Zoo Finance

Security Assessment

CertiK Assessed on Feb 28th, 2025

CertiK Assessed on Feb 28th, 2025

Zoo Finance

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

Executive Summary

TYPES	ECOSYSTEM	METHODS
Vault	Berachain | Ethereum (ETH)	Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 02/28/2025	N/A

CODEBASE	COMMITS
base	d8dc59779bd93e46a5c8387fc53ea2606d359f9c
update_20250217	079e855417628425b2416c23592cd4da353c0d3d
View All in Codebase Page	View All in Codebase Page

Highlighted Centralization Risks

- Transfers can be paused
- Withdraws can be disabled

Vulnerability Summary

12	11	1	0	0	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 Mitigated	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
1	Medium	1 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
6	Minor	6 Resolved	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
4	Informational	4 Resolved	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | ZOO FINANCE

CODEBASE | ZOO FINANCE

Repository

base

update_20250217

Commit

d8dc59779bd93e46a5c8387fc53ea2606d359f9c

079e855417628425b2416c23592cd4da353c0d3d

AUDIT SCOPE | ZOO FINANCE

87 files audited ● 5 files with Mitigated findings ● 7 files with Resolved findings ● 75 files without findings

ID	Repo	File	SHA256 Checksum
PSB	zoofiio/zoo-bribe-vault	contracts/settings/ProtocolSettings.sol	fc96809f0ad3bf52818b043d4cde4ee551980226995e093d2f5b68b931f26293
PTB	zoofiio/zoo-bribe-vault	contracts/tokens/PToken.sol	34216b079038fd6cb643c4d38be898a12c7cabfaacee4a4756265b75f2a8d623
VAL	zoofiio/zoo-bribe-vault	contracts/vaults/Vault.sol	3ce1a9b3b33497838921842cd0d34eb49d7d632fba1ea04e4c7d7c2e3dd55574
BQB	zoofiio/zoo-bribe-vault	contracts/BQuery.sol	b5b3b2ae367eb24192e18d99c2be2dedbd5ce2ce0e9bb164163bc1195b528d55
ZPB	zoofiio/zoo-bribe-vault	contracts/ZooProtocol.sol	44de784488f3268d46623c3cfd6177a8c0dc1264845fe6ec22adfa23f047281e
ABP	zoofiio/zoo-bribe-vault	contracts/bribes/AdhocBribesPool.sol	dc0318585c2b683c5ffb02e1e527efcf18dc4b2251461a0ae83a9562c49ab41e
SBP	zoofiio/zoo-bribe-vault	contracts/bribes/StakingBribesPool.sol	b13cd3291df5c699e414bc7389c5894d16867d7ae28a778767c9b53dd37121e8
CON	zoofiio/zoo-bribe-vault	contracts/libs/Constants.sol	450349f749d32529f43793620f3955f64060d03964e2b40eff277eb75b0bd4f8
VCB	zoofiio/zoo-bribe-vault	contracts/libs/VaultCalculator.sol	5437c5fa7c8f6047ac3a1b7bcfd9f60f3264f8171c106ca69477f2ccb6669a55
POB	zoofiio/zoo-bribe-vault	contracts/settings/ProtocolOwner.sol	698c7178f0393284cac8b5d30f1876cc553a55f94eb95e9099c4737a28b422a5
ERC	zoofiio/zoo-bribe-vault	contracts/vaults/ERC4626BribeVault.sol	71f5287fdda1b5f331c891926dc75714bca2cb680cc37c9f6ae60db67d6ab107
RPB	zoofiio/zoo-bribe-vault	contracts/vaults/RedeemPool.sol	c5b32ea5b511a8bc7137fbca8b31f283fac0bbb2d4725bdfdaf3df7acab4ba0a
BPF	zoofiio/zoo-bribe-vault	contracts/bribes/BribesPoolFactory.sol	202f612b9482340a5d3c40b385e6d129cfc692fa818b2bf0d48c027125f5b7b8

