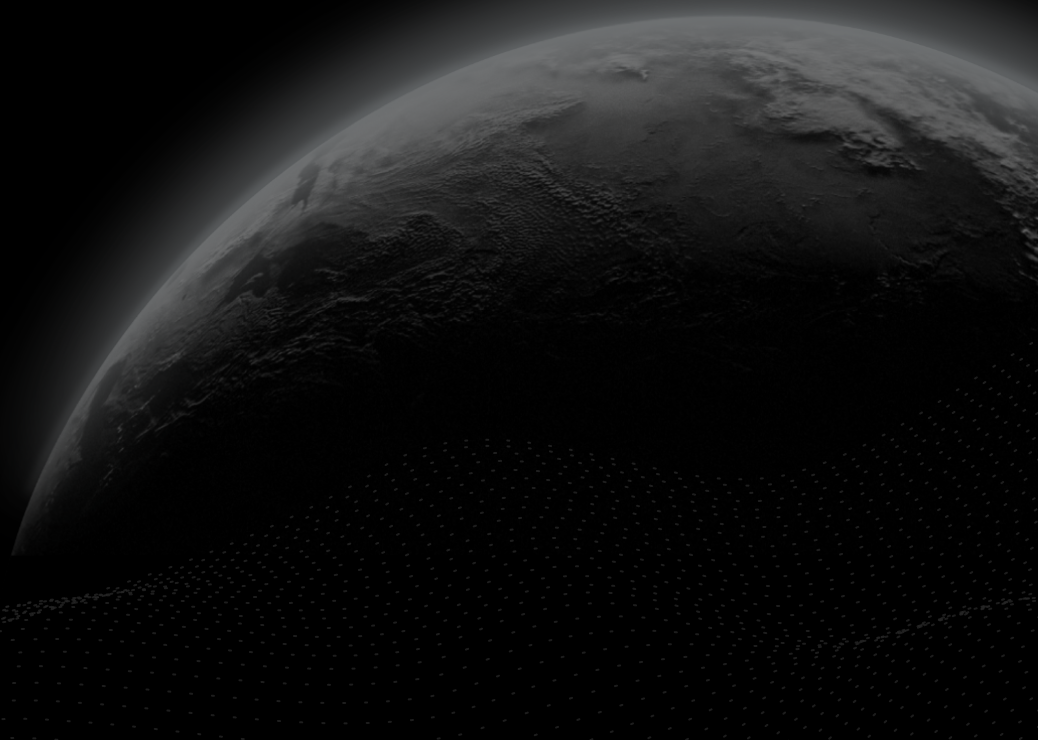
CERTIK

Security Assessment

IDRX

CertiK Assessed on Jul 7th, 2023



CertiK Assessed on Jul 7th, 2023

IDRX

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

Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	Polygon (MATIC)	Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 07/07/2023	N/A

CODEBASE

https://polygonscan.com/address/0x20fad183dc35f4ae0d1d125ae2f3c4c43a53bbd0

https://polygonscan.com/address/0x649a2da7b28e0d54c13d5eff95d3a

View All in Codebase Page

Vulnerability Summary

10	6	0	0	4	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	1 Resolved, 4 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.
2	Medium	2 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
1	Minor	1 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.
2	Informational	2 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 | IDRX

CODEBASE | IDRX

Repository

https://polygonscan.com/address/0x20fad183dc35f4ae0d1d125ae2f3c4c43a53bbd0

https://polygonscan.com/address/0x649a2da7b28e0d54c13d5eff95d3a660652742cc

AUDIT SCOPE | IDRX

4 files audited ● 1 file with Acknowledged findings ● 1 file with Resolved findings ● 2 files without findings

ID	Repo	File	SHA256 Checksum
● IDR	mainnet	contracts/IDRX.sol	751efda268d626a30bb262eb6863543b89c778e6d821cdc44c4599ceec2a63e1
● IDX	mainnet	contracts/IDRXBasicToken.sol	06dcceb6d2c79b11fdee9978f39e688019cfd31bc04c75106336d33f77a1f1a3
● ERC	mainnet	contracts/ERC20BurnableUpgradeable.sol	a58320c20fe5c162397b34cf6c145c7b041b7cb51338805d0f23a2b5ed5c7198
● IMP	mainnet	contracts/import.sol	fbd2dbc1a472e4e58973c7554b906b2fb5012114018ce69bf6f13a0de5b949fa

