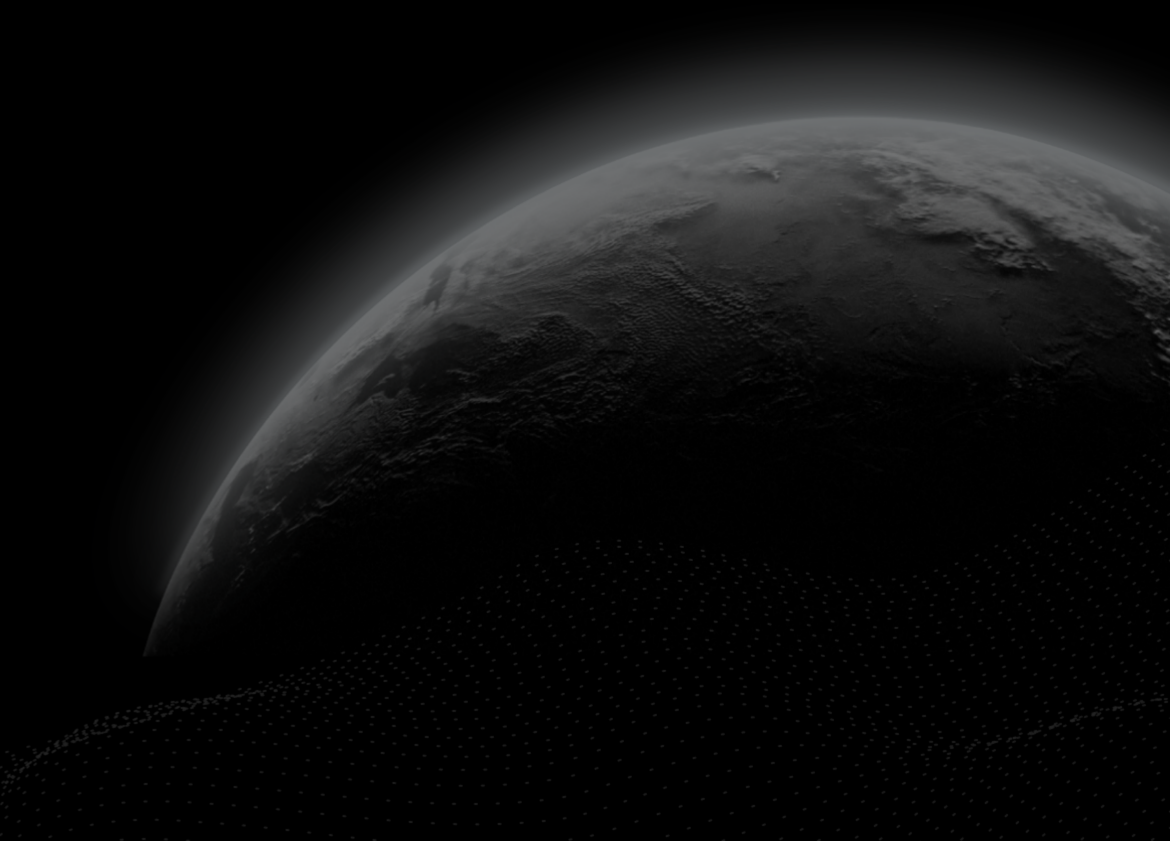
CERTIK

Security Assessment

USDL

CertiK Assessed on Mar 1st, 2024



CertiK Assessed on Mar 1st, 2024

USDL

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

Executive Summary

TYPES	ECOSYSTEM	METHODS
ERC-20	Ethereum (ETH)	Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 03/01/2024	N/A

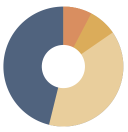
CODEBASE
ybs-contract-internal
View All in Codebase Page

COMMITS
base: e0bccf00d7fa82622d46376e4f5c1c74f05b9711
update 1: 3101c55803ac178042db5ddf89b4b3130c5d84ac
update 2: 51e931918b27daa3b33caceade03bdf5106d67ab
View All in Codebase Page

Highlighted Centralization Risks

- Contract upgradeability
- Privileged role can mint tokens
- Transfers can be paused
- Has blacklist/whitelist
- Privileged role can remove users' tokens

Vulnerability Summary



13	10	0	1	2	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.
🟧 1	Major	1 Acknowledged	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
🟨 1	Medium	1 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
🟨 5	Minor	4 Resolved, 1 Partially 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.

■ 6	Informational	5 Resolved, 1 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 | USDL

CODEBASE | USDL

Repository

ybs-contract-internal

Commit

base: e0bccf00d7fa82622d46376e4f5c1c74f05b9711

update 1: 3101c55803ac178042db5ddf89b4b3130c5d84ac

update 2: 51e931918b27daa3b33caceade03bdf5106d67ab

update3: 6267d04f417c85996791691687e0ba86e0bfbc09

AUDIT SCOPE | USDL

7 files audited ● 3 files with Acknowledged findings ● 3 files with Resolved findings ● 1 file without findings