ID	Repo	File	SHA256 Checksum
IBP	zoofiio/zoo-bribe-vault	contracts/interfaces/IBribesPool.sol	c2c4585515de4fcf98c808bc946d64004a6db9ad9ba6fcbf48f35601bd7d5f93
IBF	zoofiio/zoo-bribe-vault	contracts/interfaces/IBribesPoolFactory.sol	5b0ae6b816d17813ba5e27488e9a2603eb89c1370f9d50444d2ca9dd650b0a52
IIB	zoofiio/zoo-bribe-vault	contracts/interfaces/IInfrared.sol	b4e476fa7859346d3b0c12b6bdf29a9264fb2b94e52d7cb25d7145fd1ee5d834
IPT	zoofiio/zoo-bribe-vault	contracts/interfaces/IPToken.sol	bbfe671a581aea5db4b5a1286cb2c8ca014243e8dddc8a0027d0e2c5835c3825
IPS	zoofiio/zoo-bribe-vault	contracts/interfaces/IProtocolSettings.sol	e26f742d4c48f5357f7879ec16f58be8b067ef244b847637717f076f081ae1a2
IRP	zoofiio/zoo-bribe-vault	contracts/interfaces/IRedeemPool.sol	5a6b17d3a7a5ae7e36d371183d270e0d55d26b6e464de537bc35ffcb36929477
IRF	zoofiio/zoo-bribe-vault	contracts/interfaces/IRedeemPoolFactory.sol	6f789af36250a697dbc5f3f3366d6eda576a79ed34962b18db4b78198d24327c
ISP	zoofiio/zoo-bribe-vault	contracts/interfaces/IStakingPool.sol	cbf16c38485fd1cd900721b7f42bca9a07aa3e2ea6f0b76d0ca780100722875e
IVB	zoofiio/zoo-bribe-vault	contracts/interfaces/IVault.sol	ad13837dfe46ce4880de5f1e314b8ecb743565da314162be6712de7e3b568b0b
IYT	zoofiio/zoo-bribe-vault	contracts/interfaces/IYeetTrifectaVault.sol	88b3f3a0021d7ac81fc4443ee93e337ae1563b031c5e3bbba0a930790b1476a4
IZP	zoofiio/zoo-bribe-vault	contracts/interfaces/IZooProtocol.sol	495fab3f90865f79ee732651275446b0e3abcbabeadc354643cb6be1a0161552
CLB	zoofiio/zoo-bribe-vault	contracts/libs/CalcLiq.sol	19fb22adb5b988ab333859ccd8415d4259e3fe4177a5b89fef8e0dc687e60d52
TTB	zoofiio/zoo-bribe-vault	contracts/libs/TokensTransfer.sol	f14c63db9f058f62be460811e3908120f13aebb486e61c0a4333a9fc465cace1
BEB	zoofiio/zoo-bribe-vault	contracts/vaults/BriberExtension.sol	e5938caaddda0934472f54a77d214cf167726f6d414714699908b8a0de206de4
IBV	zoofiio/zoo-bribe-vault	contracts/vaults/InfraredBribeVault.sol	6db78ab7505e3a8a1a8e9b8700e8d45efaa0572fc70914fc08ef07b211f7dc6d

ID	Repo	File	SHA256 Checksum
RPF	zoofiio/zoo-bribe-vault	contracts/vaults/RedeemPoolFactory.sol	49e74367445ad645a9d7ffe5fcf679847aa80d279718374375e2cc82af9b46ae
ADH	zoofiio/zoo-bribe-vault	contracts/bribes/AdhocBribesPool.sol	0d771091eb124774df9400a122cfec5fcbb5c7b3fd705c2a0f8c1601d73cc0f6
BRE	zoofiio/zoo-bribe-vault	contracts/bribes/BribesPoolFactory.sol	202f612b9482340a5d3c40b385e6d129cfc692fa818b2bf0d48c027125f5b7b8
STA	zoofiio/zoo-bribe-vault	contracts/bribes/StakingBribesPool.sol	766b0fc4a1ec8d048eb60c16e6af0cff6b518610793345c233026ea1e8120c5c
IBR	zoofiio/zoo-bribe-vault	contracts/interfaces/IBribesPool.sol	c2c4585515de4fcf98c808bc946d64004a6db9ad9ba6fcbf48f35601bd7d5f93
IPF	zoofiio/zoo-bribe-vault	contracts/interfaces/IBribesPoolFactory.sol	5b0ae6b816d17813ba5e27488e9a2603eb89c1370f9d50444d2ca9dd650b0a52
IIU	zoofiio/zoo-bribe-vault	contracts/interfaces/IInfrared.sol	b4e476fa7859346d3b0c12b6bdf29a9264fb2b94e52d7cb25d7145fd1ee5d834
IPO	zoofiio/zoo-bribe-vault	contracts/interfaces/IPToken.sol	bbfe671a581aea5db4b5a1286cb2c8ca014243e8dddc8a0027d0e2c5835c3825
IPR	zoofiio/zoo-bribe-vault	contracts/interfaces/IProtocolSettings.sol	e26f742d4c48f5357f7879ec16f58be8b067ef244b847637717f076f081ae1a2
IRE	zoofiio/zoo-bribe-vault	contracts/interfaces/IRedeemPool.sol	5a6b17d3a7a5ae7e36d371183d270e0d55d26b6e464de537bc35ffcb36929477
IRD	zoofiio/zoo-bribe-vault	contracts/interfaces/IRedeemPoolFactory.sol	6f789af36250a697dbc5f3f3366d6eda576a79ed34962b18db4b78198d24327c
IST	zoofiio/zoo-bribe-vault	contracts/interfaces/IStakingPool.sol	cbf16c38485fd1cd900721b7f42bca9a07aa3e2ea6f0b76d0ca780100722875e
IVU	zoofiio/zoo-bribe-vault	contracts/interfaces/IVault.sol	29477f8f8ab6966d17d14e892f65d256a9a4e5fa158add50999425f5c75490fa
IYV	zoofiio/zoo-bribe-vault	contracts/interfaces/IYeetTrifectaVault.sol	88b3f3a0021d7ac81fc4443ee93e337ae1563b031c5e3bbba0a930790b1476a4
IZO	zoofiio/zoo-bribe-vault	contracts/interfaces/IZooProtocol.sol	495fab3f90865f79ee732651275446b0e3abcbabeadc354643cb6be1a0161552