APPROACH & METHODS | IDRX

This report has been prepared for IDRX to discover issues and vulnerabilities in the source code of the IDRX 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 | IDRX

10	0	5	2	1	2
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for IDRX. Through this audit, we have uncovered 10 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
GLOBAL-01	**Centralized Control Of Contract Upgrade**	**Centralization**	**Major**	**Acknowledged**
IDR-01	**Centralized Balance Manipulation**	**Centralization**	**Major**	**Acknowledged**
IDR-05	**Arbitrary Account Blacklisting Leading To Potential Fund Loss**	**Centralization**	**Major**	**Acknowledged**
IDR-08	**Centralization Related Risks**	**Centralization**	**Major**	**Acknowledged**
IDR-09	Potential Bypass Of Blacklisted Users	Logical Issue	Major	Resolved
IDR-02	No Upper Limit In `setMarketplaceFee` Function	Logical Issue	Medium	Resolved
IDR-10	Zero Address Can Be Blacklisted	Logical Issue	Medium	Resolved
IDR-07	Missing Zero Address Validation	Volatile Code	Minor	Resolved
IDR-03	Inconsistency In Bridge Nonce Incrementing	Inconsistency	Informational	Resolved
IDX-01	Usage Of Hardhat's Console	Coding Style	Informational	Resolved

GLOBAL-01 | CENTRALIZED CONTROL OF CONTRACT UPGRADE

Category	Severity	Location	Status
Centralization	**Major**		**Acknowledged**

Description

The privileged role has the authority to update the implementation contract behind the proxy contract.

Any compromise to the privileged 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

`[IDRX, 20230626]`: The multi-sig proxy has been given the roles Proposer, Executor and Canceler to interact with TImeLock.

Once the IDRX contract changes are completely audited and deployed, we will assign the Upgrader role to TimeLock

`[CertiK, 20230626]`: The team deployed timelock and multisig wallet on polygon:

Timelock: https://polygonscan.com/address/0x58aa9720f456667c97093aaf87623d656f1ee6fa

CANCELLER_ROLE: 0xf80fdF246928B7862B23e094b3a14C4E36eE117E EXECUTOR_ROLE: 0xf80fdF246928B7862B23e094b3a14C4E36eE117E PROPOSER_ROLE: 0xf80fdF246928B7862B23e094b3a14C4E36eE117E TIMELOCK_ADMIN_ROLE: 0xcac3cf6b226317d91c2e72ac7193a83c34728b2c, 0x58aa9720f456667c97093aaf87623d656f1ee6fa

Multisig Wallet: https://polygonscan.com/address/0xf80fdF246928B7862B23e094b3a14C4E36eE117E

There are 3 signers:

- matic:0xE51b67864A38F42126231d30eD8f3e72Ec7F32f4
- matic:0xb4fcA8725a9E65B4cD1Ef27d71e7C7537148061b
- matic:0x0f9f4cbBEc64524DEB0D743F9CAF34be81BD1576

Any transaction requires the confirmation of 2 out of 3 owners.

IDR-01 | CENTRALIZED BALANCE MANIPULATION

Category	Severity	Location	Status
Centralization	**Major**	**contracts/IDRX.sol (IDRX): 64, 68**	**Acknowledged**

Description

In the contract `IDRX` , the role `MINTER_ROLE` has the authority to update the token balance of an arbitrary account without sanity restriction.

Any compromise to the `MINTER_ROLE` account may allow a hacker to take advantage of this authority and manipulate users' balances by calling `mint()` and/or `mintBridge()` functions

Recommendation

We recommend the team makes efforts to restrict access to the private key of the privileged account. A strategy of 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 mint more tokens or engage in similar balance-related operations.
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 multi signature (⅔, ⅗) wallet *mitigate* the risk by avoiding a single point of key management failure.

- 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 **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 DAO for controlling the operation *mitigate* the risk by applying transparency and decentralization.

- Introduction of a DAO, governance, or voting module to increase decentralization, transparency, and user involvement;
  AND
- A medium/blog link for sharing the multi-signers' addresses, and DAO information with the community.

