DEFIMOON

be secure

Smart Contract Audit Report

September, 2022

Ayoken

DEFIMOON PROJECT

Audit and Development

CONTACTS

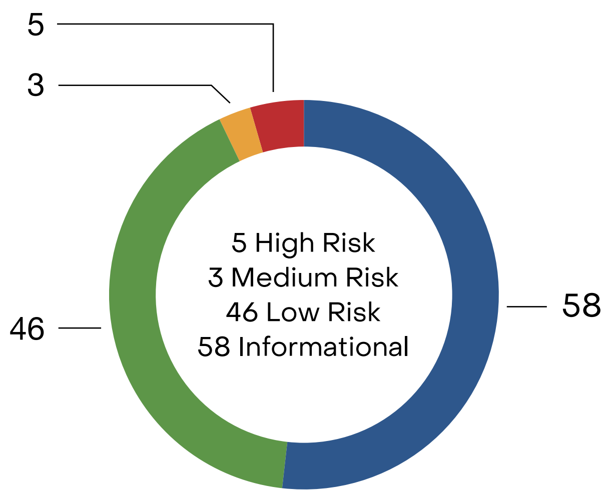
https://defimoon.org
audit@defimoon.org
defimoon_org
defimoonorg
defimoon
defimoonorg

DEFIMOON
be secure

September 15th 2022
This audit report was prepared by Defimoon for Ayoken

Audit information

Timeline	15th September
Audited by	Cyrill Novoseletskyi
Approved by	Artur Makhnach, Cyrill Minyaev
Languages	Solidity
Methods	Automated Testing, Manual Review
Source code	https://bscscan.com/address/0x5d9a10221938Bbe1b86D84d10e9b92C6E14007C4#code
Network	Binance smart chain
Status	Risks Acknowledged



🟥	High Risk	A fatal vulnerability that can cause the loss of all Tokens / Funds.
🟧	Medium Risk	A vulnerability that can cause the loss of some Tokens / Funds.
🟩	Low Risk	A vulnerability which can cause the loss of protocol functionality.
🔷	Informational	Non-security issues such as functionality, style, and convention.

Disclaimer

This audit is not financial, investment, or any other kind of advice and could be used for informational purposes only. This report is not a substitute for doing your own research and due diligence should always be paid in full to any project. Defimoon is not responsible or liable for any loss, damage, or otherwise caused by reliance on this report for any purpose. Defimoon has based this audit report solely on the information provided by the audited party and on facts that existed before or during the audit being conducted. Defimoon is not responsible for any outcome, including changes done to the contract/contracts after the audit was published. This audit is fully objective and only discerns what the contract is saying without adding any opinion to it. Defimoon has no connection to the project other than the conduction of this audit and has no obligations other than to publish an objective report. Defimoon will always publish its findings regardless of the outcome of the findings. The audit only covers the subject areas detailed in this report and unless specifically stated, nothing else has been audited. Defimoon assumes that the provided information and materials were not altered, suppressed, or misleading. This report is published by Defimoon, and Defimoon has sole ownership of this report. Use of this report for any reason other than for informational purposes on the subjects reviewed in this report including the use of any part of this report is prohibited without the express written consent of Defimoon. In instances where an auditor or team member has a personal connection with the audited project, that auditor or team member will be excluded from viewing or impacting any internal communication regarding the specific audit.

Audit Information

Defimoon utilizes both manual and automated auditing approach to cover the most ground possible. We begin with generic static analysis automated tools to quickly assess the overall state of the contract. We then move to a comprehensive manual code analysis, which enables us to find security flaws that automated tools would miss. Finally, we conduct an extensive unit testing to make sure contract behaves as expected under stress conditions.

In our decision making process we rely on finding located via the manual code inspection and testing. If an automated tool raises a possible vulnerability, we always investigate it further manually to make a final verdict. All our tests are run in a special test environment which matches the "real world" situations and we utilize exact copies of the published or provided contracts.

While conducting the audit, the Defimoon security team uses best practices to ensure that the reviewed contracts are thoroughly examined against all angles of attack. This is done by evaluating the codebase and whether it gives rise to significant risks. During the audit, Defimoon assesses the risks and assigns a risk level to each section together with an explanatory comment.

Files Report

DropERC20.sol:

Severity: High Risk

Status: Acknowledged

Reentrancy in DropERC20.`claim(address,uint256,address,uint256,bytes32[],uint256)` (contracts/DropERC20.sol#177-235):

- External calls:
  - `collectClaimPrice(_quantity,_currency,_pricePerToken)`(contracts/DropERC20.sol#229)
    - `returndata = address(token).functionCall(data,SafeERC20: low-level call failed)` (contracts/openzeppelin-presets/token/ERC20/utils/SafeERC20.sol#93)
    - `(success) = to.call{value: value}()` (contracts/lib/CurrencyTransferLib.sol#84)
    - `(success,returndata) = target.call{value: value}(data)` (contracts/lib/TWAddress.sol#137)
    - `IERC20(_currency).safeTransfer(_to,_amount)` (contracts/lib/CurrencyTransferLib.sol#74)
    - `IERC20(_currency).safeTransferFrom(_from,_to,_amount)` (contracts/lib/CurrencyTransferLib.sol#76)
    - `currencyTransferLib.transferCurrency(_currency,_msgSender(),platformFeeRecipient,platformFees)` (contracts/DropERC20.sol#324)
    - `CurrencyTransferLib.transferCurrency(_currency,_msgSender(),primarySaleRecipient,totalPrice - platformFees)` (contracts/DropERC20.sol#325)
- External calls sending eth:
  - `collectClaimPrice(_quantity,_currency,_pricePerToken)` (contracts/DropERC20.sol#229)
    - `(success) = to.call{value: value}()` (contracts/lib/CurrencyTransferLib.sol#84)
    - `(success,returndata) = target.call{value: value}(data)` (contracts/lib/TWAddress.sol#137)
- State variables written after the call(s):
  - `transferClaimedTokens(_receiver,activeConditionId,_quantity)` (contracts/DropERC20.sol#232)
    - `_totalSupply += amount` (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#267)
  - `transferClaimedTokens(_receiver,activeConditionId,_quantity)` (contracts/DropERC20.sol#232)
    - `claimCondition.phases[_conditionId].supplyClaimed += _quantityBeingClaimed` (contracts/DropERC20.sol#335)
    - `claimCondition.limitLastClaimTimestamp[_conditionId][_msgSender()] = block.timestamp` (contracts/DropERC20.sol#339)
  - `transferClaimedTokens(_receiver,activeConditionId,_quantity)` (contracts/DropERC20.sol#232)
    - `walletClaimCount[_msgSender()] += _quantityBeingClaimed` (contracts/DropERC20.sol#340)