ID	Repo	File	SHA256 Checksum
CLU	zoofiio/zoo-bribe-vault	contracts/libs/CalcLiq.sol	19fb22adb5b988ab333859ccd8415d4259e3fe4177a5b89fef8e0dc687e60d52
COS	zoofiio/zoo-bribe-vault	contracts/libs/Constants.sol	450349f749d32529f43793620f3955f64060d03964e2b40eff277eb75b0bd4f8
TTU	zoofiio/zoo-bribe-vault	contracts/libs/TokensTransfer.sol	f14c63db9f058f62be460811e3908120f13aebb486e61c0a4333a9fc465cace1
VCU	zoofiio/zoo-bribe-vault	contracts/libs/VaultCalculator.sol	f81a4e640eedc273212d0450ee383e38400d7937937717399e6a351c7f1e391f
POU	zoofiio/zoo-bribe-vault	contracts/settings/ProtocolOwner.sol	f7234c8db7a0b58b512934e96ca0bf1f2d2d08a75b8c4198f8d44b676fb82777
PSU	zoofiio/zoo-bribe-vault	contracts/settings/ProtocolSettings.sol	24af7810d9fc64375f389769bb029608d27608302c75b28adf1a2edcc5d568de
PTU	zoofiio/zoo-bribe-vault	contracts/tokens/PToken.sol	a44defc6ed52d90d0acb768bcdadbeb752f375d89e1b22f9157873ca6fac9205
BEU	zoofiio/zoo-bribe-vault	contracts/vaults/BriberExtension.sol	e5938caaddda0934472f54a77d214cf167726f6d414714699908b8a0de206de4
ERB	zoofiio/zoo-bribe-vault	contracts/vaults/ERC4626BribeVault.sol	812df528ef42778e3e016c108c0874d0ea229dba1f5e0ecd84bb00834abe4e80
INF	zoofiio/zoo-bribe-vault	contracts/vaults/InfraredBribeVault.sol	2152d36ae5020d01fd9ebea7dad73e99e40a0e8880061b8abee8cd0fea019764
RPU	zoofiio/zoo-bribe-vault	contracts/vaults/RedeemPool.sol	b73d7febeca2a0887802a2c5096ed94f46ceee7d5dfc0113fdfb5004d232de35
RED	zoofiio/zoo-bribe-vault	contracts/vaults/RedeemPoolFactory.sol	49e74367445ad645a9d7ffe5fcf679847aa80d279718374375e2cc82af9b46ae
VAV	zoofiio/zoo-bribe-vault	contracts/vaults/Vault.sol	f4b13fa2d009033ffa264850ea4ad2ab99b3fa70d0a5709c438407afcfa666f5
BQU	zoofiio/zoo-bribe-vault	contracts/BQuery.sol	3dfd664968475a8349edaf7a6d5fce91c075be8d4100c2942daf64198cb71c7c
ZPU	zoofiio/zoo-bribe-vault	contracts/ZooProtocol.sol	44de784488f3268d46623c3cfd6177a8c0dc1264845fe6ec22adfa23f047281e

ID	Repo	File	SHA256 Checksum
ADO	zoofiio/zoo-bribe-vault	contracts/bribes/AdhocBribesPool.sol	e6857836c0fa016d127b56c8a22ddfb5d97dde89d1a36b5de5eeeaebda7f6d6e
BRP	zoofiio/zoo-bribe-vault	contracts/bribes/BribesPoolFactory.sol	202f612b9482340a5d3c40b385e6d129cfc692fa818b2bf0d48c027125f5b7b8
STK	zoofiio/zoo-bribe-vault	contracts/bribes/StakingBribesPool.sol	8ddd48d379f209ed0e302c91a13f2dca5e73c61f83de5961060be959fe444e39
IBI	zoofiio/zoo-bribe-vault	contracts/interfaces/IBribesPool.sol	c2c4585515de4fcf98c808bc946d64004a6db9ad9ba6fcbf48f35601bd7d5f93
IBB	zoofiio/zoo-bribe-vault	contracts/interfaces/IBribesPoolFactory.sol	5b0ae6b816d17813ba5e27488e9a2603eb89c1370f9d50444d2ca9dd650b0a52
IIH	zoofiio/zoo-bribe-vault	contracts/interfaces/IInfrared.sol	b4e476fa7859346d3b0c12b6bdf29a9264fb2b94e52d7cb25d7145fd1ee5d834
IPK	zoofiio/zoo-bribe-vault	contracts/interfaces/IPToken.sol	bbfe671a581aea5db4b5a1286cb2c8ca014243e8dddc8a0027d0e2c5835c3825
IPC	zoofiio/zoo-bribe-vault	contracts/interfaces/IProtocolSettings.sol	e26f742d4c48f5357f7879ec16f58be8b067ef244b847637717f076f081ae1a2
IRM	zoofiio/zoo-bribe-vault	contracts/interfaces/IRedeemPool.sol	5a6b17d3a7a5ae7e36d371183d270e0d55d26b6e464de537bc35ffcb36929477
IRO	zoofiio/zoo-bribe-vault	contracts/interfaces/IRedeemPoolFactory.sol	6f789af36250a697dbc5f3f3366d6eda576a79ed34962b18db4b78198d24327c
ISA	zoofiio/zoo-bribe-vault	contracts/interfaces/IStakingPool.sol	cbf16c38485fd1cd900721b7f42bca9a07aa3e2ea6f0b76d0ca780100722875e
IVH	zoofiio/zoo-bribe-vault	contracts/interfaces/IVault.sol	29477f8f8ab6966d17d14e892f65d256a9a4e5fa158add50999425f5c75490fa
ITV	zoofiio/zoo-bribe-vault	contracts/interfaces/IYeetTrifectaVault.sol	88b3f3a0021d7ac81fc4443ee93e337ae1563b031c5e3bbba0a930790b1476a4
IZR	zoofiio/zoo-bribe-vault	contracts/interfaces/IZooProtocol.sol	495fab3f90865f79ee732651275446b0e3abcbabeadc354643cb6be1a0161552
CLH	zoofiio/zoo-bribe-vault	contracts/libs/CalcLiq.sol	19fb22adb5b988ab333859ccd8415d4259e3fe4177a5b89fef8e0dc687e60d52

