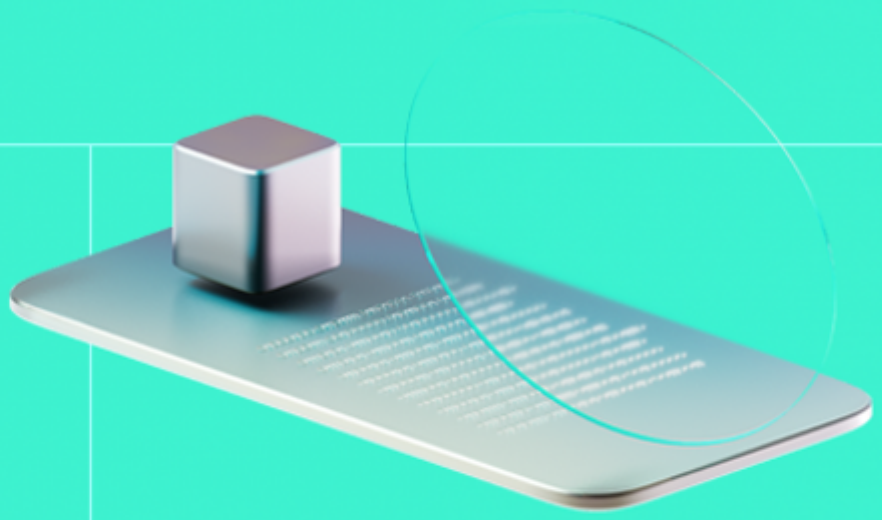
HACKEN

Smart Contract Code Review And Security Analysis Report

Customer: Vechain Foundation

Date: 21/06/2024



We express our gratitude to the Vechain Foundation team for the collaborative engagement that enabled the execution of this Smart Contract Security Assessment.

VeBetterDAO is a decentralized autonomous organization aiming at taking sustainability mainstream by unlocking the value of collective sustainable actions, through its utility token B3TR.

Platform: VeBetterDAO

Language: Solidity

Tags: DAO, ERC20, ERC721

Timeline: 13/05/2024 - 21/06/2024

Methodology: https://hackenio.cc/sc_methodology

Review Scope

Repository	https://github.com/vechain/b3tr
Commit	d2b16905e767de5ea42291393dd9b5e071d33a9c

Audit Summary

10/10	10/10	89.22%	10/10
Security score	Code quality score	Test coverage	Documentation quality score

Total 9.6/10

The system users should acknowledge all the risks summed up in the risks section of the report

5	4	1	0
Total Findings	Resolved	Accepted	Mitigated

Findings by severity

Critical	0
High	1
Medium	0
Low	3

Vulnerability	Status
F-2024-3494 - Overuse of DEFAULT_ADMIN_ROLE could lead to security risks	Accepted
F-2024-3427 - Absence of voting threshold check and missing vote registration in castVoteWithReason()	Fixed
F-2024-3444 - Missing voting period check in setCycleDuration	Fixed
F-2024-3491 - Initialization not disabled on implementation contract	Fixed
F-2024-3495 - Potential front-running in GalaxyMember token upgrade process	Fixed

This report may contain confidential information about IT systems and the intellectual property of the Customer, as well as information about potential vulnerabilities and methods of their exploitation.

The report can be disclosed publicly after prior consent by another Party. Any subsequent publication of this report shall be without mandatory consent.

Document

Name	Smart Contract Code Review and Security Analysis Report for Vechain Foundation
Audited By	Niccolò Pozzolini, Kornel Światłowski
Approved By	Przemyslaw Swiatowiec
Website	https://vechain.org/
Changelog	10/06/2024 - Preliminary Report; 18/06/2024 - Remediation Report; 21/06/2024 - Final Report

Table of Contents

System Overview

VeBetterDAO is a DAO protocol with the following contracts:

B3TR — Simple incentive ERC-20 token with 1 billion cap. Minting is possible to addresses with MINTER_ROLE.

B3TRGovernor - Main governance contract for the VeBetterDAO ecosystem. Anyone can create a proposal. In order for the proposal to become active, the community needs to deposit a certain amount of VOT3 tokens. Once a proposal succeeds, it can be queued and executed. The execution is done through the timelock contract.

Emissions- Manages the periodic distribution of B3TR tokens to XAllocation, Vote2Earn, and Treasury allocations.

GalaxyMember - Upgradeable NFT system to determine the level of user privilege. The greater the level, the greater the B3TR token rewards.

TimeLock - Performs the actions of the B3TRGovernor contract with a time delay.

Treasury - Holds all assets belonging to the DAO. Such assets include `VET`, `VTHO`, `B3TR`, `VOT3` and any other `ERC-20` or `ERC-721` tokens that have been transferred to the Treasury contract address. Tokens can only be transferred from the Treasury to another address via a governance proposal. The same goes for converting to `B3TR`/ `VOT3` .

VOT3 - Governance token of the VeBetterDAO.

VoterRewards - Handles the rewards for voters in the VeBetterDAO ecosystem. It calculates the rewards for voters based on their voting power and the level of their Galaxy Member NFT.

X2EarnApps - Handles the x-2-earn applications of the VeBetterDAO ecosystem. The contract allows the insert, management and eligibility of apps for the B3TR allocation rounds.

XAllocationPool - Receiver and distributor of weekly B3TR emissions for x2earn apps. Funds can be claimed by the X2Earn apps at the end of each allocation round

XAllocationVoting - This contract handles the voting for the most supported x2Earn applications through periodic allocation rounds. The user's voting power is calculated on his VOT3 holdings at the start of each round, using a "Quadratic Funding" formula.