Client Comment:

The fact that this is an automated security tool, it will read any transfer done by a msg.sender (connected wallet other than the owner of the contract) as a threat for the contract but in fact without a Claim Phase no one can interact with these two functions, and these functions are added to do a vesting for the investors over multiple months.

Severity: Medium Risk

Status: Acknowledged

- DropERC20.`collectClaimPrice(uint256,address,uint256)` (contracts/DropERC20.sol#305-326) performs a multiplication on the result of a division:
  - `totalPrice = (_quantityToClaim * _pricePerToken) / 1000000000000000000` (contracts/DropERC20.sol#315)
  - `platformFees = (totalPrice * platformFeeBps) / MAX_BPS` (contracts/DropERC20.sol#318)
- DropERC20`.setClaimConditions(IDropClaimCondition.ClaimCondition[],bool)` (contracts/DropERC20.sol#238-302) deletes BitMapsUpgradeable.`BitMap` (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/BitMapsUpgradeable.sol#10-12) which contains a mapping:
  - delete `claimCondition.limitMerkleProofClaim[i_scope_0]` (contracts/DropERC20.sol#290)
- DropERC20`.setClaimConditions(IDropClaimCondition.ClaimCondition[],bool)` (contracts/DropERC20.sol#238-302) deletes BitMapsUpgradeable.`BitMap` (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/BitMapsUpgradeable.sol#10-12) which contains a mapping:
  - delete c`laimCondition.limitMerkleProofClaim[newStartIndex + i_scope_1]` (contracts/DropERC20.sol#296)

Severity: Low Risk

Status: Acknowledged

- DropERC20.`initialize(address,string,string,string,address[],address,address,uint256)._name` (contracts/DropERC20.sol#98) shadows:
  - ERC20Upgradeable.`_name` (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#43) (state variable)
- DropERC20`.initialize(address,string,string,string,address[],address,address,uint256)._symbol` (contracts/DropERC20.sol#99) shadows:
  - ERC20Upgradeable.`_symbol` (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#44) (state variable)
- Low level call in TWAddress.`sendValue(address,uint256)` (contracts/lib/TWAddress.sol#60-65):
  - `(success) = recipient.call{value: amount}()` (contracts/lib/TWAddress.sol#63)
- Low level call in TWAddress.`functionCallWithValue(address,bytes,uint256,string)` (contracts/lib/TWAddress.sol#128-139):
  - `(success,returndata) = target.call{value: value}(data)` (contracts/lib/TWAddress.sol#137)
- Low level call in TWAddress.`functionStaticCall(address,bytes,string)` (contracts/lib/TWAddress.sol#157-166):
  - `(success,returndata) = target.staticcall(data)` (contracts/lib/TWAddress.sol#164)
- Low level call in TWAddress`.functionDelegateCall(address,bytes,string)` (contracts/lib/TWAddress.sol#184-193):
  - `(success,returndata) = target.delegatecall(data)` (contracts/lib/TWAddress.sol#191
- Parameter DropERC20.initialize(address,string,string,string,address[],address,address,uint256)._defaultAdmin (contracts/DropERC20.sol#97) is not in mixedCase
- Parameter DropERC20.initialize(address,string,string,string,address[],address,address,uint256)._name (contracts/DropERC20.sol#98) is not in mixedCase

