



Auditing Report

Hardening Blockchain Security with Formal Methods

FOR



GlazeCorp Coin Launchpad



Veridise Inc.
September 7, 2026

► **Prepared For:**

GlazeCorp
<https://glazecorp.io>

► **Prepared By:**

Alberto Gonzalez
Aayushman Thapa Magar

► **Contact Us:**

contact@veridise.com

► **Version History:**

September 15, 2026 V1
September 17, 2026 V2

Contents

Contents	iii
1 Executive Summary	1
2 Project Dashboard	3
3 Security Assessment Goals and Scope	4
3.1 Security Assessment Goals	4
3.2 Security Assessment Methodology & Scope	4
3.3 Classification of Vulnerabilities	5
4 Trust Model	6
4.1 Operational Assumptions	6
4.2 Privileged Roles	6
5 Vulnerability Report	8
5.1 Detailed Description of Issues	9
5.1.1 V-GLZC-VUL-001: No minimum USDG amount on Strategy buy	9
5.1.2 V-GLZC-VUL-002: Direct USDG transfers inflate Strategy Purchased events	10
5.1.3 V-GLZC-VUL-003: Maintainability Concerns	11

About Veridise

Veridise is an independent security firm founded by academics and security researchers with deep expertise in formal methods, programming languages, and applied cryptography. The firm focuses on high-assurance security for blockchain and cryptographic systems, combining rigorous theory with practical adversarial analysis.

Veridise provides full-stack blockchain security services spanning smart contracts, zero-knowledge circuits and proving systems, consensus and execution clients, cryptographic libraries, and supporting infrastructure. The team routinely analyzes systems at the boundary between protocol design, implementation, and cryptography, where failures are both subtle and high-impact. To date, Veridise has conducted hundreds of security assessments for protocols securing billions of dollars in TVL.

Dynamic Security Veridise treats security as an ongoing engineering discipline rather than a one-time validation exercise. Effective security requires iterative review, continuous feedback, and close collaboration between auditors and system designers. Throughout the engagement, Veridise analysts communicated regularly with the GlazeCorp team using [AuditHub](#), enabling rapid clarification of design intent, prompt discussion of findings, and efficient validation of fixes.

In parallel with manual review, Veridise maintains active research programs focused on automated and formal analysis of blockchain systems. These efforts have produced tools such as *Vanguard* for static analysis and *Picus* for formal verification, which are available through [AuditHub](#). AuditHub integrates these techniques directly into CI/CD workflows, supporting continuous vulnerability detection and regression analysis beyond the scope of a single audit.

Additional details and case studies are available at <https://audithub.dev/case-studies/>.

Veridise Reports Veridise reports are published in the public audit archive linked below. Only reports obtained from this archive or confirmed directly by a representative of [Veridise](#) should be considered official Veridise reports.

<https://veridise.com/audits-archive/>



From Sep. 7, 2026 to Sep. 10, 2026, GlazeCorp engaged Veridise to conduct a security assessment of their GlazeCorp Coin Launchpad. Specifically, the assessment covered the path from how each launched coin is mined, through how it is used to signal strategies, to how holders receive a pro-rata share of the protocol's accumulated funds. Veridise conducted the assessment over 8 person-days, with 2 security analysts reviewing the project over 4 days on commit af56858. The review strategy involved a tool-assisted analysis of the program source code performed by Veridise security analysts as well as thorough code review.

Project Summary. The protocol is mainly divided into the following contracts:

- ▶ **IndexCoin:** The transferable ERC-20 created by each launch. Holders can lock it to choose which Strategies receive protocol revenue, or burn it to redeem a proportional share of that launch's treasury. After a one-time genesis mint into the launch liquidity pool, only Mine may create new supply.
- ▶ **Mine:** Issuance mechanism for each launched IndexCoin. It exposes a single Dutch auction slot: paying USDG* replaces the current miner, and the incumbent receives newly minted IndexCoin over their tenure. Each paid replacement sends 80% of the USDG to the outgoing miner, small creator and protocol fees, and the remainder into Resonance as protocol revenue.
- ▶ **Resonance:** This is the signaling and revenue hub. Holders allocate IndexCoin to live Strategies, and Resonance streams incoming USDG over seven days to those Strategies in proportion to their signal weight.
- ▶ **Strategy:** Each Strategy is a Dutch auction that sells USDG accrued from Resonance for a configured payment token. A purchase sends that token to the Fund and, optionally, a bounded share to the Strategy's Bribe.
- ▶ **Fund:** This is the treasury of assets acquired through Strategy purchases. IndexCoin holders redeem by burning their token balance and taking a pro-rata share of the Fund's assets.
- ▶ **Bribe:** Each Strategy has a paired Bribe that streams reward tokens over seven days to accounts signaling that Strategy. Automatic funding comes from the Strategy purchase share; additional reward tokens can be registered and notified separately.

Code Assessment. The GlazeCorp Coin Launchpad developers provided the contracts' source code for review. The codebase appears to be largely original, although portions of the auction logic in the Mine and Strategy contracts are mentioned to have been adapted from Euler Fee Flow, with the Strategy contract also crediting Liquid Signal Governance.

The codebase contains substantial documentation in the form of READMEs and documentation comments on contracts, functions, and storage variables. To facilitate the Veridise security analysts' understanding of the codebase, the GlazeCorp Coin Launchpad developers also provided a helpful walkthrough of the protocol's main components and workflows during a meeting.

* USD-pegged stablecoin, for more information visit: <https://docs.paxos.com/guides/stablecoin/usdg>

The source code included a comprehensive test suite covering the scoped contracts, cross-contract workflows, adversarial scenarios, fuzzed inputs, and stateful invariants.

Summary of Issues Detected. The security assessment uncovered 3 issues, none of which were assessed to be of high or critical severity by the Veridise analysts. The review identified 1 low-severity issue ([V-GLZC-VUL-001](#)), which highlights the lack of a minimum USDG purchase amount, potentially causing strategy auction buyers to receive smaller USDG lots than expected. Additionally, the Veridise analysts identified 1 warning issue ([V-GLZC-VUL-002](#)), which highlights the possibility of an attacker misleading off-chain systems by manipulating the USDG purchased event. The review also flagged 1 informational finding ([V-GLZC-VUL-003](#)) involving minor maintainability recommendations.

Recommendations. After conducting the assessment of the protocol, the security analysts had a few suggestions to improve the GlazeCorp Coin Launchpad robustness.

Inline Bribe and Strategy deployments in Resonance. Since `BribeFactory` and `StrategyFactory` are stateless, consider deploying `Bribe`, `Strategy`, and `BribeRouter` directly within `Resonance's addStrategy`. This would eliminate two deployed contracts and reduce the protocol's maintenance burden.

Use interfaces as enforceable contract specifications As discussed in [V-GLZC-VUL-003](#), each contract should inherit its corresponding interface, which should declare all expected external functionality. Compiler-enforced compliance reduces integration errors and provides a stable specification for future maintenance, refactoring, and code reuse.

Disclaimer. We hope that this report is informative but provide no warranty of any kind, explicit or implied. The contents of this report should not be construed as a complete guarantee that the system is secure in all dimensions. In no event shall Veridise or any of its employees be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the results reported here.



Table 2.1: Application Summary.

Name	Version	Type	Platform
GlazeCorp Coin Launchpad	af56858	Solidity	Ethereum

Table 2.2: Engagement Summary.

Dates	Method	Consultants Engaged	Level of Effort
Sep. 7–Sep. 10, 2026	Manual & Tools	2	8 person-days

Table 2.3: Vulnerability Summary.

Name	Number	Acknowledged	Fixed
Critical-Severity Issues	0	0	0
High-Severity Issues	0	0	0
Medium-Severity Issues	0	0	0
Low-Severity Issues	1	1	0
Warning-Severity Issues	1	1	0
Informational-Severity Issues	1	1	1
TOTAL	3	3	1

Table 2.4: Category Breakdown.