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

- 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:

The following actions can *fully* resolve the risk:

- Renounce the ownership and never claim back the privileged role.
  OR
- Remove the risky functionality.
  ORa
- Add minting logic (such as a vesting schedule) to the contract instead of allowing the owner account to call the sensitive function directly.

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

`[IDRX, 20230626]` : IDRX is a stable token and our platform has to mint the token in an instant, therefore we can not use time-lock for this particular function.

MINTER_ROLE will be assigned to an MPC wallet, rather than Multisig wallet.

IDR-05 ARBITRARY ACCOUNT BLACKLISTING LEADING TO POTENTIAL FUND LOSS

Category	Severity	Location	Status
Centralization	**Major**	**contracts/IDRX.sol (IDRX): 120~130, 130**	**Acknowledged**

Description

The contract provides an arbitrary account blacklisting feature, where a privileged role (holder of `BLACKLIST_ROLE`) can blacklist a user's account, and then destroy any funds held in the blacklisted account.

Scenario

If the privilege is wrongly assigned or gets into malicious hands, it can lead to huge fund loss for users as the bad actor can blacklist any account and destroy its funds.

Proof of Concept

An account with the `BLACKLIST_ROLE` can call `addBlackList(address _evilUser)` to add any user to the blacklist, and then call `destroyBlackFunds(address _blackListedUser)` to destroy all funds of that blacklisted user.

Recommendation

Consider removing the ability to blacklist users and destroy their funds or add further checks and balances to this process to prevent misuse. For the BLACKLIST_ROLE role accounts, in order to avoid single point of failure, 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.

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

`[IDRX, 20230626]`: For law enforcement, we are required to have blacklist feature.

We also need the function to run instantly therefore we cannot implement timelock.

The BLACKLIST_ROLE will be assigned to an MPC wallet on Qredo platform

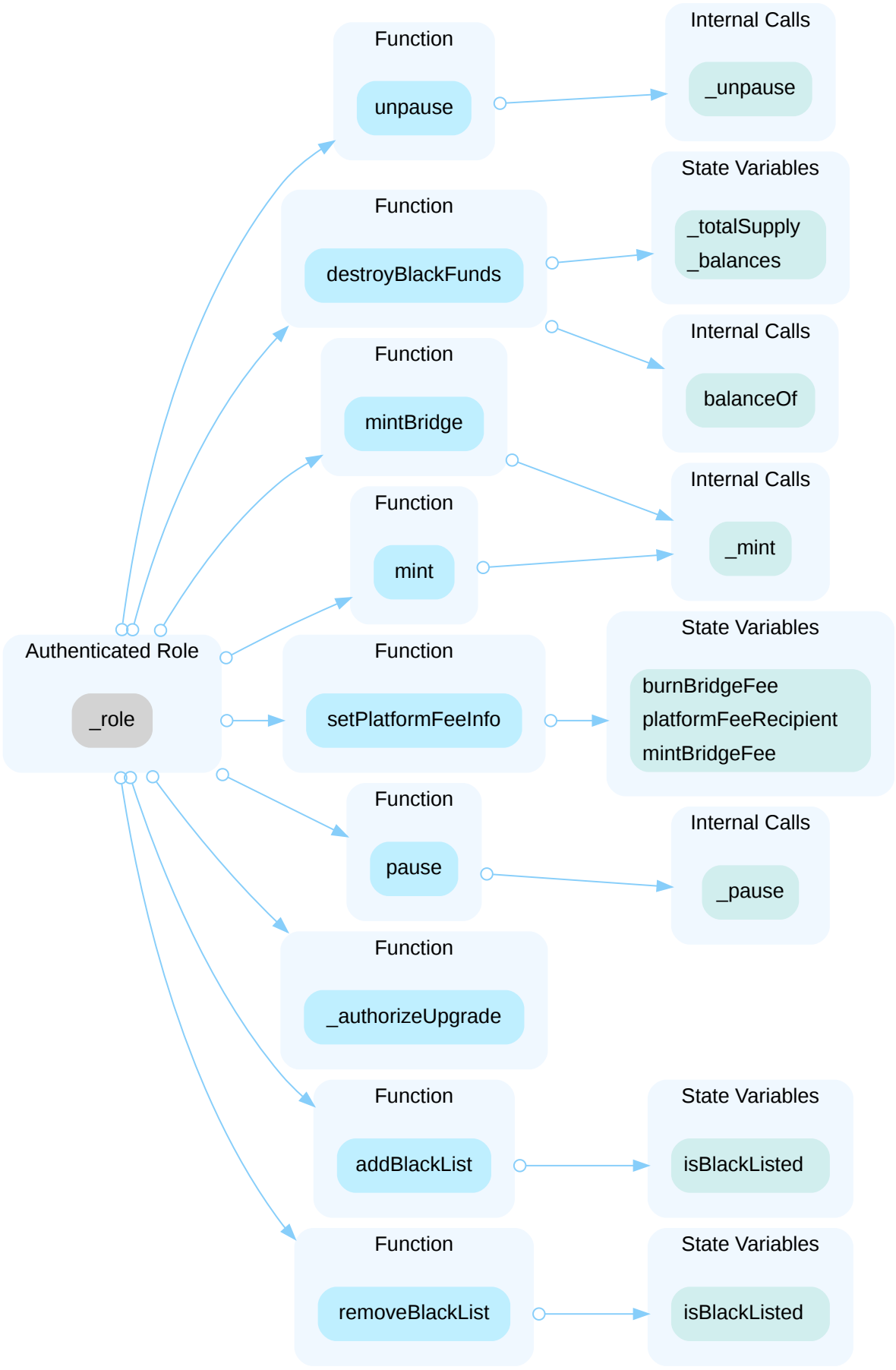
`[CertiK, 20230626]`: The team clarified the context and will remain the current setting