- Parameter DropERC20.initialize(address,string,string,string,address[],address,address,uint256)._symbol (contracts/DropERC20.sol#99) is not in mixedCase
- Parameter DropERC20.initialize(address,string,string,string,address[],address,address,uint256)._contractURI (contracts/DropERC20.sol#100) is not in mixedCase
- Parameter DropERC20.`initialize(address,string,string,string,address[],address,address,uint256)`._trustedForwarders (contracts/DropERC20.sol#101) is not in mixedCase
- Parameter DropERC20.`initialize(address,string,string,string,address[],address,address,uint256)`._primarySaleRecipient (contracts/DropERC20.sol#102) is not in mixedCase
- Parameter DropERC20.`initialize(address,string,string,string,address[],address,address,uint256)`._platformFeeRecipient (contracts/DropERC20.sol#103) is not in mixedCase
- Parameter DropERC20`initialize(address,string,string,string,address[],address,address,uint256)`._platformFeeBps (contracts/DropERC20.sol#104) is not in mixedCase
- Parameter DropERC20`.claim(address,uint256,address,uint256,bytes32[],uint256).`_receiver (contracts/DropERC20.sol#178) is not in mixedCase
- Parameter `DropERC20.claim(address,uint256,address,uint256,bytes32[],uint256)._quantity` (contracts/DropERC20.sol#179) is not in mixedCase
- Parameter DropERC20.`claim(address,uint256,address,uint256,bytes32[],uint256)._currency` (contracts/DropERC20.sol#180) is not in mixedCase
- Parameter DropERC20.`claim(address,uint256,address,uint256,bytes32[],uint256)._pricePerToken` (contracts/DropERC20.sol#181) is not in mixedCase
- Parameter DropERC20.`claim(address,uint256,address,uint256,bytes32[],uint256)._proofs` (contracts/DropERC20.sol#182) is not in mixedCase
- Parameter DropERC20.`claim(address,uint256,address,uint256,bytes32[],uint256)._proofMaxQuantityPerTransaction` (contracts/DropERC20.sol#183) is not in mixedCase
- Parameter DropERC20.`setClaimConditions(IDropClaimCondition.ClaimCondition[],bool)._phases` (contracts/DropERC20.sol#238) is not in mixedCase
- Parameter DropERC20.`setClaimConditions(IDropClaimCondition.ClaimCondition[],bool)._resetClaimEligibility` (contracts/DropERC20.sol#238) is not in mixedCase
- Parameter DropERC20`.collectClaimPrice(uint256,address,uint256)._quantityToClaim` (contracts/DropERC20.sol#306) is not in mixedCase

- Parameter DropERC20.`collectClaimPrice(uint256,address,uint256)._currency` (contracts/DropERC20.sol#307) is not in mixedCase
- Parameter DropERC20.`collectClaimPrice(uint256,address,uint256)._pricePerToken` (contracts/DropERC20.sol#308) is not in mixedCase
- Parameter DropERC20.`transferClaimedTokens(address,uint256,uint256)._to` (contracts/DropERC20.sol#330) is not in mixedCase
- Parameter DropERC20`.transferClaimedTokens(address,uint256,uint256)._conditionId` (contracts/DropERC20.sol#331) is not in mixedCase
- Parameter DropERC20.`transferClaimedTokens(address,uint256,uint256)._quantityBeingClaimed` (contracts/DropERC20.sol#332) is not in mixedCase
- Parameter DropERC20.`verifyClaim(uint256,address,uint256,address,uint256,bool)._conditionId` (contracts/DropERC20.sol#347) is not in mixedCase
- Parameter DropERC20.`verifyClaim(uint256,address,uint256,address,uint256,bool)._claimer` (contracts/DropERC20.sol#348) is not in mixedCase
- Parameter DropERC20.`verifyClaim(uint256,address,uint256,address,uint256,bool)._quantity` (contracts/DropERC20.sol#349) is not in mixedCase
- Parameter DropERC20.`verifyClaim(uint256,address,uint256,address,uint256,bool)._currency` (contracts/DropERC20.sol#350) is not in mixedCase
- Parameter DropERC20`.verifyClaim(uint256,address,uint256,address,uint256,bool)._pricePerToken` (contracts/DropERC20.sol#351) is not in mixedCase
- Parameter DropERC20.`verifyClaimMerkleProof(uint256,address,uint256,bytes32[],uint256)._conditionId` (contracts/DropERC20.sol#385) is not in mixedCase
- Parameter DropERC20.`verifyClaimMerkleProof(uint256,address,uint256,bytes32[],uint256)._claimer` (contracts/DropERC20.sol#386) is not in mixedCase
- Parameter DropERC20`.verifyClaimMerkleProof(uint256,address,uint256,bytes32[],uint256)._quantity` (contracts/DropERC20.sol#387) is not in mixedCase
- Parameter DropERC20`.verifyClaimMerkleProof(uint256,address,uint256,bytes32[],uint256)._proofs` (contracts/DropERC20.sol#388) is not in mixedCase
- Parameter DropERC20.`verifyClaimMerkleProof(uint256,address,uint256,bytes32[],uint256)._proofMaxQuantityPerTransaction` (contracts/DropERC20.sol#389) is not in mixedCase
- Parameter DropERC20.`getClaimTimestamp(uint256,address)._conditionId` (contracts/DropERC20.sol#424) is not in mixedCase
- Parameter DropERC20.`getClaimTimestamp(uint256,address)._claimer` (contracts/DropERC20.sol#424) is not in mixedCase

- Parameter DropERC20.`getClaimConditionById(uint256)._conditionId` (contracts/DropERC20.sol#445) is not in mixedCase
- Parameter DropERC20.`setWalletClaimCount(address,uint256)._claimer` (contracts/DropERC20.sol#459) is not in mixedCase
- Parameter DropERC20.`setWalletClaimCount(address,uint256)._count` (contracts/DropERC20.sol#459) is not in mixedCase
- Function ERC2771ContextUpgradeable.`__ERC2771Context_init(address[])` (contracts/openzeppelin-presets/metatx/ERC2771ContextUpgradeable.sol#15-18) is not in mixedCase
- Function ERC2771ContextUpgradeable.__ERC2771Context_init_unchained(address[]) (contracts/openzeppelin-presets/metatx/ERC2771ContextUpgradeable.sol#20-24) is not in mixedCase
- Variable ERC2771ContextUpgradeable.`__gap` (contracts/openzeppelin-presets/metatx/ERC2771ContextUpgradeable.sol#49) is not in mixedCase

