CERTIK

Security Assessment

Draft (Internal Use Only)

ApeIron - Token Contract

CertiK Verified on Sept 19th, 2022

CertiK Verified on Sept 19th, 2022

ApeIron - Token Contract

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

Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	Ethereum	Manual Review, Static Analysis
LANGUAGE	**TIMELINE**	**KEY COMPONENTS**
Solidity	Delivered on 09/19/2022	N/A

CODEBASE
https://github.com/FoonieMagus/ApeironTokenContract
...View All

COMMITS
26f30a14af91f2cf58697d51cd0753bf8603952d
...View All

Vulnerability Summary

28	15	0	2	11	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Unresolved

3	Critical	1 Resolved, 2 Partially Resolved	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.
14	Major	3 Resolved, 11 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.
1	Medium	1 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
6	Minor	6 Resolved	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.
4	Informational	4 Resolved	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 | APEIRON - TOKEN CONTRACT

CODEBASE | APEIRON - TOKEN CONTRACT

Repository

https://github.com/FoonieMagus/ApeironTokenContract

Commit

26f30a14af91f2cf58697d51cd0753bf8603952d

AUDIT SCOPE | APEIRON - TOKEN CONTRACT

20 files audited ● 11 files with Acknowledged findings ● 2 files with Resolved findings ● 7 files without findings

ID	File	SHA256 Checksum
APT	contracts/utils/AccessProtected.sol	0abb9595e25aabef3df6bac7def069c9315626e24c62dd6069d494c36a295c96
APU	contracts/utils/AccessProtectedUpgradable.sol	fe158483892edcae570ae5bcc12b0f797a854444247e3d132cdf39d404f60507
ANM	contracts/ANIMAchild.sol	f1d8eac77d846a44ffa26e69e821f8416804a4d6dcbedb9d240773d74a3c6b47
ANA	contracts/ANIMAroot.sol	e2247007d6cde41ef434f4bb22187148ba43004f6d0825b18434c3bb6d9cb80d
APR	contracts/APRSVesting.sol	6a4371a4295c453a663edc004bb4374b0c779fc7bc468f4ebea73e21bb7cac9e
APS	contracts/APRSchild.sol	6de5e9ca23d3df91f913c46b419e400e4563e4a25e0daaac58bfdd8420e9eb0a
APA	contracts/APRSroot.sol	72d4bb0c5bc49d0b1732d30f0e7ddc693f135a58c0db09e06c8b2c1b1bee3f04
EMA	contracts/ExpeditionMeta.sol	7aea742b6ae36b6248283b4c2ec2d88fa2578731f46a07ef5bf5793cc7a7583b
EPA	contracts/ExpeditionPrizes.sol	82ce4271788ccc3d1f91cadc9c2d2eec7a02ef982e67216273d4d474f2fbf2d9
ESA	contracts/ExpeditionStakingAndKeys.sol	ca367b89649c22bde5056df35f44b90c1c70847d5eadca11a619a6b1cc884f7e
NEA	contracts/NebulaExpedition.sol	05fba26cdc6396125f788bd91a680c5bd484de83b66cb7d4d06ac9990a42a345
ANI	contracts/ANIMA.sol	6abd46cea3360c3911a535332bbf2b15334d4ee75552c1c218cc0a87d5408e87
AGC	contracts/ApeironGodiverseCollection.sol	4226d8b5d18081bb6bf693717b0475446542bab092a7c3aa9cc8b23f18614578
IAG	contracts/interfaces/IApeironGodiverseCollection.sol	7258e0e1886526d21884a0d94930b848b2e336772f3b4c6f6ec7f71d38d02d92

ID	File	SHA256 Checksum
IAP	contracts/interfaces/IApeironPlanet.sol	dc3610b046a7dfdc33ef4db73e5ac808b6f0874607e5f4393281ed75c1d8bf14
IAS	contracts/interfaces/IApeironStar.sol	709ebc6adbe878666db63f838c91a874ceef5d4c43a12988bebe439069fd8511
ACM	contracts/utils/AccessControlMixin.sol	e7694bae6e32e7d5ee0833e2792d5510d5ba0d7998f9f827212e17624574613f
ICT	contracts/utils/IChildToken.sol	53785792a073ca4735bbfe8cdaea7397fe78d4795c1044a7d6461febad4d4c5a
IME	contracts/utils/IMintableERC20.sol	3f594a69416816e1ded1d314b7e563d8b3b9ba40d381bf756ff20f6fe81e357a
SAM	contracts/StakeAssetMeta.sol	3dfd2b5b23d3c2b4dceba229959d01c437d01d88df742f3316703bc497a9c4b2

APPROACH & METHODS | APEIRON - TOKEN CONTRACT

This report has been prepared for ApeIron - Token Contract to discover issues and vulnerabilities in the source code of the ApeIron - Token Contract 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 | APEIRON - TOKEN CONTRACT

28	3	14	1	6	4
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Apelron - Token Contract. Through this audit, we have uncovered 28 issues ranging from different severity levels. Utilizing Static Analysis techniques to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
AGC-01	**Centralization Risks In ApeironGodiverseCollection.Sol**	**Centralization / Privilege**	**Major**	**Resolved**
ANA-01	**Centralization Risks In ANIMAroot.Sol**	**Centralization / Privilege**	**Major**	**Acknowledged**
ANM-01	**Centralization Risks In ANIMAchild.Sol**	**Centralization / Privilege**	**Major**	**Acknowledged**
APA-01	**Centralization Risks In APRSroot.Sol**	**Centralization / Privilege**	**Major**	**Acknowledged**
APR-01	**Centralization Risks In APRSVesting.Sol**	**Centralization / Privilege**	**Major**	**Acknowledged**
APR-02	Potential Reentrancy Attack (Events)	Volatile Code	Minor	Resolved
APR-03	Unchecked ERC-20 `transfer()` / `transferFrom()` Call	Volatile Code	Minor	Resolved
APS-01	**Centralization Risks In APRSchild.Sol**	**Centralization / Privilege**	**Major**	**Acknowledged**
APT-01	**Centralization Risks In AccessProtected.Sol**	**Centralization / Privilege**	**Major**	**Acknowledged**
APU-01	**Centralization Risks In AccessProtectedUpgradable.Sol**	**Centralization / Privilege**	**Major**	**Acknowledged**
ATC-01	**Centralized Control Of Contract Upgrade**	**Centralization / Privilege**	**Major**	**Acknowledged**

ID	Title	Category	Severity	Status
ATC-02	Missing Zero Address Validation	Volatile Code	Minor	Resolved
EPA-01	A Winner Can Lock All Treasures	Logical Issue	Critical	Resolved
EPA-02	**Centralization Risks In ExpeditionPrizes.Sol**	**Centralization / Privilege**	**Major**	**Acknowledged**
ESA-01	**Centralization Risks In ExpeditionStakingAndKeys.Sol**	**Centralization / Privilege**	**Major**	**Acknowledged**
ESA-02	Missing Zero Amount Check	Logical Issue	Medium	Resolved
ESA-03	Lack Of Specified Rate Range Restriction	Logical Issue	Minor	Resolved
ESA-04	Missing Zero Address Validation	Volatile Code	Minor	Resolved
MER-01	**Centralization Risks In MockERC1155.Sol**	**Centralization / Privilege**	**Major**	**Resolved**
NEA-01	Users Can Join Multiple Expeditions While Only Stake Onetime	Logical Issue	Critical	Partially Resolved
NEA-02	Users Can Join An Expedition By Staking Zero Asset	Logical Issue	Critical	Partially Resolved
NEA-03	**Centralization Risks In NebulaExpedition.Sol**	**Centralization / Privilege**	**Major**	**Acknowledged**
NEA-04	Users Can Join An Expedition Right Before It Ends	Logical Issue	Major	Resolved
NEA-05	Lack Of Check For Start Time	Logical Issue	Minor	Resolved
ATC-06	Use Enum Data Structure	Language Specific	Informational	Resolved
ATC-07	Missing Error Messages	Coding Style	Informational	Resolved