X2EarnRewardsPool - This smart contract is designed to manage the distribution of rewards to users performing sustainable actions within x2Earn applications. Funds are deposited into the contract, earmarked for specific apps, and withdrawn by app administrators to reward teams.

Privileged roles

- **DEFAULT_ADMIN_ROLE**: The superuser role with the ability to grant, revoke, and renounce roles.
- **PAUSER_ROLE**: Controls the ability to pause and unpause contract functionality.
- **MINTER_ROLE**: Allows for token minting.
- **UPGRADER_ROLE**: Permits contract upgrades and modifications.
- **CONTRACTS_ADDRESS_MANAGER_ROLE**: Manages critical contract addresses.
- **GOVERNANCE_ROLE**: Involved in governance decisions, proposal management, and voting.

Executive Summary

This report presents an in-depth analysis and scoring of the customer's smart contract project. Detailed scoring criteria can be referenced in the scoring methodology.

Documentation quality

The total Documentation quality score is **10** out of **10**.

- Functional requirements are detailed.
- Technical description is detailed.

Code quality

The total Code quality score is **10** out of **10**.

Test coverage

Code coverage of the project is **89.22%** (branch coverage).

- Deployment and basic user interactions are covered with tests.
- Negative cases and some functions coverage is missed.

Security score

Upon auditing, the code was found to contain **0** critical, **1** high, **0** medium, and **3** low severity issues. Out of these, **4** issues have been addressed and resolved, leading to a security score of **10** out of **10**.

All identified issues are detailed in the “Findings” section of this report.

Summary

The comprehensive audit of the customer's smart contract yields an overall score of **9.6**. This score reflects the combined evaluation of documentation, code quality, test coverage, and security aspects of the project.

Risks

- Fixed Total Supply Post-Deployment: The token’s total supply is determined at deployment and cannot be verified beforehand, potentially limiting the project’s adaptability and economic model flexibility.
- Single Points of Failure and Control: The project is fully or partially centralized, introducing single points of failure and control. This centralization can lead to vulnerabilities in decision-making and operational processes, making the system more susceptible to targeted attacks or manipulation.
- Flexibility and Risk in Contract Upgrades: The project's contracts are upgradable, allowing the administrator to update the contract logic at any time. While this provides flexibility in addressing issues and evolving the project, it also introduces risks if upgrade processes are not properly managed or secured, potentially allowing for unauthorized changes that could compromise the project's integrity and security.

Findings

Vulnerability Details

F-2024-3427 - Absence of voting threshold check and missing vote registration in castVoteWithReason() - High

Description:

The `castVoteWithReason()` lacks two important parts: check against voting threshold and registration of user vote.

To determine voting power and distribute rewards to users, a quadratic formula is used. This formula aims to protect the interests of smaller groups of voters by giving them slightly greater impact. To prevent exploitation of this formula with small amounts of tokens, a voting threshold check has been implemented in the `castVote()` function of the `GovernorUpgradable` contract. However, the `castVoteWithReason()` function lacks this check, allowing an attacker to spread votes among multiple accounts and inflate voting power due to the quadratic factor.

According to documentation, users participating in voting should receive rewards in tokens. Each vote should be registered with the `registerVote()` function from the `VoterRewards` contract. The `GovernorUpgradable` contract contains two functions that allow users to vote: `castVote()` and `castVoteWithReason()`. The `castVoteWithReason()` function lacks an invocation of the `registerVote()` function. This omission results in users voting with this function not receiving the promised rewards for participation in voting.

```
function castVote(uint256 proposalId, uint8 support) public virtual returns (uint256) {
  address voter = _msgSender();
  uint256 weight = _castVote(proposalId, voter, support, "");

  if (weight < votingThreshold()) {
    revert GovernorVotingThresholdNotMet(weight, votingThreshold());
  }

  voterRewards().registerVote(proposalSnapshot(proposalId), msg.sender, weight, Math.sqr
  return weight;
}


function castVoteWithReason(
  uint256 proposalId,
  uint8 support,
  string calldata reason
) public virtual returns (uint256) {
  address voter = _msgSender();
  return _castVote(proposalId, voter, support, reason);
}
```

Assets:

- governance/GovernorUpgradeable.sol [https://github.com/vechain/b3tr]

Status: Fixed

Classification

Impact: 4/5

Likelihood: 4/5

Exploitability: Independent

Complexity: Simple

Severity: High

Recommendations

Remediation: It is recommended to incorporate the voting threshold check and include a call to the `registerVote()` function within the `castVoteWithReason()` function or delete the `castVoteWithReason()` function.

Resolution: The Finding was fixed in commit **53b6993**. The `castVote()` function has been introduced inside the `GovernorVotesLogic` library. This function includes a voting threshold check and externally registers a vote. The newly added function is used in both `castVote()` and `castVoteWithReason()` inside the `B3TRGovernor` contract.

F-2024-3444 - Missing voting period check in setCycleDuration - Low

Description: In the `Emissions.sol` contract, the `setCycleDuration` function is currently not checking that the new cycle duration is greater than the current voting period. This could potentially break the contracts and lead to unexpected consequences if a cycle duration lower than the voting period is set.

Assets:

- Emissions.sol [https://github.com/vechain/b3tr]

Status: Fixed

Classification

Impact: 3/5

Likelihood: 2/5

Exploitability: Dependent

Complexity: Simple

Severity: Low

Recommendations

Remediation: The function should include a check to ensure that the new cycle duration is greater than the current voting period. This can be done using the `require` sttement, as follows:

```
require(_cycleDuration > IXAllocationVotingGovernor(_xAllocationsGovernor).votingPeriod(),
```

Adding this check will prevent the cycle duration from being set to a value that could disrupt the functioning of the contracts.