ID	Repo	File	SHA256 Checksum
COT	zoofiio/zoo-bribe-vault	contracts/libs/Constants.sol	f55e593b8ccbeedc05f108f14b694de9f1a219cb0779202d6e6d43b6ca0ead55
TTH	zoofiio/zoo-bribe-vault	contracts/libs/TokensTransfer.sol	f14c63db9f058f62be460811e3908120f13aebb486e61c0a4333a9fc465cace1
VCH	zoofiio/zoo-bribe-vault	contracts/libs/VaultCalculator.sol	a0aea8356e2adf0d8b92aca3a3f1cf21589bffb6de57565377c2c2fb144ab397
POH	zoofiio/zoo-bribe-vault	contracts/settings/ProtocolOwner.sol	ddfd817e1a6767b91088029a4727864de749c4b9de88a8ce943ea0b6e2d25872
PSH	zoofiio/zoo-bribe-vault	contracts/settings/ProtocolSettings.sol	6b9d69929c58486babe0a173cdc41b8fcdfd18762d88ff500d48021e433e675c
PTH	zoofiio/zoo-bribe-vault	contracts/tokens/PToken.sol	e27ac0c455c35008e042206446d1d7bbc04efba59f14a0b78fd7b0284d231672
BEH	zoofiio/zoo-bribe-vault	contracts/vaults/BriberExtension.sol	e5938caaddda0934472f54a77d214cf167726f6d4147146999908b8a0de206de4
ERV	zoofiio/zoo-bribe-vault	contracts/vaults/ERC4626BribeVault.sol	b1940759a79422732a2de758c1fb3a57ac30be8c47b90d4f59490f08745064ee
INA	zoofiio/zoo-bribe-vault	contracts/vaults/InfraredBribeVault.sol	2152d36ae5020d01fd9ebea7dad73e99e40a0e8880061b8abee8cd0fea019764
RPH	zoofiio/zoo-bribe-vault	contracts/vaults/RedeemPool.sol	62d0ed1b12a42efb50b1965f4cc4e094c03cbc8e18d3146fcb5557e57e16005f
REE	zoofiio/zoo-bribe-vault	contracts/vaults/RedeemPoolFactory.sol	49e74367445ad645a9d7ffe5fcf679847aa80d279718374375e2cc82af9b46ae
VAA	zoofiio/zoo-bribe-vault	contracts/vaults/Vault.sol	21d8cd15028d2585085af241800621e7895dd6e8761a9fb70e075c3736a755f0
BQH	zoofiio/zoo-bribe-vault	contracts/BQuery.sol	3dfd664968475a8349edaf7a6d5fce91c075be8d4100c2942daf64198cb71c7c
ZPH	zoofiio/zoo-bribe-vault	contracts/ZooProtocol.sol	44de784488f3268d46623c3cfd6177a8c0dc1264845fe6ec22adfa23f047281e

APPROACH & METHODS | ZOO FINANCE

This report has been prepared for Zoo Finance to discover issues and vulnerabilities in the source code of the Zoo Finance 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 | ZOO FINANCE

12	0	1	1	6	4
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Zoo Finance. Through this audit, we have uncovered 12 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
CON-02	**Centralization Risks**	**Centralization**	**Major**	**Mitigated**
VAL-01	Memory Data Not Save On Chain	Logical Issue	Medium	Resolved
BRI-02	Possible Run Out Of Gas Due To Iteration Over Excessive `_bribeTokens`	Denial of Service	Minor	Resolved
CON-03	Missing Zero Address Validation	Volatile Code	Minor	Resolved
PTB-02	Allowing Token Name And Symbol Changes After Deployment	Logical Issue	Minor	Resolved
RPB-01	Invalid Use Of Access Control Modifier	Logical Issue	Minor	Resolved
RPB-02	`withdrawRedeem()` Rounds In Favor Of User	Financial Manipulation	Minor	Resolved
VAL-02	Deprecated Usage Of `Counters.sol`	Logical Issue	Minor	Resolved
ERC-01	Using Unlimited Token Approval In Constructor Is A Bad Practice	Coding Style	Informational	Resolved
POB-01	Redundant Code Components	Volatile Code	Informational	Resolved
PSB-01	Usage Of Hardhat/Foundry's Console	Coding Issue	Informational	Resolved

ID	Title	Category	Severity	Status
VCB-01	Redundant Safe Math	Volatile Code	Informational	Resolved

CON-02 | CENTRALIZATION RISKS

Category	Severity	Location	Status
Centralization	Major	contracts/BQuery.sol (base): 288, 296; contracts/ZooProtocol.sol (base): 29; contracts/settings/ProtocolSettings.sol (base): 91, 100, 121; contracts/tokens/PToken.sol (base): 125, 129; contracts/vaults/Vault.sol (base): 236, 257, 261, 265, 271, 277, 281, 285, 289	Mitigated

Description

The `_owner` role has significant authority across multiple contracts, giving it the ability to alter key aspects of the protocol's functionality. If the `_owner` account is compromised, a hacker could take advantage of this access to perform malicious actions. These actions could include modifying liquidity provider (LP) status, changing the address of the `CrocQuery` contract, or executing other critical operations like adding or updating `vaults`, managing the `treasury`, or pausing/unpausing specific features such as `redeem pools`. The hacker could also manipulate token details like the `name` and `symbol`, or modify `vault` and `bribes` pools, thus compromising the integrity of the system. This highlights the importance of securing the `_owner` account, as any loss of control could lead to severe consequences, including unauthorized fund transfers, system disruptions, or loss of liquidity.