Name	Number
Data Validation	1
Event Manipulation	1
Maintainability	1



3.1 Security Assessment Goals

The engagement was scoped to provide a security assessment of GlazeCorp Coin Launchpad's source code. During the assessment, the security analysts aimed to answer questions such as:

- ▶ Are privileged operations restricted to their intended authorities, and are permanent administrative transitions truly irreversible?
- ▶ Can IndexCoin be minted by an unauthorized party?
- ▶ Can a miner receive more or fewer IndexCoin emissions than intended for their tenure?
- ▶ Can a miner be unfairly replaced or replaced at an incorrectly calculated Dutch auction price?
- ▶ Can a miner prevent their replacement and retain the mining position indefinitely?
- ▶ Can an IndexCoin holder manipulate the redemption process to withdraw more assets from the Fund than they are entitled to?
- ▶ Can IndexCoin holders manipulate the signaling system to assign a strategy more signal weight than the IndexCoin they have committed?
- ▶ Can killing a Strategy forfeit previously accrued revenue or allow it to continue receiving signal weight or accruing revenue afterward?
- ▶ Can Strategy signalers receive more or fewer bribe rewards than intended for their signal weight and participation period?
- ▶ Do USDG and reward assets follow their intended flow across protocol contracts, with each transfer and split correctly routed and accounted for?

3.2 Security Assessment Methodology & Scope

Security Assessment Methodology. To address the questions above, the security assessment involved a combination of human experts and automated program analysis. In particular, the security assessment was conducted with the aid of the following techniques:

- ▶ *Manual review.* Security analysts performed a line-by-line review of the in-scope code and supporting documentation to evaluate design assumptions and identify logic flaws.
- ▶ *Static analysis.* To identify potential common vulnerabilities, security analysts leveraged Veridise's custom smart contract analysis tool [Vanguard](#) to detect classes of issues such as reentrancy, unchecked external calls, front-running, and flash loan vulnerabilities. Vanguard is designed to surface these issues with low false-positive rates and is run through [AuditHub](#).

Scope. The scope of the security assessment was limited to the following files of the source code provided by the GlazeCorp Coin Launchpad developers:

1. `packages/contracts/src/core/Bribe.sol`
2. `packages/contracts/src/core/BribeFactory.sol`
3. `packages/contracts/src/core/BribeRouter.sol`
4. `packages/contracts/src/core/Fund.sol`

5. packages/contracts/src/core/FundController.sol
6. packages/contracts/src/core/IndexCoin.sol
7. packages/contracts/src/core/Mine.sol
8. packages/contracts/src/core/Resonance.sol
9. packages/contracts/src/core/ResonanceRouter.sol
10. packages/contracts/src/core/SignalIndexCoin.sol
11. packages/contracts/src/core/Strategy.sol
12. packages/contracts/src/core/StrategyFactory.sol

Methodology. Veridise security analysts inspected the provided tests, and read the GlazeCorp Coin Launchpad documentation. They then began a review of the code assisted by both static analyzers and automated testing.

3.3 Classification of Vulnerabilities

When Veridise security analysts discover a possible security vulnerability, they must estimate its severity by weighing its potential impact against the likelihood that a problem will arise.

The severity of a vulnerability is evaluated according to the Table 3.1.

Table 3.1: Severity Breakdown.

	Somewhat Bad	Bad	Very Bad	Protocol Breaking
Not Likely	Info	Warning	Low	Medium
Likely	Warning	Low	Medium	High
Very Likely	Low	Medium	High	Critical

The likelihood of a vulnerability is evaluated according to the Table 3.2.