CurrencyTransferLib.sol

Severity: Low Risk
Status: Acknowledged

- TWAddress.`verifyCallResult(bool,bytes,string)` (contracts/lib/TWAddress.sol#201-221) uses assembly
  - INLINE ASM (contracts/lib/TWAddress.sol#213-216)
- CurrencyTransferLib.`safeTransferERC20(address,address,address,uint256)` (contracts/lib/CurrencyTransferLib.sol#63-78) is never used and should be removed
- CurrencyTransferLib.`safeTransferNativeToken(address,uint256)` (contracts/lib/CurrencyTransferLib.sol#81-86) is never used and should be removed
- CurrencyTransferLib.safeTransferNativeTokenWithWrapper(address,uint256,address) (contracts/lib/CurrencyTransferLib.sol#89-101) is never used and should be removed
- CurrencyTransferLib.`transferCurrency(address,address,address,uint256)` (contracts/lib/CurrencyTransferLib.sol#16-31) is never used and should be removed
- CurrencyTransferLib.`transferCurrencyWithWrapper(address,address,address,uint256,address)` (contracts/lib/CurrencyTransferLib.sol#34-60) is never used and should be removed
  - `(success,returndata) = target.delegatecall(data)` (contracts/lib/TWAddress.sol#191)
- Pragma version^0.8.0 (contracts/interfaces/IWETH.sol#2) allows old versions
- Pragma version^0.8.0 (contracts/lib/CurrencyTransferLib.sol#2) allows old versions
- Pragma version^0.8.0 (contracts/lib/TWAddress.sol#4) allows old versions
- Pragma version^0.8.0 (contracts/openzeppelin-presets/token/ERC20/utils/IERC20.sol#2) allows old versions
- Pragma version^0.8.0 (contracts/openzeppelin-presets/token/ERC20/utils/SafeERC20.sol#4) allows old versions

FeeType.sol

Severity: Low Risk
Status: Acknowledged

- Pragma version^0.8.11 (contracts/lib/FeeType.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7

TWAddress.sol

Severity: Low Risk
Status: Acknowledged

- TWAddress `.functionCall(address,bytes)` (contracts/lib/TWAddress.sol#85-87) is never used and should be removed
- TWAddress.`functionCall(address,bytes,string)` (contracts/lib/TWAddress.sol#95-101) is never used and should be removed
- TWAddress.`functionCallWithValue(address,bytes,uint256)` (contracts/lib/TWAddress.sol#114-120) is never used and should be removed
- TWAddress.`functionCallWithValue(address,bytes,uint256,string)` (contracts/lib/TWAddress.sol#128-139) is never used and should be removed
- TWAddress `.functionDelegateCall(address,bytes)` (contracts/lib/TWAddress.sol#174-176) is never used and should be removed
- TWAddress.`functionDelegateCall(address,bytes,string)` (contracts/lib/TWAddress.sol#184-193) is never used and should be removed
- TWAddress.`functionStaticCall(address,bytes)` (contracts/lib/TWAddress.sol#147-149) is never used and should be removed
- TWAddress.`functionStaticCall(address,bytes,string)` (contracts/lib/TWAddress.sol#157-166) is never used and should be removed
- TWAddress.`isContract(address)` (contracts/lib/TWAddress.sol#36-42) is never used and should be removed
- TWAddress.`sendValue(address,uint256)` (contracts/lib/TWAddress.sol#60-65) is never used and should be removed
- TWAddress.`verifyCallResult(bool,bytes,string)` (contracts/lib/TWAddress.sol#201-221) is never used and should be removed
- Low level call in TWAddress `.sendValue(address,uint256)` (contracts/lib/TWAddress.sol#60-65):
  - `(success) = recipient.call{value: amount}()` (contracts/lib/TWAddress.sol#63)
- Low level call in TWAddress.`functionCallWithValue(address,bytes,uint256,string)` (contracts/lib/TWAddress.sol#128-139):

  - (`success,returndata) = target.call{value: value}(data)` (contracts/lib/TWAddress.sol#137)
- Low level call in TWAddress.`functionStaticCall(address,bytes,string)` (contracts/lib/TWAddress.sol#157-166):
  - `(success,returndata) = target.staticcall(data)` (contracts/lib/TWAddress.sol#164)
- Low level call in TWAddress.`functionDelegateCall(address,bytes,string)` (contracts/lib/TWAddress.sol#184-193):

IERC20.sol

It is an interface and there is no logic in this file.

IPlatformFee.sol

It is an interface and there is no logic in this file.

IPrimarySale.sol

It is an interface and there is no logic in this file.

ITWFee.sol

It is an interface and there is no logic in this file.

IThirdwebContract.sol

It is an interface and there is no logic in this file.

IWETH.sol

It is an interface and there is no logic in this file.

IVotesUpgradeable.sol

It is an interface and there is no logic in this file.

IDropClaimCondition.sol

It is an interface and there is no logic in this file.

IDropERC20.sol

It is an interface and there is no logic in this file.

IAccessControlEnumerableUpgradeable.sol

It is an interface and there is no logic in this file.

IAccessControlUpgradeable.sol

It is an interface and there is no logic in this file.

IVotesUpgradeable.sol

It is an interface and there is no logic in this file.

IERC20Upgradeable.sol

It is an interface and there is no logic in this file.

draft-IERC20PermitUpgradeable.sol

It is an interface and there is no logic in this file.

IERC165Upgradeable.sol

It is an interface and there is no logic in this file.