ID	Repo	File	SHA256 Checksum
● EI3	paxosglobal/ybs-contract-internal	contracts/lib/EIP3009.sol	7b50958fdc503754f5bb820db4a4e6a80da53b5062f50595c747f1245583787a
● PBA	paxosglobal/ybs-contract-internal	contracts/lib/PaxosBaseAbstract.sol	e79f6d824d15ccf5f81b8cbf17ed03b2602b9485c75d1ea938f31256139b42d0
● YBS	paxosglobal/ybs-contract-internal	contracts/YBS.sol	2531b78ef38efe7a1ffc7056f02d4c414a4f54cbb3312c46be1e273ddbd9d488
● ECR	paxosglobal/ybs-contract-internal	contracts/lib/ECRecover.sol	05a64986f78875ce2044fae79a4cdfa277eebffea8f8cde0c9a3c9825dbc3e52
● EIP	paxosglobal/ybs-contract-internal	contracts/lib/EIP2612.sol	4919b0e1f9307fcc543ebadec8c12b6e81bf9d2817ff06041d79ba3654d10150
● EI7	paxosglobal/ybs-contract-internal	contracts/lib/EIP712.sol	170455996b9bc6ac3431e8ed98d2c6d47451cffca3e3d17c824452f0fe04f883
● YBV	paxosglobal/ybs-contract-internal	contracts/YBSV1.sol	08a21c2a1dddfb4b5ab7e08205f0770de0209b203ea51cbff2abaf55a75ce918

APPROACH & METHODS | USDL

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

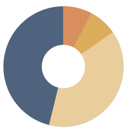
The auditing process pays special attention to the following considerations:

- 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 | USDL



13	0	1	1	5	6
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for USDL. Through this audit, we have uncovered 13 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
YBS-11	**Centralization Risks In YBS.Sol**	**Centralization**	**Major**	**● Acknowledged**
YBC-01	Wrong Multiplier Is Set By `increaseRebaseMultiplier()` If `rebaseRate == 0`	Logical Issue	Medium	● Resolved
YBS-01	Wrong Arguments Order Of `InsufficientSupply` Event	Inconsistency	Minor	● Resolved
YBS-02	Inaccurate `_fixedShares` Updating	Volatile Code	Minor	● Resolved
YBS-04	Potential Divide By Zero	Logical Issue	Minor	● Partially Resolved
YBS-09	Missing Input Validation	Volatile Code	Minor	● Resolved
YBS-13	Missing Zero Address Validation	Volatile Code	Minor	● Resolved
CON-01	Inaccurate Comments	Inconsistency	Informational	● Resolved
EI3-02	Missing NatSpec Comments	Inconsistency	Informational	● Resolved
PBA-01	Lack Of Storage Gap In Upgradeable Parent Contract	Logical Issue	Informational	● Acknowledged

ID	Title	Category	Severity	Status
YBS-03	`__gap_YBS` Size Is Inaccurate	Inconsistency	Informational	● Resolved
YBS-14	Discrepancy Between `totalSupply()` And The Total Sum Of All Balances	Inconsistency	Informational	● Resolved
YBS-15	Unblocked Accounts May Lose Dust Amounts Of Previous Fixed Funds	Volatile Code	Informational	● Resolved

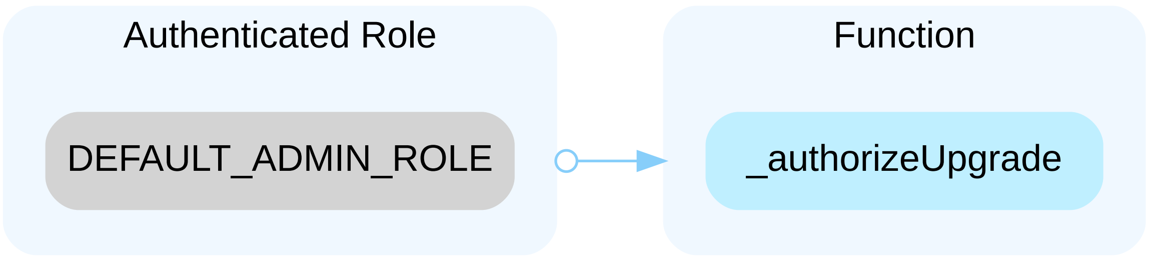
YBS-11 | CENTRALIZATION RISKS IN YBS.SOL

Category	Severity	Location	Status
Centralization	● **Major**	**contracts/YBS.sol (base): 170, 179, 188, 202, 217, 231, 246, 257, 271, 286, 386, 415, 576, 673**	● **Acknowledged**

