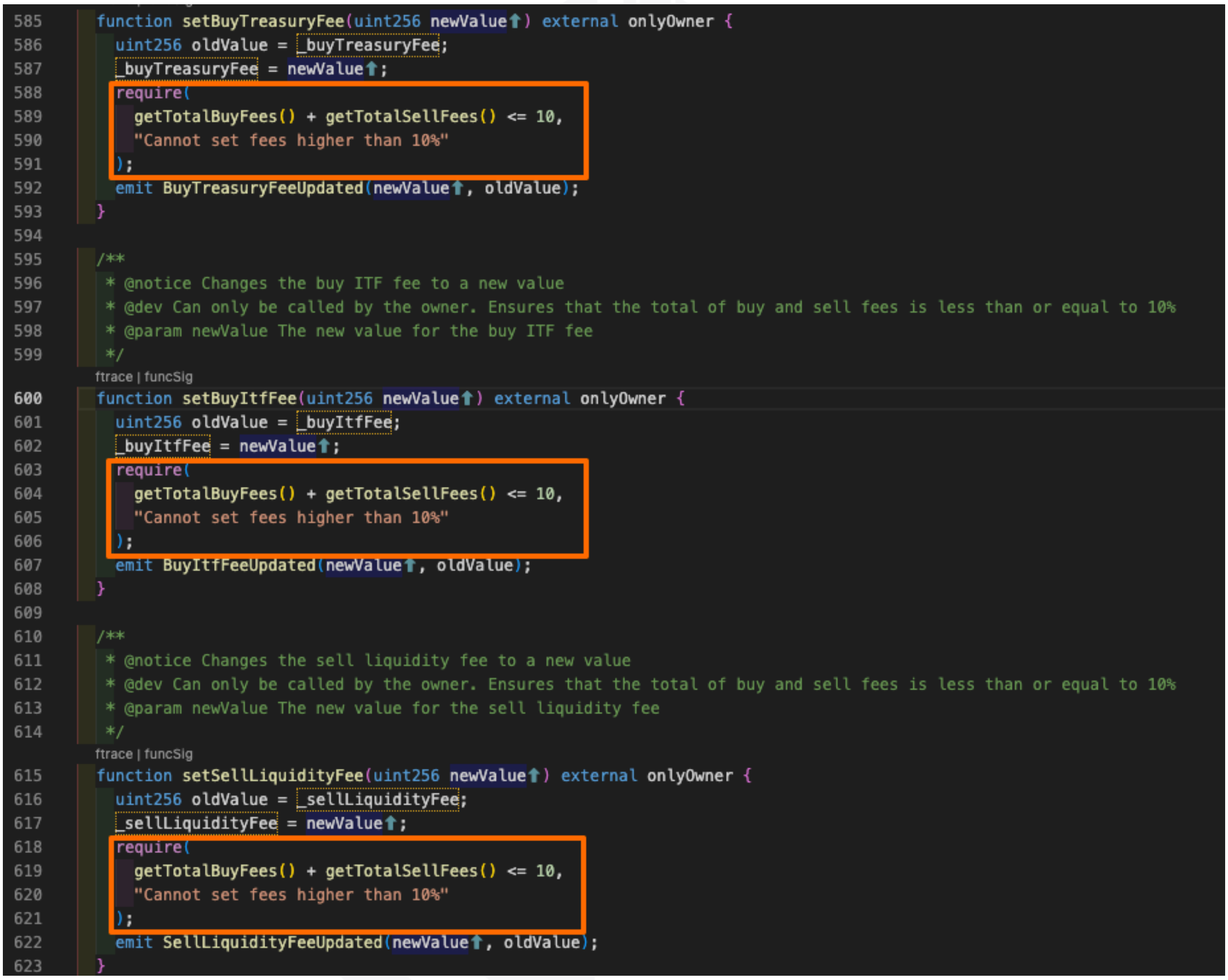
```
        if (!_isFeeExempt[from]) {
            if (getTotalSellFees() > 0) {
                fee = (amount * getTotalSellFees()) / 100;
                _balances[address(_prismaDividendTracker)] += fee;
                if (overMinSwapFees) {
                    _isInternalTransaction = true;
                    _prismaDividendTracker.swapFees();
                    _isInternalTransaction = false;
                }
            }
        }
    } else {
        // Token Transfer
        if (_stakedPrisma[from] > 0) {
            uint256 nonStakedAmount = fromBalance - _stakedPrisma[
            require(nonStakedAmount >= amount, "You need to unst
        }
    }
}

uint256 amountReceived = amount - fee;
```