Table 3.2: Likelihood Breakdown

Not Likely	A small set of users must make a specific mistake
Likely	Requires a complex series of steps by almost any user(s) - OR - Requires a small set of users to perform an action
Very Likely	Can be easily performed by almost anyone

The impact of a vulnerability is evaluated according to the Table 3.3:

Table 3.3: Impact Breakdown

Somewhat Bad	Inconveniences a small number of users and can be fixed by the user
Bad	Affects a large number of people and can be fixed by the user - OR - Affects a very small number of people and requires aid to fix
Very Bad	Affects a large number of people and requires aid to fix - OR - Disrupts the intended behavior of the protocol for a small group of users through no fault of their own
Protocol Breaking	Disrupts the intended behavior of the protocol for a large group of users through no fault of their own



4.1 Operational Assumptions

In addition to assuming that any out-of-scope components behave correctly, Veridise analysts assumed the following properties held when modeling security for GlazeCorp Coin Launchpad.

- ▶ The highly privileged roles described below are assumed to behave honestly, within the privilege allotted to them.
- ▶ Strategies are not configured with payment tokens that fee-on-transfer, rebase, or expose multiple entry points*.

4.2 Privileged Roles

Roles. This section describes in detail the specific roles present in the system, and the actions each role is trusted to perform. The roles are grouped based on two characteristics: privilege-level and time-sensitivity. *Highly-privileged* roles may have a critical impact on the protocol if compromised, while *limited-authority* roles have a negative, but manageable impact if compromised. Time-sensitive *emergency* roles may be required to perform actions quickly based on real-time monitoring, while *non-emergency* roles perform actions like deployments and configurations which can be planned several hours or days in advance.

The privileged functionality in the reviewed contracts is concentrated in FundController, which is controlled by an external account. That account is a *limited-authority, non-emergency* role: it may add and kill Strategies, register additional Bribe reward tokens, and adjust bounded parameters such as the Strategy Bribe share and the Mine creator-fee recipient. It cannot pause the protocol, redirect revenue, or drain user or contract funds.

Operational Recommendations. Full validation of operational security practices is beyond the scope of this review. Users of the protocol should ensure they are confident that the operators of privileged keys are following best practices such as:

- ▶ Never storing a protocol key in plaintext, on a regularly used phone, laptop, or device, or relying on a custom solution for key management.
- ▶ Using separate keys for each separate function.
- ▶ Storing multi-sig keys in a diverse set of key management software/hardware services and geographic locations.
- ▶ Enabling 2FA for key management accounts. SMS should *not* be used for 2FA, nor should any account which uses SMS for 2FA. Authentication apps or hardware are preferred.
- ▶ Validating that no party has control over multiple multi-sig keys.
- ▶ Performing regularly scheduled key rotations for high-frequency operations.
- ▶ Securely storing physical, non-digital backups for critical keys.

* For more information about these behaviors visit: <https://github.com/d-xo/weird-erc20>

- ▶ Actively monitoring for unexpected invocation of critical operations and/or deployed attack contracts.
- ▶ Regularly drilling responses to situations requiring emergency response such as pausing/unpausing.

5

Vulnerability Report

This section presents the vulnerabilities found during the security assessment. For each issue found, the type of the issue, its severity, location in the code base, and its current status (i.e., acknowledged, fixed, etc.) is specified. Table 5.1 summarizes the issues discovered:

Table 5.1: Summary of Discovered Vulnerabilities.

ID	Description	Severity	Status
V-GLZC-VUL-001	No minimum USDG amount on Strategy buy	Low	Acknowledged
V-GLZC-VUL-002	Direct USDG transfers inflate Strategy ...	Warning	Acknowledged
V-GLZC-VUL-003	Maintainability Concerns	Info	Fixed

5.1 Detailed Description of Issues

5.1.1 V-GLZC-VUL-001: No minimum USDG amount on Strategy buy

Severity	Low	Commit	af56858
Type	Data Validation	Status	Acknowledged
Location(s)	src/core/Strategy.sol:160-161		
Confirmed Fix At	N/A		

Description A Strategy purchase lets the buyer cap how much payment token they will spend, which limits overpayment on that side of the trade. However, the other side is unprotected: the buyer cannot set a minimum amount of USDG they are willing to receive.

