CERTIK

BlockStreet Protocol

Security Assessment

CertiK Assessed on Aug 6th, 2025

CertiK Assessed on Aug 6th, 2025

BlockStreet Protocol

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

Executive Summary

TYPES	ECOSYSTEM	METHODS
Lending	EVM Compatible	Manual Review, Static Analysis, Testnet Deployment

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 08/06/2025	N/A

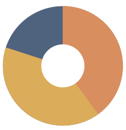
CODEBASE
https://github.com/BlockStreet-finance/BlockStreet-protocol
View All in Codebase Page

COMMITS
ec4ffb88c5df589afc7f4772887bd80393443161
View All in Codebase Page

Highlighted Centralization Risks

Transfers can be paused

Vulnerability Summary



5	2	0	3	0
Total Findings	Resolved	Partially Resolved	Acknowledged	Declined

1	Centralization	1 Acknowledged		Centralization findings highlight privileged roles & functions and their capabilities, or instances where the project takes custody of users' assets.
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 Resolved		Major risks may include logical errors that, under specific circumstances, could result in fund losses or loss of project control.
2	Medium	1 Resolved, 1 Acknowledged		Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
0	Minor			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.

1	Informational	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 | BLOCKSTREET PROTOCOL

CODEBASE | BLOCKSTREET PROTOCOL

Repository

https://github.com/BlockStreet-finance/BlockStreet-protocol

Commit

ec4ffb88c5df589afc7f4772887bd80393443161

AUDIT SCOPE | BLOCKSTREET PROTOCOL

3 files audited ● 2 files with Acknowledged findings ● 1 file without findings

ID	Repo	File	SHA256 Checksum
BBB	BlockStreet-finance/BlockStreet-protocol	src/Blotroller.sol	fedfe3653e404dc5f0a759341a4237d5454f4f1bb7a32759304f542f6a3cfc55
BPB	BlockStreet-finance/BlockStreet-protocol	src/BloPriceOracle.sol	6d3d3bd94e4424c3bcb4c941d3b15faed063e5aa665aa886178613b06bca5d08
BSS	BlockStreet-finance/BlockStreet-protocol	src/BlotrollerStorage.sol	4c0fb506fef7a2ca04bcfd14e82bc68d28642e746051df7b62ab388485df7317

APPROACH & METHODS | BLOCKSTREET PROTOCOL

This report has been prepared for BlockStreet to discover issues and vulnerabilities in the source code of the BlockStreet Protocol 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, Static Analysis, and Testnet Deployment techniques.

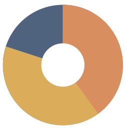
The auditing process pays special attention to the following considerations:

- 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 | BLOCKSTREET PROTOCOL



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

This report has been prepared to discover issues and vulnerabilities for BlockStreet Protocol. Through this audit, we have uncovered 5 issues ranging from different severity levels. Utilizing the techniques of Manual Review, Static Analysis & Testnet Deployment to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
BLP-01	**Centralization Risks**	**Centralization**	**Centralization**	**Acknowledged**
BLP-02	Cross-Bucket Redemption Logic Is Flawed	Logical Issue	Major	Resolved
BLP-03	Pyth Confidence Intervals Is Ignored	Logical Issue	Medium	Resolved
BLP-04	Empty Pool Attack Vector Of Compound V2 Forks	Volatile Code, Financial Manipulation	Medium	Acknowledged
BLP-05	External Functions With `_` As Name Prefix	Coding Style	Informational	Acknowledged

BLP-01 | CENTRALIZATION RISKS

Category	Severity	Location	Status
Centralization	**Centralization**	**src/BloPriceOracle.sol: 148; src/Blotroller.sol: 1009, 1033, 1051, 1092, 1116, 1153, 1171, 1189, 1206, 1216, 1226, 1235, 1244, 1290, 1318**	**Acknowledged**

Description

In the contract `Blotroller`, the privileged `admin` role has authority over the following functions:

- `_setPriceOracle`
- `_setCollateralFactor`
- `_setLiquidationIncentive`
- `_supportMarket`
- `_setPauseGuardian`
- `_setTokenType`
- `_setSeparationMode`

The privileged `admin` and `pauseGuardian` roles have authority over the following functions:

- `_setMintPaused`
- `_setBorrowPaused`
- `_setTransferPaused`
- `_setSeizePaused`

In the `BloPriceOracle` contract, the `Owner` role has authority over the `setAssetConfigs()` function.

Any compromise to the privileged account may allow the hacker to take advantage of this authority and change critical configurations in the contracts, such as pausing / unpausing the protocol, and changing `bToken` asset configuration.