Description

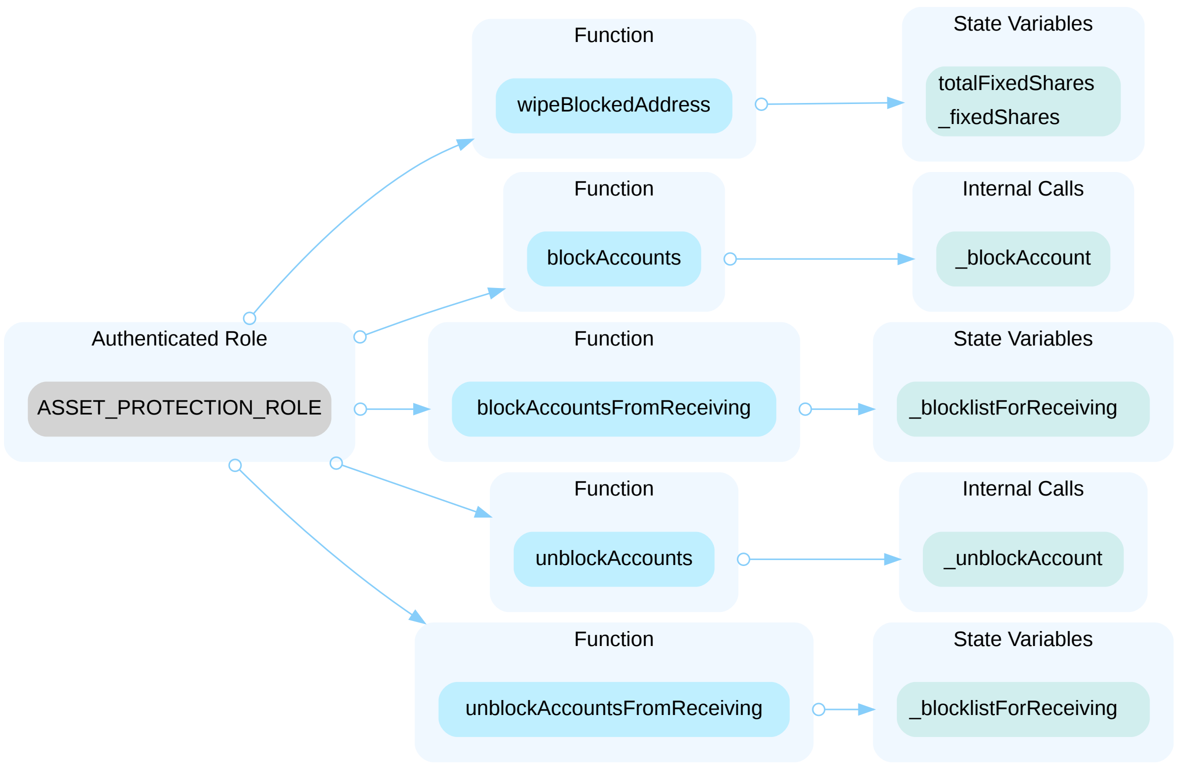
In the contract `YBS` the role `DEFAULT_ADMIN_ROLE` has authority over the functions shown in the diagram below. Any compromise to the `DEFAULT_ADMIN_ROLE` account may allow the hacker to take advantage of this authority and

- change the implementation contract pointed by proxy and therefore execute potential malicious functionality in the implementation contract
- give an account they control any of the roles outlined within this finding
- remove access to any of the roles outlined within this finding from other accounts

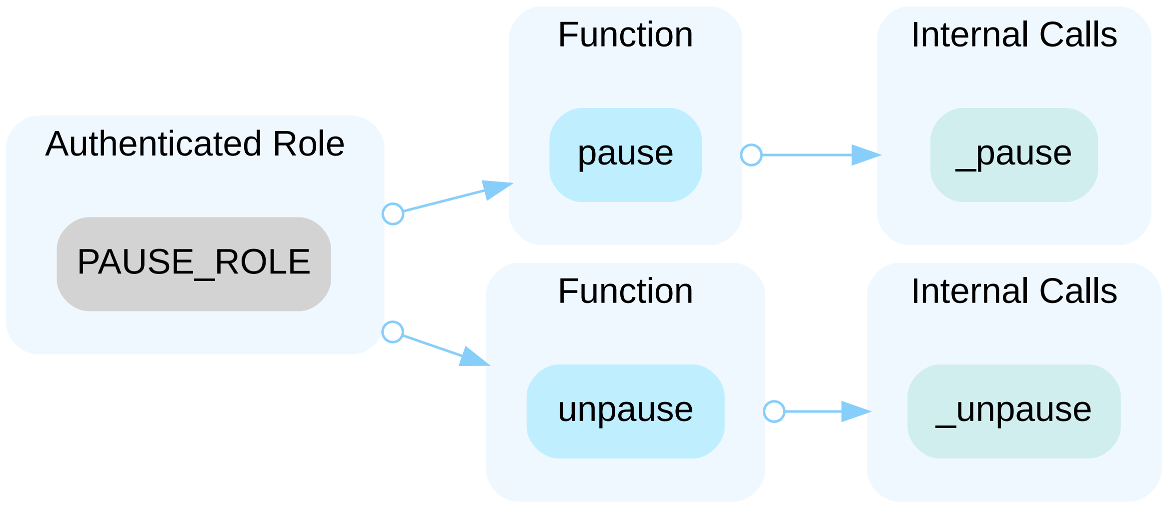


In the contract `YBS` the role `ASSET_PROTECTION_ROLE` has authority over the functions shown in the diagram below. Any compromise to the `ASSET_PROTECTION_ROLE` account may allow the hacker to take advantage of this authority and

- block any account, preventing the sending or receiving of tokens
- unblock accounts previously blocked from sending or receiving tokens
- remove all fixed shares from any blocked account



In the contract `YBS` the role `PAUSE_ROLE` has authority over the functions shown in the diagram below. Any compromise to the `PAUSE_ROLE` account may allow the hacker to take advantage of this authority and pause or unpause the contract's key user endpoint functionality during time-sensitive periods.