Resolution: The Finding was fixed in commit **387e2cf**. The suggested check has been added to the `setCycleDuration()` function, ensuring that the new cycle duration is greater than the current voting period.

F-2024-3494 - Overuse of DEFAULT_ADMIN_ROLE could lead to security risks - Low

Description:

The contracts in scope implement proper sets of roles. However, in many other contracts throughout the codebase, the `DEFAULT_ADMIN_ROLE` is used to perform administrative operations such as setters.

Due to its wide permissions, the `DEFAULT_ADMIN_ROLE` should not be used for common operations. A compromise of its private key could lead to significant damage to the protocol and its users. For example, in many of the affected contracts, an attacker gaining access to a `DEFAULT_ADMIN_ROLE` account would be able to obtain the `UPGRADER_ROLE`, opening to more severe consequences.

Assets:

- B3TRGovernor.sol [https://github.com/vechain/b3tr]
- Emissions.sol [https://github.com/vechain/b3tr]
- GalaxyMember.sol [https://github.com/vechain/b3tr]
- Treasury.sol [https://github.com/vechain/b3tr]
- VoterRewards.sol [https://github.com/vechain/b3tr]
- X2EarnApps.sol [https://github.com/vechain/b3tr]

Status: Accepted

Classification

Impact: 4/5

Likelihood: 1/5

Exploitability: Independent

Complexity: Simple

Severity: Low

Recommendations

Remediation:

It is recommended to define and use a dedicated administrator role for these operations, while the `DEFAULT_ADMIN_ROLE` should only be used to assign roles. This would limit the potential damage in case of a security breach and improve the overall security of the contracts.

Resolution:

The Finding was accepted. The client responded: "Additional roles were added to the `Emissions` contract. After team discussions, it was decided that adding more roles to other contracts would be excessive for the project."

However, in commit **8fc5f31**. the additional roles have been added to the `Emission` contract.

F-2024-3495 - Potential front-running in GalaxyMember token upgrade process - Low

Description:

The `GalaxyMember` ERC721 tokens can be upgraded by depositing a defined amount of ERC20 `B3RT` tokens. These amounts are stored in the `_b3trToUpgradeToLevel` mapping. During the `GalaxyMember` contract initialization, the maximum level is saved to the `MAX_LEVEL` variable, and `_b3trToUpgradeToLevel` is updated with the specified amounts. An address with the `DEFAULT_ADMIN_ROLE` can add new levels (`setMaxLevel()`) and set the amount of `B3RT` required to upgrade to each level (`setB3TRtoUpgradeToLevel()`).

However, if the update actions are executed in the wrong order by the `DEFAULT_ADMIN_ROLE`, it can be front-run, allowing NFT owners to upgrade their tokens for free. This front-run can occur if the `setMaxLevel()` function is executed before the `setB3TRtoUpgradeToLevel()` function. This order of execution leaves a window for the user making it possible to upgrade to a new level with the required amount of `B3RT` tokens set to the default value of the `uint256` type, which is 0.

Assets:

- GalaxyMember.sol [https://github.com/vechain/b3tr]

Status: Fixed

Classification

Impact: 3/5

Likelihood: 2/5

Exploitability: Independent

Complexity: Medium

Severity: Low

Recommendations

Remediation:

It is recommended to implement a mechanism that ensures the required `B3RT` amount for the upgrade is set before increasing the maximum level. A second approach could be to combine the actions of setting the maximum level and updating the required `B3RT` amount into a single function to prevent the possibility of front-running.

Resolution: The Finding was fixed in commit **dabe4fb**. The `require` check has been added to the `setMaxLevel()` function, disallowing the addition of a new level if the corresponding B3RT amount needed to upgrade is not set. Additionally, checks have been added to `initialize()` and `setB3TRtoUpgradeToLevel()` functions to prevent the amount needed to upgrade from being set to 0.

F-2024-3491 - Initialization not disabled on implementation contract - Info

Description:

In the `X2EarnApps.sol` file, the `X2EarnApps` contract is not properly initialized. The initialization has not been disabled on the implementation contract, which could lead to a dangerous state.

An attacker could potentially initialize the implementation contract, appointing themselves upgrade privileges. This could allow them to perform a DelegateCall to a contract implementing the SELFDISTRUCT operation, effectively bricking the contract.

Assets:

- X2EarnApps.sol [https://github.com/vechain/b3tr]

Status: Fixed

Classification

Impact: 3/5

Likelihood: 1/5

Exploitability: Independent

Complexity: Medium

Severity: Info

Recommendations

Remediation:

Although this issue has been addressed in an upgrade to the OpenZeppelin contract library, which now checks that `UUPSUpgradeable.upgradeToAndCall()` can only be called by a proxy, it is still advised to follow OpenZeppelin's guidelines and disable the initialization on the implementation contract. This would further secure the contract and prevent potential vulnerabilities.

Resolution:

The Finding was fixed in commit **55501a0**. The constructor with the `_disableInitializers()` function has been added to the `X2EarnApps` contract.

Observation Details

F-2024-3428 - Floating pragma - Info

Description: The project uses floating pragmas `^0.8.18`

This may result in the contracts being deployed using the wrong pragma version, which is different from the one they were tested with. For example, they might be deployed using an outdated pragma version which may include bugs that affect the system negatively

Assets:

- B3TR.sol [https://github.com/vechain/b3tr]
- B3TRGovernor.sol [https://github.com/vechain/b3tr]
- B3TRProxy.sol [https://github.com/vechain/b3tr]
- Emissions.sol [https://github.com/vechain/b3tr]
- GalaxyMember.sol [https://github.com/vechain/b3tr]
- TimeLock.sol [https://github.com/vechain/b3tr]
- Treasury.sol [https://github.com/vechain/b3tr]
- VOT3.sol [https://github.com/vechain/b3tr]
- VoterRewards.sol [https://github.com/vechain/b3tr]
- X2EarnApps.sol [https://github.com/vechain/b3tr]
- XAllocationPool.sol [https://github.com/vechain/b3tr]
- XAllocationVoting.sol [https://github.com/vechain/b3tr]
- governance/GovernorUpgradeable.sol [https://github.com/vechain/b3tr]
- governance/libraries/GovernorDescriptionValidator.sol [https://github.com/vechain/b3tr]
- governance/libraries/GovernorQuorumFraction.sol [https://github.com/vechain/b3tr]
- governance/modules/GovernorCountingSimpleUpgradeable.sol [https://github.com/vechain/b3tr]
- governance/modules/GovernorDepositUpgradeable.sol [https://github.com/vechain/b3tr]
- governance/modules/GovernorExternalContractsUpgradeable.sol [https://github.com/vechain/b3tr]
- governance/modules/GovernorFunctionsSettingsUpgradeable.sol [https://github.com/vechain/b3tr]
- governance/modules/GovernorSettingsUpgradeable.sol [https://github.com/vechain/b3tr]
- 31 more asset(s) affected

Status: Fixed

Recommendations

Remediation: Consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment. Consider known bugs for the compiler version that is chosen.

Resolution: The Finding was fixed in commit **b313672**. The pragma was locked to `0.8.20`.

The pragma in `X2EarnRewardsPool` contract was locked in commit **4b429a0.**

F-2024-3429 - Some public functions should be declared as external - Info

Description:

Functions that are meant to be exclusively invoked from external sources should be designated as `external` rather than `public`. This is essential to enhance both the gas efficiency and the overall security of the contract.

external visibility can be added:

- B3TR: pause(), unpause()
- B3TRGovernor: initialize(), pause(), unpause()
- Emissions: initialize(), bootstrap(), start(), distribute(), setXallocationsAddress(), setVote2EarnAddress(), setTreasuryAddress(), setCycleDuration(), setXAllocationsDecay(), setVote2EarnDecay(), setXAllocationsDecayPeriod(), setVote2EarnDecayPeriod(), setTreasuryPercentage(), setScalingFactor(), setMaxVote2EarnDecay(), setXAllocationsGovernorAddress()
- GalaxyMember: initialize(), pause(), unpause(), freeMint(), mint(), upgrade(), selectHighestLevel(), setMaxLevel(), setXAllocationsGovernorAddress(), setB3trGovernorAddress(), setBaseURI(), setB3TRtoUpgradeToLevel(), setIsPublicMintingPaused(), getPastHighestLevel(), numCheckpoints(), getB3TRtoUpgradeToLevel(), getNextLevel(), getB3TRtoUpgrade(), CLOCK_MODE(), xAllocationsGovernor(), b3trGovernor(), b3tr(), treasury(), MAX_LEVEL(), levelOf(),
- TimeLock: initialize()
- Treasury: initialize(), pause(), unpause(), transferVTHO(), transferB3TR(), transferVOT3(), transferVET(), transferTokens(), transferNFT(), convertB3TR(), convertVOT3(), setTransferLimitVET(), setTransferLimitToken(), getVTHOBalance(), getB3TRBalance(), getVOT3Balance(), getVETBalance(), getCollectionNFTBalance(), getTransferLimitVET(), getTransferLimitToken()
- VOT3: initialize(), pause(), unpause(), convertedB3trOf()
- VoterRewards: initialize(), registerVote(), claimReward(), cycleToVoterToTotal(), cycleToTotal(), levelToMultiplier(), galaxyMember(), emissions(), scalingFactor(), b3tr(), setGalaxyMember(), setLevelToMultiplier(), setEmissions(), setScalingFactor()
- X2EarnApps: initialize(), setXAllocationVotingAddress(), setEmissionsAddress(), setTreasuryAddress(), setX2EarnAppsAddress(), claim(), claimed(), getMaxAppAllocation(), treasury(), b3tr(), x2EarnApps(),
- GovernorDepositUpgradeable: withdraw(), getProposalDeposits(), proposalDepositThreshold(), getUserDeposit()
- GovernorQuorumFraction: quorumNumerator()
- GovernorFunctionsSettingsUpgradeable: isFunctionWhitelisted()
- RoundsStorageUpgradeable: getRound(), getAppsOfRound()
- X2EarnRewardsPool : initialize(), distributeReward(), setX2EarnApps(), availableFunds(), version(), b3tr(), x2EarnApps()

Assets:

- B3TR.sol [https://github.com/vechain/b3tr]
- B3TRGovernor.sol [https://github.com/vechain/b3tr]
- Emissions.sol [https://github.com/vechain/b3tr]
- GalaxyMember.sol [https://github.com/vechain/b3tr]
- TimeLock.sol [https://github.com/vechain/b3tr]
- Treasury.sol [https://github.com/vechain/b3tr]
- VOT3.sol [https://github.com/vechain/b3tr]
- VoterRewards.sol [https://github.com/vechain/b3tr]

- X2EarnApps.sol [https://github.com/vechain/b3tr]
- governance/libraries/GovernorQuorumFraction.sol [https://github.com/vechain/b3tr]
- governance/modules/GovernorDepositUpgradeable.sol [https://github.com/vechain/b3tr]
- governance/modules/GovernorFunctionsSettingsUpgradeable.sol [https://github.com/vechain/b3tr]
- x-allocation-voting-governance/modules/RoundsStorageUpgradeable.sol [https://github.com/vechain/b3tr]
- X2EarnRewardsPool.sol [https://github.com/vechain/b3tr/commit/12a4fbb6dbc887fcfdec2091ea0cd02d6ac2a543]

Status: Fixed

Recommendations

Remediation: To optimize gas usage and improve code clarity, declare functions that are not called internally within the contract and are intended for external access as `external` rather than `public`. This ensures that these functions are only callable externally, reducing unnecessary gas consumption.

Resolution: The Finding was fixed in commit **b3776b4**. The visibility of all of the mentioned functions has been changed from `public` to `external`.

F-2024-3439 - Assignment of default value to variables increases gas consumption - Info

Description:

The contract's variables, upon deployment or declaration in functions, are automatically assigned default values based on their types. This redundancy results in increased gas consumption during contract deployment.

Variables that are redundantly reassigned to its default value:

- GalaxyMember: initialize(isPublicMintingPaused, i)
- VoterRewards: initialize(i)
- X2EarnApps: initialize(i)
- XAllocationPool: roundEarnings(unallocatedAmount), getAppShares(unallocatedShare)
- XAllocationVoting: initialize(i)
- GovernorDescriptionValidator: isValidDescriptionForProposer(recovered)
- GovernorFunctionsSettingsUpgradeable: setWhitelistFunctions(i), _checkFunctionsRestriction(i)
- RoundsStorageUpgradeable: getAppsOfRound(i)
- RoundVotesCountingUpgradeable: _countVote(totalWeight, totalQFVotesAdjustment, i)
- GovernorUpgradeable: execute(i), _executeOperations(i)
- AdministrationUpgradeable: _removeAppModerator(i), isAppModerator(i)
- AppsStorageUpgradeable: apps(i)

Assets:

- GalaxyMember.sol [https://github.com/vechain/b3tr]
- VoterRewards.sol [https://github.com/vechain/b3tr]
- X2EarnApps.sol [https://github.com/vechain/b3tr]
- XAllocationPool.sol [https://github.com/vechain/b3tr]
- XAllocationVoting.sol [https://github.com/vechain/b3tr]
- governance/GovernorUpgradeable.sol [https://github.com/vechain/b3tr]
- governance/libraries/GovernorDescriptionValidator.sol [https://github.com/vechain/b3tr]
- governance/modules/GovernorFunctionsSettingsUpgradeable.sol [https://github.com/vechain/b3tr]
- x-2-earn-apps/modules/AdministrationUpgradeable.sol [https://github.com/vechain/b3tr]
- x-2-earn-apps/modules/AppsStorageUpgradeable.sol [https://github.com/vechain/b3tr]
- x-allocation-voting-governance/modules/RoundsStorageUpgradeable.sol [https://github.com/vechain/b3tr]
- x-allocation-voting-governance/modules/RoundVotesCountingUpgradeable.sol [https://github.com/vechain/b3tr]

Status: Fixed

Recommendations

Remediation:

It is suggested to remove the assignments to the default values in order to reduce the deployment gas cost and improve the code quality.

Resolution: The Finding was fixed in commit **3e69762**. The assignments of default values have been removed.

F-2024-3440 - Parent initializer call inside __GovernorDeposit_init_unchained - Info

Description:

In the `GovernorDepositUpgradeable.sol` contract, the `__ReentrancyGuard_init()` function is currently being called in the `__GovernorDeposit_init_unchained()` function.

Initializer functions are not linearized by the compiler like constructors; each `__{ContractName}_init` function embeds the linearized calls to all parent initializers. Therefore, calling two of these init functions can potentially initialize the same contract twice.

To avoid potential double initialization, the parent initializer functions should be moved to the `__GovernorDeposit_init()` function. The `__{ContractName}_init_unchained` function is the initializer function minus the calls to parent initializers and should be used to avoid the double initialization problem.

The impact here is negligible, since `ReentrancyGuard` gets properly initialized, and a double initialization of ReentrancyGuard would have no consequences besides gas overhead.

Assets:

- governance/modules/GovernorDepositUpgradeable.sol [https://github.com/vechain/b3tr]

Status: Fixed

Recommendations

Remediation:

It is suggested to move `__ReentrancyGuard_init` invocation to `__GovernorDeposit_init` to follow established best practices and improve the contract's maintainability.

Resolution:

The Finding was fixed in commit **53b6993**. The `GovernorDepositUpgradeable` module contract has been transformed into the library `GovernorDepositLogic`. Therefore, the initialization of `ReentrancyGuard` is not necessary and has been removed.

F-2024-3441 - The apps getter is missing a pagination feature - Info

Description: In the `AppsStorageUpgradeable.sol` contract, the `apps()` function currently returns all apps in a single call. This approach could potentially exceed the block gas limit if the number of apps grows significantly.

Assets:

- x-2-earn-apps/modules/AppsStorageUpgradeable.sol [https://github.com/vechain/b3tr]

Status: Fixed

Recommendations

Remediation: To improve the user experience and avoid potential block gas limit issues, it is recommended to implement a pagination feature in the `apps()` function. This would allow the function to return a subset of apps at a time, reducing the amount of data processed in a single call and making the function more scalable.

Implementing pagination typically involves adding parameters to the function to specify the index of the first app and the number of apps to return. The function would then only loop over the specified subset of apps, reducing the amount of computation and storage required. Function overloading can be used to keep the current integrations unchanged.

Resolution: The Finding was fixed in commit **18b42b8**. The `getPaginatedApps()` function has been added, allowing for retrieval of a selected range of apps instead of all.

F-2024-3443 - Redundant imports - Info

Description:

Several redundant imports were identified:

- The contract `Initializable` is imported in `B3TRGovernor` contract, but `Initializable` is already part of `UUPSUpgradeable`.
- The contract `Initializable` is imported in `Emissions` contract, but `Initializable` is already part of `UUPSUpgradeable`.
- The contract `Initializable` is imported in `GalaxyMember` contract, but `Initializable` is already part of `UUPSUpgradeable`.
- The contract `Initializable` is imported in `TimeLock` contract, but `Initializable` is already part of `UUPSUpgradeable`.
- The contract `Initializable` is imported in `Treasury` contract, but `Initializable` is already part of `UUPSUpgradeable`.
- The contract `Initializable` is imported in `VOT3` contract, but `Initializable` is already part of `UUPSUpgradeable`.
- The contract `Initializable` is imported in `VoterRewards` contract, but `Initializable` is already part of `UUPSUpgradeable`.
- The contract `Initializable` is imported in `XAllocationVoting` contract, but `Initializable` is already part of `UUPSUpgradeable`.
- The contract `Initializable` is imported in `GovernorDepositUpgradeable` contract, but `Initializable` is already part of `UUPSUpgradeable`.
- The contract `Initializable` is imported in `X2EarnRewardsPool` contract, but `Initializable` is already part of `UUPSUpgradeable`.

This redundancy in import operations has the potential to result in unnecessary gas consumption during deployment and could potentially impact the overall code quality.

Assets:

- B3TRGovernor.sol [https://github.com/vechain/b3tr]
- Emissions.sol [https://github.com/vechain/b3tr]
- GalaxyMember.sol [https://github.com/vechain/b3tr]
- TimeLock.sol [https://github.com/vechain/b3tr]
- Treasury.sol [https://github.com/vechain/b3tr]
- VOT3.sol [https://github.com/vechain/b3tr]
- VoterRewards.sol [https://github.com/vechain/b3tr]
- XAllocationVoting.sol [https://github.com/vechain/b3tr]
- governance/modules/GovernorDepositUpgradeable.sol [https://github.com/vechain/b3tr]
- X2EarnRewardsPool.sol [https://github.com/vechain/b3tr/commit/12a4fbb6dbc887fcfdec2091ea0cd02d6ac2a543]

Status: Fixed

Recommendations

Remediation:

Remove redundant imports, and ensure that the contract is imported only in the required locations, avoiding unnecessary duplications.

Resolution:

The Finding was fixed in commit **93ce885**. The redundant import of the `Initializable` library has been removed.

The redundant import of the `Initializable` library in `X2EarnRewardsPool` contract has been removed in commit **6ac5dc5.**

F-2024-3445 - Scaling factor should be declared as constant - Info

Description: In the `Emissions.sol` and `VoterRewards.sol` contracts, a scaling factor is used to enhance precision during mathematical operations. Typically, scaling factors are set as constants, as the required precision is not expected to change. However, in this contract, the `scalingFactor` is stored, resulting in an overhead of storage reads. This leads to an increased gas cost for operations.

Assets:

- Emissions.sol [https://github.com/vechain/b3tr]
- VoterRewards.sol [https://github.com/vechain/b3tr]

Status: Fixed

Recommendations

Remediation: It is recommended to use a constant scaling factor. The `scalingFactor` should be defined as a constant at the top of the contract, and all references to the `scalingFactor` storage variable should be replaced with this constant.

Resolution: The Finding was fixed in commit **0631c7f**. The scaling factor has been declared constant.

F-2024-3446 - Missing events emitting for critical functions - Info

Description:

Events allow capturing the changed parameters so that off-chain tools/interfaces can register such changes with timelocks that allow users to evaluate them and consider if they would like to engage/exit based on how they perceive the changes as affecting the trustworthiness of the protocol or profitability of the implemented financial services. The alternative of directly querying the on-chain contract state for such changes is not considered practical for most users/usages.

Missing events do not promote transparency and if such changes immediately affect users' perception of fairness or trustworthiness, they could exit the protocol causing a reduction in liquidity which could negatively impact protocol TVL and reputation.

The following functions do not emit any events:

- Emissions: setXallocationsAddress(), setVote2EarnAddress(), setTreasuryAddress(), setCycleDuration(), setXAllocationsDecay(), setVote2EarnDecay(), setXAllocationsDecayPeriod(), setVote2EarnDecayPeriod(), setTreasuryPercentage(), setScalingFactor(), setMaxVote2EarnDecay(), setXAllocationsGovernorAddress()
- GalaxyMember: setMaxLevel(), setXAllocationsGovernorAddress(), setB3trGovernorAddress(), setBaseURI(), setB3TRtoUpgradeToLevel(), setIsPublicMintingPaused()
- Treasury: setTransferLimitVET(), setTransferLimitToken()
- VoterRewards: setGalaxyMember(), setLevelToMultiplier(), setEmissions(), setScalingFactor()
- GovernorFunctionsSettingsUpgradeable: setWhitelistFunction(), setIsFunctionRestrictionEnabled()

Assets:

- Emissions.sol [https://github.com/vechain/b3tr]
- GalaxyMember.sol [https://github.com/vechain/b3tr]
- Treasury.sol [https://github.com/vechain/b3tr]
- VoterRewards.sol [https://github.com/vechain/b3tr]
- governance/modules/GovernorFunctionsSettingsUpgradeable.sol [https://github.com/vechain/b3tr]

Status: Fixed

Recommendations

Remediation: Consider emitting the corresponding events in the reported functions.

Resolution: The Finding was fixed in commit **882cf9f**. The event emissions have been added to the important setter functions.

F-2024-3447 - Lack of usage of named parameters for cycleToVoterToTotal mapping - Info

Description:

In the `VoterRewards.sol` contract, the `cycleToVoterToTotal` mapping is currently declared without named parameters:

```
mapping(uint256 => mapping(address => uint256)) cycleToVoterToTotal;
```

Assets:

- VoterRewards.sol [https://github.com/vechain/b3tr]

Status: Fixed

Recommendations

Remediation:

While the purpose of the mapping is explained through a clear name and a comment, the code could be made more readable by using named parameters for mappings, a feature introduced in Solidity 0.8.18.

The updated declaration would look like this:

```
mapping(uint256 cycle => mapping(address voter => uint256 total)) cycleToVoterToTotal;
```

This change would make the code more self-explanatory and easier to understand, reducing the need for comments and improving maintainability.

Resolution:

The Finding was fixed in commit **a875f2e**. The parameter names inside `cycleToVoterToTotal` have been added.

F-2024-3481 - Missing validation of cap input parameters in the constructor() of the B3TR contract - Info

Description: According to documentation:

```
The total supply of B3TR is capped at 1,000,000,000 tokens, with a weekly issuance schedul
carried out over 12 years.
```

However, the constructor of the B3TR contract lacks a check against this value. This omission allows for the deployment of the B3TR contract with a different cap, potentially leading to a violation of the requirements defined in the documentation and disrupting the planned emission schedule.

```
constructor(
  address _admin,
  address _defaultMinter,
  address _pauser,
  uint256 _cap
) ERC20("B3TR", "B3TR") ERC20Capped(_cap * 1e18) {
  _grantRole(DEFAULT_ADMIN_ROLE, _admin);
  _grantRole(MINTER_ROLE, _defaultMinter);
  _grantRole(PAUSER_ROLE, _pauser);
}
```

Assets:

- B3TR.sol [https://github.com/vechain/b3tr]

Status: Fixed

Recommendations

Remediation: It is recommended to implement a constant variable storing the desired cap amount (1,000,000,000) and use it for the `ERC20Capped constructor()`.

Resolution: The Finding was fixed in commit **b875b05**. The constant variable named `B3TR_CAP` has been added to the B3RT contract and used inside the `ERC20Capped constructor()`.

F-2024-3496 - Missing parameters validation in initializations and setters - Info

Description:

Multiple functions throughout the codebase lack the validation of their input parameters:

- In the `B3TRGovernor` contract, the `initialize` function is not performing zero address checks against: `data.vot3Token`, `data.timelock`, `data.voterRewards`, `data.xAllocationVoting`, `data.b3tr`. In the same contract, the setters `setVoterRewards`, `setXAllocationVoting`, `setWhitelistFunction`, `setWhitelistFunctions` are lacking the zero address check on their parameters.
- In the `Emissions` contract, the `initialize` function is not performing zero address checks against: `data.b3trAddress`, `data.admin`. In the same contract, the setters `setVoterRewards`, `setXAllocationVoting`, `setWhitelistFunction`, `setWhitelistFunctions` are lacking the zero address check on their parameters.
- In the `GalaxyMember` contract, the `initialize` function is not performing zero address checks against `data.admin`. In the `initialize` function, `data.maxLevel` should also be checked to be equal to `data.b3trToUpgradeToLevel.length + 2`, to avoid free upgrades due to misconfiguration.
- In the `Treasury` contract, the `initialize` function is not performing zero address checks against `_b3tr` and `_vot3`.
- In the `VOT3` contract, the `initialize` function is not performing zero address checks against `_admin`.
- In the `VoterRewards` contract, the `initialize` function is not performing zero address checks against `admin`,
- In the `X2EarnApps` contract, the `initialize` function is not performing zero address checks against `_admins`.
- In the `XAllocationPool` contract, the `initialize` function is not performing zero address checks against `_admin`.
- In the `XAllocationVoting` contract, the `initialize` function is not performing zero address checks against: `data.x2EarnAppsAddress, data.emissions, data.voterReward, data.vot3Token, data.admins`.
- In the `X2EarnRewardsPool` contract, the `setX2EarnApps` function is not performing zero address checks against `_x2EarnApps`.

Assets:

- B3TRGovernor.sol [https://github.com/vechain/b3tr]
- Emissions.sol [https://github.com/vechain/b3tr]
- GalaxyMember.sol [https://github.com/vechain/b3tr]
- Treasury.sol [https://github.com/vechain/b3tr]
- VOT3.sol [https://github.com/vechain/b3tr]
- VoterRewards.sol [https://github.com/vechain/b3tr]
- X2EarnApps.sol [https://github.com/vechain/b3tr]
- XAllocationPool.sol [https://github.com/vechain/b3tr]
- XAllocationVoting.sol [https://github.com/vechain/b3tr]
- X2EarnRewardsPool.sol [https://github.com/vechain/b3tr/commit/12a4fbb6dbc887fcfdec2091ea0cd02d6ac2a543]

Status: Fixed

Recommendations

Remediation: It is suggested to implement the aforementioned parameter validations.

Resolution: The Finding was fixed in commit **1da12d9**. Validation of the mentioned input parameters has been added.

The validation of the mentioned input parameter in `X2EarnRewardsPool` contract has been added in commit **5a7ef29.**

F-2024-3503 - View functions in GalaxyMember contract lack validation for maximum level - Info

Description:

The `GalaxyMember` ERC721 tokens can be upgraded by depositing a defined amount of ERC20 `B3RT` tokens. During the `GalaxyMember` contract initialization, the maximum level is set alongside the required amounts to upgrade ERC721 tokens. For easier access to values related to the leveling feature, the `GalaxyMember` contract contains several view functions. However, some of these view functions lack a check against the maximum level and can return misleading values beyond the defined maximum level:

- The `getB3TRtoUpgradeToLevel()` function can return the amount of B3RT tokens required for the next level, incorrectly indicating that the next level exists.
- The `getNextLevel()` function can return the next level of a given `tokenId`, incorrectly suggesting that an upgrade beyond the maximum level is possible.
- The `getB3TRtoUpgrade()` function can return the required `B3RT` amount to upgrade to the next level of a given `tokenId`, incorrectly indicating that an upgrade beyond the maximum level is possible.

Assets:

- GalaxyMember.sol [https://github.com/vechain/b3tr]

Status: Accepted

Recommendations

Remediation:

It is recommended to implement a validation check in the view functions to ensure that they do not return values beyond the defined maximum level. This can be achieved by verifying that the requested level or `tokenId` does not exceed the maximum level before returning values.

Resolution:

Client response: “This is intended behavior and it gives more flexibility to view functions".

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the Consultant cannot guarantee the explicit security of the audited smart contracts.

Appendix 1. Severity Definitions

When auditing smart contracts, Hacken is using a risk-based approach that considers **Likelihood**, **Impact**, **Exploitability** and **Complexity** metrics to evaluate findings and score severities.

Reference on how risk scoring is done is available through the repository in our Github organization:

hknio/severity-formula

Severity	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation.
High	High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation.
Medium	Medium vulnerabilities are usually limited to state manipulations and, in most cases, cannot lead to asset loss. Contradictions and requirements violations. Major deviations from best practices are also in this category.
Low	Major deviations from best practices or major Gas inefficiency. These issues will not have a significant impact on code execution, do not affect security score but can affect code quality score.

Appendix 2. Scope

The scope of the project includes the following smart contracts from the provided repository:

Scope Details

Repository	https://github.com/vechain/b3tr
Commit	d2b16905e767de5ea42291393dd9b5e071d33a9c
Whitepaper	https://vechain.org/assets/whitepaper/VeBetterDAO-whitepaper.pdf
Requirements	https://vechain.org/assets/whitepaper/VeBetterDAO-whitepaper.pdf; https://vechain-foundation-san-marino.gitbook.io/vebetter-dao/treasury
Technical Requirements	https://vechain.org/assets/whitepaper/VeBetterDAO-whitepaper.pdf; https://vechain-foundation-san-marino.gitbook.io/vebetter-dao/treasury

Contracts in Scope

B3TR.sol

B3TRGovernor.sol

B3TRProxy.sol

Emissions.sol

GalaxyMember.sol

TimeLock.sol

Treasury.sol

VOT3.sol

VoterRewards.sol

X2EarnApps.sol

XAllocationPool.sol

XAllocationVoting.sol

governance/GovernorUpgradeable.sol

governance/libraries/GovernorDescriptionValidator.sol

governance/libraries/GovernorQuorumFraction.sol

governance/modules/GovernorCountingSimpleUpgradeable.sol

governance/modules/GovernorDepositUpgradeable.sol

governance/modules/GovernorExternalContractsUpgradeable.sol

governance/modules/GovernorFunctionsSettingsUpgradeable.sol

Contracts in Scope

governance/modules/GovernorSettingsUpgradeable.sol

governance/modules/GovernorTimelockControlUpgradeable.sol

governance/modules/GovernorVotesQuorumFractionUpgradeable.sol

governance/modules/GovernorVotesUpgradeable.sol

libraries/X2EarnAppsDataTypes.sol

x-2-earn-apps/X2EarnAppsUpgradeable.sol

x-2-earn-apps/modules/AdministrationUpgradeable.sol

x-2-earn-apps/modules/AppsStorageUpgradeable.sol

x-2-earn-apps/modules/SettingsUpgradeable.sol

x-2-earn-apps/modules/VoteEligibilityUpgradeable.sol

x-allocation-voting-governance/XAllocationVotingGovernor.sol

x-allocation-voting-governance/modules/ExternalContractsUpgradeable.sol

x-allocation-voting-governance/modules/RoundEarningsSettingsUpgradeable.sol

x-allocation-voting-governance/modules/RoundFinalizationUpgradeable.sol

x-allocation-voting-governance/modules/RoundsStorageUpgradeable.sol

x-allocation-voting-governance/modules/RoundVotesCountingUpgradeable.sol

x-allocation-voting-governance/modules/VotesQuorumFractionUpgradeable.sol

x-allocation-voting-governance/modules/VotesUpgradeable.sol

x-allocation-voting-governance/modules/VotingSettingsUpgradeable.sol

interfaces/IB3TR.sol

interfaces/IB3TRGovernor.sol

interfaces/IEmissions.sol

interfaces/IERC20.sol

interfaces/IERC721.sol

interfaces/IGalaxyMember.sol

interfaces/ITokenAuction.sol

interfaces/ITreasury.sol

interfaces/IVOT3.sol

interfaces/IVoterRewards.sol

interfaces/IX2EarnApps.sol

interfaces/IXAllocationPool.sol

Contracts in Scope

interfaces/IXAllocationVotingGovernor.sol

X2EarnRewardsPool.sol (**added** to scope in 12a4fbb6dbc887fcfdec2091ea0cd02d6ac2a543)