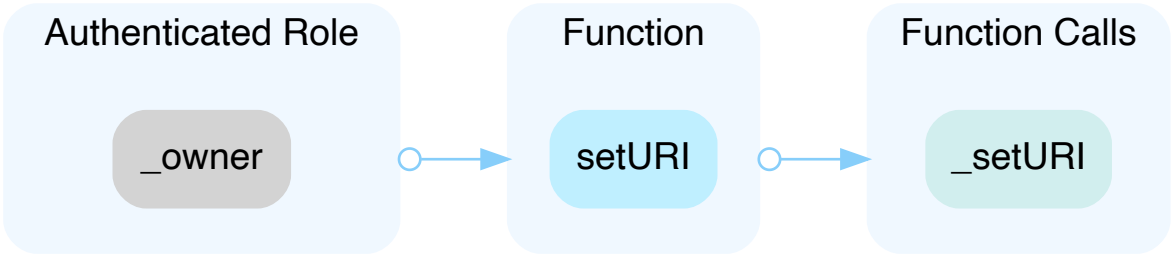
ID	Title	Category	Severity	Status
EPA-03	Typo In Comments	Coding Style	Informational	Resolved
ESA-06	Unnecessary Checks	Logical Issue	Informational	Resolved

AGC-01 | CENTRALIZATION RISKS IN APEIRONGODIVERSECOLLECTION.SOL

Category	Severity	Location	Status
Centralization / Privilege	Major	contracts/ApeironGodiverseCollection.sol (base): 18	Resolved

Description

In the contract `ApeironGodiverseCollection` the role `_owner` has authority over the function `setURI` as shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and set the new URI.



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

[Apeiron]
Multi-sign proxy address: https://polygonscan.com/address/0xb0B432420d27645c24C774fe3e3d7efF1AA1dc81

Transaction proof for transferring ownership to a multi-signature proxy:
https://polygonscan.com/tx/0xde01d120734c7084f91c5617378b27d0e0cc0b842a6fe711020cc00de06a4b23

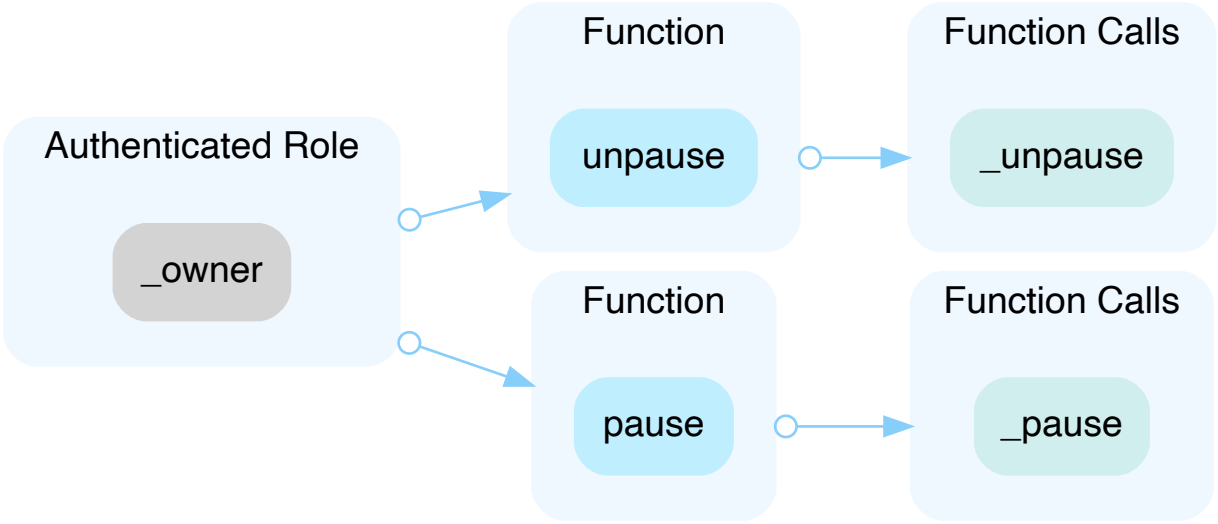
[CertiK] The `Apeiron` team resolved the issue by transferring ownership to a gnosis Proxy.

ANA-01 | CENTRALIZATION RISKS IN ANIMAROOT.SOL

Category	Severity	Location	Status
Centralization / Privilege	Major	contracts/ANIMAroot.sol (base): 36, 40	Acknowledged

Description

In the contract `AnimaRoot` the role `_owner` has authority over the functions `pause` and `unpause` as shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and pause and unpause the contract.



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

[Apeiron] - Will transfer ownership to a multi-sign wallet after deployment like the expedition one.

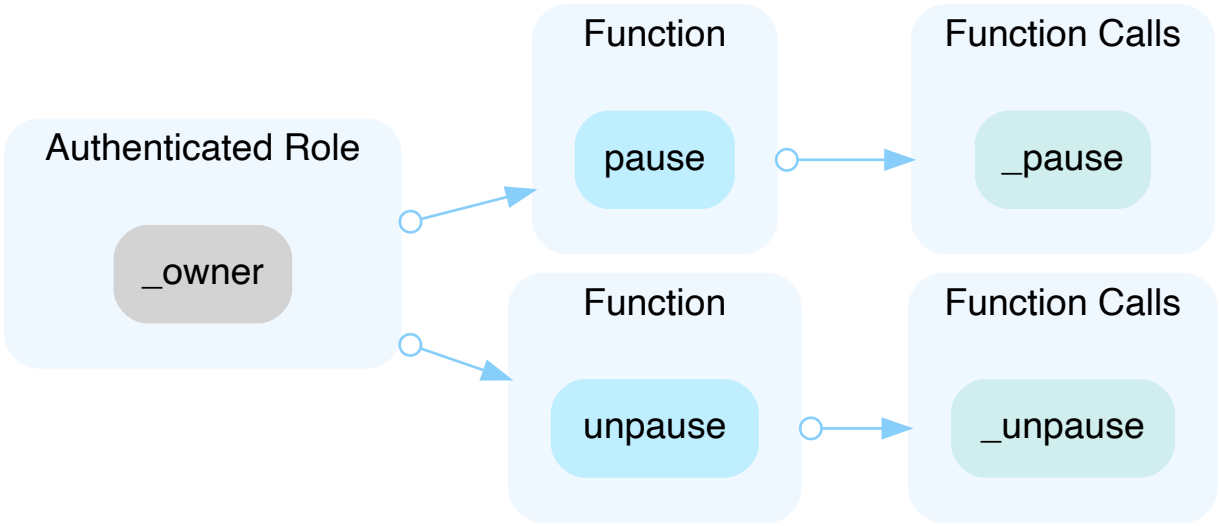
[CertiK] - The Apeiron team acknowledged the issue and will transfer ownership once the contracts have been deployed.

ANM-01 | CENTRALIZATION RISKS IN ANIMACHILD.SOL

Category	Severity	Location	Status
Centralization / Privilege	Major	contracts/ANIMAchild.sol (base): 37, 41	Acknowledged

Description

In the contract `AnimaChild` the role `_owner` has authority over the functions `pause` and `unpause` as shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and pause and unpause the contract.



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

[Apeiron] - Will transfer ownership to a multi-sign wallet after deployment like the expedition one.