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

[**Zoo Finance Team** 13 Feb 2025]: Issue acknowledged. I won't make any changes for the current version. The plan is to transfer the ownership to a multisig account in the future, like a contract account created via https://safe.global/.

[**Zoo Finance Team** 17 Feb 2025]: Issue acknowledged. I will fix the issue in the future, which will not be included in this audit engagement.

[**Certik team** 28 Feb 2025]: The ZooProtocol contract has been successfully deployed on Berachain at:
0x4737c3bab13a1ad94ede8b46bc6c22fb8bbe9c81

The new owner is assigned to **GnosisSafeProxy**, located at: **0x5C9B9C19ccaC7925157Cc60aCc289e78839c5D70**, which is part of the multi-signature wallet **Good Berachain Safe**

Multi-Signature Wallet Details: Threshold: 2 out of 3 signers required.

Signers:

- berachain:0x69b29756A231F771cc59E1CCA2A0ebb313F70F49
- berachain:0x1851CbB368C7c49B997064086dA94dBAD90eB9b5
- berachain:0xD0354daC1ca9418B1daBd6D1e5e55935d2a6eFB8

VAL-01 | MEMORY DATA NOT SAVE ON CHAIN

Category	Severity	Location	Status
Logical Issue	Medium	contracts/vaults/Vault.sol (base): 241	Resolved

Description

The `currentEpoch` is defined as a memory-type struct. Consequently, any changes to its internal attribute values are not stored on the blockchain. As a result, these changes are ineffective. In certain situations, this could lead to significant issues, including inaccurate balance recordings.

Recommendation

We recommend reconsidering the use of this modifier. If the intention is to update data on the blockchain, we suggest using the storage modifier.

Alleviation

[**Zoo Finance Team** 13 Feb 2025]: Issue acknowledged. Changes have been reflected in the commit 4f682e38.

BRI-02 POSSIBLE RUN OUT OF GAS DUE TO ITERATION OVER EXCESSIVE `_bribeTokens`

Category	Severity	Location	Status
Denial of Service	Minor	contracts/bribes/AdhocBribesPool.sol (base): 91, 137; contracts/bribes/StakingBribesPool.sol (base): 62, 109	Resolved

Description

The issue arises from the `updateAllBribes` modifier, which iterates over all `_bribeTokens` and calls `_updateBribes` for each token before executing the function logic. Since `getBribes` also applies the `updateAllBribes` modifier and then iterates over `_bribeTokens` again, this leads to two consecutive loops over the same data set in a single transaction. If `_bribeTokens` contains a large number of entries, the transaction may run out of gas, making `updateAllBribes` and `getBribes` impractical to execute. This design limits scalability and usability, especially in environments with high gas constraints.

Recommendation

To mitigate the issue, you can limit the number of `_bribeTokens` that can be processed in a single transaction. By imposing a cap on the maximum number of tokens iterated over in the `updateAllBribes` modifier and the `getBribes` function, you can prevent gas exhaustion. If the number of tokens exceeds this limit, the contract can throw an error.

Alleviation

[**Zoo Finance Team** 17 Feb 2025]: Normally, there should be limited number of bribe tokens, since: Bribe tokens are synced from Infrared Staking Pool contract, or added by Admin. Not added by normal users. The BribesPool contract only lives for an epoch like 30 days. But anyway, just in case there are too many bribe tokens, a new `getBribe(address bribeToken)` function is added to the contract, so that users are still able to claim bribes in case `getBribes()` runs out of gas. Issue acknowledged. Changes have been reflected in the commit hash.

CON-03 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	Minor	contracts/BQuery.sol (base): 297; contracts/bribes/AdhocBribesPool.sol (base): 38; contracts/bribes/StakingBribesPool.sol (base): 34; contracts/settings/ProtocolSettings.sol (base): 31	Resolved

Description

The cited address input is missing a check that it is not `address(0)` .

Recommendation

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

Alleviation

[**Zoo Finance Team** 13 Feb 2025]: Issue acknowledged. Changes have been reflected in the commit 4f682e38.

PTB-02 ALLOWING TOKEN NAME AND SYMBOL CHANGES AFTER DEPLOYMENT

Category	Severity	Location	Status
Logical Issue	Minor	contracts/tokens/PToken.sol (base): 125~131	Resolved

Description

The issue lies in the functions `setName()` and `setSymbol()`, which allow the token's name and symbol to be changed after the token has been deployed. While these functions are restricted to the owner of the contract, enabling such changes post-deployment can create potential risks, such as confusion or misleading information for users interacting with the token. Token names and symbols are often considered immutable once a token is deployed, and allowing them to be changed could undermine trust or violate expectations in decentralized applications or other systems relying on the token's identity.