IDR-08 | CENTRALIZATION RELATED RISKS

Category	Severity	Location	Status
Centralization	**Major**	**contracts/IDRX.sol (IDRX): 56, 60, 64, 68, 110, 120, 125, 130, 138**	**Acknowledged**

Description

In the contract `IDRX` the priviledge roles have authority over the functions shown in the diagram below.


Function
unpause
Internal Calls
_unpause
Function
destroyBlackFunds
State Variables
_totalSupply
_balances
Internal Calls
balanceOf
Function
mintBridge
Internal Calls
_mint
Function
mint
Authenticated Role
_role
Function
setPlatformFeeInfo
State Variables
burnBridgeFee
platformFeeRecipient
mintBridgeFee
Function
pause
Internal Calls
_pause
Function
_authorizeUpgrade
Function
addBlackList
State Variables
isBlackListed
Function
removeBlackList
State Variables
isBlackListed

- `pause()` : Pauses all token transfer functionality. Only accessible to addresses with the `PAUSER_ROLE` .
- `unpause()` : Unpauses the token transfer functionality. Only accessible to addresses with the `PAUSER_ROLE` .
- `mint(address to, uint256 amount)` : Mints the specified amount of tokens to the specified address. Only accessible to addresses with the `MINTER_ROLE` .
- `mintBridge(address to, uint256 amount, uint fromChain, uint fromChainBridgeNonce)` : Mints tokens and transfers them to a bridge after deducting a fee. Only accessible to addresses with the `MINTER_ROLE` .
- `_authorizeUpgrade(address newImplementation)` : Authorizes a new implementation for the contract. Only accessible to addresses with the `UPGRADER_ROLE` .
- `addBlackList(address _evilUser)` : Adds the specified address to the blacklist. Only accessible to addresses with the `BLACKLIST_ROLE` .
- `removeBlackList(address _clearedUser)` : Removes the specified address from the blacklist. Only accessible to addresses with the `BLACKLIST_ROLE` .
- `destroyBlackFunds(address _blackListedUser)` : Destroys the funds of a blacklisted address. Only accessible to addresses with the `BLACKLIST_ROLE` .
- `setPlatformFeeInfo(address _platformFeeRecipient, uint64 _burnBridgeFee, uint64 _mintBridgeFee)` : Sets the platform fee recipient and the fees for burning and minting on the bridge. Only accessible to addresses with the `MINTER_ROLE` .

Any compromise to the `_role` account may allow the hacker to take advantage of this authority and update the sensitive settings and execute sensitive functions of the project.

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

`[IDRX, 20230626]`: The multi sig proxy has been given the roles Proposer, Executor and Canceler to interact with TImeLock.