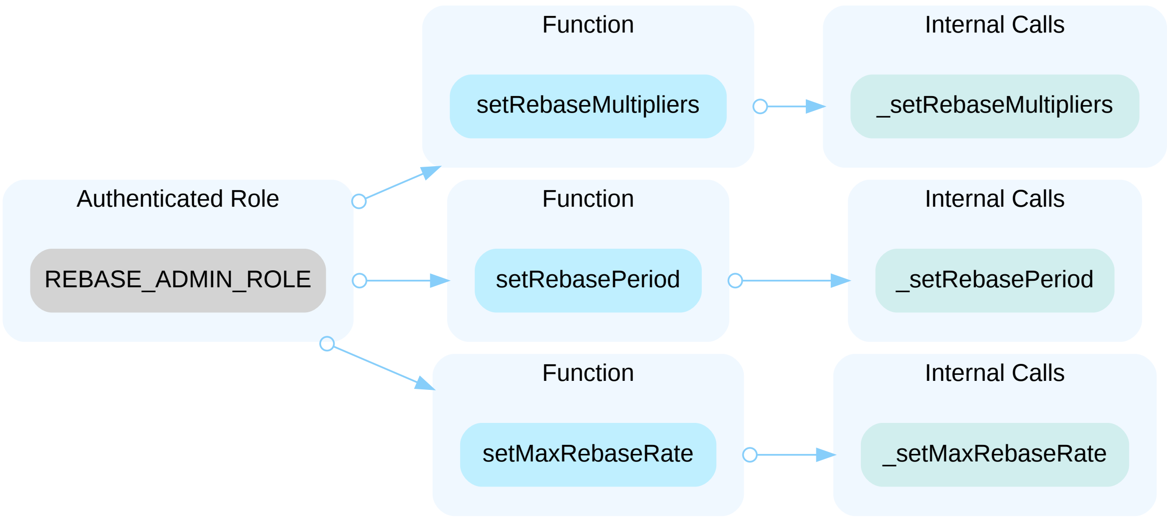


In the contract `YBS` the role `REBASE_ADMIN_ROLE` has authority over the functions shown in the diagram below. Any compromise to the `REBASE_ADMIN_ROLE` account may allow the hacker to take advantage of this authority and

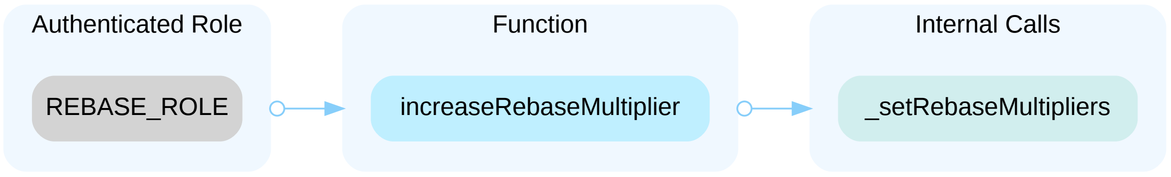
- use function `setRebaseMultipliers()` to immediately update the rebase multiplier to any value. They could use this to temporarily scale the worth of their own shares for gain. This function can also be used to set the rebase multiplier to a value lower than it previously was, causing negative rebasing.
- update the `rebasePeriod` used in determining the next timestamp for incrementing the rebase multiplier. This can be used to either block a subsequent update to the rebase multiplier by the `REBASE_ROLE` (see below) if set to a

large enough value, or allow for multiple rebase multiplier updates in a short span of time if set low enough.

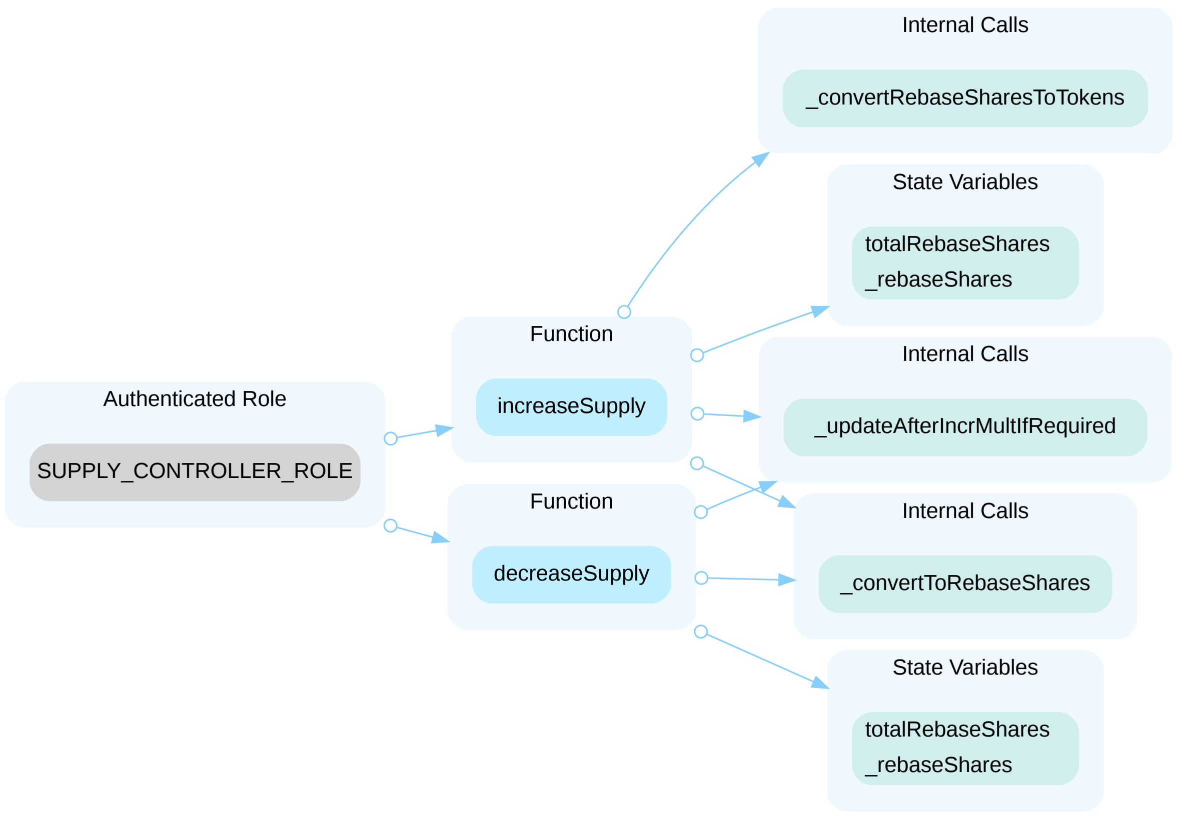
- update the `maxRebateRate` to be any value between 0 and 1e18, inclusive. This value determines the cap on the increase to the rebase multiplier. Setting this value to 0 prevents a successful update of the rebate multiplier via functions controlled by the `REBASE_ROLE`.