Recommendation

We recommend making the token's `name` and `symbol` immutable after deployment to prevent any changes post-launch. This can be achieved by removing the setter functions `setName()` and `setSymbol()`

Alleviation

[**Zoo Finance Team** 17 Feb 2025]: Issue acknowledged. Changes have been reflected in the commit hash.

RPB-01 | INVALID USE OF ACCESS CONTROL MODIFIER

Category	Severity	Location	Status
Logical Issue	Minor	contracts/vaults/RedeemPool.sol (base): 75~76, 80~81, 91~92, 95~96	Resolved

Description

The functions marked as `view` or `pure` are unnecessarily restricted by the `onlyBeforeSettlement` modifier. These functions are designed to be read-only, meaning they do not modify the state on the blockchain. However, they are restricted so that only a specific address can call them.

It's important to note that even private state variables can be read off-chain, rendering the access restriction on these functions ineffective.

Recommendation

We recommend that access restrictions are not used on `view` or `pure` functions, as they do not improve security for read-only operations. Instead, these getter functions should be made public to allow transparency and follow best practice. If there is sensitive information that should not be disclosed, the way in which this data is managed and stored should be reconsidered, as restricting access in this way does not provide effective security.

Alleviation

[**Zoo Finance Team** 13 Feb 2025]: I'd like to keep the `onlyBeforeSettlement` modifier for these view/pure functions, since they are only meaningful before settlement. When a RedeemPool contract is settled on epoch end, all user deposited PToken held by the contract will be burned, and user could not deposit or withdraw PTokens any more. They could only withdraw the distributed asset tokens. The `onlyBeforeSettlement` modifier ensures these `view` / `pure` functions are not mis-called after settlement.

RPB-02 | `withdrawRedeem()` ROUNDS IN FAVOR OF USER

Category	Severity	Location	Status
Financial Manipulation	Minor	contracts/tokens/PToken.sol (base): 158; contracts/vaults/RedeemPool.sol (base): 123	Resolved

Description

```
123  uint256 sharesAmount = getRedeemingSharesByBalance(amount);
124  _totalRedeemingShares = _totalRedeemingShares - sharesAmount;
125
_userRedeemingShares[_msgSender()] = _userRedeemingShares[_msgSender()] -
sharesAmount;
```

`RedeemPool.getRedeemingSharesByBalance()` uses `_convertToShares()` which rounds down. The resulting `sharesAmount` is subtracted from the `_userRedeemingShares`. As a result, the rounding is done in favor of the user.

`PToken.burn()` also performs `getSharesByBalance()` which rounds down. PToken also doesn't use decimals offset, making it vulnerable to inflation attack. `PToken.burn()` is called by `_redeemOnClose()` in `InfraredBribeVault` and `ERC4626BribeVault`.

Recommendation

We recommend always rounding in favor of project.

Alleviation

[**Zoo Finance Team** 17 Feb 2025]: Issue acknowledged. Changes have been reflected in the commit hash.

VAL-02 | DEPRECATED USAGE OF `Counters.sol`

Category	Severity	Location	Status
Logical Issue	Minor	contracts/vaults/Vault.sol (base): 29	Resolved

Description

The linked contracts import and use OpenZeppelin's `Counters` contract. OpenZeppelin has deprecated the usage of the `Counters` contract: https://github.com/OpenZeppelin/openzeppelin-contracts/issues/4233

Recommendation

Consider removing the usage of deprecated 3rd party contracts.

Alleviation

[**Zoo Finance Team** 13 Feb 2025]: Issue acknowledged. Changes have been reflected in the commit 4f682e38.

ERC-01 USING UNLIMITED TOKEN APPROVAL IN CONSTRUCTOR IS A BAD PRACTICE

Category	Severity	Location	Status
Coding Style	Informational	contracts/vaults/ERC4626BribeVault.sol (base): 26	Resolved

Description

The issue lies in the use of `IERC20(assetToken).approve(address(stakingPool), type(uint256).max)` within the constructor. Approving an unlimited amount of tokens for a contract from the start is considered a bad practice, as it exposes the contract to potential risks if the approved contract is later compromised or behaves unexpectedly.

Recommendation

We recommend approving a specific amount of tokens before transferring them, rather than approving an unlimited amount. This can be done by calling `IERC20(assetToken).approve(address(stakingPool), amount)` before using `TokensTransfer.transferTokens` to transfer the tokens. By approving only the necessary amount, you limit potential exposure to risks and improve the overall security of the contract.

Alleviation

[**Zoo Finance Team** 17 Feb 2025]: Issue acknowledged. Changes have been reflected in the commit hash.

POB-01 | REDUNDANT CODE COMPONENTS

Category	Severity	Location	Status
Volatile Code	Informational	contracts/settings/ProtocolOwner.sol (base): 15	Resolved

Description

The linked statements do not affect the functionality of the codebase and appear to be either leftovers from test code or older functionality.

Recommendation

We advise to remove the redundant statements for production environments.

Alleviation

[**Zoo Finance Team** 13 Feb 2025]: Issue acknowledged. Changes have been reflected in the commit 4f682e38.

