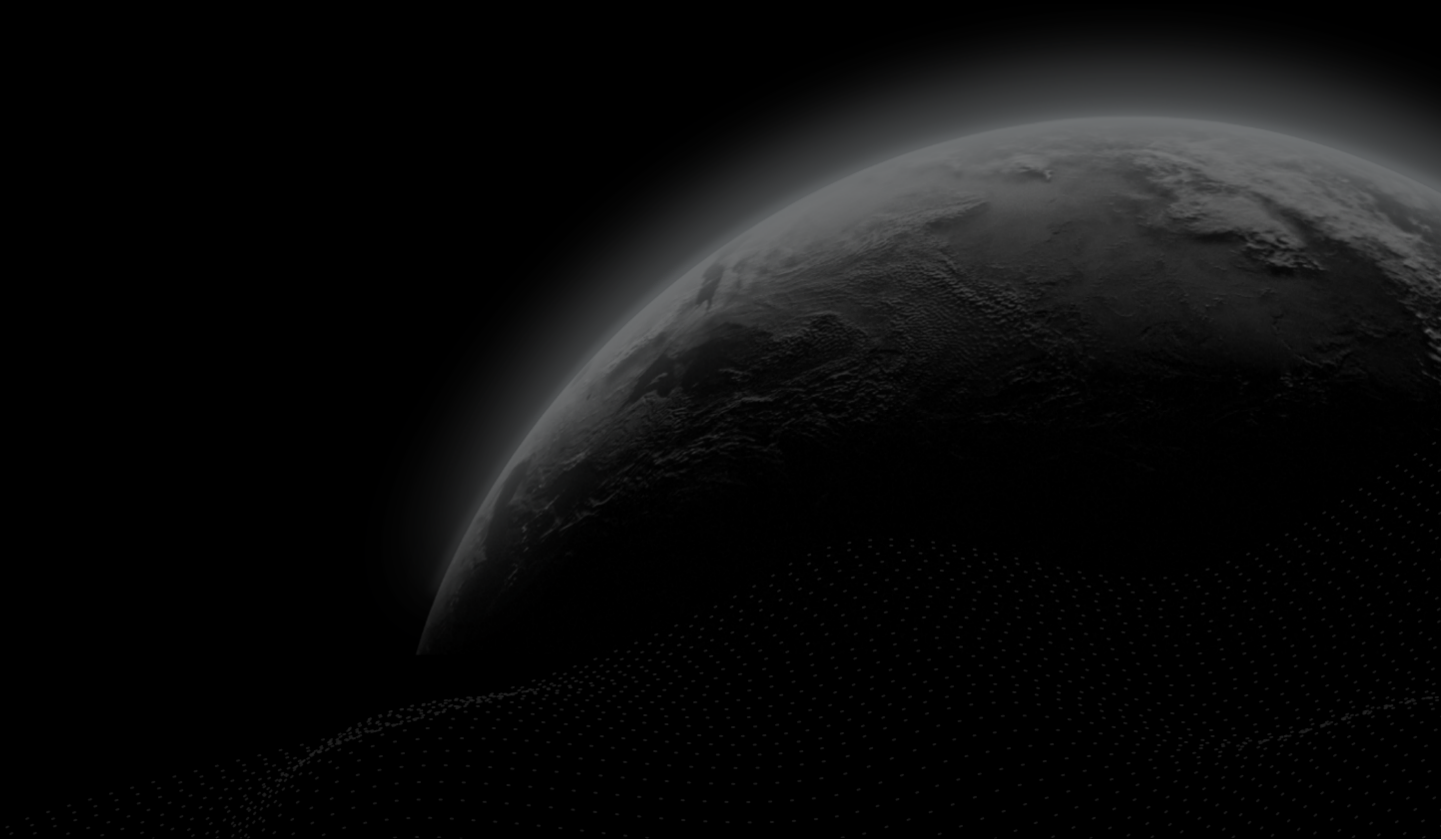
CERTIK

Security Assessment

CyberConnect - Audit

CertiK Assessed on Aug 24th, 2023



CertiK Assessed on Aug 24th, 2023

CyberConnect - Audit

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

Executive Summary

TYPES	ECOSYSTEM	METHODS
Platform	Ethereum (ETH)	Manual Review, Static Analysis
LANGUAGE	**TIMELINE**	**KEY COMPONENTS**
Solidity	Delivered on 08/24/2023	N/A

CODEBASE

cybergraph

cyberid

View All in Codebase Page

COMMITS

cybergraph

- acc4d08f51c684690afd92a92bcf8aa669411309
- a62adbf51918973af2255f2d9a2e81f3ef6234bb

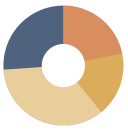
View All in Codebase Page

Highlighted Centralization Risks

! Contract upgradeability

! Privileged role can mint tokens

Vulnerability Summary



23	7	0	2	14	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined

🟥	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.
🟧	5	Major	5 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.
🟨	4	Medium	4 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
🟨	8	Minor	1 Resolved, 7 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.
🟦	6	Informational	2 Resolved, 2 Partially Resolved, 2 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 | CYBERCONNECT - AUDIT

CODEBASE | CYBERCONNECT - AUDIT

Repository

cybergraph

cyberid

Commit

cybergraph

- acc4d08f51c684690afd92a92bcf8aa669411309
- a62adbf51918973af2255f2d9a2e81f3ef6234bb

cyberid

- 22bb9567f4240c3f31b639f176122b9c44e6f966
- 26ba7731e79531edf4b1414c730b292672c4b6a9
- 48f18ec9af2d3f9e5c67d0f4bcd9a6e131823dfc
- cf6f7712d49220b9fa931e8f81228ddfa22c9260

AUDIT SCOPE | CYBERCONNECT - AUDIT

40 files audited ● 16 files with Acknowledged findings ● 5 files with Resolved findings ● 19 files without findings

ID	File	SHA256 Checksum
● CON	src/core/Content.sol	7ab4099d54a909543859c6e18fb70bb0d54f58c2851fc9bfd9f9cd49d29d7990
● CEB	src/core/CyberEngine.sol	140ef8e3c7be701e5d9f49adba9e632fbd899c89cf2da5985370e8b97d9548ba
● ESS	src/core/Essence.sol	0604ce1737fb7818afc3dfe34c530285449b0e1f612af66084e09464ec136635
● MMB	src/core/MiddlewareManager.sol	95c7ff7d781b2e06021f71bed6832e5f6adc04231858c6f48c119858edcd700c
● SOU	src/core/Soul.sol	c5b1d88dc48ce0eeb7bc49ee3cfd5884281467cbf4ff42249251fd2a3d7b8185
● SUB	src/core/Subscribe.sol	0661fd9c7564a5b948c3c19a70488e41fb90fc136e1b942c3c7e55be4b8a4853
● W3S	src/core/W3st.sol	a60226bad4b33988d8903b235e738e84b502ef223a439e3798b2b43f15e7e1a3
● OWN	src/dependencies/solmate/Owned.sol	3473214199566358695223b967149a6fa6cc12093957744b51db7d8502a1b5e6
● CAF	src/factory/CyberAccountFactory.sol	0d5a3cae9fa0ab3082f829eb9d694216feb30dea5a4a30880fe216b4ed0adef6
● TRE	src/middlewares/base/Treasury.sol	9409472a36fbdcdfd7d5c6af05644ca5117f8543c110c06603a9e03b504129db
● LTP	src/middlewares/LimitedTimePaidMw.sol	a2cce920619f73f9d073a70de389a1fece94b1a2786d02e0b7932c5063aadcd2
● TRB	src/periphery/TokenReceiver.sol	2e077caaca673c8b2cd83a7da07d741afbe7e1bd3e522a65f46d61c95385850f
● CIB	src/core/CyberId.sol	8f773743c98688ac432b3aa06f4b28b09f57eddd761e3bb9d64dd0fe3bc814de
● MIB	src/core/MocaId.sol	765103e6ac3c4c25f67aef94ea72903f397553a0537b6df64a20214abefe3739