ERC20BurnableUpgradeable.sol

Severity: Low Risk

Status: Acknowledged

- Function ERC20BurnableUpgradeable.`__ERC20Burnable_init()` (contracts/ERC20BurnableUpgradeable.sol#16-17) is not in mixedCase
- Function ERC20BurnableUpgradeable.`__ERC20Burnable_init_unchained()` (contracts/ERC20BurnableUpgradeable.sol#19-20) is not in mixedCase
- Variable ERC20BurnableUpgradeable.`__gap` (contracts/ERC20BurnableUpgradeable.sol#51) is not in mixedCase
- Function ERC20Upgradeable.`__ERC20_init(string,string)` (contracts/ERC20Upgradeable.sol#55-57) is not in mixedCase
- Function ERC20Upgradeable.`__ERC20_init_unchained(string,string)` (contracts/ERC20Upgradeable.sol#59-62) is not in mixedCase
- Variable ERC20Upgradeable.`__gap` (contracts/ERC20Upgradeable.sol#394) is not in mixedCase

Initializable.sol

Severity: Low Risk

Status: Acknowledged

Different versions of Solidity are used:

- Version used: ['^0.8.0', '^0.8.1']
- ^0.8.1 (contracts/AddressUpgradeable.sol#4)
- ^0.8.0 (contracts/Initializable.sol#4)

ReentrancyGuardUpgradeable.sol

Severity: Low Risk

Status: Acknowledged

- Function ReentrancyGuardUpgradeable.`__ReentrancyGuard_init()` (contracts/ReentrancyGuardUpgradeable.sol#40-42) is not in mixedCase
- Function ReentrancyGuardUpgradeable.`__ReentrancyGuard_init_unchained()` (contracts/ReentrancyGuardUpgradeable.sol#44-46) is not in mixedCase
- Variable ReentrancyGuardUpgradeable.`__gap` (contracts/ReentrancyGuardUpgradeable.sol#74) is not in mixedCase

ERC20VotesUpgradeable.sol

Severity: Low Risk

Status: Acknowledged

- ERC20VotesUpgradeable.`__ERC20Votes_init()` (contracts/ERC20VotesUpgradeable.sol#29-30) is never used and should be removed
- ERC20VotesUpgradeable.`__ERC20Votes_init_unchained()` (contracts/ERC20VotesUpgradeable.sol#32-33) is never used and should be removed
- ERC20VotesUpgradeable.`_add(uint256,uint256)` (contracts/ERC20VotesUpgradeable.sol#248-250) is never used and should be removed
- ERC20VotesUpgradeable.`_burn(address,uint256)` (contracts/ERC20VotesUpgradeable.sol#178-182) is never used and should be removed
- ERC20VotesUpgradeable.`_maxSupply()` (contracts/ERC20VotesUpgradeable.sol#161-163) is never used and should be removed
- ERC20VotesUpgradeable.`_mint(address,uint256)` (contracts/ERC20VotesUpgradeable.sol#168-173) is never used and should be removed
- ERC20VotesUpgradeable.`_subtract(uint256,uint256)` (contracts/ERC20VotesUpgradeable.sol#252-254) is never used and should be removed

ERC20Upgradeable.sol

Severity: Low Risk

Status: Acknowledged

- ERC20Upgradeable.`__gap` (contracts/ERC20Upgradeable.sol#394) is never used in ERC20Upgradeable (contracts/ERC20Upgradeable.sol#36-396)
- name() should be declared external:
  - ERC20Upgradeable`.name()` (contracts/ERC20Upgradeable.sol#67-69)
- symbol() should be declared external:
  - ERC20Upgradeable`.symbol()` (contracts/ERC20Upgradeable.sol#75-77)
- decimals() should be declared external:
  - ERC20Upgradeable`.decimals()` (contracts/ERC20Upgradeable.sol#92-94)
- totalSupply() should be declared external:
  - ERC20Upgradeable`.totalSupply()` (contracts/ERC20Upgradeable.sol#99-101)
- balanceOf(address) should be declared external:
  - ERC20Upgradeable.`balanceOf(address)` (contracts/ERC20Upgradeable.sol#106-108)
- transfer(address,uint256) should be declared external:
  - ERC20Upgradeable`.transfer(address,uint256)` (contracts/ERC20Upgradeable.sol#118-122)
- approve(address,uint256) should be declared external:
  - ERC20Upgradeable.`approve(address,uint256)` (contracts/ERC20Upgradeable.sol#141-145)
- transferFrom(address,address,uint256) should be declared external:
  - ERC20Upgradeable.`transferFrom(address,address,uint256)` (contracts/ERC20Upgradeable.sol#163-172)
- increaseAllowance(address,uint256) should be declared external:
  - ERC20Upgradeable.`increaseAllowance(address,uint256)` (contracts/ERC20Upgradeable.sol#186-190)
- decreaseAllowance(address,uint256) should be declared external:
  - ERC20Upgradeable.`decreaseAllowance(address,uint256)` (contracts/ERC20Upgradeable.sol#206-215)

MerkleProof.sol

Severity: Low Risk

Status: Acknowledged

- MerkleProof.`verify(bytes32[],bytes32,bytes32)` (contracts/lib/MerkleProof.sol#25-49) is never used and should be removed
- Pragma version^0.8.0 (contracts/lib/MerkleProof.sol#5) allows old versions

ERC2771ContextUpgradeable.sol

Severity: Low Risk

Status: Acknowledged

- ERC2771ContextUpgradeable.`_msgSender()` (contracts/ERC2771ContextUpgradeable.sol#30-39) uses assembly
  - INLINE ASM (contracts/ERC2771ContextUpgradeable.sol#33-35)
- AddressUpgradeable.`verifyCallResult(bool,bytes,string)` (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#174-194) uses assembly
  - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#186-189)
- ERC2771ContextUpgradeable.`__ERC2771Context_init(address[])` (contracts/ERC2771ContextUpgradeable.sol#15-18) is never used and should be removed
- ERC2771ContextUpgradeable`.__ERC2771Context_init_unchained(address[])` (contracts/ERC2771ContextUpgradeable.sol#20-24) is never used and should be removed
- ERC2771ContextUpgradeable`._msgData()` (contracts/ERC2771ContextUpgradeable.sol#41-47) is never used and should be removed
- ERC2771ContextUpgradeable.`_msgSender()` (contracts/ERC2771ContextUpgradeable.sol#30-39) is never used and should be removed