v1.1

Resolved

The set fee functions have been modified with “require” statements which prevent setting overall fees above the value of 10%. That causes that the lock above is not possible anymore because the fee cannot be higher than the set amount.



```
function setBuyTreasuryFee(uint256 newValue) external onlyOwner {
    uint256 oldValue = _buyTreasuryFee;
    _buyTreasuryFee = newValue;
    require(
        getTotalBuyFees() + getTotalSellFees() <= 10,
        "Cannot set fees higher than 10%"
    );
    emit BuyTreasuryFeeUpdated(newValue, oldValue);
}

/**
 * @notice Changes the buy ITF fee to a new value
 * @dev Can only be called by the owner. Ensures that the total of buy and sell fees is less than or equal to 10%
 * @param newValue The new value for the buy ITF fee
 */
ftrace | funcSig
function setBuyItfFee(uint256 newValue) external onlyOwner {
    uint256 oldValue = _buyItfFee;
    _buyItfFee = newValue;
    require(
        getTotalBuyFees() + getTotalSellFees() <= 10,
        "Cannot set fees higher than 10%"
    );
    emit BuyItfFeeUpdated(newValue, oldValue);
}

/**
 * @notice Changes the sell liquidity fee to a new value
 * @dev Can only be called by the owner. Ensures that the total of buy and sell fees is less than or equal to 10%
 * @param newValue The new value for the sell liquidity fee
 */
ftrace | funcSig
function setSellLiquidityFee(uint256 newValue) external onlyOwner {
    uint256 oldValue = _sellLiquidityFee;
    _sellLiquidityFee = newValue;
    require(
        getTotalBuyFees() + getTotalSellFees() <= 10,
        "Cannot set fees higher than 10%"
    );
    emit SellLiquidityFeeUpdated(newValue, oldValue);
}
```

Deployer can blacklist/antisnipe addresses

Name	Exist	Tested	Status
Deployer cannot blacklist/antisnipe addresses	–	–	–

Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

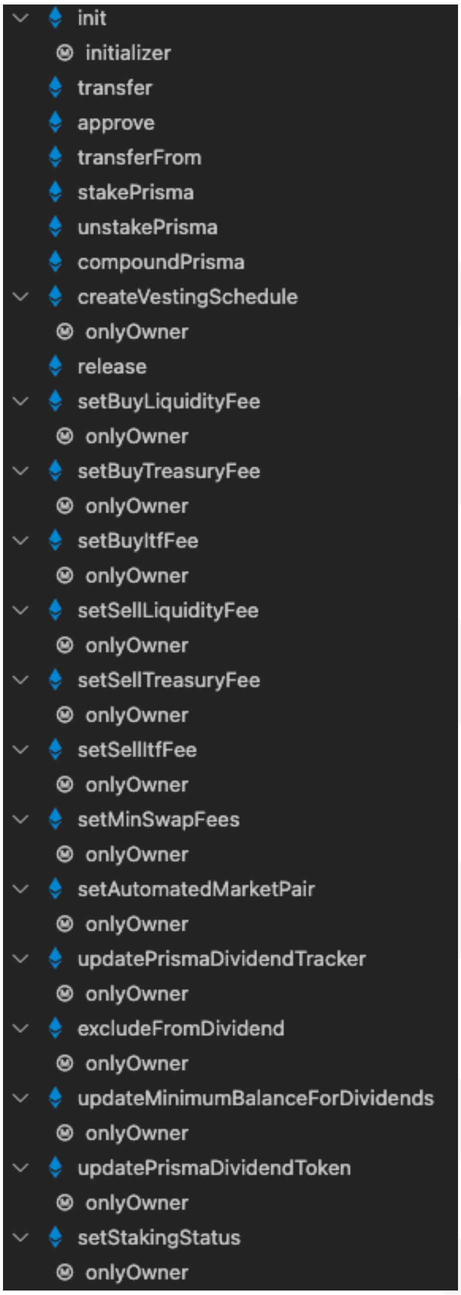
Legend

Attribute	Symbol
Verified / Checked	✓
Partly Verified	⚑
Unverified / Not checked	✗
Not available	–