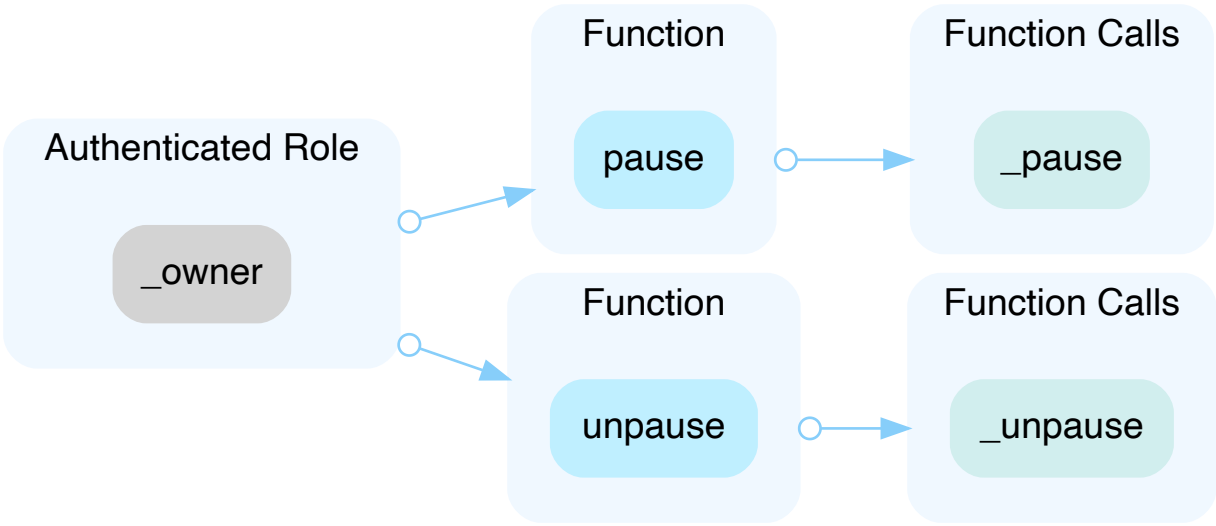
[CertiK] - The Apeiron team acknowledged the issue and will transfer ownership once the contracts have been deployed.

APA-01 | CENTRALIZATION RISKS IN APRSROOT.SOL

Category	Severity	Location	Status
Centralization / Privilege	**Major**	**contracts/APRSroot.sol (base): 36, 40**	**Acknowledged**

Description

In the contract `ApeirosRoot` the role `_owner` has authority over the functions `pause` and `unpause` as shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and pause and unpause the contract.



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

[Apeiron] - Will transfer ownership to a multi-sign wallet after deployment like the expedition one.

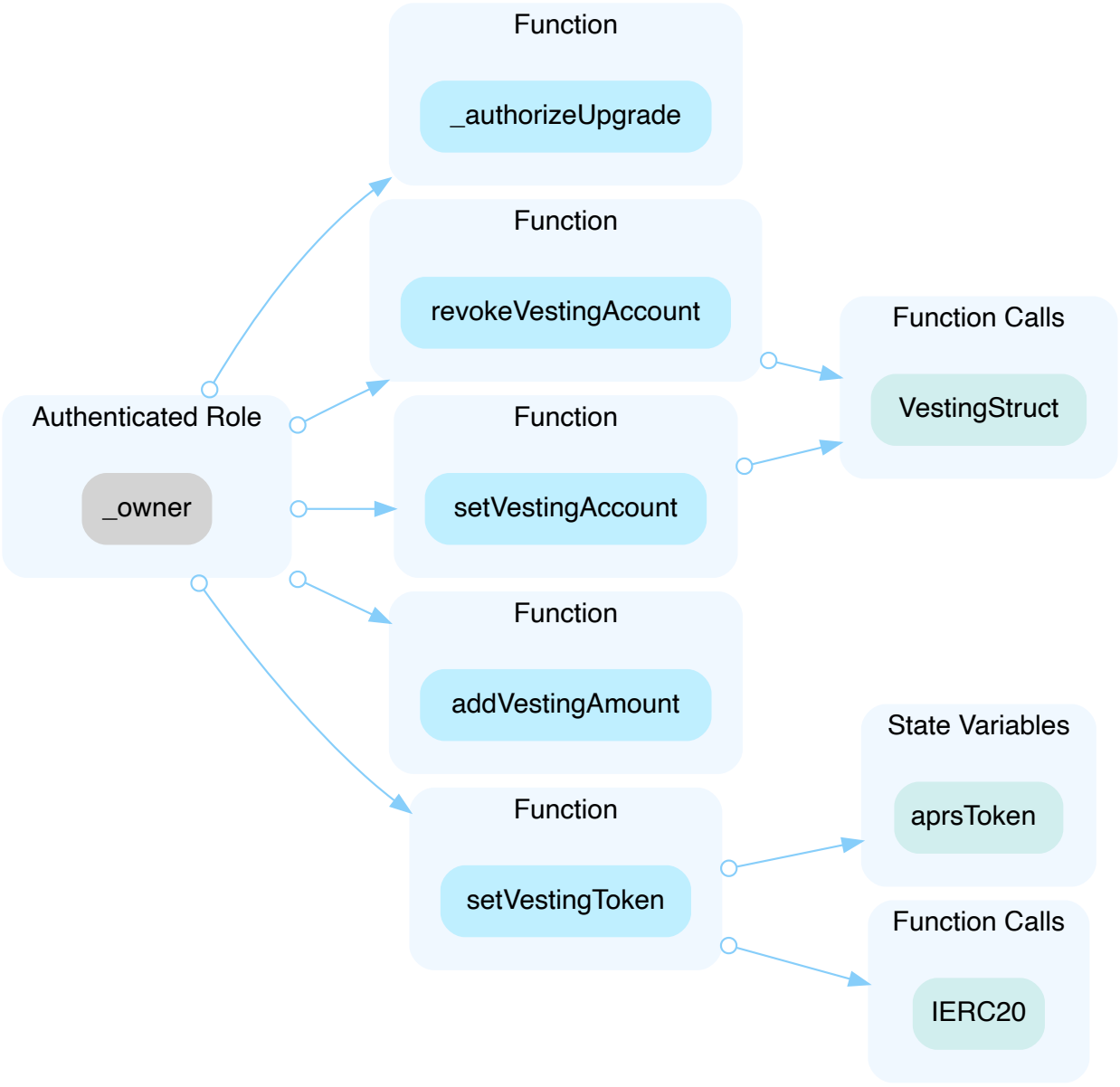
[CertiK] - The Apeiron team acknowledged the issue and will transfer ownership once the contracts have been deployed.

APR-01 | CENTRALIZATION RISKS IN APRSVESTING.SOL

Category	Severity	Location	Status
Centralization / Privilege	**Major**	**contracts/APRSVesting.sol (base): 37, 44, 49, 68, 80**	**Acknowledged**

Description

In the contract `ApeirosVesting` the role `_owner` has authority over the functions `_authorizeUpgrade`, `revokeVestingAccount`, `setVestingAccount`, `addVestingAmount` and `setVestingToken` as shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and authorize the upgrade. In addition, this compromise will also allow the hacker to take advantage of this authority and set and/or revoke the vesting account, as well as set the vesting token and add the vesting amount.



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

[Apeiron] - Will transfer ownership to a multi-sign wallet after deployment like the expedition one.

[CertiK] - The Apeiron team acknowledged the issue and will transfer ownership once the contracts have been deployed.

APR-02 | POTENTIAL REENTRANCY ATTACK (EVENTS)

Category	Severity	Location	Status
Volatile Code	Minor	contracts/APRSVesting.sol (base): 136, 137	Resolved

Description

This finding has a minor impact because the reentrancy only causes out-of-order events.

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)

```
136         aprsToken.transfer(msg.sender, releasable);
```

Events emitted after the call(s)

```
137         emit VestingReleased(msg.sender, releasable);
```

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

[Apeiron] fixed on 9669ba77f21f6aaac1d4920d819a6542f087bfad

[CertiK] The `Apeiron` team resolved the issue by applying OpenZeppelin ReentrancyGuard library. The changes can be seen in commit hash 9669ba77f21f6aaac1d4920d819a6542f087bfad at line 140 of APRSVesting.sol.

APR-03 | UNCHECKED ERC-20 `transfer()` / `transferFrom()` CALL

Category	Severity	Location	Status
Volatile Code	Minor	contracts/APRSVesting.sol (base): 136	Resolved

Description

The return value of the transfer()/transferFrom() call is not checked.

```
136         aprsToken.transfer(msg.sender, releasable);
```

Recommendation

Since some ERC-20 tokens return no values and others return a `bool` value, they should be handled with care. We advise using the OpenZeppelin's `SafeERC20.sol` implementation to interact with the `transfer()` and `transferFrom()` functions of external ERC-20 tokens. The OpenZeppelin implementation checks for the existence of a return value and reverts if `false` is returned, making it compatible with all ERC-20 token implementations.