SafeERC20.sol

Severity: Low Risk
Status: Acknowledged

- SafeERC20.`_callOptionalReturn(IERC20,bytes)` (contracts/openzeppelin-presets/token/ERC20/utils/SafeERC20.sol#88-98) is never used and should be removed
- SafeERC20.`safeApprove(IERC20,address,uint256)` (contracts/openzeppelin-presets/token/ERC20/utils/SafeERC20.sol#45-58) is never used and should be removed
- SafeERC20.`safeDecreaseAllowance(IERC20,address,uint256)` (contracts/openzeppelin-presets/token/ERC20/utils/SafeERC20.sol#69-80) is never used and should be removed
- SafeERC20.`safeIncreaseAllowance(IERC20,address,uint256)` (contracts/openzeppelin-presets/token/ERC20/utils/SafeERC20.sol#60-67) is never used and should be removed
- SafeERC20.`safeTransfer(IERC20,address,uint256)` (contracts/openzeppelin-presets/token/ERC20/utils/SafeERC20.sol#21-27) is never used and should be removed
- SafeERC20.`safeTransferFrom(IERC20,address,address,uint256)` (contracts/openzeppelin-presets/token/ERC20/utils/SafeERC20.sol#29-36) is never used and should be removed

AccessControlEnumerableUpgradeable.sol

Severity: Low Risk
Status: Acknowledged

- `getRoleMember(bytes32,uint256)` should be declared external:
  - AccessControlEnumerableUpgradeable.getRoleMember(bytes32,uint256) (contracts/AccessControlEnumerableUpgradeable.sol#43-45)
- `getRoleMemberCount(bytes32)` should be declared external:
  - AccessControlEnumerableUpgradeable.`getRoleMemberCount(bytes32)` (contracts/AccessControlEnumerableUpgradeable.sol#51-53)
- `grantRole(bytes32,address)` should be declared external:
  - AccessControlUpgradeable.`grantRole(bytes32,address)` (contracts/AccessControlUpgradeable.sol#136-138)
- `revokeRole(bytes32,address)` should be declared external:
  - AccessControlUpgradeable.`revokeRole(bytes32,address)` (contracts/AccessControlUpgradeable.sol#149-151)
- `renounceRole(bytes32,address)` should be declared external:
  - AccessControlUpgradeable.`renounceRole(bytes32,address)` (contracts/AccessControlUpgradeable.sol#167-171)

MathUpgradeable.sol

Severity: Low Risk

Status: Acknowledged

- MathUpgradeable.`average(uint256,uint256)` (contracts/MathUpgradeable.sol#28-31) is never used and should be removed
- MathUpgradeable.`ceilDiv(uint256,uint256)` (contracts/MathUpgradeable.sol#39-42) is never used and should be removed
- MathUpgradeable.`max(uint256,uint256)` (contracts/MathUpgradeable.sol#13-15) is never used and should be removed
- MathUpgradeable`.min(uint256,uint256)` (contracts/MathUpgradeable.sol#20-22) is never used and should be removed

SafeCastUpgradeable.sol

Severity: Low Risk
Status: Acknowledged

- SafeCastUpgradeable.`toInt128(int256)` (contracts/SafeCastUpgradeable.sol#152-155) is never used and should be removed
- SafeCastUpgradeable.`toInt16(int256)` (contracts/SafeCastUpgradeable.sol#206-209) is never used and should be removed
- SafeCastUpgradeable.`toInt256(uint256)` (contracts/SafeCastUpgradeable.sol#236-240) is never used and should be removed
- SafeCastUpgradeable.`toInt32(int256)` (contracts/SafeCastUpgradeable.sol#188-191) is never used and should be removed
- SafeCastUpgradeable.`toInt64(int256)` (contracts/SafeCastUpgradeable.sol#170-173) is never used and should be removed
- SafeCastUpgradeable.`toInt8(int256)` (contracts/SafeCastUpgradeable.sol#224-227) is never used and should be removed
- SafeCastUpgradeable.`toUint128(uint256)` (contracts/SafeCastUpgradeable.sol#47-50) is never used and should be removed
- SafeCastUpgradeable.`toUint16(uint256)` (contracts/SafeCastUpgradeable.sol#107-110) is never used and should be removed
- SafeCastUpgradeable.`toUint224(uint256)` (contracts/SafeCastUpgradeable.sol#32-35) is never used and should be removed
- SafeCastUpgradeable.`toUint256(int256)` (contracts/SafeCastUpgradeable.sol#134-137) is never used and should be removed
- SafeCastUpgradeable.`toUint32(uint256)` (contracts/SafeCastUpgradeable.sol#92-95) is never used and should be removed
- SafeCastUpgradeable.`toUint64(uint256)` (contracts/SafeCastUpgradeable.sol#77-80) is never used and should be removed
- SafeCastUpgradeable.`toUint8(uint256)` (contracts/SafeCastUpgradeable.sol#122-125) is never used and should be removed
- SafeCastUpgradeable.`toUint96(uint256)` (contracts/SafeCastUpgradeable.sol#62-65) is never used and should be removed