Modifiers and public functions

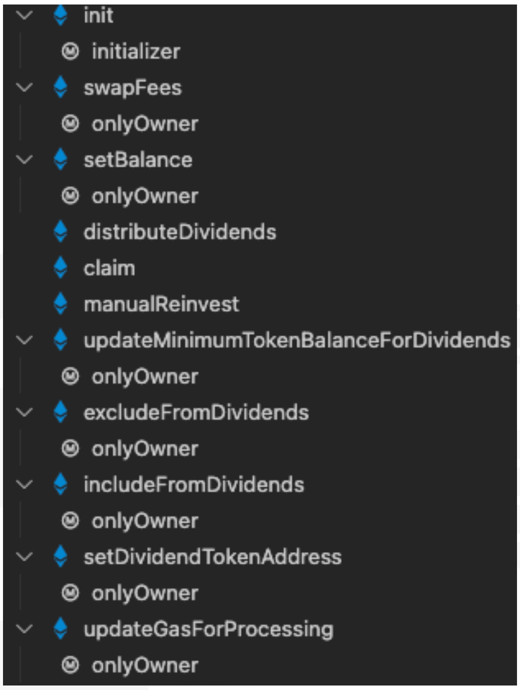
v1.0

PrismaToken



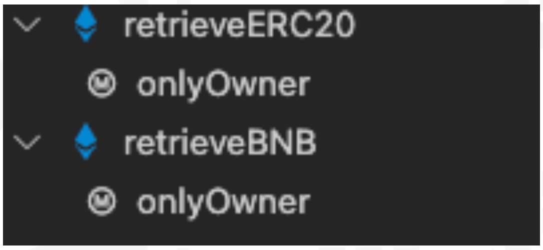
```
init
    initializer
transfer
approve
transferFrom
stakePrisma
unstakePrisma
compoundPrisma
createVestingSchedule
    onlyOwner
release
setBuyLiquidityFee
    onlyOwner
setBuyTreasuryFee
    onlyOwner
setBuyItfFee
    onlyOwner
setSellLiquidityFee
    onlyOwner
setSellTreasuryFee
    onlyOwner
setSellItfFee
    onlyOwner
setMinSwapFees
    onlyOwner
setAutomatedMarketPair
    onlyOwner
updatePrismaDividendTracker
    onlyOwner
excludeFromDividend
    onlyOwner
updateMinimumBalanceForDividends
    onlyOwner
updatePrismaDividendToken
    onlyOwner
setStakingStatus
    onlyOwner
```

PrismaDividendTracker



```
init
    initializer
swapFees
    onlyOwner
setBalance
    onlyOwner
distributeDividends
claim
manualReinvest
updateMinimumTokenBalanceForDividends
    onlyOwner
excludeFromDividends
    onlyOwner
includeFromDividends
    onlyOwner
setDividendTokenAddress
    onlyOwner
updateGasForProcessing
    onlyOwner
```