That can matter if the purchase is front-run:

1. An attacker buys the current lot.
2. The attacker sends 1 wei of USDG back to the Strategy so the next auction is not empty.
3. A new epoch starts at a reset price.
4. If that price is still within the victim's `maximumPayment`, the victim's transaction continues and they pay for essentially nothing (1 wei of USDG).

Callers who bind `expectedEpochId` to the epoch they observed before signing are largely covered: the front-run increments the epoch, so the victim's transaction reverts. The risk is sharper for contract integrators that buy on a user's behalf and read the current epoch at execution time, then pass that value in. Those callers attach to the attacker's new epoch, so the epoch check does not save the user.

Impact The buyer could end up paying the auction price while receiving less USDG than expected.

Recommendation Let the buyer specify a minimum USDG amount and revert if the lot is smaller.

Developers Response The developers have acknowledged issue and indicated they do not plan to provide a fix. The design decision to maintain the current function interface is made explicit under the protocol documentation.

5.1.2 V-GLZC-VUL-002: Direct USDG transfers inflate Strategy Purchased events

Severity	Warning	Commit	af56858
Type	Event Manipulation	Status	Acknowledged
Location(s)	src/core/Strategy.sol:175-176		
Confirmed Fix At	N/A		

Description When a Strategy auction is filled, the purchase emits a Purchased event. Among other fields, that event reports how much USDG the buyer received and how much of the Strategy's payment token they paid.

The lot is meant to be USDG the Strategy earned as part of the Resonance revenue. A purchase first pulls any pending Resonance revenue into the Strategy, then treats the Strategy's entire USDG balance as the amount sold. That balance can include USDG that never came from Resonance, for example, tokens sent directly to the Strategy.

An attacker can inflate the reported lot in a single transaction without losing the extra USDG. From a helper contract they send an arbitrary amount of USDG to the target Strategy (their own funds or a flash loan) and immediately buy the auction. They still pay the normal auction price, but they receive both the genuine Resonance revenue and the donated USDG. The donated amount therefore appears in Purchased as USDG acquired through the auction, while the attacker recovers it in the same transaction.

Impact Off-chain systems that treat the USDG amount in Purchased as revenue actually earned and sold through the Strategy auction can record washed USDG as legitimate purchase volume. How damaging that is depends on whether such listeners exist and on how they use the figure.

Recommendation Use the USDG received when pending Resonance revenue is pulled into the Strategy as the lot transferred to the buyer and reported in Purchased, rather than the Strategy's full USDG balance.

If selling the entire USDG balance remains the intended behavior, report both amounts in the event: the USDG pulled from Resonance, and any USDG that was already sitting in the Strategy. Listeners can then treat leftover or donated balances separately from protocol revenue.

Developers Response The developers have acknowledged issue and indicated they do not plan to provide a fix. This design decision is recorded under the protocol documentation.

5.1.3 V-GLZC-VUL-003: Maintainability Concerns

Severity	Info	Commit	af56858
Type	Maintainability	Status	Fixed
Location(s)	src/core/ ▶ Bribe.sol ▶ Mine.sol ▶ Resonance.sol:589-592		
Confirmed Fix At	741d05b0		

Description The following maintainability concerns were identified throughout the code-base:

1. The Mine and Strategy contracts comments and docs describe reverse Dutch auctions (price starts at the floor and rises until a seller accepts). The code implements standard Dutch auctions: price starts at the ceiling and falls until a buyer accepts.
2. The Mine, Bribe, Resonance and IndexCoin contracts do not inherit their associated interfaces under the interfaces/ directory. Those interfaces should be part of the contracts' inheritance trees.
3. The `_lastApplicableRevenueTime()` from Resonance and the `_lastApplicableRewardTime()` from Bribe implement equivalent behavior but with different logic. Standardizing the implementations would improve consistency and maintainability.

Impact The mentioned maintainability concerns may degrade code quality, negatively impact long term maintainability or mislead users.

Recommendation It is recommended to address the mentioned maintainability concerns.

Developers Response The developers implemented the suggested recommendations.