BitMapsUpgradeable.sol

Severity: Low Risk
Status: Acknowledged

- BitMapsUpgradeable`.get(BitMapsUpgradeable.BitMap,uint256)` (contracts/BitMapsUpgradeable.sol#17-21) is never used and should be removed
- BitMapsUpgradeable.`set(BitMapsUpgradeable.BitMap,uint256)` (contracts/BitMapsUpgradeable.sol#41-45) is never used and should be removed
- BitMapsUpgradeable.`setTo(BitMapsUpgradeable.BitMap,uint256,bool)` (contracts/BitMapsUpgradeable.sol#26-36) is never used and should be removed
- BitMapsUpgradeable.`unset(BitMapsUpgradeable.BitMap,uint256)` (contracts/BitMapsUpgradeable.sol#50-54) is never used and should be removed

draft-ERC20PermitUpgradeable.sol

Severity: Low Risk

Status: Acknowledged

- Variable ERC20Upgradeable.`__gap` (contracts/ERC20Upgradeable.sol#394) is not in mixedCase
- Function EIP712Upgradeable.`__EIP712_init(string,string)` (contracts/draft-EIP712Upgradeable.sol#48-50) is not in mixedCase
- Function EIP712Upgradeable.`__EIP712_init_unchained(string,string)` (contracts/draft-EIP712Upgradeable.sol#52-57) is not in mixedCase
- Function EIP712Upgradeable.`_EIP712NameHash()` (contracts/draft-EIP712Upgradeable.sol#99-101) is not in mixedCase
- Function EIP712Upgradeable`._EIP712VersionHash()` (contracts/draft-EIP712Upgradeable.sol#109-111) is not in mixedCase
- Variable EIP712Upgradeable`._HASHED_NAME` (contracts/draft-EIP712Upgradeable.sol#30) is not in mixedCase
- Variable EIP712Upgradeable.`_HASHED_VERSION` (contracts/draft-EIP712Upgradeable.sol#31) is not in mixedCase
- Variable EIP712Upgradeable.`__gap` (contracts/draft-EIP712Upgradeable.sol#118) is not in mixedCase
- Function ERC20PermitUpgradeable`.__ERC20Permit_init(string)` (contracts/draft-ERC20PermitUpgradeable.sol#36-39) is not in mixedCase
- Function ERC20PermitUpgradeable.`__ERC20Permit_init_unchained(string)` (contracts/draft-ERC20PermitUpgradeable.sol#41-42) is not in mixedCase
- Function ERC20PermitUpgradeable.`DOMAIN_SEPARATOR()` (contracts/draft-ERC20PermitUpgradeable.sol#79-81) is not in mixedCase
- Variable ERC20PermitUpgradeable.`__gap` (contracts/draft-ERC20PermitUpgradeable.sol#99) is not in mixedCase
- Variable ERC20PermitUpgradeable.`_PERMIT_TYPEHASH` (contracts/draft-ERC20PermitUpgradeable.sol#29) is not in mixedCase
- Function IERC20PermitUpgradeable.`DOMAIN_SEPARATOR()` (contracts/draft-IERC20PermitUpgradeable.sol#59) is not in mixedCase

MulticallUpgradeable.sol

Severity: Low Risk

Status: Acknowledged

- Function MulticallUpgradeable.`__Multicall_init()` (contracts/MulticallUpgradeable.sol#15-16) is not in mixedCase
- Function MulticallUpgradeable.`__Multicall_init_unchained()` (contracts/MulticallUpgradeable.sol#18-19) is not in mixedCase
- Variable MulticallUpgradeable.`__gap` (contracts/MulticallUpgradeable.sol#50) is not in mixedCase

AccessControlUpgradeable.sol

Severity: Low Risk
Status: Acknowledged

- AccessControlUpgradeable.__gap (contracts/AccessControlUpgradeable.sol#235) is never used in AccessControlUpgradeable (contracts/AccessControlUpgradeable.sol#50-237)
- AccessControlUpgradeable.__AccessControl_init() (contracts/AccessControlUpgradeable.sol#51-52) is never used and should be removed
- AccessControlUpgradeable.__AccessControl_init_unchained() (contracts/AccessControlUpgradeable.sol#54-55) is never used and should be removed
- AccessControlUpgradeable._setRoleAdmin(bytes32,bytes32) (contracts/AccessControlUpgradeable.sol#200-204) is never used and should be removed
- AccessControlUpgradeable._setupRole(bytes32,address) (contracts/AccessControlUpgradeable.sol#191-193) is never used and should be removed
- AddressUpgradeable.functionCall(address,bytes) (contracts/AddressUpgradeable.sol#85-87) is never used and should be removed
- AddressUpgradeable.functionCall(address,bytes,string) (contracts/AddressUpgradeable.sol#95-101) is never used and should be removed
- AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (contracts/AddressUpgradeable.sol#114-120) is never used and should be removed
- AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (contracts/AddressUpgradeable.sol#128-139) is never used and should be removed
- AddressUpgradeable.functionStaticCall(address,bytes) (contracts/AddressUpgradeable.sol#147-149) is never used and should be removed
- AddressUpgradeable.functionStaticCall(address,bytes,string) (contracts/AddressUpgradeable.sol#157-166) is never used and should be removed
- AddressUpgradeable.isContract(address) (contracts/AddressUpgradeable.sol#36-42) is never used and should be removed

draft-EIP712Upgradeable.sol

Severity: Low Risk
Status: Acknowledged

- Function EIP712Upgradeable.`__EIP712_init(string,string)` (contracts/draft-EIP712Upgradeable.sol#48-50) is not in mixedCase
- Function EIP712Upgradeable.`__EIP712_init_unchained(string,string)` (contracts/draft-EIP712Upgradeable.sol#52-57) is not in mixedCase
- Function EIP712Upgradeable.`_EIP712NameHash()` (contracts/draft-EIP712Upgradeable.sol#99-101) is not in mixedCase
- Function EIP712Upgradeable.`_EIP712VersionHash()` (contracts/draft-EIP712Upgradeable.sol#109-111) is not in mixedCase
- Variable EIP712Upgradeable.`_HASHED_NAME` (contracts/draft-EIP712Upgradeable.sol#30) is not in mixedCase
- Variable EIP712Upgradeable.`_HASHED_VERSION` (contracts/draft-EIP712Upgradeable.sol#31) is not in mixedCase