Once the IDRX contract changes are completely audited and upgraded, we will assign the Upgrader role and PlatformFeeSetter role to TimeLock

`[CertiK, 20230626]`: The team deployed timelock and multisig wallet on polygon:

Timelock: https://polygonscan.com/address/0x58aa9720f456667c97093aaf87623d656f1ee6fa

CANCELLER_ROLE: 0xf80fdF246928B7862B23e094b3a14C4E36eE117E EXECUTOR_ROLE: 0xf80fdF246928B7862B23e094b3a14C4E36eE117E PROPOSER_ROLE: 0xf80fdF246928B7862B23e094b3a14C4E36eE117E TIMELOCK_ADMIN_ROLE: 0xcac3cf6b226317d91c2e72ac7193a83c34728b2c, 0x58aa9720f456667c97093aaf87623d656f1ee6fa

Multisig Wallet: https://polygonscan.com/address/0xf80fdF246928B7862B23e094b3a14C4E36eE117E

There are 3 signers:

- matic:0xE51b67864A38F42126231d30eD8f3e72Ec7F32f4
- matic:0xb4fcA8725a9E65B4cD1Ef27d71e7C7537148061b

- matic:0x0f9f4cbBEc64524DEB0D743F9CAF34be81BD1576

Any transaction requires the confirmation of 2 out of 3 owners.

IDR-09 | POTENTIAL BYPASS OF BLACKLISTED USERS

Category	Severity	Location	Status
Logical Issue	Major	contracts/IDRX.sol (IDRX): 106	Resolved

Description

The blacklist feature is designed to prevent the blacklisted users from transferring the `IDRX` to others by validating the following **require** statement in the `_beforeTokenTransfer()` of `IDRX` contract:

```
106  require(!isBlackListed[msg.sender], "Blacklist: account is blacklisted");
```

The **require** statement prohibits the blacklisted users from calling `transfer()` and `transferFrom()` to move their funds by requiring the `msg.sender` not blacklisted. However, this feature can be bypassed by using a combination of the `approve()` / `increaseAllowance()` and `transferFrom()` functions. The blacklisted user could approve another account to transfer his tokens on his behalf.

Recommendation

We recommend validating the `msg.sender` is not blacklisted in the function `approve()`. Or we recommend checking the `from` address is not blacklisted in the function `transferFrom()`.

Alleviation

`[IDRX, 20230705]`: We have followed your recommendation for validating the msg.sender is not blacklisted in the function approve().

Please find the changes in the following link

https://github.com/nusa-idrx/idrx-backend/blob/345baa4c7ddc2dcf8bdab90ed6cb9d1aef386247/packages/smartcontract/contracts/IDRX.sol#L177

`[CertiK, 20230705]`: The team heeded the advice and resolved the finding in the commit 345baa4c7ddc2dcf8bdab90ed6cb9d1aef386247

IDR-02 | NO UPPER LIMIT IN `setMarketplaceFee` FUNCTION

Category	Severity	Location	Status
Logical Issue	Medium	contracts/IDRX.sol (IDRX): 143, 144	Resolved

Description

There are no upper boundaries for `_burnBridgeFee` and `_mintBridgeFee` which are used to calculate fees that would be charged in the bridge. It is possible to set the total fee rate up to any arbitrary amount.

```
138 function setPlatformFeeInfo(
139     address _platformFeeRecipient,
140     uint64 _burnBridgeFee,
141     uint64 _mintBridgeFee
142 ) external onlyRole(MINTER_ROLE) {
143     burnBridgeFee = _burnBridgeFee;
144     mintBridgeFee = _mintBridgeFee;
145     ...
146 }
```

Recommendation

We recommend adding reasonable boundaries for the fees.

Alleviation

`[IDRX, 20230705]`: Please find the changes in the following link

https://github.com/nusa-idrx/idrx-backend/blob/345baa4c7ddc2dcf8bdab90ed6cb9d1aef386247/packages/smartcontract/contracts/IDRX.sol#L26C28-L26C42

`[CertiK, 20230705]`: The team heeded the advice and resolved the finding in the commit 345baa4c7ddc2dcf8bdab90ed6cb9d1aef386247

IDR-10 | ZERO ADDRESS CAN BE BLACKLISTED

Category	Severity	Location	Status
Logical Issue	Medium	contracts/IDRX.sol (IDRX): 101	Resolved

Description

If the zero address is blacklisted, calls to internal ERC20 functions `_mint()` and `_burn()` will fail because the `beforeTokenTransfer()` or `afterTokenTransfer()` hook disallow transferring to/from the zero address.