In the contract `YBS` the role `REBASE_ROLE` has authority over the functions shown in the diagram below. Any compromise to the `REBASE_ROLE` account may allow the hacker to take advantage of this authority and call function `increaseRebaseMultiplier()` to update the rebase multiplier to any value 1e18 and the upper bound, determined by the `REBASE_ADMIN_ROLE` (see above). This can only be called once time period, determined by the `REBASE_ADMIN_ROLE`.



In the contract `YBS` the role `SUPPLY_CONTROLLER_ROLE` has authority over the functions shown in the diagram below. Any compromise to the `SUPPLY_CONTROLLER_ROLE` account may allow the hacker to take advantage of this authority and increase or decrease the total token supply by minting tokens to their account or burning tokens from their account. When the token supply is increased or decreased, it also may update the rebase multiplier if within the appropriate window of time. The role may mint themselves a large amount of tokens outside the window for updating the rebase multiplier, in order to maintain the current worth of the tokens, and use the minted tokens for gain.



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, 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.

Permanent:

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

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

Alleviation

`[CertiK, 02/22/2024]` : The client acknowledges the finding and states that all roles defined in the contract will be controlled by multisig wallets.

We recommend a timelock is used in conjunction with any multisig wallets for the handling of privileged roles. If a privileged account gets compromised and no timelock is used to delay their corresponding operations, the lack of delay will leave no reaction time for the team to take action. We suggest at least 24 hours delay for any sensitive transactions/operations within the protocol.

`[Paxos, 03/01/2024]` : "Paxos uses multi-signature wallets for all roles defined in the contract. Furthermore, Paxos has strict key management processes to protect against unauthorized access. Paxos will consider using a timelock controller for administrative roles. The contract used for multisig can be found here: https://github.com/paxosglobal/simple-multisig"

YBC-01 WRONG MULTIPLIER IS SET BY `increaseRebaseMultiplier()` IF `rebaseRate == 0`

Category	Severity	Location	Status
Logical Issue	● Medium	contracts/YBS.sol (update): 302	● Resolved

Description

```
301         if (rebaseRate == 0) {
302             _setRebaseMultipliers(beforeIncrMult, afterIncrMult, multIncrTime_,
expectedTotalSupply);
303         } else {
304             uint256 afterIncrMult_ = (afterIncrMult * (_BASE + rebaseRate)) /
_BASE;
305             _setRebaseMultipliers(afterIncrMult, afterIncrMult_, multIncrTime_,
expectedTotalSupply);
306         }
```

`beforeIncrMult` should be replaced with `afterIncrMult` to prevent multiplier rolling back.

Scenario

Let's assume:

1. `beforeIncrMult = 1.5e18`
2. `afterIncrMult = 1.7e18`
3. `multIncrTime = 2 am`
4. `rebasePeriod = 1 hour`
5. The current time is 2:30 am.

Then `_getActiveMultiplier()` returns 1.7e18.

But calling `increaseRebaseMultiplier(rebaseRate = 0)` will give:

1. `beforeIncrMult = 1.5e18`
2. `afterIncrMult = 1.7e18`
3. `multIncrTime = 3 am`
4. `rebasePeriod = 1 hours`

Now `_getActiveMultiplier()` returns 1.5e18. The expected value is 1.7e18.

Recommendation

We recommend rewriting the code this way:

```
        uint256 afterIncrMult_ = (afterIncrMult * (_BASE + rebaseRate)) / _BASE;
        _setRebaseMultipliers(afterIncrMult, afterIncrMult_, multIncrTime_,
expectedTotalSupply);
```

So, even if `rebaseRate == 0`, the correct values are set.

Alleviation

`[CertiK, 02/21/2024]`: The client made changes resolving the finding in commit 1bad3a12e607421ec18ad37dc465cd09a15073a0.

YBS-01 | WRONG ARGUMENTS ORDER OF `InsufficientSupply` EVENT

Category	Severity	Location	Status
Inconsistency	● Minor	contracts/YBS.sol (base): 421	● Resolved

Description

`InsufficientSupply` event has `shares` as the second argument and `sharesNeeded` as the third. However, `decreaseSupply()` provides requested shares and existing shares as the second and the third arguments respectively.