PrismaCharity



```
retrieveERC20
    onlyOwner
retrieveBNB
    onlyOwner
```

Note: The functions from official libraries haven't been listed here

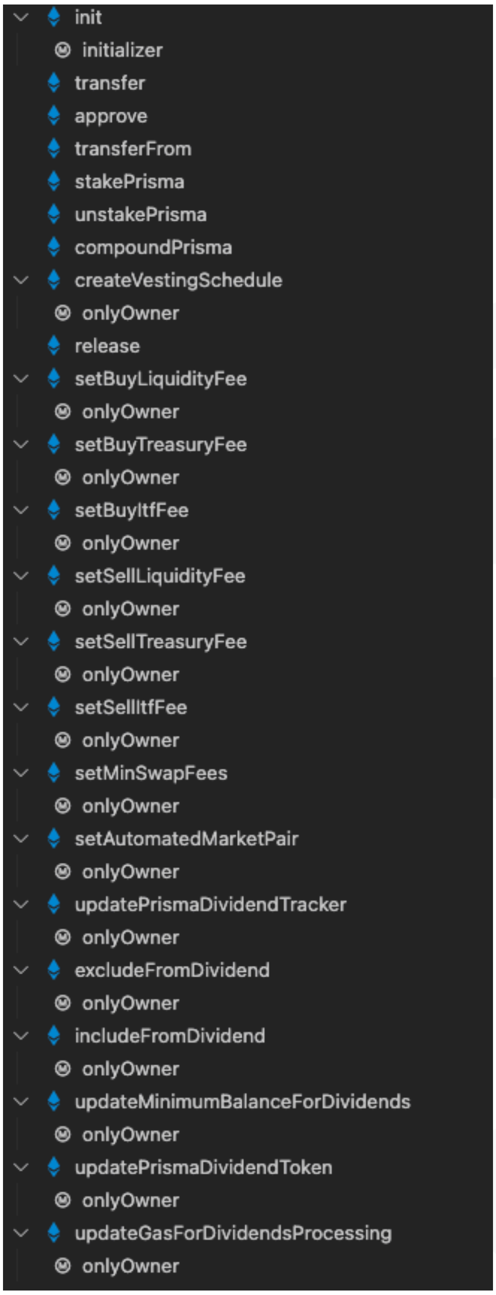
Comments

- PrismaToken
    - setStakingStatus
        - Enable/disable staking of prisma
    - updatePrismaDividendToken
        - Update the current dividend token address
    - updateMinimumBalanceForDividends
        - Update the minimum balance of the tracker. This can be set to an arbitrary value without limitation
    - excludeFromDividend
        - Excluded an address from dividends
    - updatePrismaDividendTracker
        - Update the current dividend tracker address. Make sure that the dividend token address is set while updating the tracker. Ensure that the owner of the dividend tracker should be always the prismaToken address otherwise the swapFees function will not work properly
    - setAutomatedMarketPair
        - Set an automated market pair
    - setMinSwapFees
        - Min swap fees
    - setSellItfFee
        - Sell fees
    - setSellTreasuryFee
        - Treasury fees
    - setSellLiquidityFee
        - Liquidity fees for sells
    - setBuyItfFee
        - Buy fees
    - setBuyTreasuryFee
        - Buy fees of treasury
    - setBuyLiquidityFee
        - Liquidity buy fees
    - createVestingSchedule
        - Creates a new vesting schedule for a beneficiary
    - init
        - Initialize function while deploying a new contract to set the variables
- PrismaCharity
    - retrieveERC20
        - The owner is able to take out every token that is held by the charity contract. The tokens of the prisma proxy can only be retrieved when the balance of the prismaproxy address minus the amount is higher than 200_000 * 10^18
    - retrieveBNB

        - The owner is able to sen native funds of the charity contract to an arbitrary destination.
- PrismaDividendTracker
    - Owner should be the prismaToken itself
    - updateGasForProcessing
        - Update gas for processing
    - setDividendTokenAddress
        - Update dividend token
    - updateMinimumTokenBalanceForDividends
        - Update minimum token balance for the dividends
    - excludeFromDividends
        - Excludes addresses from dividends
    - includeFromDividends
        - Include addresses in dividends