ERC165Upgradeable.sol

Severity: Low Risk
Status: Acknowledged

- Function ERC165Upgradeable.`__ERC165_init()` (contracts/ERC165Upgradeable.sol#24-25) is not in mixedCase
- Function ERC165Upgradeable.`__ERC165_init_unchained()` (contracts/ERC165Upgradeable.sol#27-28) is not in mixedCase
- Variable ERC165Upgradeable.`__gap` (contracts/ERC165Upgradeable.sol#41) is not in mixedCase

AddressUpgradeable.sol

Severity: Low Risk
Status: Acknowledged

- AddressUpgradeable.`functionCall(address,bytes)` (contracts/AddressUpgradeable.sol#85-87) is never used and should be removed
- AddressUpgradeable.`functionCall(address,bytes,string)` (contracts/AddressUpgradeable.sol#95-101) is never used and should be removed
- AddressUpgradeable.`functionCallWithValue(address,bytes,uint256)` (contracts/AddressUpgradeable.sol#114-120) is never used and should be removed
- AddressUpgradeable.`functionCallWithValue(address,bytes,uint256,string)` (contracts/AddressUpgradeable.sol#128-139) is never used and should be removed
- AddressUpgradeable.`functionStaticCall(address,bytes)` (contracts/AddressUpgradeable.sol#147-149) is never used and should be removed
- AddressUpgradeable.`functionStaticCall(address,bytes,string)` (contracts/AddressUpgradeable.sol#157-166) is never used and should be removed

ContextUpgradeable.sol

Severity: Low Risk
Status: Acknowledged

- Function ContextUpgradeable.__Context_init() (contracts/ContextUpgradeable.sol#18-19) is not in mixedCase
- Function ContextUpgradeable.__Context_init_unchained() (contracts/ContextUpgradeable.sol#21-22) is not in mixedCase
- Variable ContextUpgradeable.__gap (contracts/ContextUpgradeable.sol#36) is not in mixedCase

CountersUpgradeable.sol

Severity: Low Risk
Status: Acknowledged

- CountersUpgradeable.current(CountersUpgradeable.Counter) (contracts/CountersUpgradeable.sol#22-24) is never used and should be removed
- CountersUpgradeable.decrement(CountersUpgradeable.Counter) (contracts/CountersUpgradeable.sol#32-38) is never used and should be removed
- CountersUpgradeable.increment(CountersUpgradeable.Counter) (contracts/CountersUpgradeable.sol#26-30) is never used and should be removed

StringsUpgradeable.sol

Severity: Low Risk
Status: Acknowledged

- StringsUpgradeable.`toHexString(uint256)` (contracts/StringsUpgradeable.sol#40-51) is never used and should be removed
- StringsUpgradeable.`toHexString(uint256,uint256)` (contracts/StringsUpgradeable.sol#56-66) is never used and should be removed
- StringsUpgradeable.`toString(uint256)` (contracts/StringsUpgradeable.sol#15-35) is never used and should be removed

ECDSAUpgradeable.sol

Severity: Low Risk
Status: Acknowledged

- ECDSAUpgradeable._throwError(ECDSAUpgradeable.RecoverError) (contracts/ECDSAUpgradeable.sol#23-35) is never used and should be removed
- ECDSAUpgradeable.recover(bytes32,bytes) (contracts/ECDSAUpgradeable.sol#102-106) is never used and should be removed
- ECDSAUpgradeable.recover(bytes32,bytes32,bytes32) (contracts/ECDSAUpgradeable.sol#130-138) is never used and should be removed
- ECDSAUpgradeable.recover(bytes32,uint8,bytes32,bytes32) (contracts/ECDSAUpgradeable.sol#181-190) is never used and should be removed
- ECDSAUpgradeable.toEthSignedMessageHash(bytes) (contracts/ECDSAUpgradeable.sol#214-216) is never used and should be removed
- ECDSAUpgradeable.toEthSignedMessageHash(bytes32) (contracts/ECDSAUpgradeable.sol#200-204) is never used and should be removed
- ECDSAUpgradeable.toTypedDataHash(bytes32,bytes32) (contracts/ECDSAUpgradeable.sol#227-229) is never used and should be removed
- ECDSAUpgradeable.tryRecover(bytes32,bytes) (contracts/ECDSAUpgradeable.sol#57-86) is never used and should be removed
- ECDSAUpgradeable.tryRecover(bytes32,bytes32,bytes32) (contracts/ECDSAUpgradeable.sol#115-123) is never used and should be removed

Methodology

Automated Analyses

Slither
Slither has reported 1086 findings. These results were either related to code from dependencies, false positives or have been integrated in the findings or best practices of this report.

Manual Code Review

We prefer to work with a transparent process and make our reviews a collaborative effort. The goal of our security audits is to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Vulnerability Analysis

Our audit techniques include manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high-level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, review open issue tickets, and investigate details other than the implementation.

Documenting Results

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system to make a final decision.

Suggested Solutions

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Appendix A — Finding Statuses

Resolved	Contracts were modified to permanently resolve the finding
Mitigated	The finding was resolved by other methods such as revoking contract ownership or updating the code to minimize the effect of the finding
Acknowledged	Project team is made aware of the finding
Open	The finding was not addressed