Recommendation

We recommend changing the arguments order.

Alleviation

`[CertiK, 02/16/2024]`: The client made changes resolving the finding in commit 1c0dd2630fe45c87104c894106f96c30c7c9cb3b.

YBS-02 | INACCURATE `_fixedShares` UPDATING

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/YBS.sol (base): 730	● Resolved

Description

```
730        _fixedShares[account] = amount;
731        totalFixedShares += amount;
```

`_fixedShares` is assigned, not increased.

`_convertRebaseSharesToFixedShares()` assumes that the user not in `_blocklist` doesn't have any `_fixedShares`.

`_convertFixedSharesToRebaseShares()` assumes that the user in `_blocklist` doesn't have any `_rebaseShares`.

These assumptions reduce the code maintainability and also can be violated. For example, blocking of `SUPPLY_CONTROLLER_ROLE` user can lead to incorrect `totalFixedShares` calculation.

Scenario

Consider the scenarios:

1. Let's assume `supply_controller` has `SUPPLY_CONTROLLER_ROLE` and `asset_protector` has `ASSET_PROTECTION_ROLE`.
2. `asset_protector` calls `blockAccounts(supply_controller)`. All their `_rebaseShares` are converted into `_fixedShares`.
3. `supply_controller` calls `increaseSupply(100)` and gets extra `_rebaseShares`.
4. `asset_protector` calls `blockAccounts(supply_controller)` again. Their `_rebaseShares` are converted into `_fixedShares`. `_fixedShares` is **overwritten** with the new value. `totalFixedShares` is updated as expected.

Recommendation

We recommend increasing of `_fixedShares` / `_rebaseShares` instead of overwriting, or preventing of `increaseSupply()` calls by blocked users.

Alleviation

`[Paxos, 02/15/2024]`: "We've decided to prevent blocked users from calling increaseSupply(), as that allows us to maintain that users cannot hold rebaseShares and fixedShares at the same time."

`[CertiK, 02/22/2024]` : The client made changes resolving the finding in commits 2da2d6e806c360d029fee547cb9b3f5e0a51d715 and f2013f5e6a8523811044c3c58e9dde92d65f4a1f.

YBS-04 | POTENTIAL DIVIDE BY ZERO

Category	Severity	Location	Status
Logical Issue	● Minor	contracts/YBS.sol (base): 861~862, 865~866	● Partially Resolved

Description

Performing division by zero would raise an error and revert the transaction. In fringe cases, the below lines of code can return a value of 0. For instance, if all accounts holding rebase shares have been blocked, then `totalRebaseShares` may be 0. It will also be 0 the first time `increaseSupply()` is called. Additionally, the `value` used may be 0, since in some cases this value is produced by a rounding-down conversion, and in other cases there are no checks on the input provided ensuring it is nonzero. As a result, the denominator may be 0 when both of these conditions are met. Last, the denominator may take on a 0-value when `value * _BASE` is equivalent to `totalRebaseShares * beforeIncrMult`, which may occur if `decreaseSupply()` is called on the entire token worth of `totalRebaseShares`, or if all accounts become blocked.

```
861            (((totalRebaseShares * afterIncrMult) + (value * _BASE)) * beforeIncrMult) /
862            (((totalRebaseShares * beforeIncrMult) + (value * _BASE)));
```

The expression `(((totalRebaseShares * afterIncrMult) + (value * _BASE)) * beforeIncrMult) / (((totalRebaseShares * beforeIncrMult) + (value * _BASE)))` may divide by zero.

```
865            (((totalRebaseShares * afterIncrMult) - (value * _BASE)) * beforeIncrMult) /
866            (((totalRebaseShares * beforeIncrMult) - (value * _BASE)));
```

The expression `(((totalRebaseShares * afterIncrMult) - (value * _BASE)) * beforeIncrMult) / (((totalRebaseShares * beforeIncrMult) - (value * _BASE)))` may divide by zero.

Recommendation

We recommend handling the possible cases which cause the denominator to be zero according to the design intent of the protocol.

Alleviation

`[CertiK, 02/22/2024]` : The client made changes partially resolving the finding in commit e971f109f852f8fa88add189951b5e53b3e01173.

`[Paxos, 03/01/2024]` : "The first fringe case has been addressed in the YBS-09 remediation. While the latter case is technically possible, Paxos has well defined processes for the listed scenarios such that these divide by zero cases will not be encountered."

YBS-09 | MISSING INPUT VALIDATION

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/YBS.sol (base): 387~388, 389~390, 416~417, 418~419, 741~742, 742~743, 745~746	● Resolved

Description

YBS.sol

- Function `increaseSupply()` is missing a check ensuring the input `value` does not correspond to a zero-value `shares` amount. If `value` is nonzero while `shares` is 0, then it may be possible for `afterIncrMult` to be updated to a new value through `_updateAfterIncrMultIfRequired()` while the overall `totalRebaseShares` has not changed.
- Similarly function `decreaseSupply()` is missing a check ensuring the input `value` does not correspond to a zero-value `shares` amount.
- Function `_convertFixedSharesToRebaseShares()` is missing a check for whether a nonzero `_fixedShares amount` corresponds to a zero-value `shares` value when input into `_convertToRebaseShares()` for conversion. If it does, then it may be possible for `afterIncrMult` to be updated to a new value through `_updateAfterIncrMultIfRequired()` while the overall `totalRebaseShares` has not changed.