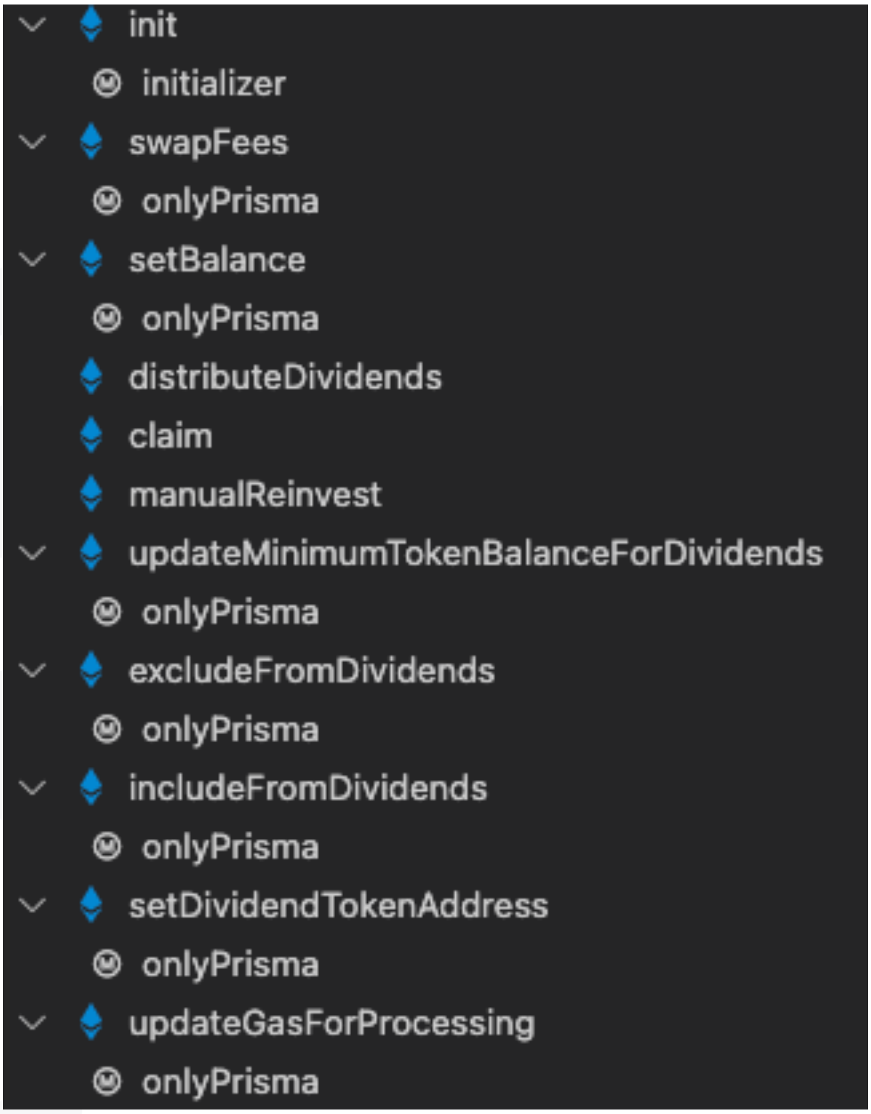
v1.1

PrismaToken



```
init
  initializer
transfer
approve
transferFrom
stakePrisma
unstakePrisma
compoundPrisma
createVestingSchedule
  onlyOwner
release
setBuyLiquidityFee
  onlyOwner
setBuyTreasuryFee
  onlyOwner
setBuyItfFee
  onlyOwner
setSellLiquidityFee
  onlyOwner
setSellTreasuryFee
  onlyOwner
setSellItfFee
  onlyOwner
setMinSwapFees
  onlyOwner
setAutomatedMarketPair
  onlyOwner
updatePrismaDividendTracker
  onlyOwner
excludeFromDividend
  onlyOwner
includeFromDividend
  onlyOwner
updateMinimumBalanceForDividends
  onlyOwner
updatePrismaDividendToken
  onlyOwner
updateGasForDividendsProcessing
  onlyOwner
```

PrismaDividendTracker



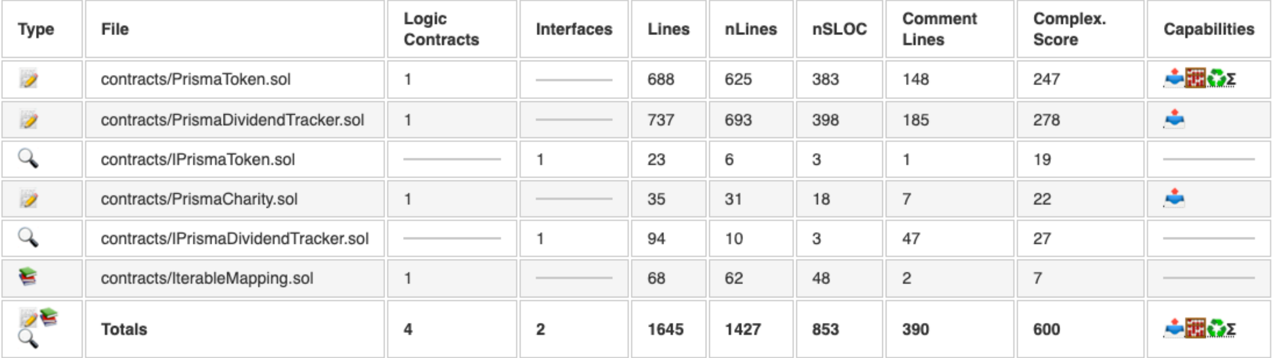
```
init
  initializer
swapFees
  onlyPrisma
setBalance
  onlyPrisma
distributeDividends
claim
manualReinvest
updateMinimumTokenBalanceForDividends
  onlyPrisma
excludeFromDividends
  onlyPrisma
includeFromDividends
  onlyPrisma
setDividendTokenAddress
  onlyPrisma
updateGasForProcessing
  onlyPrisma
```