Alleviation

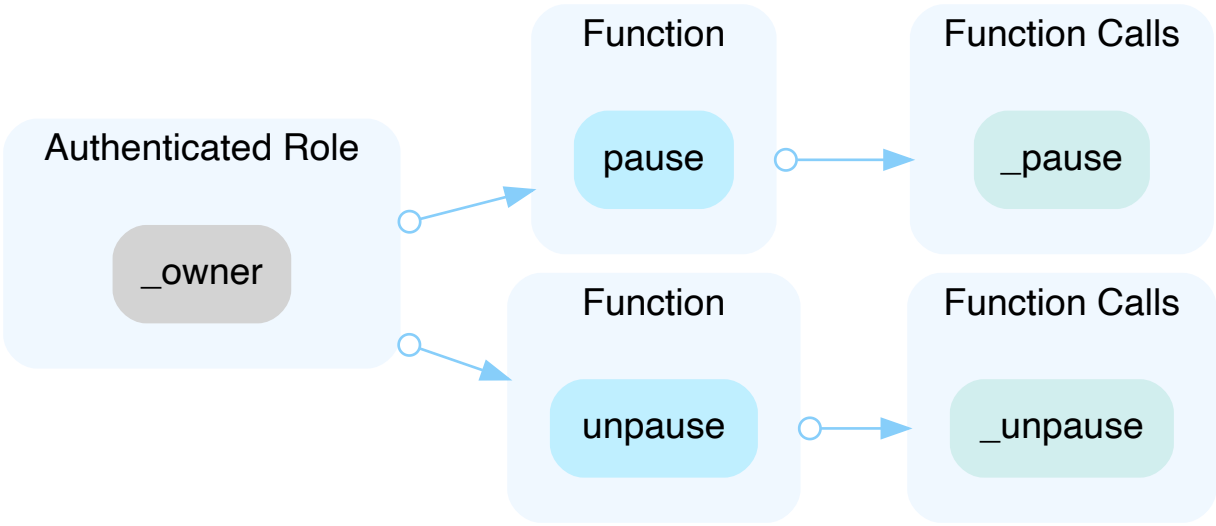
[CertiK] - The `Apeiron` team resolved this issue by implementing the `safeTransfer()` function defined in the `safeERC20Upgradeable.sol` contract. This function will make a low-level call to an an ERC20 contract and ensure that return value is `True`. The changes can be seen in the following commit 76437bb2d1d06847479eed89f1f9ce1b8a735c65f88785c5c0b1c04724459072

APS-01 | CENTRALIZATION RISKS IN APRSCHILD.SOL

Category	Severity	Location	Status
Centralization / Privilege	Major	contracts/APRSchild.sol (base): 37, 41	Acknowledged

Description

In the contract `ApeirosChild` the role `_owner` has authority over the functions `pause` and `unpause` as shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and pause and unpause the contract.



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

[Apeiron] - Will transfer ownership to a multi-sign wallet after deployment like the expedition one.

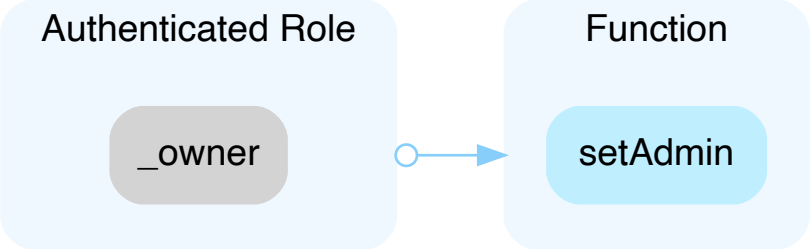
[CertiK] - The Apeiron team acknowledged the issue and will transfer ownership once the contracts have been deployed.

APT-01 | CENTRALIZATION RISKS IN ACCESSPROTECTED.SOL

Category	Severity	Location	Status
Centralization / Privilege	**Major**	**contracts/utils/AccessProtected.sol (base): 18**	**Acknowledged**

Description

In the contract `AccessProtected` the role `_owner` has authority over the function `setAdmin` as shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and set the desired account address as admin.



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

[Apeiron] - Will transfer ownership to a multi-sign wallet after deployment like the expedition one.

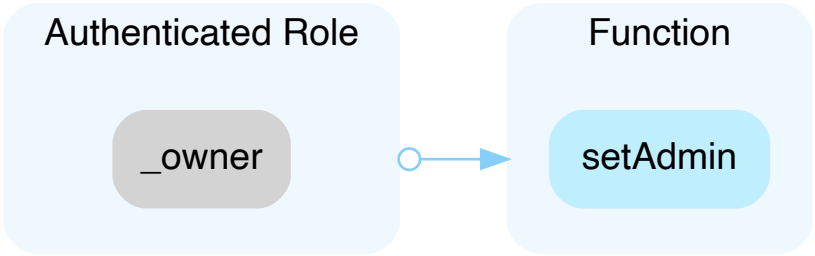
[CertiK] - The Apeiron team acknowledged the issue and will transfer ownership once the contracts have been deployed.

APU-01 CENTRALIZATION RISKS IN ACCESSPROTECTEDUPGRADABLE.SOL

Category	Severity	Location	Status
Centralization / Privilege	Major	contracts/utils/AccessProtectedUpgradable.sol (base): 18	Acknowledged

Description

In the contract `AccessProtectedUpgradable` the role `_owner` has authority over the function `setAdmin` as shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and set the desired account address as admin.



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

[Apeiron] - Will transfer ownership to a multi-sign wallet after deployment like the expedition one.

[CertiK] - The Apeiron team acknowledged the issue and will transfer ownership once the contracts have been deployed.

ATC-01 | CENTRALIZED CONTROL OF CONTRACT UPGRADE

Category	Severity	Location	Status
Centralization / Privilege	**Major**	**contracts/APRSVesting.sol (base): 10; contracts/ExpeditionPrizes.sol (base): 23; contracts/ExpeditionStakingAndKeys.sol (base): 19; contracts/NebulaExpedition.sol (base): 18; contracts/utils/AccessProtectedUpgradable.sol (base): 7**	**Acknowledged**

Description

`APRSVesting.sol`, `ExpeditionPrizes.sol`, `ExpeditionStakingAndKeys.sol`, `NebulaExpedition.sol` and `AccessProtectedUpgradable.sol` are upgradeable contracts, the owner can upgrade the contract without the community's commitment. If an attacker compromises the account, he can change the implementation of the contract and drain tokens from the contract.

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

[Apeiron] - Will transfer ownership to a multi-sign wallet after deployment like the expedition one.

[CertiK] - The Apeiron team acknowledged the issue and will transfer ownership once the contracts have been deployed.

ATC-02 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	Minor	contracts/ExpeditionPrizes.sol (base): 148; contracts/NebulaExpedition.sol (base): 162	Resolved

Description

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

File: NebulaExpedition.sol (Line 162, Function `setStakingAndKeys`)

```
stakingAndKeysAddress = _address;
```

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

File: ExpeditionPrizes.sol (Line 148, Function `setExpedition`)

```
expeditionAddress = _address;
```

- `_address` 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

[CertiK] - The `Apeiron` team resolved the issue by following the recommendation given above. The changes can be seen in the following commit a0af22bd837db162cdc3e673a7e6bb7ece21c0ac

EPA-01 | A WINNER CAN LOCK ALL TREASURES

Category	Severity	Location	Status
Logical Issue	Critical	contracts/ExpeditionPrizes.sol (base): 202~205	Resolved

Description

If a winner does not have enough keys required, then the function `stakingAndKeys.burnUserKeys()` will be reverted because of underflow. Hence admins will be unable to assign treasures to all winners.

Recommendation

We recommend putting the function `stakingAndKeys.burnUserKeys()` inside the function claimTreasure().

Alleviation

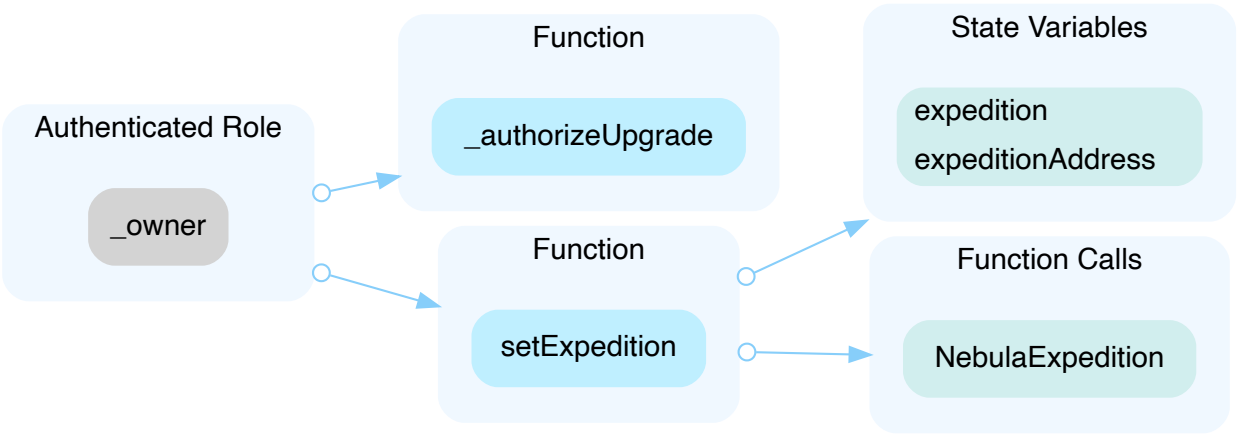
[CertiK] - The `Apeiron` team resolved the issue by following the recommendation above. The changes can be seen in the following commit a4b7f6ef843c8152e3eb964ac42fb908beea7afa

EPA-02 | CENTRALIZATION RISKS IN EXPEDITIONPRIZES.SOL

Category	Severity	Location	Status
Centralization / Privilege	Major	contracts/ExpeditionPrizes.sol (base): 97, 144, 154~157	Acknowledged

Description

In the contract `ExpeditionPrizes` the role `_owner` has authority over the functions `_authorizeUpgrade` and `setExpedition` as shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and authorize the upgrade as well as set the expedition to link up with the NebulaExpedition contract.



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

[Apeiron] - Will transfer ownership to a multi-sign wallet after deployment like the expedition one.

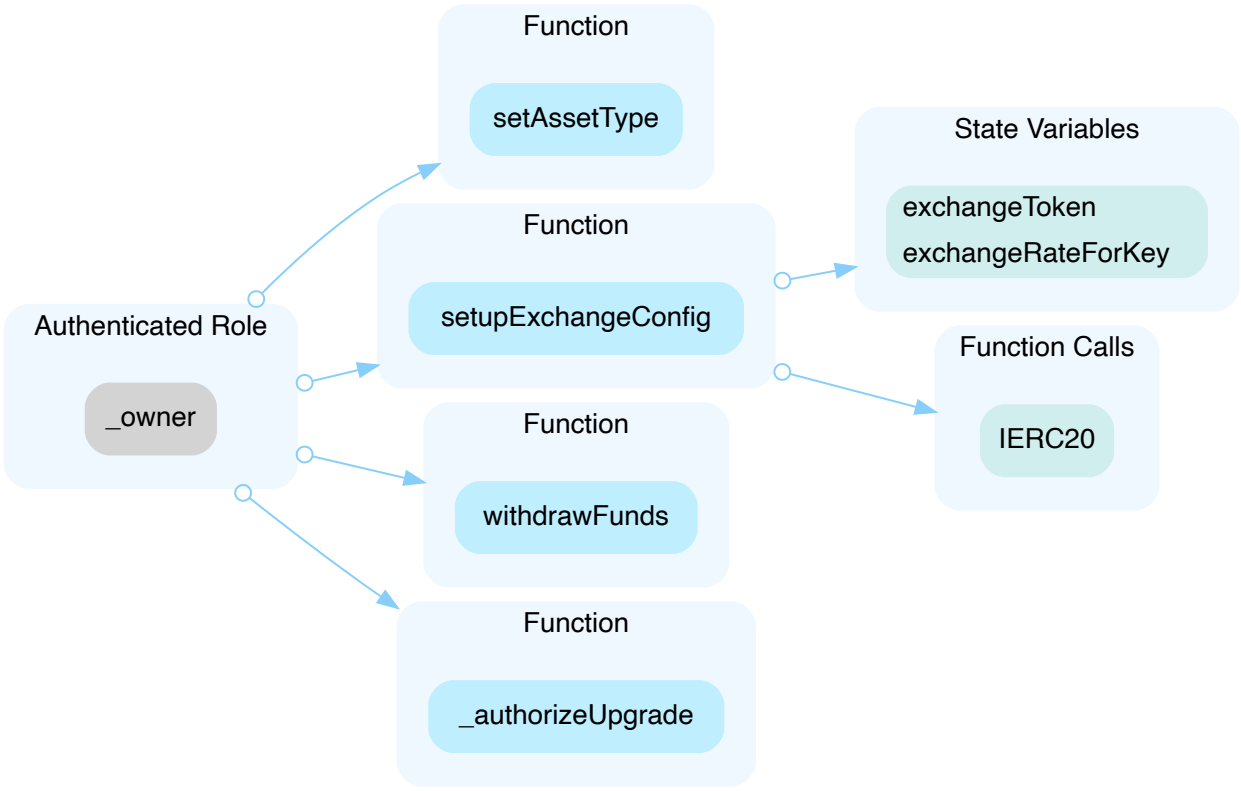
[CertiK] - The Apeiron team acknowledged the issue and will transfer ownership once the contracts have been deployed.

ESA-01 CENTRALIZATION RISKS IN EXPEDITIONSTAKINGANDKEYS.SOL

Category	Severity	Location	Status
Centralization / Privilege	Major	contracts/ExpeditionStakingAndKeys.sol (base): 102, 151, 196~201, 334~339, 417, 441~445, 486, 496	Acknowledged

Description

In the contract `ExpeditionStakingAndKeys` the role `_owner` has authority over the functions `setAssetType`, `setupExchangeConfig`, `withdrawFunds`, and `_authorizeUpgrade` as shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and permanently lock users' funds by calling the setAssetType() function and setting all assets to the type ASSET_TYPE_NONE. In addition, in case of hacker taking advantage of this authority, they will also be able to setup the exchange keys, withdraw the funds to the desired address, and authorize the upgrade.



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

[Apeiron] - Will transfer ownership to a multi-sign wallet after deployment like the expedition one.

[CertiK] - The Apeiron team acknowledged the issue and will transfer ownership once the contracts have been deployed.

ESA-02 | MISSING ZERO AMOUNT CHECK

Category	Severity	Location	Status
Logical Issue	Medium	contracts/ExpeditionStakingAndKeys.sol (base): 168, 200, 374	Resolved

Description

1. Missing zero amount check on `Asset.amount`, so the function getStakedAssets() will return assets that have been unstaked.
2.
3. Missing zero amount check on `_amounts`, so attackers can add any assets into the `stakedAssets` mapping.
4. Missing zero amount check on `_amounts[i]` so an unstaked asset is still in the `stakedAssets` mapping.

Recommendation

We recommend adding a zero amount check on all of the variables mentioned above and deleting unstaked assets from the `stakedAssets` mapping.

Alleviation

[CertiK] - The `Apeiron` team resolved the issue by including checks that ensure a zero amount is not passed. The changes can be seen in the following commit f383b27f646871d8ef16c9f1fe5d383bb40c4554

ESA-03 | LACK OF SPECIFIED RATE RANGE RESTRICTION

Category	Severity	Location	Status
Logical Issue	Minor	contracts/ExpeditionStakingAndKeys.sol (base): 432	Resolved

Description

The `owner` of the contract has permission to modify the fees without limitation and the exchange rate can change.

Therefore, in the extreme case, that fee could be a very large amount of value, which might cause unexpected loss to the project and users.

Recommendation

We advise the client to set a reasonable range restriction for the aforementioned states to ensure the fair distribution of the fees/tokens.

Alleviation

[Apeiron] - We added a timing restriction from previous expeditions.

[CertiK] - The changes can be seen in this following commit 9669ba77f21f6aaac1d4920d819a6542f087bfad

ESA-04 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	Minor	contracts/ExpeditionStakingAndKeys.sol (base): 418	Resolved

Description

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

File: ExpeditionStakingAndKeys.sol (Line 431, Function `setupExchangeConfig`)

```
exchangeToken = IERC20(_exchangeTokenAddress);
```

- `_exchangeTokenAddress` 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

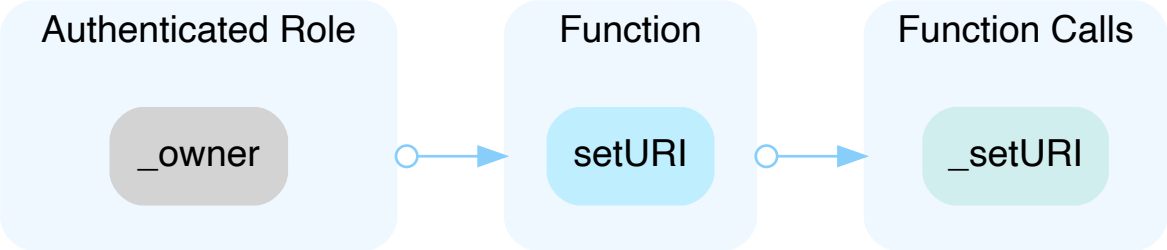
[CertiK] - The Apeiron team resolved the issue by including a check to ensure that a zero-address was not passed. The changes can be seen in the following commit c726e1593e4628f0272290c4b70210baf5d865ae

MER-01 | CENTRALIZATION RISKS IN MOCKERC1155.SOL

Category	Severity	Location	Status
Centralization / Privilege	Major	contracts/mock/MockERC1155.sol (base): 12	Resolved

Description

In the contract `MockERC1155` the role `_owner` has authority over the function `setURI` as shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and set the new URI.



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

[CertiK] - Since this contract is for testing purposes we can consider this issue as resolved.

NEA-01 USERS CAN JOIN MULTIPLE EXPEDITIONS WHILE ONLY STAKE ONETIME

Category	Severity	Location	Status
Logical Issue	Critical	contracts/NebulaExpedition.sol (base): 214~215	Partially Resolved

Description

Users can join multiple expeditions while only stake onetime. This attack is only valid if expeditions share required planets and optional assets NFT. Attack Flow:

1. call the function stakeNFTBeforeExpeditionEnd() to stake NFT.
2. call the function joinExpedition() to join an expedition.
3. call the function joinExpedition() to join another expedition.
4. call the function joinExpedition() to join another expedition. .....
5. choose one that is most profitable to claim rewards.

Recommendation

We recommend using a data structure to track staked assets for each expedition for each user.

Alleviation

[Apeiron] - According to our operation plan, we will create single expedition at the same period, so user never to join the multiple expedition with same period.

[CertiK] - The operation plan does indeed prevent this attack from being implemented; however, it does not fully resolve the issue since the vulnerability still lies within the codebase.

NEA-02 USERS CAN JOIN AN EXPEDITION BY STAKING ZERO ASSET

Category	Severity	Location	Status
Logical Issue	Critical	contracts/NebulaExpedition.sol (base): 338	Partially Resolved

Description

Users can join an expedition by staking zero asset. This attack is only valid if expeditions share required planets and optional assets NFT. Attack flow:

1. call the function stakeNFTBeforeExpeditionEnd() to stake NFT.
2. call the function joinExpedition() to join an expedition.
3. call the function unstakeNFTBeforeExpeditionEnd() to unstake NFT by passing another `_expeditionId`.

Recommendation

We recommend using a data structure to track staked assets for each expedition for each user.

Alleviation

[Apeiron] - Since only one expedition active at a time, user will not be able to unstake NFT when entering another expeditionId

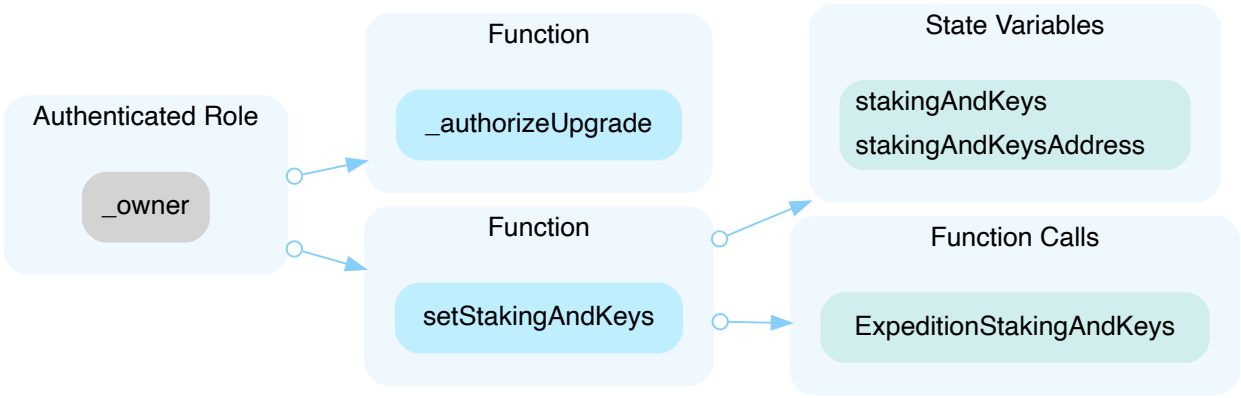
[CertiK] - This plan prevents this issue from being exploited.

NEA-03 | CENTRALIZATION RISKS IN NEBULAEXPEDITION.SOL

Category	Severity	Location	Status
Centralization / Privilege	Major	contracts/NebulaExpedition.sol (base): 150, 158, 168~171, 191	Acknowledged

Description

In the contract `NebulaExpedition` the role `_owner` has authority over the functions `_authorizeUpgrade` and `setStakingAndKeys` as shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and authorize the upgrade and set the staking and keys to link up with `ExpeditionStakingAndKeys` contract.



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

[Apeiron] - Will transfer ownership to a multi-sign wallet after deployment like the expedition one.

[CertiK] - The Apeiron team acknowledged the issue and will transfer ownership once the contracts have been deployed.

NEA-04 | USERS CAN JOIN AN EXPEDITION RIGHT BEFORE IT ENDS

Category	Severity	Location	Status
Logical Issue	Major	contracts/NebulaExpedition.sol (base): 205~206	Resolved

Description

Users can join an expedition right before it ends, which means users can join an expedition by staking assets only for a couple of minutes.

Attack Flow:

1. call the function stakeNFTBeforeExpeditionEnd() and joinExpedition() right before the expedition ends.
2. call the function unstakeNFTAfterExpeditionEnd() to unstake NFT.

Recommendation

We recommend setting a deadline for joining each expedition.

Alleviation

[Apeiron] I can confirm This is the correct behavior.

[CertiK] The `Apeiron` team comfirmed that this is a correct behavior.

NEA-05 | LACK OF CHECK FOR START TIME

Category	Severity	Location	Status
Logical Issue	Minor	contracts/NebulaExpedition.sol (base): 176	Resolved

Description

There is no check that the beginning or end of an expedition is set in the past.

Recommendation

We recommend including a check to ensure that start and end time are not set in the past.

Alleviation

[Apeiron] - We added the timing restriction from previous expedition.

[CertiK] - This change resolves the issue and can be seen here 1e920f838622eadfeebd5323c87b190c45e655f0

ATC-06 | USE ENUM DATA STRUCTURE

Category	Severity	Location	Status
Language Specific	Informational	contracts/ExpeditionPrizes.sol (base): 34~39; contracts/ExpeditionStakingAndKeys.sol (base): 29~32	Resolved

Description

Enums allow you to define a type by enumerating its possible values.

Recommendation

We recommend using enums to group these variables.

Alleviation

[CertiK] - The `Apeiron` team resolved this issue by defining the an enum type named `ASSET_TYPE` in `StakeAssetMeta.sol;` . The contract `ExpeditionStakingAndKeys` imports this contract hence it inherits the enum type `ASSET_TYPE` and can make use of it. The changes can be seen in the following commit f4e77fab40da589150c2338b6bd16c385c502fb9

ATC-07 | MISSING ERROR MESSAGES

Category	Severity	Location	Status
Coding Style	Informational	contracts/ExpeditionPrizes.sol (base): 231; contracts/ExpeditionStakingAndKeys.sol (base): 300~302, 387, 464~469, 479, 507	Resolved

Description

The **require** can be used to check for conditions and throw an exception if the condition is not met. It is better to provide a string message containing details about the error that will be passed back to the caller.

Recommendation

We advise adding error messages to the linked **require** statements.

Alleviation

[CertiK] - The `Apeiron` team resolved the issue by including error messages to the linked require statements. The changes can be seen in the following commit 19543f7b86bf46bc022a4cdc5a15dded77df37fd

EPA-03 | TYPO IN COMMENTS

Category	Severity	Location	Status
Coding Style	Informational	contracts/ExpeditionPrizes.sol (base): 143	Resolved

Description

`NebulaExpedition` is written as `NubulaExpedition`.

Recommendation

Consider rewriting `NubulaExpedition` to `NebulaExpedition` in the comment for consistency and better readability.

Alleviation

[CertiK] - The `Apeiron` fixed the typo in the following commit 5823cc86aabd130800275e5c7e3aad817eb832c3

ESA-06 | UNNECESSARY CHECKS

Category	Severity	Location	Status
Logical Issue	Informational	contracts/ExpeditionStakingAndKeys.sol (base): 250~252, 261, 268, 278, 285~286, 315	Resolved

Description

The check

```
require(erc20.allowance(_stakeholder, address(this)) >=_amounts[i] &&
erc20.balanceOf(_stakeholder) >= _amounts[I], "Not enough allowance or balance for
tokens");
```

is redundant because it happens in ERC20.sol.

```
    function _transfer(
        address from,
        address to,
        uint256 amount
    ) internal virtual {
        require(from != address(0), "ERC20: transfer from the zero address");
        require(to != address(0), "ERC20: transfer to the zero address");

        _beforeTokenTransfer(from, to, amount);

        uint256 fromBalance = _balances[from];
        require(fromBalance >= amount, "ERC20: transfer amount exceeds balance"); //
(HERE!!!)
        unchecked {
            _balances[from] = fromBalance - amount;
        }
        _balances[to] += amount;

        emit Transfer(from, to, amount);

        _afterTokenTransfer(from, to, amount);
    }
```

The checks

```
require(erc721.isApprovedForAll(_stakeholder, address(this)),"Not approved");
require(erc721.ownerOf(_tokenIds[i]) == _stakeholder);
```

are redundant because they happen in ERC721.sol.

```
    function safeTransferFrom(
        address from,
        address to,
        uint256 tokenId,
        bytes memory data
    ) public virtual override {
        require(_isApprovedOrOwner(_msgSender(), tokenId), "ERC721: caller is not
token owner nor approved"); // (HERE!!!)
        _safeTransfer(from, to, tokenId, data);
    }

    function _isApprovedOrOwner(address spender, uint256 tokenId) internal view
virtual returns (bool) {
        address owner = ERC721.ownerOf(tokenId);
        return (spender == owner || isApprovedForAll(owner, spender) ||
getApproved(tokenId) == spender); // (HERE!!!)
    }

    function _transfer(
        address from,
        address to,
        uint256 tokenId
    ) internal virtual {
        require(ERC721.ownerOf(tokenId) == from, "ERC721: transfer from incorrect
owner");  // (HERE!!!)
        require(to != address(0), "ERC721: transfer to the zero address");

        _beforeTokenTransfer(from, to, tokenId);

        // Clear approvals from the previous owner
        _approve(address(0), tokenId);

        _balances[from] -= 1;
        _balances[to] += 1;
        _owners[tokenId] = to;

        emit Transfer(from, to, tokenId);

        _afterTokenTransfer(from, to, tokenId);
    }
```

The checks

```
require(erc1155.isApprovedForAll(_stakeholder, address(this)), "Not approved");
require(erc1155.balanceOf(_stakeholder, _tokenIds[i]) >= _amounts[i], "Not enough
balance");
```

are redundant because they happen in ERC1155.sol.

```
    function safeBatchTransferFrom(
        address from,
        address to,
        uint256[] memory ids,
        uint256[] memory amounts,
        bytes memory data
    ) public virtual override {
        require(
            from == _msgSender() || isApprovedForAll(from, _msgSender()), //
(HERE!!!)
            "ERC1155: caller is not token owner nor approved"
        );
        _safeBatchTransferFrom(from, to, ids, amounts, data);
    }

  function _safeBatchTransferFrom(
        address from,
        address to,
        uint256[] memory ids,
        uint256[] memory amounts,
        bytes memory data
    ) internal virtual {
        require(ids.length == amounts.length, "ERC1155: ids and amounts length
mismatch");
        require(to != address(0), "ERC1155: transfer to the zero address");

        address operator = _msgSender();

        _beforeTokenTransfer(operator, from, to, ids, amounts, data);

        for (uint256 i = 0; i < ids.length; ++i) {
            uint256 id = ids[i];
            uint256 amount = amounts[i];

            uint256 fromBalance = _balances[id][from];
            require(fromBalance >= amount, "ERC1155: insufficient balance for
transfer"); // (HERE!!!)
            unchecked {
                _balances[id][from] = fromBalance - amount;
            }
            _balances[id][to] += amount;
        }

        emit TransferBatch(operator, from, to, ids, amounts);

        _afterTokenTransfer(operator, from, to, ids, amounts, data);
```

```
        _doSafeBatchTransferAcceptanceCheck(operator, from, to, ids, amounts, data);
    }
```

Recommendation

We recommend deleting these checks to save gas.

Alleviation

[CertiK] - The Apeiron team resolved this issue by removing the duplicated checks. The changes can be seen in the following commit f828d0373c03ab0e142f0b9ba6443d05a0003418

OPTIMIZATIONS | APEIRON - TOKEN CONTRACT

ID	Title	Category	Severity	Status
AGC-02	User-Defined Getters	Gas Optimization	Optimization	Resolved
ANI-01	Redundant Import Statement	Gas Optimization	Optimization	Resolved
ATC-03	Improper Usage Of `public` And `external` Type	Gas Optimization	Optimization	Resolved
ATC-04	Function Should Be Declared External	Gas Optimization	Optimization	Resolved
ATC-05	Unused State Variable	Gas Optimization	Optimization	Acknowledged
ESA-05	Combining For Loops	Coding Style	Optimization	Resolved

AGC-02 | USER-DEFINED GETTERS

Category	Severity	Location	Status
Gas Optimization	Optimization	contracts/ApeironGodiverseCollection.sol (base): 39~41	Resolved

Description

The linked functions are equivalent to the compiler-generated getter functions for the respective variables.

Recommendation

We advise that the linked variables are instead declared as `public` as compiler-generated getter functions are less prone to error and much more maintainable than manually written ones.

Alleviation

[CertiK] - The Apeiron team resolved this issue by removing the function that is equivalent to the compiler-generated function. The changes can be seen in the following commit 9669ba77f21f6aaac1d4920d819a6542f087bfad

ANI-01 | REDUNDANT IMPORT STATEMENT

Category	Severity	Location	Status
Gas Optimization	Optimization	contracts/ANIMA.sol (base): 10~13	Resolved

Description

The last 4 files, ERC20, ERC20Burnable, Pausable, and ownable, have been imported a few lines above.

Recommendation

We recommend removing the redundant import statements.

Alleviation

[CertiK] - The Apeiron team resolved the issue by following the recommendation above. The changes can be seen at this commit 9669ba77f21f6aaac1d4920d819a6542f087bfad

ATC-03 | IMPROPER USAGE OF `public` AND `external` TYPE

Category	Severity	Location	Status
Gas Optimization	Optimization	contracts/ANIMA.sol (base): 28; contracts/ANIMAchild.sol (base): 30; contracts/APRSchild.sol (base): 30; contracts/ExpeditionPrizes.sol (base): 312, 323; contracts/mock/MockERC20.sol (base): 10; contracts/mock/MockERC721.sol (base): 16; contracts/mock/MockExpeditionStakingAndKeysV2.sol (base): 7; contracts/utils/AccessProtected.sol (base): 29; contracts/utils/AccessProtectedUpgradable.sol (base): 29	Resolved

Description

`public` functions that are never called by the contract could be declared as `external`. `external` functions are more efficient than `public` functions.

Recommendation

Consider using the external attribute for public functions that are never called within the contract.

Alleviation

[CertiK] - The Apeiron team resolved this issue by changing the visibilities to `external`.

ATC-04 | FUNCTION SHOULD BE DECLARED EXTERNAL

Category	Severity	Location	Status
Gas Optimization	Optimization	contracts/ANIMA.sol (base): 20, 24, 28; contracts/ANIMAchild.sol (base): 30, 37, 41; contracts/ANIMAroot.sol (base): 36, 40; contracts/APRSVesting.sol (base): 32; contracts/APRSchild.sol (base): 30, 37, 41; contracts/APRSroot.sol (base): 36, 40; contracts/ApeironGodiverseCollection.sol (base): 18; contracts/ExpeditionPrizes.sol (base): 90, 312, 323; contracts/ExpeditionStakingAndKeys.sol (base): 96; contracts/NebulaExpedition.sol (base): 143; contracts/mock/MockERC1155.sol (base): 12, 16, 25; contracts/mock/MockERC20.sol (base): 10; contracts/mock/MockERC721.sol (base): 16; contracts/mock/MockExpeditionPrizesV2.sol (base): 7; contracts/mock/MockExpeditionStakingAndKeysV2.sol (base): 7; contracts/mock/MockNebulaExpeditionV2.sol (base): 7; contracts/utils/AccessProtected.sol (base): 29; contracts/utils/AccessProtectedUpgradable.sol (base): 29	Resolved

Description

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

```
20      function pause() public onlyOwner {
```

```
24      function unpause() public onlyOwner {
```

```
28      function mint(address to, uint256 amount) public onlyAdmin {
```

```
30      function mint(address user, uint256 amount)
```

```
37      function pause() public onlyOwner {
```

```
41      function unpause() public onlyOwner {
```

```
36      function pause() public onlyOwner {
```

```
40      function unpause() public onlyOwner {
```

```
32      function initialize() public initializer {
```

```
30      function mint(address user, uint256 amount)
```

```
37      function pause() public onlyOwner {
```

```
41      function unpause() public onlyOwner {
```

```
36      function pause() public onlyOwner {
```

```
40      function unpause() public onlyOwner {
```

```
18      function setURI(string memory newuri) public onlyOwner {
```

```
 90      function initialize() public initializer {
```

```
312      function getTreasureById(uint256 _treasureId)
```

```
323      function getUserTreasureIds(address _user)
```

```
 96      function initialize() public initializer {
```

```
143      function initialize() public initializer {
```

```
12      function setURI(string memory newuri) public onlyOwner {
```

```
16    function mint(
```

```
25    function mintBatch(
```

```
10    function mint(address to, uint256 amount) public onlyAdmin {
```

```
16    function safeMint(address to) public onlyAdmin {
```

```
 7    function testToWritePrizePerTreasureId(Treasure memory _treasure) public {
```

```
 7    function testToWriteAssetTypes(uint256 assetType) public {
```

```
 7    function testToWriteUserExpeditionIds(uint256[] memory ids) public {
```

```
29    function isAdmin(address admin) public view returns (bool) {
```

```
29    function isAdmin(address admin) public view returns (bool) {
```

Recommendation

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

Alleviation

[CertiK] - The Apeiron team resolved this issue by changing the visibilities to `external`.

ATC-05 | UNUSED STATE VARIABLE

Category	Severity	Location	Status
Gas Optimization	Optimization	contracts/ExpeditionMeta.sol (base): 12, 13, 15; contracts/ExpeditionPrizes.sol (base): 34	Acknowledged

Description

One or more state variables are never used in the codebase.

Variable `STATE_NOT_STARTED` in `ExpeditionMeta` is never used in `MockExpeditionPrizesV2`.

```
12      uint256 internal constant STATE_NOT_STARTED = 0;
```

```
6  contract MockExpeditionPrizesV2 is ExpeditionPrizes {
```

Variable `STATE_STARTED` in `ExpeditionMeta` is never used in `MockExpeditionPrizesV2`.

```
13      uint256 internal constant STATE_STARTED = 1;
```

```
6  contract MockExpeditionPrizesV2 is ExpeditionPrizes {
```

Variable `STATE_CLAIMABLE` in `ExpeditionMeta` is never used in `MockExpeditionPrizesV2`.

```
15      uint256 internal constant STATE_CLAIMABLE = 3;
```

```
6  contract MockExpeditionPrizesV2 is ExpeditionPrizes {
```

Variable `PRIZE_TYPE_TRANSFER_NONE` in `ExpeditionPrizes` is never used in `MockExpeditionPrizesV2`.

```
34      uint256 internal constant PRIZE_TYPE_TRANSFER_NONE = 0;
```

```
6  contract MockExpeditionPrizesV2 is ExpeditionPrizes {
```

Recommendation

We advise removing the unused variables.

Alleviation

[Aperion] - We won't make any changes, since these unused state variable are appeared on our Mock***** Contracts only, the mock contracts are used for our test case only.

[CertiK] - Since the unused variables are part of the mock contracts for testing purposes, this finding does not pose any risk to the project.

ESA-05 | COMBINING FOR LOOPS

Category	Severity	Location	Status
Coding Style	Optimization	contracts/ExpeditionStakingAndKeys.sol (base): 352, 368	Resolved

Description

The following two for loops

```
        for (uint i = 0; i < _tokenIds.length; i++) {
            bool found = false;
            for (uint j = 0; j < stakedAssets[_stakeholder].length; j++) {
                if (
                    stakedAssets[_stakeholder][j].addr == _assetAddress &&
                    stakedAssets[_stakeholder][j].tokenId == _tokenIds[i] &&
                    stakedAssets[_stakeholder][j].amount >= _amounts[i]
                ) {
                    found = true;
                }
            }

            require(found, "Not staked");
        }

        //deduce the asset from the stakedAssets
        for (uint i = 0; i < _tokenIds.length; i++) {
            for (uint j = 0; j < stakedAssets[_stakeholder].length; j++) {
                if (
                    stakedAssets[_stakeholder][j].addr == _assetAddress &&
                    stakedAssets[_stakeholder][j].tokenId == _tokenIds[i]
                ) {
                    stakedAssets[_stakeholder][j].amount -= _amounts[i];
                    break;
                }
            }
        }
```

can be combined into the follwong one

```
uint counter = 0;
for (uint i = 0; i < _tokenIds.length; i++) {
    for (uint j = 0; j < stakedAssets[_stakeholder].length; j++) {
        if (
            stakedAssets[_stakeholder][j].addr == _assetAddress &&
            stakedAssets[_stakeholder][j].tokenId == _tokenIds[i]
        ) {
          counter += 1;
          stakedAssets[_stakeholder][j].amount -= _amounts[i];
          break;
        }
    }
}
require(counter == _tokenIds.length, "Not staked");
```

Recommendation

We recommend combining these two for loops to save gas.

Alleviation

[CertiK] - The Apeiron team resolved the finding by combining the two for loops. The changes can be seen in the following commit ce6fc6227e0225dfeea794159b8c8de5f642c971

APPENDIX | APEIRON - TOKEN CONTRACT

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.