Recommendation

We recommend adding the relevant checks to the functions cited above.

Alleviation

`[CertiK, 02/21/2024]`: The client made changes resolving the finding in commit e971f109f852f8fa88add189951b5e53b3e01173.

YBS-13 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/YBS.sol (base): 141~142, 142~143, 143~144, 144~145, 145~146	● Resolved

Description

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

Recommendation

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

Alleviation

`[CertiK, 02/21/2024]` : The client made changes resolving the finding in commits 55ae7e3d4b1f6afff828f55627e4a7c0403213a9 and 6267d04f417c85996791691687e0ba86e0bfbc09.

CON-01 | INACCURATE COMMENTS

Category	Severity	Location	Status
Inconsistency	● Informational	contracts/YBS.sol (base): 50, 363~366; contracts/lib/ECRecover.sol (base): 26	● Resolved

Description

Some comments are outdated or inaccurate.

```
25
// unique. Appendix F in the Ethereum Yellow paper
(https://ethereum.github.io/yellowpaper/paper.pdf), defines

26
// the valid range for s in (281): 0 < s < secp256k1n ÷ 2 + 1, and for v in (282): v
∈ {27, 28}. Most
```

In fact, in The Yellow Paper from 2024-01-29 the `s` condition is (303) and the `v` condition (304) allows `v ∈ {0, 1}`. The `ecrecover()` function is described in Appendix E (202).

"BOCKLIST" is supposed to be "BLOCKLIST".

The comment for `isAddrBlockedForReceiving()` is the same as for `isAddrBlocked()`.

Recommendation

We recommend updating the comments.

Alleviation

`[CertiK, 02/22/2024]`: The client made changes resolving the finding in commits aaa0e58025a04f331dfe74f8fc19a91450cd83b4 and 51e931918b27daa3b33caceade03bdf5106d67ab.

EI3-02 | MISSING NATSPEC COMMENTS

Category	Severity	Location	Status
Inconsistency	● Informational	contracts/lib/EIP3009.sol (base): 91~92, 208~209	● Resolved

Description

There are missing comments above the functions `transferWithAuthorization()` and `_transferWithAuthorization()`.

Recommendation

We recommend adding the missing NatSpec comments mentioned above.

Alleviation

`[CertiK. 02/21/2024]`: The client made changes resolving the finding in commit e65c79a7589a9f3098b0bf1ef4b168c11178ec19.

PBA-01 | LACK OF STORAGE GAP IN UPGRADEABLE PARENT CONTRACT

Category	Severity	Location	Status
Logical Issue	● Informational	contracts/lib/PaxosBaseAbstract.sol (base): 10	● Acknowledged

Description

Contract `PaxosBaseAbstract` is an abstract contract that acts as a base for an upgradeable child contract. While it is understood that currently this contract does not implement logic or contain state variables, it may be beneficial to include a storage placeholder at this level in case future upgrades require the need for state variables declared within this parent contract.

Recommendation

We recommend the consideration of a storage gap of a reasonable size within contract `PaxosBaseAbstract` in case new state variables are introduced at this level in future upgrades.

Reference: https://docs.openzeppelin.com/contracts/3.x/upgradeable#storage_gaps

Alleviation

`[Paxos, 03/01/2024]` : "Paxos will use the PaxosBaseAbstract contract to declare only internal and external functions. Paxos has no intention of adding state to this contract in the future."

YBS-03 | `__gap_YBS` SIZE IS INACCURATE

Category	Severity	Location	Status
Inconsistency	● Informational	contracts/YBS.sol (base): 64	● Resolved

Description

```
61         * Expected storage slots used by this contract, 50.
62         * See https://docs.openzeppelin.com/contracts/4.x/upgradeable#storage_gaps
63         */
64        uint256[34] private __gap_YBS; // solhint-disable-line var-name-mixedcase
```

The expected storage size is 49 since the `constant _BASE` doesn't occupy the slot.

Recommendation

We recommend changing the `__gap_YBS` size to 35.

Alleviation

`[CertiK, 02/21/2024]` : The client made changes resolving the finding in commit 2d3de0c200c2b20381066cd3bf06772112627bac.

YBS-14 | DISCREPANCY BETWEEN `totalSupply()` AND THE TOTAL SUM OF ALL BALANCES

Category	Severity	Location	Status
Inconsistency	● Informational	contracts/YBS.sol (base): 318~321, 329~333	● Resolved

Description

The `totalSupply()` is meant to be an invariant representing the sum of all balances of tokens within the contract. A user's balance within the contract is represented by the user's rebase shares, converted to their token worth, summed with their fixed shares:

```
_convertRebaseSharesToTokens(_rebaseShares[account]) +
            _fixedShares[account];
```

while the `totalSupply()` is represented by the total number of rebase shares, converted to their token worth, summed with the total fixed shares:

```
_convertRebaseSharesToTokens(totalRebaseShares) + totalFixedShares;
```

Since `_convertRebaseSharesToTokens()` may round down, it is possible that the sum of all user balances may be less than the returned `totalSupply()` value.

While this may not affect the protocol directly, it should be taken into consideration when composing with other protocols.

Scenario

User A has 299 rebase shares and User B has 1 rebase shares, for a `totalRebaseShares` of 300. Assume for simplicity that `totalFixedShares` is 0. Say that the return of `_getActiveMultiplier()` is 15*1e17.