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

[BlockStreet, 08/05/2025]: We acknowledge the centralized risk, we plan to transition these privileged operations to a multi-signature wallet combined with a timelock mechanism. This will significantly reduce the risk of single-point failure or compromise and enhance the protocol's security and transparency

BLP-02 | CROSS-BUCKET REDEMPTION LOGIC IS FLAWED

Category	Severity	Location	Status
Logical Issue	Major	src/Blotroller.sol: 880~915	Resolved

Description

The Blotroller contract implements an asset separation mode, where assets are classified as either TYPE_A (e.g., liquid assets like stablecoins) or TYPE_B (e.g., equity assets like tokenized stocks). The core design principle of this mode is to isolate risk by ensuring that TYPE_A collateral can only be used to back TYPE_B debt. When A/B separation mode is enabled, a user's TYPE A collateral is intended to secure TYPE B borrows (and vice-versa).

During a hypothetical liquidity check, the helper `_applySeparatedEffects()` should therefore reduce `sumCollateralA` or add `sumBorrowPlusEffectsB`, when the user tries to redeem TYPE A bTokens, because that action removes collateral that backs the opposite bucket.

Instead, the current implementation adds the redemption value to `sumBorrowPlusEffectsA`, which is compared against TYPE B collateral later.

```
        if (tokenType == TokenType.TYPE_A) {
            sepVars.sumBorrowPlusEffectsA += mul_ScalarTruncate(
                mul_(mul_(Exp({mantissa: collateralFactorMantissa}), Exp({mantissa: exchangeRateMantissa})), Exp({mantissa: oraclePriceMantissa})),
                redeemTokens
            );
            sepVars.sumBorrowPlusEffectsA += mul_ScalarTruncate(Exp({mantissa: oraclePriceMantissa}), borrowAmount);
```

Because `sumCollateralA` is left unchanged, the redemption will pass the check even though it removes real collateral from the B bucket. An attacker can:

1. Supply large TYPE A collateral.
2. Borrow TYPE B tokens up to the maximum allowed.
3. Redeem most of the TYPE A collateral in the same block, passing `redeemAllowedInternal`.
4. Leave the protocol with an under-collateralised TYPE B debt that cannot be liquidated until prices move, effectively siphoning value.

Proof of Concept

This PoC uses minimal mock contracts (ERC-20, Oracle) for clarity:

```
import { expect } from "chai";
import { ethers } from "hardhat";
import { Signer } from "ethers";

describe("Cross-bucket redemption", () => {
  let admin: Signer, user: Signer;
  let blotroller: any;
  let tokenA: any, tokenB: any;
  let bTokenA: any, bTokenB: any;
  let oracle: any;

  async function setPrice(asset: string, price = ethers.constants.WeiPerEther) {
    await oracle.setPrice(asset, price);
  }

  beforeEach(async () => {
    [admin, user] = await ethers.getSigners();

    const Oracle = await ethers.getContractFactory("MockOracle");
    oracle = await Oracle.deploy();
    await oracle.deployed();

    const Blotroller = await ethers.getContractFactory("Blotroller");
    blotroller = await Blotroller.deploy();
    await blotroller.deployed();
    await blotroller._setPriceOracle(oracle.address);

    const ERC20 = await ethers.getContractFactory("MockERC20");
    tokenA = await ERC20.deploy("TestA", "A");
    tokenB = await ERC20.deploy("TestB",  "B");
    await Promise.all([tokenA.deployed(), tokenB.deployed()]);

    const BToken = await ethers.getContractFactory("BToken");
    bTokenA = await MockBToken.deploy(tokenA.address, blotroller.address);
    bTokenB = await MockBToken.deploy(tokenB.address, blotroller.address);
    await Promise.all([bTokenA.deployed(), bTokenB.deployed()]);

    await blotroller._supportMarket(bTokenA.address);
    await blotroller._supportMarket(bTokenB.address);
    await blotroller._setCollateralFactor(bTokenA.address,
ethers.utils.parseUnits("0.75", 18));
    await blotroller._setCollateralFactor(bTokenB.address,
ethers.utils.parseUnits("0.75", 18));
    await blotroller._setTokenType(bTokenA.address, 1);
    await blotroller._setTokenType(bTokenB.address, 2);

    await blotroller._setSeparationMode(true);

    await tokenA.mint(await user.getAddress(), ethers.utils.parseEther("1000"));
```

```
    await tokenB.mint(bTokenB.address,  ethers.utils.parseEther("100000"));
    await tokenA.connect(user).approve(bTokenA.address,
ethers.constants.MaxUint256);
    await tokenB.connect(user).approve(bTokenB.address,
ethers.constants.MaxUint256);

    await setPrice(tokenA.address);
    await setPrice(tokenB.address);
  });

  it("allows redemption that should have been blocked", async () => {
    await bTokenA.connect(user).mint(ethers.utils.parseEther("1000"));
    await bTokenB.connect(user).borrow(ethers.utils.parseEther("750"));
    await expect(
      bTokenA.connect(user).redeemUnderlying(ethers.utils.parseEther("900"))
    ).to.not.be.reverted;
    // Check liquidity after redemption
    const res = await blotroller.getAccountLiquiditySeparated(await
user.getAddress());
    const shortfallB = res[4];                          //shortfall
    expect(shortfallB).to.be.gt(0, "Account now under-collateralised");
    console.log("TYPE B shortfall:", ethers.utils.formatEther(shortfallB));
  });
});
```

Recommendation

It's recommended that the team fix the accounting so that redeeming TYPE A tokens reduces the collateral available to cover TYPE B borrows.