- setStakingStatus has been removed
- onlyOwner has been renamed to onlyPrisma in the dividend tracker

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

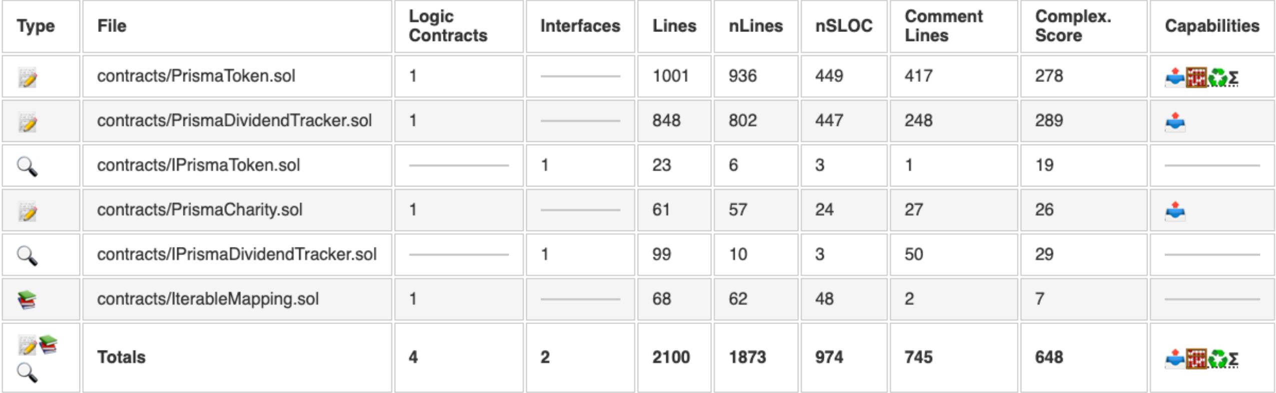
Source Units in Scope

v1.0



Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/PrismaToken.sol	1	----	688	625	383	148	247	📤🧮♻️Σ
📝	contracts/PrismaDividendTracker.sol	1	----	737	693	398	185	278	📤
🔍	contracts/IPrismaToken.sol	----	1	23	6	3	1	19	----
📝	contracts/PrismaCharity.sol	1	----	35	31	18	7	22	📤
🔍	contracts/IPrismaDividendTracker.sol	----	1	94	10	3	47	27	----
📚	contracts/IterableMapping.sol	1	----	68	62	48	2	7	----
📝📚🔍	**Totals**	**4**	**2**	**1645**	**1427**	**853**	**390**	**600**	📤🧮♻️Σ

v1.1



Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/PrismaToken.sol	1	----	1001	936	449	417	278	📤🧮♻️Σ
📝	contracts/PrismaDividendTracker.sol	1	----	848	802	447	248	289	📤
🔍	contracts/IPrismaToken.sol	----	1	23	6	3	1	19	----
📝	contracts/PrismaCharity.sol	1	----	61	57	24	27	26	📤
🔍	contracts/IPrismaDividendTracker.sol	----	1	99	10	3	50	29	----
📚	contracts/IterableMapping.sol	1	----	68	62	48	2	7	----
📝📚🔍	**Totals**	**4**	**2**	**2100**	**1873**	**974**	**745**	**648**	📤🧮♻️Σ

Legend

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

Audit Results

Critical issues

No critical issues

High issues

No high issues

Medium issues

No medium issues

Low issues

No low issues

Informational issues

No informational issues

Audit Comments

15. May 2023:

- Owner can deploy a new version of the contract which can change any limit and give owner new privileges
- Read whole report and modifiers section for more information

SWC Attacks

ID	Title	Relationships	Status
SWC-136	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SWC-135	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SWC-134	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SWC-133	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SWC-132	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	PASSED
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	PASSED
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	PASSED
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	PASSED

SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	PASSED
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED

SolidProof_io @solidproof_io

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY