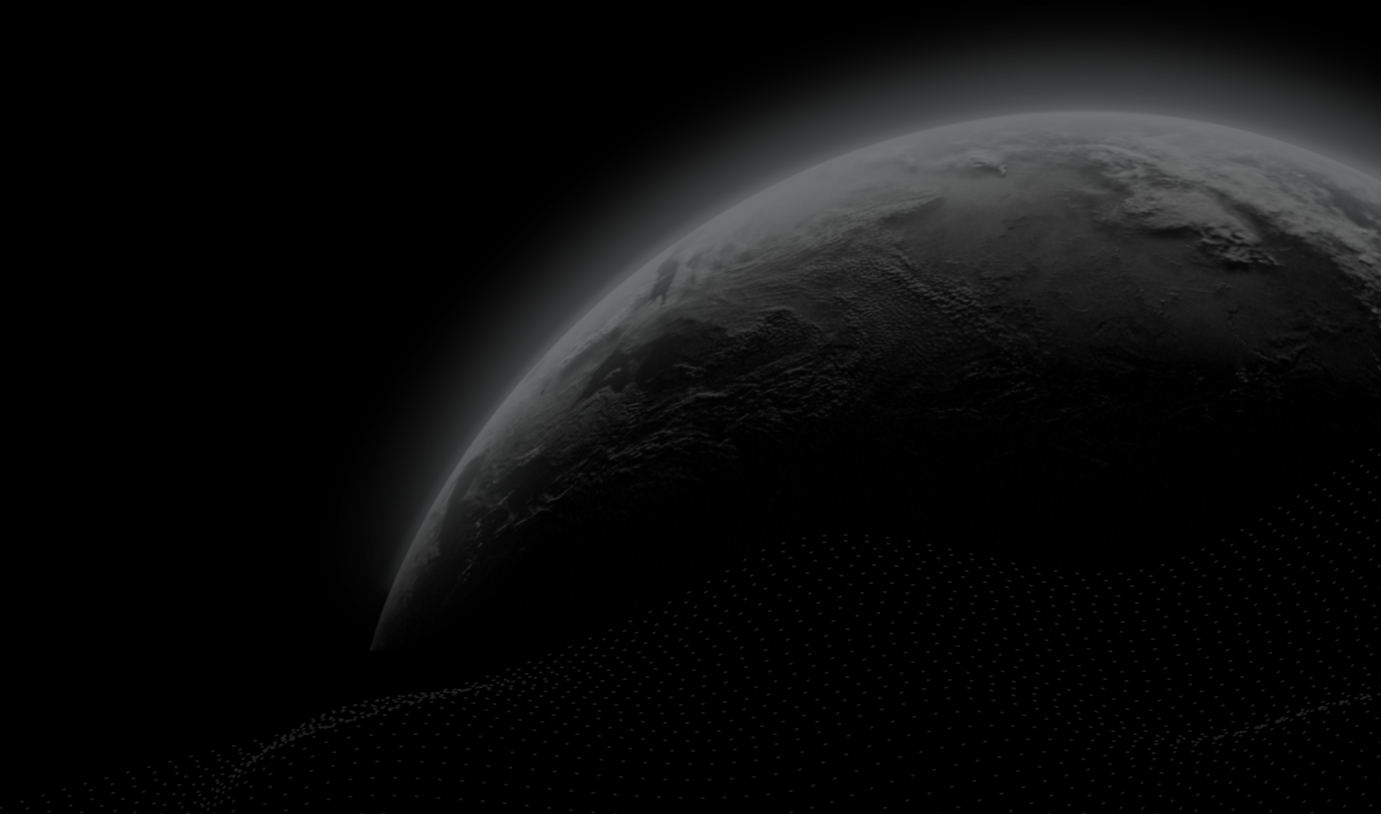
CERTIK

Security Assessment

Axes Metaverse V2(Token) - Audit

CertiK Verified on Oct 28th, 2022



CertiK Verified on Oct 28th, 2022

Axes Metaverse V2(Token) - Audit

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
Others	Binance Smart Chain (BSC)	Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 10/28/2022	N/A

CODEBASE

https://testnet.bscscan.com/address/0x0019f5355b66eb2b6c397ef258a1bb301f05bf5f

https://testnet.bscscan.com/address/0xB68c6f2758aaabD2E5154268a6

...View All

Vulnerability Summary

14	6	0	0	8	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Unresolved

🟥	0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
🟧	2	Major	2 Acknowledged	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
🟨	3	Medium	3 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
🟨	4	Minor	2 Resolved, 2 Acknowledged	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
🟦	5	Informational	1 Resolved, 4 Acknowledged	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | AXES METAVERSE V2(TOKEN) - AUDIT

CODEBASE | AXES METAVERSE V2(TOKEN) - AUDIT

Repository

https://testnet.bscscan.com/address/0x0019f5355b66eb2b6c397ef258a1bb301f05bf5f

https://testnet.bscscan.com/address/0xB68c6f2758aaabD2E5154268a6191C93FD88D5F8

https://testnet.bscscan.com/address/0x8D71062Dc809C66D688BA59C0aa7e1f126122bA2

https://testnet.bscscan.com/address/0x0fe45ffFd5510D3887a7241A3dA91cb948ff1382

https://testnet.bscscan.com/address/0xf45e1aDE23E7F4d26480bB6e025abF31de5887cf

https://testnet.bscscan.com/address/0xB83CbEcaD38a6A432B1de9B1e2DfD4703a4DCf7a

https://testnet.bscscan.com/address/0x1C5CA2AB56061A02931Dc0F11ed94A5E9fa832Aa

AUDIT SCOPE | AXES METAVERSE V2(TOKEN) - AUDIT

6 files audited ● 6 files with Acknowledged findings

ID	File	SHA256 Checksum
● AMS	AxesMetaverseShard.sol	a0681c5b7a5a6a2f32612758fb43ee4a0997ba2f0466990b28123a8d1c129952
● AUS	AxesUpgrade.sol	0d28d6b9dc82cb57d3a22228ff9ac5ce6efc34d58b8eee2ed35c66864ff09894
● ASS	AxesSummoner.sol	6190991dd9108535f41f9f5405eb0d9fcad89cf395c012bd01dbeeff36cd7335
● BAT	BattlegroundSeasonsRegistry.sol	592dcffa7c3b07461940c367b6e9938a9eea3cecb8bb8d85c6dfc7675a2ce402
● AXS	Axes721helper.sol	a4e85bd012da626c97b4c62df179c065705a2b95a5cc11f6ac3933b2404ca8a5
● IAX	IAxesHub.sol	030b35729eacc0e1be2022ba7afbd729224868075d10e8a47bc37bf79dfb8302

APPROACH & METHODS | AXES METAVERSE V2(TOKEN) - AUDIT

This report has been prepared for Axes Metaverse to discover issues and vulnerabilities in the source code of the Axes Metaverse V2(Token) - Audit project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

FINDINGS | AXES METAVERSE V2(TOKEN) - AUDIT

14	0	2	3	4	5
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Axes Metaverse V2(Token) - Audit. Through this audit, we have uncovered 14 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
AMS-01	**Initial Token Distribution**	**Centralization / Privilege**	**Major**	**Acknowledged**
BAT-01	"ClaimRewardCommission" Is Charged Twice With Different Tokens	Logical Issue	Medium	Resolved
BAT-02	Missing Zero Address Validation	Volatile Code	Minor	Resolved
IAX-01	Incorrect Recipient Address	Logical Issue	Medium	Resolved
IAX-02	Potential Reentrancy Attack	Volatile Code	Medium	Resolved
TES-01	**Centralization Related Risks**	**Centralization / Privilege**	**Major**	**Acknowledged**
TES-02	OpenZeppelin Library Code Included In Source Code	Coding Style	Minor	Acknowledged
TES-03	Third Party Dependency	Volatile Code	Minor	Acknowledged
TES-04	Locked Ether	Language Specific	Minor	Resolved
AUS-01	Potential Index Out Of Bound	Volatile Code	Informational	Acknowledged

ID	Title	Category	Severity	Status
AUS-02	Potential Request For Free	Volatile Code	Informational	● Acknowledged
AUS-03	Unchanged Upgrade Cost	Logical Issue	Informational	● Acknowledged
IAX-03	`rewardRate` And `energyRecoveryRate` Are Not Used	Logical Issue	Informational	● Acknowledged
TES-05	Hardcode Address	Logical Issue	Informational	● Resolved

AMS-01 | INITIAL TOKEN DISTRIBUTION

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	**AxesMetaverseShard.sol (1): 1607**	● **Acknowledged**

Description

All of the AMS tokens are sent to the contract deployer when deploying the contract. This could be a centralization risk as the anonymous deployer can distribute tokens without obtaining the consensus of the community. Any compromise to the deployer account that holds undistributed tokens may allow the attacker to steal and sell tokens on the market, resulting in severe damage to the project.

Recommendation

It's recommended the team be transparent regarding the initial token distribution process. The token distribution plan should be published in a public location that the community can access. The team shall make enough efforts to restrict the access of the private key. A multi-signature (⅔, ⅗) wallet can be used to prevent a single point of failure due to the private key compromise. Additionally, the team can lock up a portion of tokens, release them with a vesting schedule for long-term success, and deanonymize project teams with a third-party KYC provider to create greater accountability.

Alleviation

[Axes Team]:

All the tokens are sent to the creator on deploy event.

BAT-01 "CLAIMREWARDCOMMISSION" IS CHARGED TWICE WITH DIFFERENT TOKENS

Category	Severity	Location	Status
Logical Issue	● Medium	BattlegroundSeasonsRegistry.sol (1): 761~767	● Resolved

Description

The "claimRewardCommission" is charged twice:

1. Line 762: `require(msg.value >= claimRewardCommission, "Season: msg.value is less than claimRewardCommission");`
2. Line 764: `commissionToken.transferFrom(msg.sender, commissionRecipient, claimRewardCommission);`

Is this the intended design that the `claimRewardCommission` need to be changed with both the chain native token and the "commissionToken" token? The contract doesn't provide a way to withdraw chain native tokens; the token will be locked in the contract forever. It seems the contract should only charge the "commissionToken" as the claimRewardCommission.

Recommendation

It's recommended the team verify if the current implementation matches the intended design. If so, the contract needs to include a function for the owner to withdraw the chain native token.

Alleviation

The client revised the code and resolved this issue on address 0xF147818b2e81f55Fc18FFbC7cEbbE78E7213059C.

BAT-02 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	● Minor	BattlegroundSeasonsRegistry.sol (1): 676	● Resolved

Description

Addresses should be checked before assignment or external call to make sure they are not zero addresses.

```
676         commissionRecipient = _commissionRecipient;
```

- `_commissionRecipient` is not zero-checked before being used.

Recommendation

We advise adding a zero-check for the passed-in address value to prevent unexpected errors.

Alleviation

The client revised the code and resolved this issue on address 0x1b6287f32F0eD3EB81b7f1644a0125bB2AC727c5.

IAX-01 | INCORRECT RECIPIENT ADDRESS

Category	Severity	Location	Status
Logical Issue	● Medium	IAxesHub.sol (1): 1007	● Resolved

Description

The function `withdraw()` is used to withdraw funds to `_to` from the reward pool by the contract admin, but the recipient address is the "msg.sender" instead of the `_to` address.

Recommendation

We advise the client to check if the recipient address is correct.

Alleviation

The client revised the code and resolved this issue on address 0x9407c5F0a9643a13Dd87FE8f485856270A5F42f3.

IAX-02 | POTENTIAL REENTRANCY ATTACK

Category	Severity	Location	Status
Volatile Code	● Medium	IAxesHub.sol (1): 923	● Resolved

Description

A reentrancy attack can occur when the contract creates a function that makes an external call to another untrusted contract before resolving any effects. If the attacker can control the untrusted contract, they can make a recursive call back to the original function, repeating interactions that would have otherwise not run after the external call resolved the effects.

External call(s)

```
        recalculateReward(msg.sender);
```

- This function call executes the following external call(s).
- In `AxesStaking.recalculateReward` ,
    - `hub.register(AxesStaking:RewardUpdated,string(abi.encode(_user,usersMiningInfo[_user].rewardBalance,miningTime,rewardSum,block.timestamp)))`

```
            axes721.transferFrom(address(this), msg.sender, _tokenIds[i]);
```

State variables written after the call(s)

```
            delete stakes[_tokenIds[i]];
```

```
            totalMiningPower -= stakes[_tokenIds[i]].miningPower;
```

```
            usersMiningInfo[msg.sender] = UserMiningInfo(
                msg.sender,
                block.timestamp,
                usersMiningInfo[msg.sender].totalMiningPower -
stakes[_tokenIds[i]].miningPower,
                usersMiningInfo[msg.sender].numberOfStakes - 1,
                usersMiningInfo[msg.sender].rewardBalance,
                usersMiningInfo[msg.sender].lastRateIndex
            );
```

Recommendation

We recommend using the Checks-Effects-Interactions Pattern to avoid the risk of calling unknown contracts or applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

Alleviation

The client revised the code and resolved this issue on address 0x9407c5F0a9643a13Dd87FE8f485856270A5F42f3.

TES-01 | CENTRALIZATION RELATED RISKS

Category	Severity	Location	Status
Centralization / Privilege	● Major	AxesMetaverseShard.sol (1): 1626, 1630; IAxesHub.sol (1): 765, 894, 913, 1018, 1023, 1028, 1032, 1036, 1040; AxesSummoner.sol (1): 765, 776, 787, 823, 839; AxesUpgrade.sol (1): 1094, 1101, 1111, 1149, 1153; Axes721helper.sol (1): 501, 512; BattlegroundSeasonsRegistry.sol (1): 683, 691, 699, 703, 707, 712, 776	● Acknowledged

Description

In the contract `AxesMetaverseShard.sol` , the role `PAUSER_ROLE` has authority over the following functions:

- _pause()
- _unpause()

Any compromise to the `PAUSER_ROLE` account may allow a hacker to take advantage of this authority and pause and unpause the contract.

In the contract `AxesUpgrade.sol` , the role `PAUSER_ROLE` has authority over the following functions:

- pause()
- unpause()

Any compromise to the `PAUSER_ROLE` account may allow a hacker to take advantage of this authority and pause and unpause the contract.

In the contract `AxesUpgrade.sol` , the role `DEFAULT_ADMIN_ROLE` has authority over the following functions:

- changeSettings()
- changeType()

Any compromise to the `DEFAULT_ADMIN_ROLE` account may allow a hacker to take advantage of this authority and change settings and types.

In the contract `AxesUpgrade.sol` , the role `INFO_ROLE` has authority over the following functions:

- setUpgradeNewIds()

Any compromise to the `INFO_ROLE` account may allow a hacker to take advantage of this authority and add new token ids to upgrade.

In the contract `AxesSummoner.sol` , the role `ADMIN_ROLE` has authority over the following functions

Any compromise to the `ADMIN_ROLE` account may allow a hacker to take advantage of this authority and

- create a new option for the summoning settings through `addSummonSet()`
- edit an existing variant of the summoning settings through `setSummonSet()`
- set the activity sign of an existing summonSet through `editActive()`
- realize NFT summon through `realizeNFTSummon()`
- withdraw smartToken from the contract to himself/herself through `withdraw()`

In the contract `BattlegroundSeasonsRegistry.sol` , the role `ADMIN_ROLE` has authority over the following functions

Any compromise to the `ADMIN_ROLE` account may allow a hacker to take advantage of this authority and

- set `commissionToken` through `setCommissionToken()`
- set `claimRewardCommission` through `setClaimRewardCommission()`
- set `commissionRecipient` through `setCommissionRecipient()`

In the contract `BattlegroundSeasonsRegistry.sol` , the role `OPERATOR_ROLE` has authority over the following functions

Any compromise to the `OPERATOR_ROLE` account may allow a hacker to take advantage of this authority and

- add a new season through `addSeason()`
- change reward request status through `setClaimRewardRequests()`

In the contract `BattlegroundSeasonsRegistry.sol` , the role `RoleAdmin` has authority over the following functions

Any compromise to the `RoleAdmin` account may allow a hacker to take advantage of this authority and

- grant `role` to `account` through `grantRole()`
- revoke `role` from `account` through `revokeRole()`

In the contract `Axes721helper.sol` , the role `INFO_ROLE` has authority over the following functions

Any compromise to the `INFO_ROLE` account may allow a hacker to take advantage of this authority and

- multi NFT info set through `multiInfoSet()`
- mint NFT to anyone and set NFT info through `systemMint()`

In the contract `IAxesHub.sol` , the role `RECORDER_ROLE` has authority over the following functions

Any compromise to the `RECORDER_ROLE` account may allow a hacker to take advantage of this authority and

- register an event through `register()`

In the contract `AxesStaking.sol` , the role `OPERATOR_ROLE` has authority over the following functions

Any compromise to the `OPERATOR_ROLE` account may allow a hacker to take advantage of this authority and

- update of the mining power (hash power) for the selected NFT token through `setMiningPower()` / `setMiningPowerBatched()`

In the contract `AxesStaking.sol` , the role `ADMIN_ROLE` has authority over the following functions

Any compromise to the `ADMIN_ROLE` account may allow a hacker to take advantage of this authority and

- withdrawal of funds from the reward pool by the contract owner through `withdraw()`
- set `daysDuration` and recalculate reward rate through `setDaysDuration()`
- set `durationReward` and recalculate reward rate through `setDurationReward()`
- set `energyRecoveryRate` through `setEnergyRecoveryRate()`
- set `minStakingTime` through `setMinStakingTime()`
- set `maxNumberOfStakesPerAccount` through `setMaxNumberOfStakesPerAccount()`
- set `isActive` through `toggleActive()`

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;

  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.

  OR
- Remove the risky functionality.

Alleviation

[Axes Team]:

We have a protected registry of addresses and roles for all contracts.

TES-02 | OPENZEPPELIN LIBRARY CODE INCLUDED IN SOURCE CODE

Category	Severity	Location	Status
Coding Style	● Minor	AxesMetaverseShard.sol (1): 55~1597; IAxesHub.sol (1): 12~696; AxesSummoner.sol (1): 10~692; AxesUpgrade.sol (1): 17~964; Axes721helper.sol (1): 23~479; BattlegroundSeasonsRegistry.sol (1): 11~553	● Acknowledged

Description

Multiple OpenZeppelin libraries are included in the source code file. It is highly recommended NOT to include OpenZeppelin library code directly in the source code because even slight changes to the library code may lead to major vulnerabilities/bugs.

The other contracts have the same issue.

Recommendation

It's recommended the team remove the OpenZeppelin library from the source and import the library with the "import" statement: "import "@openzeppelin/contracts/xxx".

TES-03 | THIRD PARTY DEPENDENCY

Category	Severity	Location	Status
Volatile Code	● Minor	IAxesHub.sol (1): 785, 786; AxesSummoner.sol (1): 742, 743; AxesUpgrade.sol (1): 972, 973; Axes721helper.sol (1): 492; BattlegroundSeasonsRegistry.sol (1): 665	● Acknowledged

Description

The contract is serving as the underlying entity to interact with one or more third party protocols. The scope of the audit treats third party entities as black boxes and assume their functional correctness. However, in the real world, third parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of third parties can possibly create severe impacts, such as increasing fees of third parties, migrating to new LP pools, etc.

```
972     address public immutable contract721;
```

- The contract `AxesUpgrade` interacts with third party contract with `IERC721` interface via `contract721`.

```
973     address public contractToken;
```

- The contract `AxesUpgrade` interacts with third party contract with `IERC20` interface via `contractToken`.

```
785      IERC20 public rewardToken;
```

-The contract AxesStaking interacts with third party contract with IERC20 interface via rewardToken.

```
786     IAxes721 public axes721;
```

- The contract AxesStaking interacts with third party contract with IAxes721 interface via axes721.

```
492      address public immutable contract721;
```

- The contract Axes721helper interacts with third party contract with IERC721 interface via contract721.

```
    IERC20 public override commissionToken;
```

- The contract BattlegroundSeasonsRegistry interacts with third party contract with IERC20 interface via commissionToken.

```
    IERC20 public smartToken;
```

- The contract AxesSummoner interacts with third party contract with IERC20 interface via smartToken.

```
    IAxes721 public smartNFT;
```

- The contract AxesSummoner interacts with third party contract with IAxes721 interface via smartNFT.

Recommendation

We understand that the business logic requires interaction with the third parties. We encourage the team to constantly monitor the statuses of third parties to mitigate the side effects when unexpected activities are observed.

TES-04 | LOCKED ETHER

Category	Severity	Location	Status
Language Specific	● Minor	AxesSummoner.sol (1): 798; BattlegroundSeasonsRegistry.sol (1): 760	● Resolved

Description

The contract has one or more payable functions but does not have a function to withdraw the fund.

Recommendation

We recommend removing the `payable` attribute or adding a withdraw function.

Alleviation

The client revised the code and resolved this issue on addresses 0xF147818b2e81f55Fc18FFbC7cEbbE78E7213059C and 0x536764802c82634c2A4F627B8bbef1A1543e51f1.

AUS-01 | POTENTIAL INDEX OUT OF BOUND

Category	Severity	Location	Status
Volatile Code	● Informational	AxesUpgrade.sol (1): 1137, 1141	● Acknowledged

Description

The functions `tokenIdInfo1()` and `tokenIdInfo2()` lack checking if the `_index` out of bound.

Recommendation

We advise the client to add check on the index to prevent the "index out of bound" error.

Alleviation

[Axes Team]:

This is by design so we see no risks here.

AUS-02 | POTENTIAL REQUEST FOR FREE

Category	Severity	Location	Status
Volatile Code	● Informational	AxesUpgrade.sol (1): 1033	● Acknowledged

Description

If the tokenPrice and price of the type is 0 and `_tokenPrice` is 0 , the user can create a request through the function `request()` for free.

Recommendation

We advise the client to set a reasonable token price when calling the function `changeType()`

Alleviation

[Axes Team]:

Token information is filled by backend.

AUS-03 | UNCHANGED UPGRADE COST

Category	Severity	Location	Status
Logical Issue	● Informational	AxesUpgrade.sol (1): 1056~1072	● Acknowledged

Description

Users can provide up to 10 "_ids" when calling the upgrade function. The cost to update 1 is the same to upgrade 9.

Recommendation

We would like to check if this is the intended design.

Alleviation

[Axes Team]:

Managed by backend so it's ok.

IAX-03 | `rewardRate` AND `energyRecoveryRate` ARE NOT USED

Category	Severity	Location	Status
Logical Issue	● Informational	IAxesHub.sol (1): 791	● Acknowledged

Description

The `rewardRate` is the reward for one second of stake, but the calculation of the reward is not used , it is only set by the function `_recalculateRewardRate()` . And the `energyRecoveryRate` is also set by the function `setEnergyRecoveryRate()` .

Recommendation

We advise the client to remove it if there is no plan for further usage.

Alleviation

[Axes Team]:

These variables are used by backend which interacts with SC.

TES-05 | HARDCODE ADDRESS

Category	Severity	Location	Status
Logical Issue	● Informational	AxesUpgrade.sol (1): 984, 985; Axes721helper.sol (1): 498	● Resolved

Description

There are many hardcode addresses in this codebase.

Recommendation

We advise double check the addresses before the contract is deployed onto the blockchain.

Alleviation

The client revised the code and resolved this issue on addresses 0x2baDc2159AE1C6B2E3c18Af60E2d12E03FFef75C and 0x46c5204309172977D22a10741FB03e236494C1Fe.

OPTIMIZATIONS | AXES METAVERSE V2(TOKEN) - AUDIT

ID	Title	Category	Severity	Status
AMS-02	Function Should Be Declared External	Gas Optimization	Optimization	● Acknowledged

AMS-02 | FUNCTION SHOULD BE DECLARED EXTERNAL

Category	Severity	Location	Status
Gas Optimization	● Optimization	AxesMetaverseShard.sol (1): 1622, 1626, 1630	● Acknowledged

Description

The functions which are never called internally within the contract should have external visibility for gas optimization.

```
1622      function snapshot() public onlyRole(SNAPSHOT_ROLE) {
```

```
1626      function pause() public onlyRole(PAUSER_ROLE) {
```

```
1630      function unpause() public onlyRole(PAUSER_ROLE) {
```

Recommendation

We advise to change the visibility of the aforementioned functions to `external`.

FORMAL VERIFICATION | AXES METAVERSE V2(TOKEN) - AUDIT

Formal guarantees about the behavior of smart contracts can be obtained by reasoning about properties relating to the entire contract (e.g. contract invariants) or to specific functions of the contract. Once such properties are proven to be valid, they guarantee that the contract behaves as specified by the property. As part of this audit, we applied automated formal verification (symbolic model checking) to prove that well-known functions in the smart contracts adhere to their expected behavior.

Considered Functions And Scope

Verification of ERC-20 compliance

We verified properties of the public interface of those token contracts that implement the ERC-20 interface. This covers

- Functions `transfer` and `transferFrom` that are widely used for token transfers,
- functions `approve` and `allowance` that enable the owner of an account to delegate a certain subset of her tokens to another account (i.e. to grant an allowance), and
- the functions `balanceOf` and `totalSupply` , which are verified to correctly reflect the internal state of the contract.

The properties that were considered within the scope of this audit are as follows:

Property Name	Title
erc20-transfer-revert-zero	Function `transfer` Prevents Transfers to the Zero Address
erc20-transfer-correct-amount	Function `transfer` Transfers the Correct Amount in Non-self Transfers
erc20-transfer-succeed-normal	Function `transfer` Succeeds on Admissible Non-self Transfers
erc20-transfer-succeed-self	Function `transfer` Succeeds on Admissible Self Transfers
erc20-transfer-correct-amount-self	Function `transfer` Transfers the Correct Amount in Self Transfers
erc20-transfer-change-state	Function `transfer` Has No Unexpected State Changes
erc20-transfer-exceed-balance	Function `transfer` Fails if Requested Amount Exceeds Available Balance
erc20-transfer-recipient-overflow	Function `transfer` Prevents Overflows in the Recipient's Balance
erc20-transfer-false	If Function `transfer` Returns `false` , the Contract State Has Not Been Changed
erc20-transfer-never-return-false	Function `transfer` Never Returns `false`

Property Name	Title
erc20-transferfrom-revert-from-zero	Function `transferFrom` Fails for Transfers From the Zero Address
erc20-transferfrom-revert-to-zero	Function `transferFrom` Fails for Transfers To the Zero Address
erc20-transferfrom-succeed-normal	Function `transferFrom` Succeeds on Admissible Non-self Transfers
erc20-transferfrom-correct-amount	Function `transferFrom` Transfers the Correct Amount in Non-self Transfers
erc20-transferfrom-correct-amount-self	Function `transferFrom` Performs Self Transfers Correctly
erc20-transferfrom-succeed-self	Function `transferFrom` Succeeds on Admissible Self Transfers
erc20-transferfrom-correct-allowance	Function `transferFrom` Updated the Allowance Correctly
erc20-transferfrom-fail-exceed-balance	Function `transferFrom` Fails if the Requested Amount Exceeds the Available Balance
erc20-transferfrom-change-state	Function `transferFrom` Has No Unexpected State Changes
erc20-transferfrom-fail-exceed-allowance	Function `transferFrom` Fails if the Requested Amount Exceeds the Available Allowance
erc20-totalsupply-succeed-always	Function `totalSupply` Always Succeeds
erc20-totalsupply-correct-value	Function `totalSupply` Returns the Value of the Corresponding State Variable
erc20-transferfrom-fail-recipient-overflow	Function `transferFrom` Prevents Overflows in the Recipient's Balance
erc20-totalsupply-change-state	Function `totalSupply` Does Not Change the Contract's State
erc20-transferfrom-false	If Function `transferFrom` Returns `false`, the Contract's State Has Not Been Changed
erc20-balanceof-succeed-always	Function `balanceOf` Always Succeeds
erc20-balanceof-correct-value	Function `balanceOf` Returns the Correct Value
erc20-transferfrom-never-return-false	Function `transferFrom` Never Returns `false`
erc20-balanceof-change-state	Function `balanceOf` Does Not Change the Contract's State
erc20-allowance-succeed-always	Function `allowance` Always Succeeds
erc20-allowance-correct-value	Function `allowance` Returns Correct Value

Property Name	Title
erc20-allowance-change-state	Function `allowance` Does Not Change the Contract's State
erc20-approve-revert-zero	Function `approve` Prevents Giving Approvals For the Zero Address
erc20-approve-succeed-normal	Function `approve` Succeeds for Admissible Inputs
erc20-approve-correct-amount	Function `approve` Updates the Approval Mapping Correctly
erc20-approve-change-state	Function `approve` Has No Unexpected State Changes
erc20-approve-false	If Function `approve` Returns `false`, the Contract's State Has Not Been Changed
erc20-approve-never-return-false	Function `approve` Never Returns `false`

Verification Results

In the remainder of this section, we list all contracts where model checking of at least one property was not successful. There are several reasons why this could happen:

- Model checking reports a counterexample that violates the property. Depending on the counterexample,this occurs if
  - The specification of the property is too generic and does not accurately capture the intended behavior of the smart contract. In that case, the counterexample does not indicate a problem in the underlying smart contract. We report such instances as being "inapplicable".
  - The property is applicable to the smart contract. In that case, the counterexample showcases a problem in the smart contract and a correspond finding is reported separately in the Findings section of this report. In the following tables, we report such instances as "invalid". The distinction between spurious and actual counterexamples is done manually by the auditors.
- The model checking result is inconclusive. Such a result does not indicate a problem in the underlying smart contract. An inconclusive result may occur if
  - The model checking engine fails to construct a proof. This can happen if the logical deductions necessary are beyond the capabilities of the automated reasoning tool. It is a technical limitation of all proof engines and cannot be avoided in general.
  - The model checking engine runs out of time or memory and did not produce a result. This can happen if automatic abstraction techniques are ineffective or of the state space is too big.

Contract AxesMetaverseShard (Source File AxesMetaverseShard.sol)

Detailed results for function `transfer`

Property Name	Final Result	Remarks
erc20-transfer-revert-zero	● True	
erc20-transfer-correct-amount	● True	
erc20-transfer-succeed-normal	● False	
erc20-transfer-succeed-self	● False	
erc20-transfer-correct-amount-self	● True	
erc20-transfer-change-state	● True	
erc20-transfer-exceed-balance	● True	
erc20-transfer-recipient-overflow	● True	
erc20-transfer-false	● True	
erc20-transfer-never-return-false	● True	

Detailed results for function `transferFrom`

Property Name	Final Result	Remarks
erc20-transferfrom-revert-from-zero	🟢 True	
erc20-transferfrom-revert-to-zero	🟢 True	
erc20-transferfrom-succeed-normal	🔴 False	
erc20-transferfrom-correct-amount	🟢 True	
erc20-transferfrom-correct-amount-self	🟢 True	
erc20-transferfrom-succeed-self	🔴 False	
erc20-transferfrom-correct-allowance	🟢 True	
erc20-transferfrom-fail-exceed-balance	🟢 True	
erc20-transferfrom-change-state	🟢 True	
erc20-transferfrom-fail-exceed-allowance	🟢 True	
erc20-transferfrom-fail-recipient-overflow	🟢 True	
erc20-transferfrom-false	🟢 True	
erc20-transferfrom-never-return-false	🟢 True	

Detailed results for function `totalSupply`

Property Name	Final Result	Remarks
erc20-totalsupply-succeed-always	🟢 True	
erc20-totalsupply-correct-value	🟢 True	
erc20-totalsupply-change-state	🟢 True	

Detailed results for function `balanceOf`

Property Name	Final Result	Remarks
erc20-balanceof-succeed-always	● True	
erc20-balanceof-correct-value	● True	
erc20-balanceof-change-state	● True	

Detailed results for function `allowance`

Property Name	Final Result	Remarks
erc20-allowance-succeed-always	● True	
erc20-allowance-correct-value	● True	
erc20-allowance-change-state	● True	

Detailed results for function `approve`

Property Name	Final Result	Remarks
erc20-approve-revert-zero	● True	
erc20-approve-succeed-normal	● True	
erc20-approve-correct-amount	● True	
erc20-approve-change-state	● True	
erc20-approve-false	● True	
erc20-approve-never-return-false	● True	

APPENDIX | AXES METAVERSE V2(TOKEN) - AUDIT

Finding Categories

Categories	Description
Centralization / Privilege	Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.
Gas Optimization	Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.
Logical Issue	Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.
Language Specific	Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.
Coding Style	Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

DISCLAIMER | CERTIK

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you (“Customer” or the “Company”) in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK’s prior written consent in each instance.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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. CertiK’s position is that each company and individual are responsible for their own due diligence and continuous security. CertiK’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND “AS AVAILABLE” AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER’S OR ANY OTHER PERSON’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE

FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK’S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER’S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED “AS IS” AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK’S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

CertiK | Securing the Web3 World

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.