ID	File	SHA256 Checksum
● SFM	src/middlewares/cyberid/StableFeeMiddleware.sol	ee866c2508299003f30285bf557e26db4aaf1590b24587468f5f062da59ac03d
● TOM	src/middlewares/cyberid/TrustOnlyMiddleware.sol	81627a703e23ab922bddd15b8a662837d3a5f05d28bb7335d116f8fd619d9d85
● EIP	src/base/EIP712.sol	fa6845bbea74e09a6d89095ed98dd299ae008bfeb2928fd1f953e2a49e2e6f01
● PMB	src/middlewares/PermissionMw.sol	e275a75f6a11e0babde4fa629c393bac7fac6daa350f03b70822fa7800e562c1
● EI7	src/base/EIP712.sol	4eaff9679f9c83c78305082fd4cd22b4689ed24058644fa546a6cfe0b7d209ef
● MRU	src/base/MetadataResolver.sol	a6d3548b481f4a28e02d701249f4c6422049fca4db97f824ce99cb103b724f7d
● PMU	src/middlewares/mocaid/PermissionMw.sol	6cfd4530807b6e0a5500815677788e6fa586d85171877d4cfe862a85ef731423
● CNF	src/base/CyberNFT1155.sol	1aa52de3b863dd56e182cd9714333f2d325c0901b01666d58bcfe4a219bb2b4d
● CNT	src/base/CyberNFT721.sol	f417699fed7747ddc5eccc4b8a7ce1f3c7e79839b85557f55b36e82cf8b43ad8
● MRB	src/base/MetadataResolver.sol	11db815f1811d25e5635141cd29c7ad228849664a509e9b03017d15a72466c83
● ERC	src/dependencies/solmate/ERC1155.sol	3fdf863ee9d3bcc5ab1b82753e18d857377d252f8aee67390a326600da911ed1
● ER7	src/dependencies/solmate/ERC721.sol	3a80de35e2a98cff044b45ff097362830bb2b75b73ee974534d63f735bc79036
● SBT	src/dependencies/solmate/SBTERC721.sol	715242090c188359cdba39f206feede1dbbc5e785e7aaa82046e2f2fe26d7d11
● CDB	src/deployer/Create2Deployer.sol	ddd33bd75e7d2cea63209d9ede651cbd092740ed6178df578f0924cebf1e5d44
● DEO	src/deployer/Deployer.sol	6db4e70faac9e3bd474a9d3dda3f64f13d9a6ec4ef80c1c99e94d947f318770d
● SDB	src/deployer/SubscribeDeployer.sol	573626c8f5476496445bec702823945d82684d7803187fd2b282c56cd7a04c1e

ID	File	SHA256 Checksum
COS	src/libraries/Constants.sol	1550b9be5866b069403c6029420875a2cbf37750e2eb3feb7b47098bab5f1f21
DTB	src/libraries/DataTypes.sol	530b30003c5f68e84e7397dbf06d53979b781492beafc95c5835b93aeb1aa806
LSB	src/libraries/LibString.sol	c5ee63ee0878b1475b1a38ca263ad4fcb9f78919cdfc478b5659421e0d49cd37
FMB	src/middlewares/base/FeeMw.sol	689556be8836a257abd235eba85edcb25c00e8eccc307e961f58beb3a2487227
OEM	src/middlewares/base/OnlyEngineMw.sol	df8cddbf8a18f92fa298398e942cc891c6de036858dd43125efd55baeaab87bb
CDU	src/deployer/Create2Deployer.sol	ddd33bd75e7d2cea63209d9ede651cbd092740ed6178df578f0924cebf1e5d44
COT	src/libraries/Constants.sol	1223f8280f1b842d1be74f87e719084415e0e0ca5cc459d34cc59d2b44436aea
DTU	src/libraries/DataTypes.sol	9da08cb66b40e5e95a27891ec5e4b680e3ffaab8f7f171310927e1455f09f8ec
LSU	src/libraries/LibString.sol	f6ecb53381a19b3424f5f3afc837531d7b5f453ec3aa73a107a15b14d4e3af1c
LCC	src/middlewares/cyberid/base/LowerCaseCyberIdMiddleware.sol	7478ee8b591390a3917d9960cc73d3a5ba7971907b666c6771e8586ca2dfc734

APPROACH & METHODS | CYBERCONNECT - AUDIT

This report has been prepared for CyberConnect to discover issues and vulnerabilities in the source code of the CyberConnect - 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.

REVIEW NOTES | CYBERCONNECT - AUDIT

Overview

CyberConnect is a web3 social network that enables developers to create social applications empowering users to own their digital identity, content, connections, and interactions.

The focus of the audit is:

- cybergraph
- cyberid

External Dependencies

The project is heavily dependent on the following third-party libs:

- solmate, https://github.com/transmissions11/solmate
- kernel, https://github.com/zerodevapp/kernel
- openzeppelin, https://github.com/openzeppelin/openzeppelin-contracts
- openzeppelin-upgradeable, https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable
- chainlink, https://github.com/smartcontractkit/chainlink

The scope of the audit treats 3rd party entities as black boxes and assume their functional correctness. However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of 3rd parties can possibly create severe impacts, such as increasing fees of 3rd parties, migrating to new LP pools, etc. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

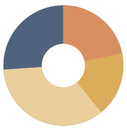
Privileged Functions

In the `CyberConnect` project, multiple roles are adopted to ensure the dynamic runtime updates of the project, which were specified in the findings **CYB-04**, **GLOBAL-01**, **GLOBAL-02**, **MIB-01** and **TOM-01**.

The advantage of this privileged role in the codebase is that the client reserves the ability to adjust the protocol according to the runtime required to best serve the community. It is also worth of note the potential drawbacks of these functions, which should be clearly stated through the client's action/plan. Additionally, if the private key of the privileged account is compromised, it could lead to devastating consequences for the project.

To improve the trustworthiness of the project, dynamic runtime updates in the project should be notified to the community. Any plan to invoke the aforementioned functions should be also considered to move to the execution queue of the `Timelock` contract.

FINDINGS | CYBERCONNECT - AUDIT



23	0	5	4	8	6
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for CyberConnect - Audit. Through this audit, we have uncovered 23 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
CYB-04	**Centralized Control Of Contract Upgrade**	**Centralization**	**Major**	● **Acknowledged**
CYB-08	**Pausing Centralization Risks**	**Centralization**	**Major**	● **Acknowledged**
GLOBAL-01	**Centralization Related Risks In CyberGraph**	**Centralization**	**Major**	● **Acknowledged**
GLOBAL-02	**Centralization Related Risks In CyberId**	**Centralization**	**Major**	● **Acknowledged**
TOM-01	**Minting Centralization Risk**	**Centralization**	**Major**	● **Acknowledged**
CYB-01	Incomplete Signature Validation	Volatile Code	Medium	● Resolved
SFM-01	Missing Validation On The Return Values Of `usdOracle.getRoundData()` And `latestRoundData()`	Volatile Code	Medium	● Resolved
SFM-02	Potential Manipulation On Registration Cost	Logical Issue	Medium	● Resolved
SR2-01	Moca Xp Set Up Validation Can Be Bypassed	Logical Issue	Medium	● Resolved
CAF-01	Missing Validation On `proxy` Address And Predicted Address	Volatile Code	Minor	● Acknowledged

ID	Title	Category	Severity	Status
CIB-01	Potential Stuck Tokens When `middleware` Is `address(0)`	Logical Issue	Minor	● Resolved
CYB-02	Missing Zero Address Validation	Volatile Code	Minor	● Acknowledged
CYB-05	Lack Of Storage Gap In Upgradeable Contract	Logical Issue	Minor	● Acknowledged
CYB-06	Lack Of Input Validation	Volatile Code	Minor	● Acknowledged
LTP-01	Insufficient Validation On `startTimestamp` And `endTimestamp`	Volatile Code	Minor	● Acknowledged
TRB-01	Missing Zero Address Validation	Volatile Code	Minor	● Acknowledged
TRB-02	Usage Of `transfer` / `send` For Sending Native Tokens	Language Version	Minor	● Acknowledged
COR-03	Potential Failure On `safeBatchTransferFrom`	Logical Issue	Informational	● Acknowledged
CYB-07	Missing Emit Events	Coding Style	Informational	● Partially Resolved
GLOBAL-03	Check-Effect-Interaction Pattern Violation	Logical Issue	Informational	● Partially Resolved
OWN-01	Unused Contract	Coding Style	Informational	● Acknowledged
SFM-03	Missing Access Restriction	Logical Issue	Informational	● Resolved
TOM-02	Multiple Ways For Owner Update In `TrustOnlyMiddleware`	Access Control	Informational	● Resolved

CYB-04 | CENTRALIZED CONTROL OF CONTRACT UPGRADE

Category	Severity	Location	Status
Centralization	● **Major**	**src/core/CyberEngine.sol (07/19-acc4d08): 30; src/core/MocaId.sol (07/19-22bb956): 18**	● **Acknowledged**

Description

In the contract `MocaId`, the role `DEFAULT_ADMIN_ROLE` has the authority to update the implementation contract behind the `MocaId` contract.

Any compromise to the `DEFAULT_ADMIN_ROLE` account may allow a hacker to take advantage of this authority and change the implementation contract which is pointed by proxy and therefore execute potential malicious functionality in the implementation contract.

Note: Update in Commit 26ba7731e79531edf4b1414c730b292672c4b6a9 The contract uses the `owner` role to control the pause functionality.

In the contract `CyberEngine`, the role `admin` has the authority to update the implementation contract behind the `CyberEngine` contract.

Any compromise to the `admin` account may allow a hacker to take advantage of this authority and change the implementation contract which is pointed by proxy and therefore execute potential malicious functionality in the implementation contract.

Recommendation

We recommend that the team make efforts to restrict access to the admin of the proxy contract. A strategy of combining a time-lock and a multi-signature (⅔, ⅗) wallet can be used to prevent a single point of failure due to a private key compromise. In addition, the team should be transparent and notify the community in advance whenever they plan to migrate to a new implementation contract.

Here are some feasible short-term and long-term suggestions that would mitigate the potential risk to a different level and suggestions that would permanently fully resolve the risk.

Short Term:

A combination of a time-lock and a multi signature (⅔, ⅗) wallet mitigate the risk by delaying the sensitive operation and avoiding a single point of key management failure.

- A time-lock with reasonable latency, such as 48 hours, for awareness of privileged operations;
  AND

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to a private key compromised;
  AND
- A medium/blog link for sharing the time-lock contract and multi-signers addresses information with the community.

For remediation and mitigated status, please provide the following information:

- Provide the deployed time-lock address.
- Provide the **gnosis** address with **ALL** the multi-signer addresses for the verification process.
- Provide a link to the **medium/blog** with all of the above information included.

Long Term:

A combination of a time-lock on the contract upgrade operation and a DAO for controlling the upgrade operation mitigate the contract upgrade risk by applying transparency and decentralization.

- A time-lock with reasonable latency, such as 48 hours, for community awareness of privileged operations;
  AND
- Introduction of a DAO, governance, or voting module to increase decentralization, transparency, and user involvement;
  AND
- A medium/blog link for sharing the time-lock contract, multi-signers addresses, and DAO information with the community.

For remediation and mitigated status, please provide the following information:

- Provide the deployed time-lock address.
- Provide the **gnosis** address with **ALL** the multi-signer addresses for the verification process.
- Provide a link to the **medium/blog** with all of the above information included.

Permanent:

Renouncing ownership of the `admin` account or removing the upgrade functionality can *fully* resolve the risk.

- Renounce the ownership and never claim back the privileged role;
  OR
- Remove the risky functionality.

Note: we recommend the project team consider the long-term solution or the permanent solution. The project team shall *make a decision based on the current state of their project, timeline, and project resources.*

Alleviation

[CyberConnect Team 08/17/2023]: The team acknowledged the finding and decided to remain unchanged.

[CertiK 08/17/2023]: CertiK strongly encourages the project team to periodically revisit the private key security management of all addresses related to privileged roles.

CYB-08 | PAUSING CENTRALIZATION RISKS

Category	Severity	Location	Status
Centralization	● **Major**	**src/core/MiddlewareManager.sol (07/19-acc4d08): 1; src/core/Mocald.sol (07/19-22bb956): 356~358, 363~365**	● **Acknowledged**

Description

In the contract `Mocald.sol` , the role `admin` has the authority to update the status of the `_paused` and further pause/resume the functionality of the `transferFrom()` , `safeTransferFrom()` and `safeTransferFrom()` functions, which effectively impact token transfers.

Note: Update in Commit 26ba7731e79531edf4b1414c730b292672c4b6a9 The contract uses the `owner` role to control the pause functionality.

Similarily, the owner of `CyberEngine` also can pause the Content, Essence, and W3st related functionalities in Cybergraph.

Any compromise to the private key of the `owner` may allow hackers to take advantage of this authority and allow/prevent external user access to token transfer functionalities.

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., multi-signature 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.

Note: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

[CyberConnect Team 08/17/2023]: The team acknowledged the finding and decided to remain unchanged.

[CertiK 08/17/2023]: CertiK strongly encourages the project team to periodically revisit the private key security management of all addresses related to privileged roles.

GLOBAL-01 | CENTRALIZATION RELATED RISKS IN CYBERGRAPH

Category	Severity	Location	Status
Centralization	● Major		● Acknowledged

Description

In the contract `Soul` , the role `owner` has authority over the functions listed below.

- `setMinter` : set/update minter status for an address.
- `setTokenURI` : update token uri.
- `transferOwnership` : set a new owner for the contract.
- `renounceOwnership` : set address(0) as the new owner.

Any compromise to the `owner` account may allow a hacker to take advantage of this authority, update minter status and modify uri.

the role `minter` has authority over the functions listed below.

- `createSoul` : mint soul token.
- `setOrg` : update org status for address.
- `clearGatedMetadatas` : clear gated metadata on a token.
- `batchSetGatedMetadatas` : batch set gated metadata.

Any compromise to the `minter` account may allow a hacker to take advantage of this authority issue and modify soul token.

In the contract `CyberEngine` , the role soul token owner has authority over the functions listed below.

- `setOperatorApproval` : set operator for the self(soul token owner).

the role soul token owner and corresponding operator have authority over the functions listed below.

- `registerEssence` : register essence and create essence contract for soul token owner.
- `registerSubscription` : register subscribe and create subscribe contract for soul token owner (can only be called once).
- `publishContent` : publish content token set for soul token owner with content parameters.
- `share` : publish content token set for soul token owner with share parameters.
- `comment` : publish content token set for soul token owner with comment parameters.
- `setEssenceData` : update essence middleware and configuration.
- `setSubscriptionData` : update subscription configuration.

- `setContentData` : update content configuration and middleware.
- `setW3stData` : update W3st configuration and middleware.

When the soul token owner is labeld as org account in the `soul` contract, the soul token owner and corresponding operator have authority over the functions listed below.

- `issueW3st` : issue W3st token set for collect.

Any compromise to the `soul` owner account corresponding operator may allow a hacker to take advantage of this authority issue and manipulate user's functionality in `CyberEngine` , like creating unexpected content.

In the contract `Essence` , the role `Engine` has authority over the functions listed below. Based on design the Engine address should be the contract `CyberEngine` :

- `mint` : mint Essence token.

In the contract `Subscribe` , the role `Engine` has authority over the functions listed below. Based on design the Engine address should be the contract `CyberEngine` :

- `mint` : mint Subscribe token
- `extend` : extend the expiration date for one token.

In the contract `Content` , the role `Engine` has authority over the functions listed below. Based on design the Engine address should be the contract `CyberEngine` :

- `mint` : mint Content token.

In the contract `W3st` , the role `Engine` has authority over the functions listed below. Based on design the Engine address should be the contract `CyberEngine` :

- `mint` : mint W3st token.

In the contract `MiddlewareManager` , the role `owner` has authority over the functions listed below.

- `allowMw` : update the status for the middleware address.
- `transferOwnership` : set a new owner for the contract.
- `renounceOwnership` : set address(0) as the new owner.

Any compromise to the owner's account may allow a hacker to take advantage of this authority issue and manipulate issue invalid middleware for malicious purposes.

In the contract `TokenReceiver` the role `owner` has authority over the functions listed below.

- `withdraw` : send native token from contract to arbitrary address
- `transferOwnership` : set new owner for the contract.
- `renounceOwnership` : set address(0) as the new owner.

Any compromise to the owner's account may allow a hacker to take advantage of this authority issue and withdraw the native token.

In the contract `PermissionMw` , the role `engine` has authority over the functions listed below. Based on design the Engine address should be the contract `CyberEngine` :

- `setMwData` : set `_signerStorage` information
- `preProcess` : validate collector signature

In the contract `LimitedTimePaidMw` , the role `engine` has authority over the functions listed below. Based on design the Engine address should be the contract `CyberEngine` :

- `setMwData` : set up limited time paid parameters
- `preProcess` : validate collector signature

In the contract `Treasury` the role `owner` has authority over the functions listed below.

- `setTreasuryAddress` : set new treasury address
- `setTreasuryFee` : set new treasury fee
- `allowCurrency` : update the currency status
- `transferOwnership` : set new owner for the contract.
- `renounceOwnership` : set address(0) as the new owner.

Any compromise to the owner's account may allow hackers to take advantage of this authority issue and update treasure settings.

In the contract `CyberAccountFactory` the role `owner` has authority over the functions listed below.

- `withdrawStake` : withdraw staked native tokens to the owner.
- `transferOwnership` : set new owner for the contract.
- `renounceOwnership` : set address(0) as the new owner.

Any compromise to the owner's account may allow a hacker to take advantage of this authority issue and withdraw the staked token.

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

[CyberConnect Team 08/17/2023]: The team acknowledged the finding and decided to remain unchanged.

[CertiK 08/17/2023]: CertiK strongly encourages the project team to periodically revisit the private key security management of all addresses related to privileged roles.

GLOBAL-02 | CENTRALIZATION RELATED RISKS IN CYBERID

Category	Severity	Location	Status
Centralization	● Major		● Acknowledged

Description

In the contract `CyberId`, the role `owner` has authority over the functions listed below.

- `setBaseTokenUri` : set token uri
- `setMiddleware` : set middleware
- `transferOwnership` : set new owner for the contract.
- `renounceOwnership` : set address(0) as the new owner.
- `clearGatedMetadatas` : clear gated metadata on a token.
- `batchSetGatedMetadatas` : batch set gated metadata(when id not is issued or when id is not expired).

Any compromise to the `owner` account may allow a hacker to take advantage of this authority and manipulate the contract setting.

In the contract `MocaId`, multiple roles have authority over the functions list below.

- `DEFAULT_ADMIN_ROLE` :
  - `setbaseTokenURI` : set base token uri
  - `setMiddleware` : set middleware address
  - `pause` : pause the contract
  - `unpause` : unpause the contract
  - `grantRole` : assign `PAUSER_ROLE` , `UPGRADER_ROLE` , `MINTER_ROLE` and `DEFAULT_ADMIN_ROLE` to new address.
  - `revokeRole` : revoke `PAUSER_ROLE` , `UPGRADER_ROLE` , `MINTER_ROLE` and `DEFAULT_ADMIN_ROLE` from address.
- `OPERATOR_ROLE` :
  - `allowNode` : set allowed node/extension
  - `setMocaXP` : sets the moca xp
  - `clearGatedMetadatas` : clear gated metadata on a token
  - `batchSetGatedMetadatas` : batch set gated metadata(when id not is issued or when id is not expired).

Any compromise to the aforementioned roles may allow a hacker to take advantage of this authority and manipulate the MocaId's setting.

Update in Commit 26ba7731e79531edf4b1414c730b292672c4b6a9

The contract using the `owner` role to control the aforementioned functions:

- `setbaseTokenURI` : set base token uri
- `setMiddleware` : set middleware address
- `pause` : pause the contract
- `unpause` : unpause the contract
- `allowNode` : set allowed node/extension
- `clearGatedMetadatas` : clear gated metadata on a token
- `batchSetGatedMetadatas` : batch set gated metadata
- `transferOwnership` : set new owner for the contract.
- `renounceOwnership` : set address(0) as the new owner.

In the contract `StableFeeMiddleware` , the role `NAME_REGISTRY` has authority over the functions listed below. The name `NAME_REGISTRY` suppose to be one of the name registry contracts.

- `setMwData` : update the information for `StableFeeMiddleware` .

In the contract `TrustOnlyMiddleware` : the role `owner` has authority over the functions listed below.

- `preRegister` : pre-check for CyberId registration.
- `preRenew` : pre-check for CyberId renew.
- `preBid` : pre-check for CyberId bid.
- `transferOwnership` : set new owner for the contract.
- `renounceOwnership` : set address(0) as the new owner. Any compromise to the `owner` account may allow a hacker to take advantage of this authority, manipulate the contract setting and bypass the validation for id minting.

the role `NAME_REGISTRY` has authority over the functions listed below. The name `NAME_REGISTRY` suppose to be one of the name registry contracts.

- `setMwData` : update the owner for `TrustOnlyMiddleware` .

In the contract `PermissionMw` , the role `NAME_REGISTRY` has authority over the functions listed below. The name `NAME_REGISTRY` suppose to be one of the name registry contracts.

- `setMwData` : set `_signerStorage` information

- `preProcess` : validate preprocess signature

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

[CyberConnect Team 08/17/2023]: The team acknowledged the finding and decided to remain unchanged.

[CertiK 08/17/2023]: CertiK strongly encourages the project team to periodically revisit the private key security management of all addresses related to privileged roles.

TOM-01 | MINTING CENTRALIZATION RISK

Category	Severity	Location	Status
Centralization	● **Major**	**src/middlewares/cyberid/TrustOnlyMiddleware.sol (07/19-22bb956): 36~39, 45~48, 54~57**	● **Acknowledged**

Description

If `TrustOnlyMiddleware` is used as a middleware in the contract `CyberId` , the role `_owner` has the authority to register/renew/bid CyberIds for free.

Any compromise to the `_owner` account may allow a hacker to take advantage of this authority and register/renew/bid any amount of CyberIds at will.

```
contract TrustOnlyMiddleware is Ownable, LowerCaseCyberIdMiddleware {
//...
    function preRegister(
        DataTypes.RegisterCyberIdParams calldata params,
        bytes calldata
    ) external payable override returns (uint256) {
        require(params.msgSender == owner(), "NOT_TRUSTED_CALLER");
        return 0;
    }

    /// @inheritdoc ICyberIdMiddleware
    function preRenew(
        DataTypes.RenewCyberIdParams calldata params,
        bytes calldata
    ) external payable override returns (uint256) {
        require(params.msgSender == owner(), "NOT_TRUSTED_CALLER");
        return 0;
    }

    /// @inheritdoc ICyberIdMiddleware
    function preBid(
        DataTypes.BidCyberIdParams calldata params,
        bytes calldata
    ) external payable override returns (uint256) {
        require(params.msgSender == owner(), "NOT_TRUSTED_CALLER");
        return 0;
    }
}
```

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 recommend carefully managing 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., multi-signature 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 of privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key being 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 of 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

[CyberConnect Team 08/17/2023]: The team acknowledged the finding and decided to remain unchanged.

[CertiK 08/17/2023]: CertiK strongly encourages the project team to periodically revisit the private key security management of all addresses related to privileged roles.

CYB-01 | INCOMPLETE SIGNATURE VALIDATION

Category	Severity	Location	Status
Volatile Code	● Medium	src/base/EIP712.sol (07/19-acc4d08): 52~60; src/middlewares/PermissionMw.sol (07/19-acc4d08): 88~109; src/base/EIP712.sol (07/19-22bb956): 44~61; src/middlewares/mocaid/PermissionMw.sol (07/19-22bb956): 53~55, 69~88	● Resolved

Description

The function `_requiresExpectedSigner()` is used to check if `expectedSigner` is the correct signer, which will be called by the function `preProcess()` from the contract `PermissionMw`.

```
    function _requiresExpectedSigner(
        bytes32 digest,
        address expectedSigner,
        uint8 v,
        bytes32 r,
        bytes32 s,
        uint256 deadline
    ) internal view {
        require(deadline >= block.timestamp, "DEADLINE_EXCEEDED");
        require(
            uint256(s) <=
                0x7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5D576E7357A4501DDFE92F46681B20A0,
            "INVALID_SIGNATURE_S_VAULE"
        );

        address recoveredAddress = ecrecover(digest, v, r, s);
        require(recoveredAddress == expectedSigner, "INVALID_SIGNATURE");
    }
```

If `ecrecover()` fails to recover an address, a zero address will be returned, which is not checked in `_requiresExpectedSigner()`. If the role `NameRegistry` assign a zero address to the signer through the function `setMwData()`, an invalid signature could bypass the permission check of the function `preProcess()` in the middleware `PermissionMw`.

```
//file src/middlewares/mocaid/PermissionMw.sol
    function setMwData(bytes calldata data) external override onlyNameRegistry {
        signer = abi.decode(data, (address));
    }
    function preProcess(
        DataTypes.RegisterNameParams calldata params,
        bytes calldata data
    ) external payable override onlyNameRegistry {
        DataTypes.EIP712Signature memory sig;

        (sig.v, sig.r, sig.s, sig.deadline) = abi.decode(
            data,
            (uint8, bytes32, bytes32, uint256)
        );

        _requiresExpectedSigner(
            _hashTypedDataV4(
                keccak256(
                    abi.encode(
                        Constants._REGISTER_TYPEHASH,
                        keccak256(bytes(params.name)),
                        params.parentNode,
                        params.to,
                        nonces[params.to]++,
                        sig.deadline
                    )
                )
            ),
            signer,
            sig.v,
            sig.r,
            sig.s,
            sig.deadline
        );
    }
```

References:

- https://docs.soliditylang.org/en/v0.8.9/units-and-global-variables.html?highlight=ecrecover#mathematical-and-cryptographic-functions

Recommendation

Recommends returns invalid if the result of `ecrecover()` is 0x0 as below.

```
require(recoveredAddress != address(0), "INVALID_RESULT");
```

Alleviation

[CertiK 08/17/2023]: Since ecrecover will return address(0) when the signature is invalid, If the role `NameRegistry` is assigned with a zero address to the signer through the function setMwData() in MocaID, the signature validation will be bypassed.

As a result, we recommend adding address(0) validation for MocaID.

Reference: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/9e3f4d60c581010c4a3979480e07cc7752f124cc/contracts/utils/cryptography/ECDSA.sol#L140C3-L140C3

[CyberConnect Team 08/23/2023]: The team heeded the advice and resolved the finding in the commit hash e3009fcea25c30c747c5b189de7ab748e951aa0a.

SFM-01 | MISSING VALIDATION ON THE RETURN VALUES OF `usdOracle.getRoundData()` AND `latestRoundData()`

Category	Severity	Location	Status
Volatile Code	● Medium	src/middlewares/cyberid/StableFeeMiddleware.sol (07/19-22bb956): 212~222, 224~234	● Resolved

Description

In the contract `StableFeeMiddleware` , the function `usdOracle.getRoundData()` returns the token's Oracle price. According to the documentation, it must validate the returned **timestamps**, **round ID**, and **price** to ensure the price is valid.

```
    function _getPriceAt(uint80 roundId) internal view returns (int256) {
        // prettier-ignore
        (
            /* uint80 roundID */,
            int price,
            /*uint startedAt*/,
            /*uint timeStamp*/,
            /*uint80 answeredInRound*/
        ) = usdOracle.getRoundData(roundId);
        return price;
    }
```

Moreover, The function _**getPrice()** is using `latestRoundData()` to get the token's price by Chainlink, but there are no validations that the data is not stale. Reference:

- https://docs.chain.link/data-feeds/historical-data#getrounddata-return-values

Recommendation

Recommend adding extra checks to the aforementioned return values to ensure the price is valid.

- price should be greater than zero.
- timestamp should not be zero
- answeredInRound should be equal to or greater than `roundID` .
- adding validation for updatedAt value and comparing it with block.timestamp + acceptableDelay to avoid old rounds.

Alleviation

[CyberConnect Team 08/17/2023]: The CyberConnect team heeded the advice and resolved the finding by checking the related value and removing the function `_getPriceAt()` in the commit hash ee71842ac499bbb704a7135de408f979d994ca5c.

```
    function _getPrice() internal view returns (int256) {
        // prettier-ignore
        (
            /* uint80 roundID */,
            int price,
            /* uint startedAt */,
            uint updatedAt,
            /*uint80 answeredInRound*/
        ) = usdOracle.latestRoundData();
        require(price > 0, "INVALID_ORACLE_PRICE");
        require(updatedAt > block.timestamp - 3 hours, "STALE_ORACLE_PRICE");
        return price;
    }
```

[CertiK 08/17/2023]: It is also recommended adding validation for `roundID` as well to avoid invalid price, for example:

```
require(roundId != 0, "ERR");
```

[CyberConnect Team 08/23/2023]: The team heeded the advice and resolved the finding by adding validation for `roundID` in the commit hash e3009fcea25c30c747c5b189de7ab748e951aa0a.

```
        (
            uint80 roundID,
            int price,
            /* uint startedAt */,
            uint updatedAt,
            /*uint80 answeredInRound*/
        ) = usdOracle.latestRoundData();
        require(roundID != 0, "INVALID_ORACLE_ROUND_ID");
```

SFM-02 | POTENTIAL MANIPULATION ON REGISTRATION COST

Category	Severity	Location	Status
Logical Issue	● Medium	src/middlewares/cyberid/StableFeeMiddleware.sol (07/19-22bb956): 87~88	● Resolved

Description

When register cyber ID, there is a pre-register process when the middleware of the contract is not address(0)

```
    function register(
        string calldata cid,
        address to,
        bytes32 secret,
        uint8 durationYear,
        bytes calldata middlewareData
    ) external payable {
    ...
        uint256 cost;
        if (middleware != address(0)) {
            cost = ICyberIdMiddleware(middleware).preRegister{
                value: msg.value
            }(
                DataTypes.RegisterCyberIdParams(
                    msg.sender,
                    cid,
                    to,
                    durationYear
                ),
                middlewareData
            );
        }

        _register(cid, to, durationYear, cost);
    }

```

During the preregister process in stable fee middleware, it will query the fee and calculate the cost:

```
    function preRegister(
        DataTypes.RegisterCyberIdParams calldata params,
        bytes calldata data
    ) external payable override returns (uint256) {
        uint80 roundId = abi.decode(data, (uint80));
        uint256 cost = getPriceWeiAt(params.cid, roundId, params.durationYear);
        _chargeAndRefundOverPayment(cost, params.msgSender);
        return cost;
    }
```

However, the `roundID` is based on the `middlewareData`, which is user's input. As a result, the user has the chance to select the price for the registration.

Recommendation

Recommend adding validation on `roundID` to avoid price manipulation during the registration.

Alleviation

[CyberConnect Team 08/17/2023]: The CyberConnect team heeded the advice and resolved the finding by removing roundId in the commit hash e565f7de009e99d7a777025b76a9cb07d9ac1bab.

```
    function preRegister(
        DataTypes.RegisterCyberIdParams calldata params,
        bytes calldata
    ) external payable override onlyNameRegistry returns (uint256) {
        uint256 cost = getPriceWei(params.cid, params.durationYear);
        _chargeAndRefundOverPayment(cost, params.msgSender);
        return cost;
    }
```

SR2-01 | MOCA XP SET UP VALIDATION CAN BE BYPASSED

Category	Severity	Location	Status
Logical Issue	● Medium	src/base/MetadataResolver.sol (07/19-22bb956): 116; src/core/MocaId.sol (07/19-22bb956): 374	● Resolved

Description

The value for moca xp can be updated via `setMocaXP` with a limitation of `_MAX_XP` by the operator.

```
    function setMocaXP(uint256 tokenId, uint256 xp) external {
        require(xp <= _MAX_XP, "XP_TOO_BIG");
        DataTypes.MetadataPair[] memory pairs = new DataTypes.MetadataPair[](1)
;
        pairs[0] = DataTypes.MetadataPair(_MOCA_XP_KEY, xp.toString());
        batchSetGatedMetadatas(tokenId, pairs);
        emit MocaXPSet(tokenId, xp);
    }
```

However, since `batchSetGatedMetadatas` is external function, the operator can invoke `batchSetGatedMetadatas` directly to bypass the validation on moca xp value.

Recommendation

Recommend refactoring the code related to the `batchSetGatedMetadatas` to avoid potential bypass.

Note: Updating the visibility of `batchSetGatedMetadatas` in `MetadataResolver` may result in changes for the children contracts.

Alleviation

[CyberConnect Team 08/17/2023]: The CyberConnect team resolved the finding by removing moca xp related functions in the commit hash 0417d547267132db8b169f3d8feeef32dc595e3f.

CAF-01 | MISSING VALIDATION ON `proxy` ADDRESS AND PREDICTED ADDRESS

Category	Severity	Location	Status
Volatile Code	● Minor	src/factory/CyberAccountFactory.sol (07/19-acc4d08): 65	● Acknowledged

Description

The function `createAccount` will compute the deploy address first and try to create the contract with the same salt. However, there is no validation between computed address and deployed address to ensure the deployment is successful.

Recommendation

Recommend adding validation between computed address and deployed address to avoid unexpected deployment.

Alleviation

[CyberConnect Team 08/17/2023]: The team acknowledged the finding and decided to remain unchanged.

CIB-01 | POTENTIAL STUCK TOKENS WHEN `middleware` IS `address(0)`

Category	Severity	Location	Status
Logical Issue	● Minor	src/core/CyberId.sol (07/19-22bb956): 208~221, 254~260, 297~308	● Resolved

Description

The function `register()` is intended to register a new cid and ultimately calls the function `preRegister()` from `middleware` , which will charge the required native tokens.

```
    function register(
        string calldata cid,
        address to,
        bytes32 secret,
        uint8 durationYear,
        bytes calldata middlewareData
    ) external payable {
//...
        uint256 cost;
        if (middleware != address(0)) {
            cost = ICyberIdMiddleware(middleware).preRegister{
                value: msg.value
            }(
                DataTypes.RegisterCyberIdParams(
                    msg.sender,
                    cid,
                    to,
                    durationYear
                ),
                middlewareData
            );
        }

        _register(cid, to, durationYear, cost);
}
```

However, if the middleware is zero address, the function `preBid()` won't be called and no cost will be calculated. As a result, the receiving token `msg.value` will be locked in the contract permanently.

Moreover, the same thing will happen in other functions `renew()` and `bid()` .

Recommendation

Recommends adding validation for the msg.value when `middleware` is zero address in the aforementioned functions to avoid unexpected deposit to the contract.

Alleviation

[CyberConnect Team 08/24/2023]: The team heeded the advice and resolved the finding in the commit hash 48f18ec9af2d3f9e5c67d0f4bcd9a6e131823dfc. The team now does not allow assigning a zero address to middleware and disable all the middleware-related function when the middleware is a default value address(0).

```
    function register(
        string calldata cid,
        address to,
        bytes32 secret,
        uint8 durationYear,
        bytes calldata middlewareData
    ) external payable {
        require(middleware != address(0), "MIDDLEWARE_NOT_SET");
//....
        cost = ICyberIdMiddleware(middleware).preRegister{ value: msg.value }(
            DataTypes.RegisterCyberIdParams(msg.sender, cid, to, durationYear),
            middlewareData
        );
        _register(cid, to, durationYear, cost);
}
    function setMiddleware(
        address _middleware,
        bytes calldata data
    ) external onlyOwner {
        require(_middleware != address(0), "ZERO_MIDDLEWARE");
        middleware = _middleware;
        ICyberIdMiddleware(_middleware).setMwData(data);
        emit MiddlewareSet(_middleware, data);
    }
```

CYB-02 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	● Minor	src/factory/CyberAccountFactory.sol (07/19-acc4d08): 27~32; src/core/CyberId.sol (07/19-22bb956): 100; src/middlewares/cyberid/StableFeeMiddleware.sol (07/19-22bb956): 61, 74; src/middlewares/cyberid/TrustOnlyMiddleware.sol (07/19-22bb956): 31	● Acknowledged

Description

The cited addresses are missing a check that they are not `address(0)` .

- `_owner`
- `_oracleAddress`
- `_recipient`
- `_entryPoint` and `_soul` in `CyberAccountFactory`

Recommendation

We recommend adding a check the passed-in address is not `address(0)` to prevent unexpected errors.

Alleviation

[CyberConnect Team 08/17/2023]: The team acknowledged the finding and decided to remain unchanged.

CYB-05 | LACK OF STORAGE GAP IN UPGRADEABLE CONTRACT

Category	Severity	Location	Status
Logical Issue	● Minor	src/core/CyberEngine.sol (07/19-acc4d08): 30; src/core/MocaId.sol (07/19-22bb956): 18~25	● Acknowledged

Description

There is no storage gap preserved in the logic contract. Any logic contract that acts as a base contract that needs to be inherited by other upgradeable child should have a reasonable size of storage gap preserved for the new state variable introduced by the future upgrades.

Recommendation

We recommend having a storage gap of a reasonable size preserved in the logic contract in case that new state variables are introduced in future upgrades. For more information, please refer to: https://docs.openzeppelin.com/contracts/3.x/upgradeable#storage_gaps.

Alleviation

[CyberConnect Team 08/17/2023]: The team acknowledged the finding and decided to remain unchanged.

CYB-06 | LACK OF INPUT VALIDATION

Category	Severity	Location	Status
Volatile Code	● Minor	src/core/CyberEngine.sol (07/19-acc4d08): 564, 578~581, 620, 646; src/core/Soul.sol (07/19-acc4d08): 102~104; src/core/CyberId.sol (07/19-22bb956): 421; src/core/MocaId.sol (07/19-22bb956): 332~337	● Acknowledged

Description

The cited functions do not check if the inputs are valid.

- The length of `uri` in `setEssenceData()`, `setContentData()`, `setW3stData()`, `setBaseTokenUri()`, `setbaseTokenURI()`, and `setTokenURI()` should be greater than zero.

Moreover, the function `setSubscriptionData()` does not check if the inputs are valid values.

- the length of `tokenURI` should be greater than zero
- `recipient` should be a nonzero address
- `pricePerSub` and `dayPerSub` should be greater than zero

Recommendation

Recommends adding checks to the aforementioned fields to avoid expected errors.

Alleviation

[CyberConnect Team 08/17/2023]: The team acknowledged the finding and decided to remain unchanged.

LTP-01 | INSUFFICIENT VALIDATION ON `startTimestamp` AND `endTimestamp`

Category	Severity	Location	Status
Volatile Code	● Minor	src/middlewares/LimitedTimePaidMw.sol (07/19-acc4d08): 119	● Acknowledged

Description

When adding/updating limited time paid information, the function `setMwData` will validate the relationship between the start and end timestamp:

```
require(endTimestamp > startTimestamp, "INVALID_TIME_RANGE");
```

However, there is no validation between the current and start/end timestamps. As a result, the input information can be invalid.

Recommendation

Recommend adding extra validation between the current timestamp and start/end timestamp to avoid stale information.

Alleviation

[CyberConnect Team 08/17/2023]: The team acknowledged the finding and decided to remain unchanged.

TRB-01 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	● Minor	src/periphery/TokenReceiver.sol (07/19-acc4d08): 44	● Acknowledged

Description

Addresses are not validated before assignment or external calls, potentially allowing the use of zero addresses and leading to unexpected behavior or vulnerabilities. For example, transferring tokens to a zero address can result in a permanent loss of those tokens.

```
44            payable(to).transfer(amount);
```

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

Recommendation

It is recommended to add a zero-check for the passed-in address value to prevent unexpected errors.

Alleviation

[CyberConnect Team 08/17/2023]: The team acknowledged the finding and decided to remain unchanged.

TRB-02 | USAGE OF `transfer` / `send` FOR SENDING NATIVE TOKENS

Category	Severity	Location	Status
Language Version	● Minor	src/periphery/TokenReceiver.sol (07/19-acc4d08): 44	● Acknowledged

Description

It is not recommended to use Solidity's `transfer()` and `send()` functions for transferring native tokens, since some contracts may not be able to receive the funds. Those functions forward only a fixed amount of gas (2300 specifically) and the receiving contracts may run out of gas before finishing the transfer. Also, EVM instructions' gas costs may increase in the future. Thus, some contracts that can receive now may stop working in the future due to the gas limitation.

```
44          payable(to).transfer(amount);
```

Recommendation

We recommend using the `Address.sendValue()` function from OpenZeppelin.

Since `Address.sendValue()` may allow reentrancy, we also recommend guarding against reentrancy attacks by utilizing the Checks-Effects-Interactions Pattern or applying OpenZeppelin ReentrancyGuard.

Alleviation

[CyberConnect Team 08/17/2023]: The team acknowledged the finding and decided to remain unchanged.

COR-03 | POTENTIAL FAILURE ON `safeBatchTransferFrom`

Category	Severity	Location	Status
Logical Issue	● Informational	src/core/Content.sol (07/19-acc4d08): 94~101; src/core/W3st.sol (07/19-acc4d08): 84~101	● Acknowledged

Description

The contract `content` and `W3st` provide `safeBatchTransferFrom` , which will try to query the transferability for each token then do the batch transfer.

```
    function safeBatchTransferFrom(
        address from,
        address to,
        uint256[] calldata ids,
        uint256[] calldata amounts,
        bytes calldata data
    ) public virtual override {
        for (uint256 i = 0; i < ids.length; i++) {
            if (
                !ICyberEngine(ENGINE).getW3stTransferability(_account, ids[i])
            ) {
                revert("TRANSFER_NOT_ALLOWED");
            }
        }

        super.safeBatchTransferFrom(from, to, ids, amounts, data);
    }
```

However, if one of the token in the list is not transferrable, the whole transaction will be reverted. As a result the user may need to create a transaction with updated input.

Recommendation

We would like to check with the team if this is intended.

Alleviation

[CyberConnect Team 08/17/2023]: The team acknowledged the finding and confirmed it is intended design.

CYB-07 | MISSING EMIT EVENTS

Category	Severity	Location	Status
Coding Style	● Informational	src/core/Soul.sol (07/19-acc4d08): 102~104; src/core/CyberId.sol (07/19-22bb956): 421; src/middlewares/cyberid/StableFeeMiddleware.sol (07/19-22bb956): 74~80	● Partially Resolved

Description

Functions that update state variables should emit relevant events as notifications.

Recommendation

We recommend adding events for state-changing actions, and emitting them in their relevant functions.

Alleviation

[CyberConnect Team 08/17/2023]: The team heeded the advice and partially resolved the finding in the commit hash 08909878b83839c8fe668c7c774052711ef72c39.

GLOBAL-03 | CHECK-EFFECT-INTERACTION PATTERN VIOLATION

Category	Severity	Location	Status
Logical Issue	● Informational		● Partially Resolved

Description

In both project `cyberId` and `cyberGraph`, the function `_chargeAndRefundOverPayment` or equivalent logic is being used to return the funds for the overpayment. Most of the return funds processes are triggered in the middle of the transaction but the actual data value has not been updated yet, for example, in middleware pre-process, which will trigger the external call for the funds transfer, which will violate the check-effect-interaction pattern.

Recommendation

We recommend using applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the functions that direct interact with the user in core folder to prevent reentrancy attack.

Alleviation

[CyberConnect Team 08/23/2023]: The team heeded the advice and resolved the finding for `CyberID`. in the commit hash e3009fcea25c30c747c5b189de7ab748e951aa0a.

[CertiK 08/17/2023]: The current codebase involves multiple external calls, which allows the user to reenter the contracts. As a result, it may end with some unexpected behavior, especially when the contract has other extensions or has a connection with other contracts out of the scope.

For the best security practice, we would recommend using the ReentrancyGuard to avoid any unexpected reentrancy.

OWN-01 | UNUSED CONTRACT

Category	Severity	Location	Status
Coding Style	● Informational	src/dependencies/solmate/Owned.sol (07/19-acc4d08): 11	● Acknowledged

Description

The project contains `Owned` contract definitions that is not used, which can lead to unnecessary complexity and reduced maintainability.

```
abstract contract Owned is Initializable {
//...
}
```

Recommendation

We advise removing the unused contracts or libraries.

Alleviation

[CyberConnect Team 08/17/2023]: The team acknowledged the finding and decided to remain unchanged.

SFM-03 | MISSING ACCESS RESTRICTION

Category	Severity	Location	Status
Logical Issue	● Informational	src/middlewares/cyberid/StableFeeMiddleware.sol (07/19-22bb956): 83, 94, 104	● Resolved

Description

The following function are missing access control in `StableFeeMiddleware` :

- `preRegister`
- `preRenew`
- `preBid`

Recommendation

It is recommend to add access control for aforementioned account to avoid mis-interaction.

Alleviation

[CyberConnect Team 08/17/2023]: The CyberConnect team heeded the advice and resolved the finding in the commit hash 6d9619e78649d0d3f811b58c27c62b1c6ad38a8d.

TOM-02 | MULTIPLE WAYS FOR OWNER UPDATE IN `TrustOnlyMiddleware`

Category	Severity	Location	Status
Access Control	● Informational	src/middlewares/cyberid/TrustOnlyMiddleware.sol (07/19-22bb956): 30~33	● Resolved

Description

In contract `TrustOnlyMiddleware` , the `_owner` can be updated by multiple parties

1. via `setMwData` (by NameRegistry)
2. via `transferOwnership` (by current owner).

Recommendation

We would like to check with the team if current access control for `_owner` address update is intended.

Alleviation

[CyberConnect Team 08/17/2023]: The CyberConnect team heeded the advice and resolved the finding by updating the logic for owner update in the commit hash fea2568a06430c1159a8ea5a088619cbb1c09c8d.

OPTIMIZATIONS | CYBERCONNECT - AUDIT

ID	Title	Category	Severity	Status
COR-02	Redundant Calls To `_disableInitializers()`	Volatile Code	Optimization	● Acknowledged
SRC-01	Missing Check For Current Values	Volatile Code	Optimization	● Acknowledged

COR-02 | REDUNDANT CALLS TO `_disableInitializers()`

Category	Severity	Location	Status
Volatile Code	● Optimization	src/core/Content.sol (07/19-acc4d08): 35; src/core/Essence.sol (07/19-acc4d08): 37; src/core/Subscribe.sol (07/19-acc4d08): 36; src/core/W3st.sol (07/19-acc4d08): 34	● Acknowledged

Description

The function `_disableInitializers()` is already called in the parent contracts, which can ensure that the derived contract is initialized. There is no need to call it again in the derived contracts, this could save gas and make the code more efficient.

Recommendation

We recommend removing redundant calls to `_disableInitializers()` in each function.

Alleviation

[CyberConnect Team 08/17/2023]: The team acknowledged the finding and decided to remain unchanged.

SRC-01 | MISSING CHECK FOR CURRENT VALUES

Category	Severity	Location	Status
Volatile Code	● Optimization	src/core/MiddlewareManager.sol (07/19-acc4d08): 34~37; src/middlewares/base/Treasury.sol (07/19-acc4d08): 51~56, 64~69, 77~81	● Acknowledged

Description

The following functions do not check if the input is equal to the current value, an equivalent value can return directly without updating the state to save gas.

- `function allowMw(address mw, bool allowed)`
- `function setTreasuryAddress(address treasuryAddress)`
- `function setTreasuryFee(uint16 treasuryFee)`
- `function allowCurrency(address currency, bool allowed)`

Recommendation

Recommends checking inputs against current values to save gas.

Alleviation

[CyberConnect Team 08/17/2023]: The team acknowledged the finding and decided to remain unchanged.

APPENDIX | CYBERCONNECT - AUDIT

Finding Categories

Categories	Description
Coding Style	Coding Style findings may not affect code behavior, but indicate areas where coding practices can be improved to make the code more understandable and maintainable.
Language Version	Language Version findings indicate that the code uses certain compiler versions or language features with known security issues.
Access Control	Access Control findings are about security vulnerabilities that make protected assets unsafe.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases and may result in vulnerabilities.
Logical Issue	Logical Issue findings indicate general implementation issues related to the program logic.
Centralization	Centralization findings detail the design choices of designating privileged roles or other centralized controls over the code.

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.