PSB-01 | USAGE OF HARDHAT/FOUNDRY'S CONSOLE

Category	Severity	Location	Status
Coding Issue	Informational	contracts/settings/ProtocolSettings.sol (base): 2~4	Resolved

Description

The current contract code includes references to the `console` contract from Hardhat/Foundry. It is important to note that the `console` contract is exclusively intended for development and testing purposes, providing developers with debugging capabilities throughout the testing phase.

While the `console` code may be commented out and thus inactive, its presence within the contract codebase can still negatively impact the overall tidiness and cleanliness of the contract. Comments intended for developers' understanding and debugging purposes should not be present in the final production-ready version of the smart contract.

Recommendation

To mitigate this issue, it is strongly advised to remove all instances of the `console` contract and any related `console.log` statements from the smart contract code before deployment to any public or production network.

Alleviation

[**Zoo Finance Team** 13 Feb 2025]: Issue acknowledged. Changes have been reflected in the commit 4f682e38.

VCB-01 | REDUNDANT SAFE MATH

Category	Severity	Location	Status
Volatile Code	Informational	contracts/libs/VaultCalculator.sol (base): 8	Resolved

Description

Solidity version >=0.8.0 includes checked arithmetic operations and underflow/overflow by default, making SafeMath redundant.

Recommendation

We recommend removing the SafeMath library and use standard arithmetic operators to reduce code complexity.

Alleviation

[**Zoo Finance Team** 13 Feb 2025]: Issue acknowledged. Changes have been reflected in the commit 4f682e38.

OPTIMIZATIONS | ZOO FINANCE

ID	Title	Category	Severity	Status
BRI-01	Redundant `updateBribes(address(0), bribeToken)` Call Due To Conditional Check	Code Optimization	Optimization	Resolved
CON-01	Variables That Could Be Declared As Immutable	Gas Optimization	Optimization	Resolved
LIB-01	Unnecessary Struct Usage For Temporary Calculation Results	Gas Optimization, Code Optimization	Optimization	Resolved
POB-02	Unnecessary Casting Of Protocol When Calling `protocolOwner()`	Code Optimization	Optimization	Resolved
PSB-02	Redundant `param.length == 0` Check In `isValidParam()` For Bytes32 Parameter	Code Optimization	Optimization	Resolved
PTB-01	Redundant Check For `_totalShares == 0` In Division Operation	Code Optimization	Optimization	Resolved
VAL-03	Redundant Use Of `_allEpochIds` For Epoch Tracking And Inefficient Access Methods	Code Optimization, Gas Optimization	Optimization	Resolved
VCB-02	Redundant Assignment Of `deltaT = 0`	Gas Optimization	Optimization	Resolved

BRI-01 REDUNDANT `updateBribes(address(0), bribeToken)` CALL DUE TO CONDITIONAL CHECK

Category	Severity	Location	Status
Code Optimization	Optimization	contracts/bribes/AdhocBribesPool.sol (base): 132; contracts/bribes/StakingBribesPool.sol (base): 85	Resolved

Description

The issue arises from the usage of `updateBribes(address(0), bribeToken)`, which is redundant because the `updateBribes` modifier internally calls `_updateBribes(user, bribeToken)`, and `_updateBribes` has a conditional check `(if (user != address(0)))`. Since user is explicitly set to `address(0)`, the function logic inside `_updateBribes` will be skipped, making the modifier call unnecessary and potentially misleading in the code.

Recommendation

To resolve this issue, simply remove the `updateBribes(address(0), bribeToken)` modifier call where it is unnecessary.

Alleviation

[**Zoo Finance Team** 17 Feb 2025]: Issue acknowledged. Changes have been reflected in the commit hash.

CON-01 | VARIABLES THAT COULD BE DECLARED AS IMMUTABLE

Category	Severity	Location	Status
Gas Optimization	Optimization	contracts/tokens/PToken.sol (base): 23; contracts/vaults/RedeemPool.sol (base): 27, 28	Resolved

Description

The linked variables assigned in the constructor can be declared as `immutable`. Immutable state variables can be assigned during contract creation but will remain constant throughout the lifetime of a deployed contract. A big advantage of immutable variables is that reading them is significantly cheaper than reading from regular state variables since they will not be stored in storage.

Recommendation

We recommend declaring these variables as immutable. Please note that the `immutable` keyword only works in Solidity version `v0.6.5` and up.

Alleviation

[**Zoo Finance Team** 13 Feb 2025]: Issue acknowledged. Changes have been reflected in the commit 4f682e38.

LIB-01 UNNECESSARY STRUCT USAGE FOR TEMPORARY CALCULATION RESULTS

Category	Severity	Location	Status
Gas Optimization, Code Optimization	Optimization	contracts/libs/Constants.sol (base): 27~37; contracts/libs/VaultCalculator.sol (base): 116~128	Resolved

Description

The issue arises from the unnecessary use of the Terms struct to store temporary calculation results. Instead of using `T.T1`, `T.T2`, and `T.T3` within the struct, the same values can be assigned directly to local `uint256` variables. This avoids unnecessary memory allocations and improves gas efficiency by reducing storage operations.

Recommendation

We recommend replacing `T.T1`, `T.T2`, and `T.T3` with simple `uint256` variables, making the computations more efficient and straightforward. Additionally, the `Terms` struct should be removed from the definition, as it is unnecessary for storing temporary calculation results.

Alleviation

[**Zoo Finance Team** 17 Feb 2025]: Issue acknowledged. Changes have been reflected in the commit hash.

POB-02 UNNECESSARY CASTING OF PROTOCOL WHEN CALLING `protocolOwner()`

Category	Severity	Location	Status
Code Optimization	Optimization	contracts/settings/ProtocolOwner.sol (base): 21, 26	Resolved

Description

The issue arises from the unnecessary casting of protocol as `IZooProtocol` when calling `protocol.protocolOwner()` in the contract. Since `protocol` is already defined as `IZooProtocol` in the constructor, there is no need to cast it again when invoking its functions. The correct and more efficient approach would be to directly call `protocol.protocolOwner()`, eliminating the redundant casting and simplifying the code. This change improves readability and reduces unnecessary operations, making the contract more efficient.

Recommendation

To mitigate this issue, we recommend removing the unnecessary casting of `protocol` in the function calls. Instead of using `IZooProtocol(protocol).protocolOwner()`, simply call `protocol.protocolOwner()`.

Alleviation

[**Zoo Finance Team** 17 Feb 2025]: Issue acknowledged. Changes have been reflected in the commit hash.

PSB-02 REDUNDANT `param.length == 0` CHECK IN `isValidParam()` FOR BYTES32 PARAMETER

Category	Severity	Location	Status
Code Optimization	Optimization	contracts/settings/ProtocolSettings.sol (base): 59	Resolved

Description

In the `isValidParam()` function, the condition `param.length == 0` is redundant because `bytes32` is a fixed-size type, always occupying `32` bytes in memory. Unlike dynamic `bytes` types, `bytes32` does not have a variable length, meaning `param.length` will always return `32`. As a result, the check `param.length == 0` is unnecessary and can be removed without affecting the function's correctness.

Recommendation

We recommend removing the redundant `param.length == 0` check from the `isValidParam()` function.

Alleviation

[**Zoo Finance Team** 17 Feb 2025]: Issue acknowledged. Changes have been reflected in the commit hash.

PTB-01 REDUNDANT CHECK FOR `_totalShares == 0` IN DIVISION OPERATION

Category	Severity	Location	Status
Code Optimization	Optimization	contracts/tokens/PToken.sol (base): 82	Resolved

Description

The issue lies in the redundant check if `(_totalSupply == 0 || _totalShares == 0)` before performing the multiplication and division operation. Specifically, the check for `_totalShares == 0` is unnecessary because if `_totalShares` is `0`, the division by `_totalSupply` would already result in a valid result

Recommendation

We recommend replacing the condition `(_totalSupply == 0 || _totalShares == 0)` with `(_totalSupply == 0)`. Since the check for `_totalShares == 0` is redundant and unnecessary, removing it simplifies the code without affecting the functionality.

Alleviation

[**Zoo Finance Team** 17 Feb 2025]: Issue acknowledged. Changes have been reflected in the commit hash.

VAL-03 REDUNDANT USE OF `_allEpochIds` FOR EPOCH TRACKING AND INEFFICIENT ACCESS METHODS

Category	Severity	Location	Status
Code Optimization, Gas Optimization	Optimization	contracts/vaults/Vault.sol (base): 45, 97, 101, 332~333	Resolved

Description

The issue lies in the usage of `_allEpochIds`, a `DoubleEndedQueue.Bytes32Deque` that tracks epoch `IDs`, which introduces redundancy and inefficiencies in the contract. In functions like `epochIdCount()` and `epochIdAt()`, the deque is accessed for operations that could be simplified. Specifically, `epochIdCount()` could directly return the current epoch count from `_currentEpochId`, and `epochIdAt()` could simply return the `index + 1` rather than converting and retrieving values from the deque. Additionally, pushing epoch `IDs` into `_allEpochIds` is unnecessary, as the current epoch `ID` can be tracked with `_currentEpochId` alone.

Recommendation

We recommend removing the `_allEpochIds` deque entirely, as it is redundant for tracking epoch `IDs`, and fixing the related functions accordingly.

Alleviation

[**Zoo Finance Team** 17 Feb 2025]: Issue acknowledged. Changes have been reflected in the commit hash.

VCB-02 | REDUNDANT ASSIGNMENT OF `deltaT = 0`

Category	Severity	Location	Status
Gas Optimization	Optimization	contracts/libs/VaultCalculator.sol (base): 47, 108	Resolved

Description

The issue lies in the redundant assignment of `deltaT = 0` within the else block, where it is already initialized to `0` before the conditional check. Since `deltaT` is set to `0` by default, assigning it again within the else block is unnecessary and does not affect the contract's behavior. This redundancy adds unnecessary code, making the logic less efficient and harder to maintain, while also potentially increasing gas costs due to the extra operation.

Recommendation

We recommend deleting the redundant line `deltaT = 0` in the `else` block, as it is unnecessary.

Alleviation

[**Zoo Finance Team** 17 Feb 2025]: Issue acknowledged. Changes have been reflected in the commit hash.

APPENDIX | ZOO FINANCE

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.
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.
Coding Issue	Coding Issue findings are about general code quality including, but not limited to, coding mistakes, compile errors, and performance issues.
Denial of Service	Denial of Service findings indicate that an attacker may prevent the program from operating correctly or responding to legitimate requests.
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.
Financial Manipulation	Financial Manipulation findings indicate issues in design that may lead to financial losses.

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.

Elevating Your Entire Web3 Journey

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.