Then the `totalSupply()` is `(300 * (15*1e17))/1e18` which is 450.

The return of `balanceOf()` for User A is 448 and the return of `balanceOf()` for User B is 1. The sum of these two values is 449.

Recommendation

We recommend keeping this information in consideration when composing with other protocols. Acknowledgment of the finding will resolve it.

Alleviation

`[Paxos, 02/21/2024]` : "Issue acknowledged.This will be made known in documentation."

`[CertiK, 02/22/2024]` : The client made changes resolving the finding in commit ccb3c81508b7e8fdc59a8f498e2dba6705533e11.

YBS-15 | UNBLOCKED ACCOUNTS MAY LOSE DUST AMOUNTS OF PREVIOUS FIXED FUNDS

Category	Severity	Location	Status
Volatile Code	● Informational	contracts/YBS.sol (base): 217~218, 738~739	● Resolved

Description

When an account is blocked, its corresponding `_rebaseShares` value is converted to a `_fixedShares` value, using the current rebase multiplier.

If an account is ever unblocked through `unblockAccounts()` , it is possible that the conversion back to `_rebaseShares` causes the account to lose value. The amount potentially lost is bounded above by `_getActiveMultiplier()/1e18` .

Scenario

User A has a `_rebaseShares` value of 100, and is subsequently blocked with a current rebase multiplier of 1.5*1e18. This converts to a `_fixedShares` value of `(100 * (1.5*1e18))/1e18 = 150` . This is the value that function `balanceOf()` returns for User A while they are blocked.

After a period of time, the rebase multiplier is now updated to 2.1*1e18 and User A is unblocked. Their `_fixedShares` of 150 is converted to `(150 * 1e18)/(2.1*1e18) = 71` . Consequently, the `balanceOf()` function returns 149 for User A instead of 150 now.

Recommendation

We recommend users are made aware of this possibility by including this information within the documentation of the protocol.

Alleviation

`[CertiK, 02/22/2024]` : The client made changes resolving the finding in commit ccb3c81508b7e8fdc59a8f498e2dba6705533e11.

OPTIMIZATIONS | USDL

ID	Title	Category	Severity	Status
CON-02	Redundant Check That `msg.sender` Is Not Blocked In `permit()`	Gas Optimization	Optimization	● Resolved
ECR-01	Use Custom Error Convention	Gas Optimization	Optimization	● Resolved
EI3-01	Arguments Should Be `calldata`	Gas Optimization	Optimization	● Acknowledged
EI7-01	`block.chainid` Can Be Used	Code Optimization	Optimization	● Resolved

CON-02 | REDUNDANT CHECK THAT `msg.sender` IS NOT BLOCKED IN `permit()`

Category	Severity	Location	Status
Gas Optimization	● Optimization	contracts/YBS.sol (base): 685~686; contracts/lib/EIP2612.sol (base): 48~49	● Resolved

Description

Function `permit()` includes in its logic a check ensuring that `isAddrBlocked(msg.sender)` is false. However, this function calls the internal `_approve()` function within the `YBS` contract, which already includes this check at the level of `_beforeApprove()` .

Recommendation

We recommend removing the redundant check from the `permit()` function.

Alleviation

`[CertiK, 02/21/2024]` : The client made changes resolving the finding in commit 312bdd7507754a81937ace615b52afe64370faa0.

ECR-01 | USE CUSTOM ERROR CONVENTION

Category	Severity	Location	Status
Gas Optimization	● Optimization	contracts/lib/ECRecover.sol (base): 37, 41, 46~47	● Resolved

Description

Library file `ECRecover` uses strings for `revert` and in one instance, `require` is used instead of `revert`.

Recommendation

We recommend keeping with the convention of the other files in the protocol by using `revert` in combination with **custom errors** for the cited locations.

Alleviation

`[CertiK, 02/21/2024]`: The client made changes resolving the finding in commit 2dd90a421e61c1321ef318d855d7036091cf7618.

EI3-01 | ARGUMENTS SHOULD BE `calldata`

Category	Severity	Location	Status
Gas Optimization	● Optimization	contracts/lib/EIP3009.sol (base): 92	● Acknowledged

Description

Non-changed arguments of external functions are declared as `memory` .

Recommendation

We recommend declaring the non-changed arguments of external functions as `calldata` where possible to save gas.

Alleviation

`[Paxos, 03/01/2024]` : "This remediation would refactor the batch function to use a struct as the sole input parameter, which would create an inconsistency between the batched and EIP-3009 non-batched function. For this reason, Paxos will not remediate this finding at this time."

EI7-01 | `block.chainid` CAN BE USED

Category	Severity	Location	Status
Code Optimization	● Optimization	contracts/lib/EIP712.sol (base): 28	● Resolved

Description

```
        uint256 chainId;
        // solhint-disable-next-line no-inline-assembly
        assembly {
            chainId := chainid()
        }
```

Since v0.8.0 Solidity allows direct access to `block.chainid`.

Recommendation

We recommend simplifying the code like `bytes32(block.chainid)`.

Alleviation

`[CertiK, 02/16/2024]`: The client made changes resolving the finding in commit a1279e1e9b2eb684dd7e7b0995f15f6aaa7093d2.

APPENDIX | USDL

Finding Categories

Categories	Description
Gas Optimization	Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.
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

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.