Recommendation

We recommend adding checks to ensure the zero address cannot be blacklisted in the contract.

Alleviation

`[IDRX, 20230705]`: Please find the changes in the following link

https://github.com/nusa-idrx/idrx-backend/blob/345baa4c7ddc2dcf8bdab90ed6cb9d1aef386247/packages/smartcontract/contracts/IDRX.sol#L128C81-L128C81

`[CertiK, 20230705]`: The team heeded the advice and resolved the finding in the commit 345baa4c7ddc2dcf8bdab90ed6cb9d1aef386247

IDR-07 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	Minor	contracts/IDRX.sol (IDRX): 145	Resolved

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.

```
145         platformFeeRecipient = _platformFeeRecipient;
```

- `_platformFeeRecipient` 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

`[IDRX, 20230626]`: Issue acknowledged. Changes have been reflected in the commit hash 8ecd9513e13be892b9a39a89ed19e2e93b32f9da

`[CertiK, 20230626]`: The team heeded the advice and resolved the finding in the updated commit

IDR-03 | INCONSISTENCY IN BRIDGE NONCE INCREMENTING

Category	Severity	Location	Status
Inconsistency	Informational	contracts/IDRX.sol (IDRX): 68, 86	Resolved

Description

The `_bridgeNonce` value is incremented in the `burnBridge()` function, but not in the `mintBridge()` function. The `_bridgeNonce` is supposed to represent the unique count of transactions that move funds between bridges, but this count is currently only updated in the `burnBridge()` function and not when minting through the bridge.

Scenario

If multiple transactions are executed using the minting bridge, these transactions will not be reflected in the `_bridgeNonce` count. This may potentially cause confusion or misalignment in the bridge transaction count, especially in scenarios where accurate transaction tracking is essential.

Proof of Concept

Run the `mintBridge()` function multiple times and observe that `_bridgeNonce` does not change. This shows that the nonce is not being incremented when `mintBridge()` is called. Conversely, executing `burnBridge()` increments the nonce as expected.

Recommendation

We would like to double confirm the intention of using `_bridgeNonce` whether it is used to count the burn transaction or any transaction(mint/burn) counts.

If its intention is to count all transactions, to maintain consistency and accurate tracking of bridge transactions, it is recommended to increment `_bridgeNonce` in the `mintBridge()` function as well, similar to how it is done in `burnBridge()`. This can be done by calling the `incrementNonce()` function inside `mintBridge()`.

Alleviation

`[IDRX, 20230626]`: The process of bridging is done manually. The user will call the burnBridge function, and the mintBridge function is done manually through the an MPC wallet. There is a high chance that the minting will not be done incrementally based on the bridgeNonce of the source chain. However we will add a condition on the mintBridge function, which checks if the bridgeNonce of the source chain has been minted yet.

The changes can be seen here:

c104b1f9ec5e0bfe8093c1f01f3c966896c84ffb

`[CertiK, 20230626]` : The team clarified the context and resolved the finding by checking the `fromChainNonceUsed` status of certain `fromChain` id and `fromChainBridgeNonce` in the updated commit

IDX-01 | USAGE OF HARDHAT'S CONSOLE

Category	Severity	Location	Status
Coding Style	Informational	contracts/IDRXBasicToken.sol (IDRX): 10, 120, 170	Resolved

Description

The contract uses the console contract from Hardhat, which is meant to be used for testing purposes.

Recommendation

It is recommended to remove the import of Hardhat's contract for better code readability and simplicity.

Alleviation

`[IDRX, 20230626]`: Issue acknowledged. Changes have been reflected in the commit hash 8ecd9513e13be892b9a39a89ed19e2e93b32f9da

`[CertiK, 20230626]`: The team heeded the advice and resolved the finding in the updated commit

APPENDIX | IDRX

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.
Inconsistency	Inconsistency findings refer to different parts of code that are not consistent or code that does not behave according to its specification.
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.