Alleviation

[BlockStreet, 08/06/2025]: The team heeded the advice and resolved the issue in commit 2101537924c532fc36215f78d794b131b5bd8391

BLP-03 | PYTH CONFIDENCE INTERVALS IS IGNORED

Category	Severity	Location	Status
Logical Issue	Medium	src/BloPriceOracle.sol: 125~136	Resolved

Description

The function `_fetchPythPrice()` ignores Pyth's confidence intervals in price calculations which will cause asset mispricing for the protocol as attackers can exploit price extremes during high volatility periods, potentially leading to incorrect liquidations and system insolvency.

```
125      try pyth.getPriceNoOlderThan(config.pythPriceId, config.maxPriceAge) returns (
126              PythStructs.Price memory pythPrice
127          ) {
128              if (pythPrice.price <= 0) {
129                  return (0, 0);
130              }
131              // Normalize price to our internal 6-decimal format.
132              price = PythUtils.convertToUint(pythPrice.price, pythPrice.expo, uint8(INTERNAL_PRICE_DECIMALS));
133              /*if (price == 0) {
134                  return (0, 0);
135              }*/
136              timestamp = pythPrice.publishTime;
137          } catch {
138              return (0, 0);
139          }
```

According to Pyth docs:

"Pyth publishes both a price and a confidence interval for each product... The aggregate confidence interval (μ-σ, μ+σ) is a good estimate of a range in which the true price lies."

Recommendation

Pyth (https://docs.pyth.network/price-feeds/best-practices#confidence-intervals) recommends to check the confidence interval the price feed:

"It is recommended to use the confidence interval to protect your users from these unusual market conditions... For lending/collateral scenarios, protocols should use price-confidence when valuing collateral and price+confidence when valuing borrowed positions."

Alleviation

[BlockStreet, 08/06/2025]: The team heeded the advice and resolved the issue. Changes have been reflected in commit 7b225fcafea3dd0cc48a3900bf9c0e30a5072a3b.

Added maxConfidenceRatio field to AssetConfig struct and implemented confidence interval validation in _fetchPythPrice() function. The oracle now:

1. Calculates confidence ratio as (confidence * 10000) / price in basis points
2. Falls back to Chainlink when Pyth confidence exceeds the configured threshold
3. Validates maxConfidenceRatio <= 10000 (100%) during configuration

This addresses the security concern by ensuring price feeds with excessive uncertainty are rejected, preventing exploitation during high volatility periods as recommended by Pyth documentation.

BLP-04 | EMPTY POOL ATTACK VECTOR OF COMPOUND V2 FORKS

Category	Severity	Location	Status
Volatile Code, Financial Manipulation	● Medium	src/BErc20.sol: 147~150; src/BToken.sol: 293~312	● Acknowledged

Description

The empty pool attack is a known vulnerability of Compound V2 forks. While the relevant contracts are outside the scope of the audit, the auditors would like to point out the attack vector to the project team.

In a newly created pool, an attacker can significantly inflate the the value of `BToken` by minting a very small amount of `BToken` and then directly transferring the underlying token to the contract. Due to the usage of `token.balanceOf(address(this))` in the `getCashPrior()` function which is used in the `exchangeRateStoredInternal()` calculation, and a very small `_totalSupply` that the attacker has minted, the exchange rate could be very high. With inflated `BToken` value, the attacker can use its `BToken` balance to borrow large amount of assets from different pools. The hundred finance post mortem includes a detailed description of a real life exploit using this attack vector: https://blog.hundred.finance/15-04-23-hundred-finance-hack-post-mortem-d895b618cf33

Recommendation

The project team should initialize pools with sufficient liquidity at deployment. This would prevent attackers from being the first to interact with an empty pool and minting a disproportionate share of pool tokens at the manipulated exchange rate.

Alleviation

[BlockStreet, 08/05/2025]: We are aware of the empty pool attack vector and acknowledge the associated risks as described in the Hundred Finance post mortem. To mitigate this issue, we will ensure that each pool is initialized with a predefined amount of initial liquidity prior to public deployment. This precaution prevents the manipulation of the exchange rate due to a low total supply, effectively addressing this attack surface.

BLP-05 | EXTERNAL FUNCTIONS WITH `_` AS NAME PREFIX

Category	Severity	Location	Status
Coding Style	● Informational	src/Blotroller.sol: 1290, 1318	● Acknowledged

Description

Functions with names starting with `_` should be `private` / `internal` functions.

Recommendation

Consider removing `_` from the start of the external function name.

APPENDIX | BLOCKSTREET PROTOCOL

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.
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.

